

HCL OneTest Performance
10.1.1 Documentation
September 2020



Special notice

Before using this information and the product it supports, read the information in [Notices on page mcccxl](#).

Contents

Special notice.....	ii	Module 4: Representing workloads.....	68
Chapter 1. Release Notes.....	13	Module 5: Running the test.....	69
Product description.....	13	Module 6: Evaluating results.....	70
What's new.....	13	Summary.....	71
Installing the product.....	14	Performance test an SAP application.....	72
Known issues.....	14	Introduction: Performance test SAP solutions.....	72
Contacting HCL support.....	14	Module 1: Creating projects and recording user actions.....	73
Chapter 2. System Requirements.....	16	Module 2: Editing an SAP test.....	74
Hardware.....	17	Module 3: Running tests.....	75
Operating systems.....	19	Summary.....	77
Host prerequisites.....	21	Performance test a Citrix application.....	78
Recording support.....	26	Introduction: Performance test Citrix applications.....	78
Supported software.....	31	Module 1: Creating projects and recording user actions.....	78
Chapter 3. Getting Started.....	37	Module 2: Editing a Citrix performance test.....	80
Task flows for performance testing.....	37	Module 3: Representing workloads.....	81
Task flow: HTTP record.....	37	Summary.....	83
Task flow: Generating workload.....	39	Chapter 5. Samples.....	84
Task flow: Recording a SAP test.....	41	Installation tuning tests for WebSphere® Application Server.....	84
Task flow: Recording a service test to test an SOA application.....	43	Installing the assets for tuning tests.....	85
Task flow: Recording a SAP batch input test.....	45	Testing with the Snoop test.....	85
Task flow: Integrating HCL OneTest™ Performance and HCL OneTest™ API.....	47	Testing the PlantsByWebSphere application.....	85
Task flow: Response time breakdown.....	49	Testing the Daytrader application.....	86
Task flow: Rate Scheduler.....	49	Chapter 6. Administrator Guide.....	87
Product overview.....	52	Installation of HCL OneTest™ Performance.....	87
HTTP performance testing overview.....	52	Installation requirements.....	87
SAP performance testing overview.....	52	Installation conventions and terminology.....	88
Citrix performance testing overview.....	53	Installation Manager overview.....	88
Service testing overview.....	55	Installation locations.....	89
Generic service client overview.....	56	Coexistence.....	90
Socket API performance testing overview.....	58	Eclipse instance overview.....	90
TN3270 performance testing overview.....	59	Increasing the number of file handles on Linux™ workstations.....	91
IBM® Engineering Test Management overview.....	59	Installation of the product by using Installation Manager.....	92
Streamlined Eclipse and full Eclipse overview.....	61	Uninstalling the product by using Installation Manager.....	99
Starting the product in full Eclipse mode.....	62	Installing the software by using stand-alone installer.....	99
Starting the product in streamlined Eclipse mode.....	62	License management.....	99
Chapter 4. Tutorials.....	64	License descriptions.....	100
Performance test a Web application.....	64	Applying the license.....	101
Introduction: Test a Web application.....	64	Collecting usage metrics data.....	103
Module 1: Creating projects and recording user actions.....	65		
Module 2: Editing a test.....	66		
Module 3: Validating a test with a single user.....	67		

Product upgrade and migration.....	104	Searching within tests.....	393
Migrating test assets to new version of the product.....	104	Exporting a test.....	396
Configuration of the product.....	105	Copying test assets with dependencies.....	396
Configuring the environment for SAP tests.....	105	Disabling portions of a test.....	399
Configuring the data collection infrastructure.....	107	Running test elements in random order.....	400
Configuring Docker containers.....	113	Renaming test assets.....	402
Integration with other products.....	114	Deleting test assets.....	403
Integration plugin compatibility matrix.....	114	Debugging custom code for tests and compound tests.....	405
Testing with Ant.....	115	Providing tests with variable data (datasets).....	405
Integration with Azure DevOps.....	118	Test variables.....	429
Integrating with Apache JMeter.....	124	Correlating response and request data.....	443
EGit integration.....	130	Data transformation.....	473
Integration with Engineering Test Management.....	131	Compound tests.....	483
Integration with IBM® Engineering Workflow Management.....	142	Creating a compound test.....	483
Integration with HCL Launch.....	145	Viewing compound tests.....	484
Testing with HCL OneTest™ API.....	154	Adding tests into a compound test.....	485
Integration of Jaeger with the product.....	167	Modifying a compound test.....	485
Testing with Jenkins.....	168	Running compound tests.....	486
Testing with Maven.....	174	Generating compound test result reports.....	486
Integrating and running performance test scripts in Micro Focus ALM.....	176	Adding a compound test to a Test Workbench project.....	488
Chapter 7. Test Author Guide.....	180	Adding compound tests to schedule.....	489
Creating tests.....	180	Simulating services with stubs.....	490
Performance testing tips.....	180	Service stub overview.....	490
Creating a project.....	181	Creating a service stub.....	491
Recording HTTP tests.....	182	Editing a service stub.....	492
Recording Citrix tests.....	209	Deploying service stubs.....	493
Recording service tests.....	225	Recording service stub activity in a log file.....	494
Recording socket and TN3270 tests.....	249	Setting log level for service stubs.....	495
Digital certificates overview.....	258	Sending service requests with the generic service client.....	495
Kerberos overview.....	267	Creating transport protocol configurations.....	495
Annotating a test during recording.....	271	Sending service requests with WSDL files.....	508
Recording sensitive session data.....	272	Sending HTTP endpoint requests.....	510
Splitting an HTTP test during recording.....	273	Sending a JMS endpoint request.....	511
Generating a new test from a recorded session.....	275	Sending a WebSphere® MQ endpoint request.....	512
Organizing test assets by type.....	275	Sending OData endpoint batch requests.....	514
Editing tests.....	276	Sending WebSphere Java MQ endpoint requests.....	515
Editing HTTP tests.....	276	Testing all operations in a WSDL file.....	518
Editing SAP tests.....	313	Viewing message content.....	519
Editing Citrix tests.....	324	Synchronizing a remote WSDL file.....	520
Editing service tests.....	337	Synchronizing a local WSDL file with GSC.....	521
Editing Socket tests.....	371	Adding static XML headers to a service request.....	521
Editing Kerberos tests.....	383		
Adding test elements.....	383		

Opening file attachments.....	522	Extending HCL OneTest™ Performance to support other protocols.....	688
Emulating workloads.....	522	Protocol extension structure.....	688
Schedule overview.....	523	Extending the test recorder.....	690
Creating a VU Schedule.....	524	Extending the test generation framework.....	702
Creating a Rate Schedule.....	542	Contributing annotations.....	708
Think time overview.....	549	Extending the load test behavior model.....	711
Working with agents.....	552	Extending data correlation	724
Adding a test to a schedule.....	556	Extending the test editor.....	728
Adding must run tests.....	556	Contributing error handlers.....	741
Assigning variables.....	557	Extending the schedule component.....	747
Defining performance requirements in schedules.....	558	Extending code generation.....	750
Repeating tests in a schedule.....	560	Extending the run-time environment.....	756
Creating rate generators in user groups.....	562	Extending the test log viewer.....	763
Running tests at a set rate.....	563	Extending evaluation results.....	764
Running tests in random order.....	564	Chapter 9. Test Manager Guide.....	773
Adding a transaction to a schedule.....	566	Evaluating results in web analytic reports.....	773
Emulating network traffic from multiple hosts.....	567	Comparing results among runs.....	773
Monitoring resource data.....	570	Comparing schedule stages.....	773
Resource Monitoring Service.....	591	Comparing results from various regions or agent locations.....	774
Monitoring response time breakdown.....	592	Generating functional test reports.....	775
Setting log and statistic levels.....	597	Publishing test results to the server.....	776
Chapter 8. Test Execution Specialist Guide.....	608	Customizing reports.....	778
Running schedules with performance testing.....	608	Export test results.....	792
Running a local schedule or test.....	608	Viewing response time breakdown.....	795
Running a long run mode SAP GUI test.....	609	Logs overview.....	797
Running long duration Citrix tests.....	610	Viewing test logs.....	799
Testing with Docker images.....	611	Viewing errors while running tests.....	799
Adjusting delays in HTTP tests.....	618	Viewing reports after a run.....	800
Setting a launch configuration.....	621	Accessing reports remotely.....	800
Running a configured schedule.....	622	Exporting test logs.....	801
Configuring multiple host names for a location.....	622	Exporting event log.....	802
Automating tests from command line.....	623	Exporting event console output.....	802
Controlling caches size.....	637	Viewing adjustments to page response times.....	802
Increasing memory allocation.....	638	Viewing resource monitoring data.....	803
Controlling execution of web analytic reports.....	639	Reports and counters.....	805
Debugging HTTP tests.....	644	Requirements report.....	805
Debugging Citrix tests.....	654	Synchronization Point report.....	806
Extending test execution with custom code.....	657	Loops report.....	806
Creating custom Java™ code.....	657	Agents Health Report.....	807
Test execution services interfaces and classes.....	660	Rate Runner report.....	808
Reducing the performance impact of custom code.....	662	Transaction report.....	809
Custom code examples.....	663	Transaction Percentile report.....	811
Migrating custom code from previous versions.....	687	Transaction Net Server Time Percentile report.....	812
		Rate Generator report.....	813
		HTTP performance test reports.....	814

SAP performance test reports.....	829	RPHE0117W.....	938
Citrix performance test reports.....	832	RPHE0118W.....	939
Web service performance test reports.....	837	RPHE0119E.....	940
Socket performance test reports.....	846	RPHE0120E.....	941
HTTP counters.....	847	RPHE0121E.....	942
SAP counters.....	866	RPHE0122W.....	943
Citrix counters.....	873	RPHE0123W.....	944
Service counters.....	883	RPHE0124W.....	945
Socket counters.....	904	RPIB0007E.....	945
Chapter 10. Troubleshooting Guide.....	908	RPKG0090E.....	945
Troubleshooting performance testing.....	908	RPKG0100E.....	946
Performance testing error messages.....	912	RPKG0101E.....	946
PRXE0101W.....	913	RPKG0110E.....	946
PRXE4943W.....	913	RPSF0172E.....	947
PRXE4951I.....	913	RPTA0001I.....	947
RMSE0003W.....	913	RPTA0002E.....	948
RMSE0004W.....	914	RPTA0003E.....	948
RMSE0005W.....	914	RPTA0004E.....	949
RMSE0006W.....	915	RPTA0009E.....	950
RPAC0001W.....	915	RPTA0010E.....	951
RPHD1032E.....	916	RPTA0011E.....	952
RPHD1034E.....	917	RPTA0012E.....	953
RPHE0001E.....	918	RPTA0013E.....	954
RPHE0010W.....	918	RPTA0014E.....	955
RPHE0011W.....	918	RPTA0015E.....	956
RPHE0012W.....	919	RPTA0016E.....	957
RPHE0013W.....	919	RPTA0017E.....	958
RPHE0014W.....	920	RPTA0018E.....	958
RPHE0100W.....	920	RPTA0019E.....	958
RPHE0101W.....	921	RPTA0020E.....	959
RPHE0102W.....	922	RPTA0021E.....	959
RPHE0103W.....	923	RPTA0023E.....	960
RPHE0104W.....	923	RPTA0024E.....	961
RPHE0105W.....	924	RPTA0025E.....	962
RPHE0106W.....	925	RPTA0026E.....	963
RPHE0107W.....	927	RPTA0025I.....	964
RPHE0108W.....	929	RPTA0026I.....	964
RPHE0109W.....	930	RPTA0027I.....	964
RPHE0110W.....	931	RPTA0031E.....	964
RPHE0111W.....	932	RPTA0032I.....	965
RPHE0112W.....	933	RPTA0033I.....	965
RPHE0113E.....	934	RPTA0034E.....	965
RPHE0113W.....	935	RPTA0035E.....	966
RPHE0114E.....	935	RPTA0036E.....	966
RPHE0114W.....	936	RPTA0037E.....	966
RPHE0115E.....	936	RPTA0038E.....	967
RPHE0115W.....	937	RPTA0039E.....	967

RPTA0040E.....	967	RPTI0142E.....	985
RPTA0041E.....	968	RPTI0143E.....	985
RPTA0042E.....	968	RPTI0144W.....	986
RPTA0043E.....	968	RPTI0145E.....	987
RPTA0100W.....	969	RPTI0146E.....	988
RPTA0518E.....	969	RPTJ0063E.....	988
RPTC0003E.....	970	RPTJ0075E.....	989
RPTC0004E.....	970	RPTJ1002E.....	989
RPTC0005E.....	971	RPTJ1003E.....	990
RPTC0006E.....	971	RPTJ1004E.....	990
RPTC0008I.....	972	RPTJ1005E.....	991
RPTC00020E.....	972	RPTJ1006E.....	992
RPTC1001W.....	972	RPTJ1007E.....	993
RPTC1002W.....	973	RPTJ1008E.....	995
RPTC1009I.....	973	RPTJ1009E.....	996
RPTC1011I.....	973	RPTJ1010E.....	996
RPTC1012I.....	973	RPTJ1011E.....	997
RPTC1013I.....	973	RPTJ1012E.....	998
RPTC1014I.....	974	RPTJ1013E.....	999
RPTC1015I.....	974	RPTJ1015E.....	1000
RPTC1016I.....	974	RPTJ1016E.....	1001
RPTC1017I.....	974	RPTJ1017E.....	1002
RPTC1018I.....	974	RPTJ1018E.....	1003
RPTC1019I.....	974	RPTJ1019E.....	1004
RPTC1020I.....	975	RPTJ1020E.....	1005
RPTC1021I.....	975	RPTJ1021E.....	1006
RPTC1030E.....	975	RPTJ0121I.....	1007
RPTC1031E.....	976	RPTJ1022E.....	1007
RPTC1032E.....	976	RPTJ1023E.....	1008
RPTE0005W.....	977	RPTJ1024E.....	1009
RPTE0011W.....	978	RPTJ1025I.....	1009
RPTE0147E.....	979	RPTJ1026I.....	1009
RPTH0130I.....	980	RPTJ1030E.....	1009
RPTH049E.....	980	RPTJ1041E.....	1010
RPTI0069E.....	981	RPTJ1042E.....	1010
RPTI0070E.....	981	RPTJ1043E.....	1011
RPTI0071I.....	981	RPTJ1044E.....	1012
RPTI0072E.....	982	RPTJ1100I.....	1013
RPTI0072I.....	982	RPTJ1101E.....	1013
RPTI0073E.....	982	RPTJ1102W.....	1014
RPTI0074E.....	982	RPTJ1103W.....	1015
RPTI0075E.....	983	RPTJ1104E.....	1015
RPTI0110I.....	983	RPTJ1141E.....	1016
RPTI0111I.....	983	RPTJ1142E.....	1016
RPTI0112I.....	983	RPTJ1200W.....	1016
RPTI0113I.....	983	RPTJ1220E.....	1017
RPTI0141E.....	984	RPTJ1221E.....	1017

RPTJ1240E.....	1018	RPTX0008E.....	1039
RPTJ1241E.....	1019	RPTX0009E.....	1039
RPTJ1242E.....	1019	RPTX0010E.....	1039
RPTJ1244E.....	1020	RPXD0022W.....	1040
RPTJ1245E.....	1021	RPXE0061I.....	1040
RPTJ1261E.....	1022	RPTX1010I.....	1041
RPTJ1270E.....	1022	RPTX1011I.....	1041
RPTJ1271E.....	1022	RPTX1012I.....	1041
RPTJ1280E.....	1023	RPTX1017I.....	1042
RPTJ1400I.....	1023	RPTX1018I.....	1042
RPTK0000I.....	1023	RPTX1019I.....	1042
RPTK1001E.....	1024	RPTX1081E.....	1042
RPTK1016E.....	1025	RPTX1082E.....	1043
RPTK1019E.....	1025	RPTX2001E.....	1043
RPTK1020E.....	1026	RPTX2002E.....	1044
RPTK1021E.....	1026	RPTX2003E.....	1044
RPTK1022E.....	1026	RPTX2004E.....	1045
RPTK1023E.....	1027	RPTX2005E.....	1045
RPTL0001W.....	1027	RPTX2006W.....	1045
RPTL0002W.....	1027	RPTX2007I.....	1046
RPTL0003W.....	1028	RPTX2008I.....	1046
RPTL0004W.....	1028	RPTX2009I.....	1046
RPTL0005W.....	1028	RPTX2010I.....	1046
RPTL0006W.....	1029	RPTX2011E.....	1047
RPTL0007W.....	1029	RPTX2012E.....	1048
RPTL0008E.....	1029	RPTX2013E.....	1048
RPTL0009I.....	1030	RPTX2014E.....	1049
RPTL0010E.....	1030	RPTX2015E.....	1049
RPTL0011E.....	1030	RPTX2016I.....	1050
RPTR0000W.....	1030	RPTX2017E.....	1050
RPTR0001W.....	1030	RPTX2018W.....	1050
RPTR0002W.....	1031	RPTX2019I.....	1050
RPTR0003W.....	1031	RPTX2020I.....	1050
RPTR0004W.....	1031	RPTX2021E.....	1050
RPTR2001E.....	1031	RPTX2022E.....	1051
RPTR2003W.....	1032	RPTX2023W.....	1051
RPTS1000E.....	1032	RPTX2024E.....	1052
RPTS1002E.....	1033	RPTX2025E.....	1052
RPTS1510E.....	1034	RPTX2026E.....	1053
RPTS1001I.....	1034	RPTX2027W.....	1053
RPTX0001E.....	1035	RPTX2029W.....	1054
RPTX0002E.....	1036	RPTX2030I.....	1054
RPTX0003E.....	1036	RPTX2031I.....	1055
RPTX0004E.....	1037	RPTX2032I.....	1055
RPTX0005E.....	1037	RPTX2033E.....	1056
RPTX0006E.....	1038	RPTX2034E.....	1056
RPTX0007E.....	1038	RPTX2035E.....	1057

RPTX2036E.....	1057	RPWF0130W.....	1080
RPTX2037E.....	1058	RPWF0131W.....	1080
RPTX2050E.....	1058	RPWF0132E.....	1080
RPTX2051E.....	1059	RPWF0140E.....	1081
RPTX2055E.....	1059	RPWH0007W.....	1081
RPTX2056E.....	1060	RPWH0009W.....	1081
RPTX2057E.....	1060	RPWH0010W.....	1081
RPTX2058E.....	1061	RPWH0012E.....	1081
RPTX2060E.....	1062	RPWH0014E.....	1081
RPTX2061W.....	1063	RPWH0015E.....	1082
RPTX2062W.....	1064	RPWH0016E.....	1082
RPTX2063W.....	1065	RPWH0017E.....	1082
RPTX2070E.....	1066	RPWS0001E.....	1083
RPTX2071E.....	1067	RPWS0002E.....	1083
RPTX2072E.....	1067	RPWS0003E.....	1083
RPTX2073E.....	1068	RPWS0004E.....	1084
RPTX2074E.....	1069	RPWS0005E.....	1084
RPTX2075E.....	1070	RPWS0006E.....	1084
RPTX2077E.....	1071	RPWS0007E.....	1085
RPWF0011E.....	1072	RPWS0008E.....	1085
RPWF0012E.....	1073	RPWY0002E.....	1085
RPWF0021E.....	1073	RPWY0003I.....	1086
RPWF0032E.....	1073	RPWY0004W.....	1086
RPWF0051E.....	1073	RPWY0005E.....	1086
RPWF0052E.....	1074	RPWY0006E.....	1086
RPWF0056E.....	1074	RPWY0007E.....	1086
RPWF0066E.....	1074	RPWZI0002E.....	1087
RPWF0071E.....	1075	RPXD0001E.....	1087
RPWF0072E.....	1075	RPXD0002E.....	1087
RPWF0074E.....	1075	RPXD0003E.....	1087
RPWF0075E.....	1075	RPXD0004E.....	1088
RPWF0076W.....	1076	RPXD0005E.....	1088
RPWF0081W.....	1076	RPXD0006E.....	1088
RPWF0082W.....	1076	RPXD0007F.....	1088
RPWF0083E.....	1077	RPXD0017W.....	1089
RPWF0084E.....	1077	RPXD0018E.....	1089
RPWF0085E.....	1077	RPXD0019E.....	1090
RPWF0101E.....	1077	RPXD0020E.....	1092
RPWF0102E.....	1078	RPXD0021E.....	1093
RPWF0103E.....	1078	RPXD0021W.....	1093
RPWF0104E.....	1078	RPXE0001W.....	1093
RPWF0111E.....	1078	RPXE0010W.....	1094
RPWF0112E.....	1079	RPXE0011W.....	1094
RPWF0121W.....	1079	RPXE0012W.....	1094
RPWF0122W.....	1079	RPXE0013W.....	1094
RPWF0123W.....	1079	RPXE0014W.....	1094
RPWF0124W.....	1080	RPXE0015W.....	1095

RPXE0016W.....	1095	RPXE2900E.....	1108
RPXE0017W.....	1095	RPXE2901W.....	1109
RPXE0018W.....	1096	RPXE4000W.....	1110
RPXE0019W.....	1096	RPXE4001E.....	1110
RPXE0021W.....	1096	RPXE4002E.....	1110
RPXE0023W.....	1096	RPXE4003E.....	1110
RPXE0024W.....	1096	RPXE4004E.....	1111
RPXE0025W.....	1097	RPXE4005E.....	1111
RPXE0027W.....	1097	RPXE4006E.....	1111
RPXE0028W.....	1097	RPXE4007E.....	1112
RPXE0029W.....	1097	RPXE4008E.....	1112
RPXE0030W.....	1097	RPXE4008I.....	1113
RPXE0031W.....	1097	RPXE4009I.....	1113
RPXE0033W.....	1098	RPXE4010I.....	1114
RPXE0035W.....	1098	RPXE4011E.....	1114
RPXE0036W.....	1098	RPXE4013I.....	1114
RPXE0037W.....	1098	RPXE4014E.....	1115
RPXE0038W.....	1099	RPXE4015E.....	1115
RPXE0039W.....	1099	RPXE4016E.....	1115
RPXE0040W.....	1099	RPXE4017I.....	1116
RPXE0041W.....	1099	RPXE4018E.....	1117
RPXE0042I.....	1100	RPXE4019E.....	1117
RPXE0043I.....	1100	RPXE4020E.....	1117
RPXE0044W.....	1100	RPXE4021E.....	1118
RPXE0045W.....	1100	RPXE4022E.....	1118
RPXE0046W.....	1101	RPXE4023E.....	1118
RPXE0047E.....	1101	RPXE4024E.....	1119
RPXE0048W.....	1101	RPXE4025E.....	1119
RPXE0049W.....	1102	RPXE4026E.....	1119
RPXE0050W.....	1102	RPXE4027E.....	1119
RPXE0051W.....	1102	RPXE4028E.....	1120
RPXE0052W.....	1103	RPXE4029E.....	1120
RPXE0053W.....	1103	RPXE4050I.....	1121
RPXE0054W.....	1103	RPXE4100W.....	1121
RPXE0055W.....	1103	RPXE4101E.....	1121
RPXE0056W.....	1104	RPXE4102E.....	1122
RPXE0057E.....	1104	RPXE4103E.....	1122
RPXE0058E.....	1104	RPXE4104E.....	1122
RPXE0059E.....	1104	RPXE4105E.....	1123
RPXE0060E.....	1105	RPXE4106E.....	1123
RPXE0100W.....	1105	RPXE4107E.....	1123
RPXE0102W.....	1105	RPXE4108E.....	1124
RPXE0103W.....	1106	RPXE4109E.....	1124
RPXE0104W.....	1106	RPXE4110E.....	1124
RPXE2501E.....	1107	RPXE4111W.....	1125
RPXE2550E.....	1108	RPXE4112W.....	1125
RPXE2552I.....	1108	RPXE4120E.....	1126

RPXE4150E.....	1126	RPXE4931I.....	1142
RPXE4151E.....	1126	RPXE4932I.....	1142
RPXE4152E.....	1126	RPXE4940I.....	1143
RPXE4153E.....	1127	RPXE4941I.....	1143
RPXE4200W.....	1127	RPXE4942I.....	1143
RPXE4201W.....	1127	RPXE4944W.....	1143
RPXE4202E.....	1128	RPXE4945W.....	1144
RPXE4203E.....	1128	RPXE4948W.....	1144
RPXE4204W.....	1128	RPXE4950I.....	1144
RPXE4205E.....	1129	RPXE4952E.....	1144
RPXE4208E.....	1129	RPXE5301E.....	1145
RPXE4209I.....	1129	RPXE5305E.....	1145
RPXE4210E.....	1130	RPXE5330E.....	1145
RPXE4211E.....	1130	RPXE5500W.....	1146
RPXE4212E.....	1130	RPXE5501W.....	1147
RPXE4213E.....	1131	RRIT0001E.....	1148
RPXE4214W.....	1131	RRIT0002E.....	1148
RPXE4215E.....	1131	RRIT0003E.....	1149
RPXE4215I.....	1132	RRIT0004E.....	1149
RPXE4216E.....	1132	RRIT0005E.....	1149
RPXE4217E.....	1133	RRITUI1002W.....	1150
RPXE4218E.....	1134	DCRC0001E.....	1150
RPXE4219E.....	1134	DCRC0002E.....	1150
RPXE4220E.....	1135	DCRC0003E.....	1151
RPXE4221E.....	1135	DCRC0008W.....	1151
RPXE4900I.....	1136	DCRC0009W.....	1151
RPXE4901I.....	1136	DCRC0010E.....	1152
RPXE4902I.....	1136	DCUI0001E.....	1152
RPXE4903I.....	1136	DCUI0003E.....	1152
RPXE4904I.....	1136	DCUI0004E.....	1152
RPXE4905I.....	1137	DCUI0006E.....	1152
RPXE4906I.....	1137	DCUI0007W.....	1153
RPXE4907I.....	1137	DCUI0008W.....	1153
RPXE4908I.....	1137	DCUI0009E.....	1153
RPXE4909I.....	1137	DCUI0010E.....	1153
RPXE4910I.....	1137	DCUI0011E.....	1154
RPXE4911I.....	1138	DCUI0012E.....	1154
RPXE4912I.....	1138	DCUI0013E.....	1154
RPXE4913I.....	1138	DCUI0014E.....	1155
RPXE4914I.....	1138	DCUI0015E.....	1155
RPXE4915I.....	1139	DCUI0016E.....	1155
RPXE4916I.....	1139	DCUI0017E.....	1155
RPXE4917I.....	1139	DCUI0998E.....	1155
RPXE4918I.....	1139	Additional error messages.....	1155
RPXE4920I.....	1140	Address already in use.....	1156
RPXE4921I.....	1141	Browser profile in use.....	1157
RPXE4930I.....	1142	Cannot open test.....	1158

Connection closed.....	1159
Dataset accessed using different modes...	1159
Error binding to port.....	1160
Error connecting to license server.....	1160
No local agent controller.....	1161
Performance Test Errors were found in the project.....	1162
Test run aborted.....	1163
Test run aborted due to error.....	1163
Testgen completed with warnings.....	1164
Variable not initialized.....	1165
Chapter 11. Reference Guide.....	1166
Accessibility features.....	1166
Keyboard shortcuts for performance and service testing.....	1166
General reference for performance testing.....	1168
Data correlation rules.....	1168
Error conditions.....	1169
Resource monitoring data sources.....	1171
Response time breakdown data sources...	1172
UI preferences.....	1173
HTTP preferences.....	1173
SAP test preferences.....	1179
Citrix recorder preferences.....	1182
Socket Test Generation preferences.....	1187
Citrix monitoring panel reference.....	1189
Proxy recording preferences.....	1190
Test editor preferences.....	1190
Report preferences.....	1192
Test editor references.....	1194
VU Schedule editor reference.....	1226
WSDL security editor reference.....	1233
Security Considerations.....	mccxxxix
Notices.....	mccxli
Index.....	1244

Chapter 1. Release notes for HCL OneTest™ Performance

This document contains information about what's new, installation, and known issues in HCL OneTest™ Performance and contact information of HCL Customer Support.

Contents

- [Product description on page 13](#)
- [What's new on page 13](#)
- [Installing the product on page 14](#)
- [Known issues on page 14](#)
- [Contacting HCL support on page 14](#)

Product description

You can find the description of HCL OneTest™ Performance.

HCL OneTest™ Performance is a scripting-free environment for automating load and scalability testing of web, ERP, and server-based software applications. HCL OneTest™ Performance provides rich and customizable reporting to help you identify the presence and cause of system bottlenecks. It captures the network traffic that is rendered when the application under test interacts with a server. This network traffic is then emulated on multiple virtual users while playing back the test. See the [Product overview on page 52](#).

What's new

You can find information about the features introduced in this release of HCL OneTest™ Performance.

• HCL Launch integration

You can now integrate HCL OneTest™ Performance with HCL Launch by using the HCL OneTest™ Performance Launch plugin to run tests. See [Integration with HCL Launch on page 145](#).

• Enhancements to HCL OneTest™ Performance integrations

- You can now use the Resource Monitoring Override Labels parameter when you choose to run tests from other applications such as Jenkins, Ant, Maven, Micro Focus Application Lifecycle Management (ALM), Azure DevOps, and UrbanCode Deploy (UCD). See [Integration with other applications on page 114](#).
- After you configure the HCL OneTest™ Server URL and if the **Publish result after execution** field is set to **Always**, then when you run tests from other applications, you can now view hyperlinks that point to the corresponding test reports as follows:

Applications	Location of the report
Jenkins	Console page

Applications	Location of the report
Ant	Log file
Maven	Console page
Micro Focus Application Lifecycle Management	Output window
Azure DevOps	Task page
UrbanCode Deploy	Output log

- **Bug fixes**

Defects reported by clients and internal defects are fixed.

Installing the product

You can find information about the installation and upgrade instructions for HCL OneTest™ Performance.

For installation instructions, see [Installation of the product by using Installation Manager on page 92](#).



Note: You cannot upgrade to the latest version of the product. You must uninstall the existing version of the product before installing the latest version of the product.

Known issues

You can find information about the known issues identified in this release of HCL OneTest™ Performance.

Table 1.

Product	Download document	Knowledge Base
HCL OneTest™ Performance	Release document	Knowledge articles

The knowledge base is continually updated as problems are discovered and resolved. By searching the knowledge base, you can quickly find workarounds or solutions to problems.

Contacting HCL support

You can find information about HCL technical support assistance for HCL OneTest™ Performance.

- For technical assistance, contact [HCL Customer Support](#).
- Before you contact HCL support, you must gather the background information that you might need to describe your problem. When you describe a problem to the HCL support specialist, be as specific as possible and

include all relevant background information so that the specialist can help you solve the problem efficiently.

To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, or messages that are related to the problem?
- Can you reproduce the problem? If so, what steps do you take to reproduce it?
- Is there a workaround for the problem? If so, be prepared to describe the workaround.

Chapter 2. System Requirements

This document includes information about hardware and software requirements for HCL OneTest™ Performance.

Contents

- [Hardware on page 17](#)
 - [Linux on page 17](#)
 - [Mac on page 18](#)
 - [Windows on page 18](#)
- [Operating Systems on page 19](#)
 - [Linux on page 19](#)
 - [Mac on page 20](#)
 - [Windows on page 20](#)
- [Host prerequisites on page 21](#)
 - [Java SDK on page 21](#)
 - [Terminal services on page 22](#)
 - [Virtualization Management on page 23](#)
 - [Web Browsers on page 23](#)
- [Recording support on page 26](#)
- [Supported software on page 31](#)
 - [Application servers on page 31](#)
 - [Business process management on page 32](#)
 - [Development tools on page 33](#)
 - [DevOps tools on page 35](#)

Disclaimers

This report is subject to the Terms of Use and the following disclaimers:

The information contained in this report is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied, including but not limited to the implied warranties of merchantability, non-infringement, and fitness for a particular purpose. In addition, this information is based on HCL's current product plans and strategy, which are subject to change by HCL without notice. HCL shall not be responsible for any direct, indirect, incidental, consequential, special or other damages arising out of the use of, or otherwise related to, this report or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from HCL or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of HCL software.

References in this report to HCL products, programs, or services do not imply that they will be available in all countries in which HCL operates. Product release dates and/or capabilities referenced in this presentation may change at any time at HCL's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Discrepancies found between reports and other

HCL documentation sources may or may not be attributed to different publish and refresh cycles for this tool and other sources. Nothing contained in this report is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results. You assume sole responsibility for any results you obtain or decisions you make as a result of this report.

Notwithstanding the Terms of Use users of this site are permitted to copy and save the reports generated from this tool for such users own internal business purpose. No other use shall be permitted.

Hardware

You can find information about the hardware requirements for HCL OneTest™ Performance.

Contents

- [Linux on page 17](#)
- [Mac on page 18](#)
- [Windows on page 18](#)

Linux

Hardware	Components	Requirement	Notes
Disk space	<ul style="list-style-type: none"> • HCL OneTest™ Performance Agent • HCL OneTest™ Performance 	10 GB	<ul style="list-style-type: none"> • An additional 500 MB of disk space is required in the /tmp directory.
Memory	<ul style="list-style-type: none"> • HCL OneTest™ Performance Agent • HCL OneTest™ Performance 	8 GB	
Processor	<ul style="list-style-type: none"> • HCL OneTest™ Performance Agent • HCL OneTest™ Performance 	Minimum: 1.86 GHz Intel Pentium 4 or higher	

Mac

Hardware	Components	Requirement	Notes
Disk space	<ul style="list-style-type: none"> • HCL OneTest™ Performance Agent • HCL OneTest™ Performance 	10 GB	
Memory	<ul style="list-style-type: none"> • HCL OneTest™ Performance Agent • HCL OneTest™ Performance 	8 GB	
Processor	<ul style="list-style-type: none"> • HCL OneTest™ Performance Agent • HCL OneTest™ Performance 	Minimum: 1.86 GHz Intel Pentium 4 or higher	

Windows

Hardware	Components	Requirement	Notes
Disk space	<ul style="list-style-type: none"> • HCL OneTest™ Performance Agent • HCL OneTest™ Performance 	10 GB	<ul style="list-style-type: none"> • Additional disk space is required if you use FAT32 instead of NTFS. An additional 500 MB of disk space is required in the directory that you specify in the environment variable TEMP.

Hardware	Components	Requirement	Notes
Memory	<ul style="list-style-type: none"> • HCL OneTest™ Performance Agent • HCL OneTest™ Performance 	8 GB	
Processor	<ul style="list-style-type: none"> • HCL OneTest™ Performance Agent • HCL OneTest™ Performance 	Minimum: 1.86 GHz Intel Pentium 4 or higher	

Operating systems

You can find the operating systems that are supported, organized by operating system family for HCL OneTest™ Performance.

Contents

- [Linux on page 19](#)
- [Mac on page 20](#)
- [Windows on page 20](#)

Linux

Operating system	Hardware	Bitness	Components		Notes
			Desktop	Agent	
Red Hat Enterprise Linux (RHEL) 8.1	x86-64	64-Exploit	✓	✓	
Red Hat Enterprise Linux (RHEL) 8	x86-64	64-Exploit	✓	✓	
Red Hat Enterprise Linux (RHEL) Client 7	x86-64	64-Exploit	✓	✓	

Operating system	Hardware	Bitness	Components		Notes
			Desktop	Agent	
Red Hat Enterprise Linux (RHEL) Workstation 7	x86-64	64-Exploit	✓	✓	
Red Hat Enterprise Linux (RHEL) Server 7	x86-64	64-Exploit	✓	✓	
Ubuntu 20.04.1 LTS	x86-64	64-Exploit	✓	✓	
Ubuntu 18.04 LTS	x86-64	64-Exploit	✓	✓	

Mac

Operating system	Hardware	Bitness	Components		Notes
			Desktop	Agent	
macOS Mojave 10.15	x86-64	64-Exploit	✓	✓	
macOS Catalina 10.14	x86-64	64-Exploit	✓	✓	

Windows

Operating system	Hardware	Bitness	Components		Notes
			Desktop	Agent	
Windows 10 Enterprise	x86-64	32, 64-Exploit	✓	✓	
Windows 10 Pro	x86-64	32, 64-Exploit	✓	✓	

Operating system	Hardware	Bitness	Components		Notes
			Desktop	Agent	
Windows 7 Enterprise	x86-32, x86-64	32, 64-Ex- ploit	✓	✓	
Windows 7 Professional	x86-32, x86-64	32, 64-Ex- ploit	✓	✓	
Windows 7 Ultimate	x86-32, x86-64	32, 64-Ex- ploit	✓	✓	
Windows Server 2019	x86-64	32, 64-Ex- ploit	✓	✓	
Windows Server 2016	x86-64	32, 64-Ex- ploit	✓	✓	

Host prerequisites

You can find the host prerequisites that support the operating capabilities for HCL OneTest™ Performance.

Contents

- [Java SDK on page 21](#)
- [Terminal services on page 22](#)
- [Virtualization Management on page 23](#)
- [Web Browsers on page 23](#)

Java SDK

Prerequisite	Version	Components		Notes
		Desktop	Agent	
Oracle Java SDK/JRE/JDK 32 or 64 bit	8 and update 162	✓	✓	

Prerequisite	Version	Components		Notes
		Desktop	Agent	
	8 and update 172	✓	✓	
	8 and update 201	✓	✓	
	8 and update 251	✓	✓	
	8 and update 261	✓	✓	

Terminal services

Supported software	Version	Components		Notes
		Desk-top	Agent	
Microsoft Windows Server	2003 Terminal Services and future fix packs	✓	Not applicable	For remote terminal access
Citrix XenApp		✓	Not applicable	
XenDesktop		✓	Not applicable	

Virtualization Management

Prerequisite	Version	Components		Notes
		Desktop	Agent	
Docker Community Edition (CE)	19.3.8 and future fix packs	✓	✓	
Docker Compose	1.25.5 and future fix packs	✓	✓	

Web Browsers

The following versions of web browsers support the viewing of performance reports. See [Recording support on page 26](#) to know the browsers that are supported to record the HTTP tests.

Prerequisite	Version	Components		Notes
		Desktop	Agent	
Apple Safari	13 and future fix packs	✓	Not applicable	
	12 and future fix packs	✓	Not applicable	
Google Chrome	84 and future fix packs	✓	Not applicable	
	83 and future fix packs	✓	Not applicable	
	81 and future fix packs	✓	Not applicable	
	80 and future fix packs	✓	Not applicable	

Prerequisite	Version	Components		Notes
		Desktop	Agent	
	79 and future fix packs	✓	Not applicable	
	78 and future fix packs	✓	Not applicable	
Microsoft Edge	Latest release	✓	Not applicable	
	44 and future fix packs	✓	Not applicable	
	42 and future fix packs	✓	Not applicable	
Microsoft Internet Explorer	11 and future fix packs	✓	Not applicable	
Mozilla Firefox	79 and future fix packs	✓	Not applicable	
	77 and future fix packs	✓	Not applicable	
	76 and future fix packs	✓	Not applicable	
	75 and future fix packs	✓	Not applicable	
	74 and future fix packs	✓	Not applicable	

Prerequisite	Version	Components		Notes
		Desktop	Agent	
	73 and future fix packs	✓	Not applicable	
	72 and future fix packs	✓	Not applicable	
	71 and future fix packs	✓	Not applicable	
	70 and future fix packs	✓	Not applicable	
	69 and future fix packs	✓	Not applicable	
	68 and future fix packs	✓	Not applicable	
	67 and future fix packs	✓	Not applicable	
	66 and future fix packs	✓	Not applicable	
	65 and future fix packs	✓	Not applicable	
	64 and future fix packs	✓	Not applicable	
	63 and future fix packs	✓	Not applicable	

Prerequisite	Version	Components		Notes
		Desktop	Agent	
	62 and future fix packs	✓	Not applicable	
	61 and future fix packs	✓	Not applicable	
	60 and future fix packs	✓	Not applicable	
Mozilla Firefox ESR	68 and future fix packs	✓	Not applicable	
	60 and future fix packs	✓	Not applicable	

Recording support

You can find information about the web browsers that supports recording capability of HTTP tests for HCL OneTest™ Performance.

Web browsers

The following versions of web browsers support the recording of HTTP tests. See [Web Browsers on page 23](#) to know the browsers that are supported to view the performance reports.

Supported software	Version	Components		Recording capability	Notes
		Desktop	Agent		
Apple Safari	13 and future fix packs	✓	Not applicable	✓	To record HTTP tests

Supported software	Version	Components		Recording capability	Notes
		Desktop	Agent		
	12 and future fix packs	✓	Not applicable	✓	
Google Chrome	84 and future fix packs	✓	Not applicable	✓	
	83 and future fix packs	✓	Not applicable	✓	
	81 and future fix packs	✓	Not applicable	✓	
	80 and future fix packs	✗	Not applicable	✗	
	79 and future fix packs	✗	Not applicable	✗	
	78 and future fix packs	✗	Not applicable	✗	
	Latest release	✓	Not applicable	✓	
Microsoft Edge	Latest release	✓	Not applicable	✓	

Supported software	Version	Components		Recording capability	Notes
		Desktop	Agent		
	44 and future fix packs	✓	Not applicable	✓	
	42 and future fix packs	✗	Not applicable	✗	
Microsoft Internet Explorer	11 and future fix packs	✓	Not applicable	✓	
Mozilla Firefox	79 and future fix packs	✓	Not applicable	✓	
	77 and future fix packs	✓	Not applicable	✓	
	76 and future fix packs	✓	Not applicable	✓	
	75 and future fix packs	✓	Not applicable	✓	
	74 and future fix packs	✓	Not applicable	✓	

Supported software	Version	Components		Recording capability	Notes
		Desktop	Agent		
	73 and future fix packs	✘	Not applicable	✘	
	72 and future fix packs	✘	Not applicable	✘	
	71 and future fix packs	✘	Not applicable	✘	
	70 and future fix packs	✘	Not applicable	✘	
	69 and future fix packs	✘	Not applicable	✘	
	68 and future fix packs	✘	Not applicable	✘	
	67 and future fix packs	✘	Not applicable	✘	
	66 and future fix packs	✘	Not applicable	✘	

Supported software	Version	Components		Recording capability	Notes
		Desktop	Agent		
	65 and future fix packs	✗	Not applicable	✗	
	64 and future fix packs	✗	Not applicable	✗	
	63 and future fix packs	✗	Not applicable	✗	
	62 and future fix packs	✗	Not applicable	✗	
	61 and future fix packs	✗	Not applicable	✗	
	60 and future fix packs	✗	Not applicable	✗	
Mozilla Firefox ESR	68 and future fix packs	✓	Not applicable	✓	
	60 and future fix packs	✗	Not applicable	✗	

Supported software

You can find the additional software that are supported for HCL OneTest™ Performance.

Contents

- [Application servers on page 31](#)
- [Business process management on page 32](#)
- [Development tools on page 33](#)
- [DevOps tools on page 35](#)

Application servers

Support for the following application servers is in reference only to the HTTP Response Time Break Down capability:

Supported software	Version	Components		Notes
		Desktop	Agent	
Oracle/BEA WebLogic Server	12 and future fix packs	✓	Not applicable	To collect response time breakdown data
	10.3 and future fix packs	✓	Not applicable	
	9 and future fix packs	✓	Not applicable	
WebSphere Application Server	9.0	✓	Not applicable	
	8.5.5	✓	Not applicable	
	8.5	✓	Not applicable	
	8.0	✓	Not applicable	
	7.5	✓	Not applicable	
WebSphere Liberty	17.0.0.1	✓	Not applicable	

Supported software	Version	Components		Notes
		Desktop	Agent	
	16.0.0.2	✓	Not applicable	

Business process management

Supported software	Version	Components		Notes
		Desktop	Agent	
SAP GUI	7.6 and future fix packs	✓	✓	To record and playback tests of SAP applications built with the SAP GUI client
	7.5 and future fix packs	✓	✓	
	7.4 and future fix packs	✓	✓	
	7.3 and future fix packs	✓	✓	
	7.2 and future fix packs	✓	✓	
	7.1 and future fix packs	✓	✓	

Development tools

Supported software	Version	Components		Notes
		Desktop	Agent	
Google Web Toolkit	2.8 and future fix packs	✓	✓	To record and playback http tests that against web applications built with Google Web Toolkit
	2.7 and future fix packs	✓	✓	
	2.6 and future fix packs	✓	✓	
	2.5 and future fix packs	✓	✓	
	2.4 and future fix packs	✓	✓	
	2.3 and future fix packs	✓	✓	
JMeter	5.3 and future fix packs	✓	Not applicable	To integrate and run JMeter tests
	5.2 and future fix packs	✓	Not applicable	
	5.1 and future fix packs	✓	Not applicable	
Rational Application Developer for WebSphere Software	9.7.0.2	✓	Not applicable	Eclipse shell sharing
	9.7	✓	Not applicable	

Supported software	Version	Components		Notes
		Desktop	Agent	
	9.6.1	✓	Not applicable	
Rational ClearCase	8.0.1 and future fix packs	✓	Not applicable	Eclipse source control
HCL OneTest™ UI	10.1.1	✓	Not applicable	Eclipse shell sharing and to run WebUI integrations
Engineering Test Management	7.0.1	✓	Not applicable	To initiate the test runs from Engineering Test Management
Rational Quality Manager	6.0.6	✓	Not applicable	To initiate the test runs from Rational Quality Manager
	6.0.5	✓	Not applicable	
	6.0.4	✓	Not applicable	
	6.0.3	✓	Not applicable	
Engineering Workflow Management	7.0.1	✓	Not applicable	To perform integrations with Engineering Workflow Management
Rational Team Concert	6.0.6	✓	Not applicable	To perform integrations with
	6.0.5	✓	Not applicable	

Supported software	Version	Components		Notes
		Desktop	Agent	
	6.0.4	✓	Not applicable	Rational Team Concert
	6.0.3	✓	Not applicable	
Rational® Software Architect Designer	9.7.0.2	✓	Not applicable	Eclipse shell sharing
	9.7	✓	Not applicable	
	9.6.1	✓	Not applicable	
Rational Software Architect Designer for Web-Sphere Software	9.7	✓	Not applicable	Eclipse shell sharing
HCL OneTest™ Studio	10.1.1	✓	✓	To integrate and run HCL OneTest™ API tests
eGit	4.10	✓	Not applicable	Eclipse source control

DevOps tools

Supported software	Version	Components		Notes
		Desktop	Agent	
IBM UrbanCode Deploy	7.0.2	✓	Not applicable	To initiate the test runs from UrbanCode Deploy

Supported software	Version	Components		Notes
		Desktop	Agent	
HCL Launch	7.1.0.1	✓	Not applicable	To initiate the test runs from HCL Launch

Chapter 3. Getting Started

This guide provides an overview and describes the task flows to get you started with HCL OneTest™ Performance. This guide is intended for new users.

Task flows for performance testing

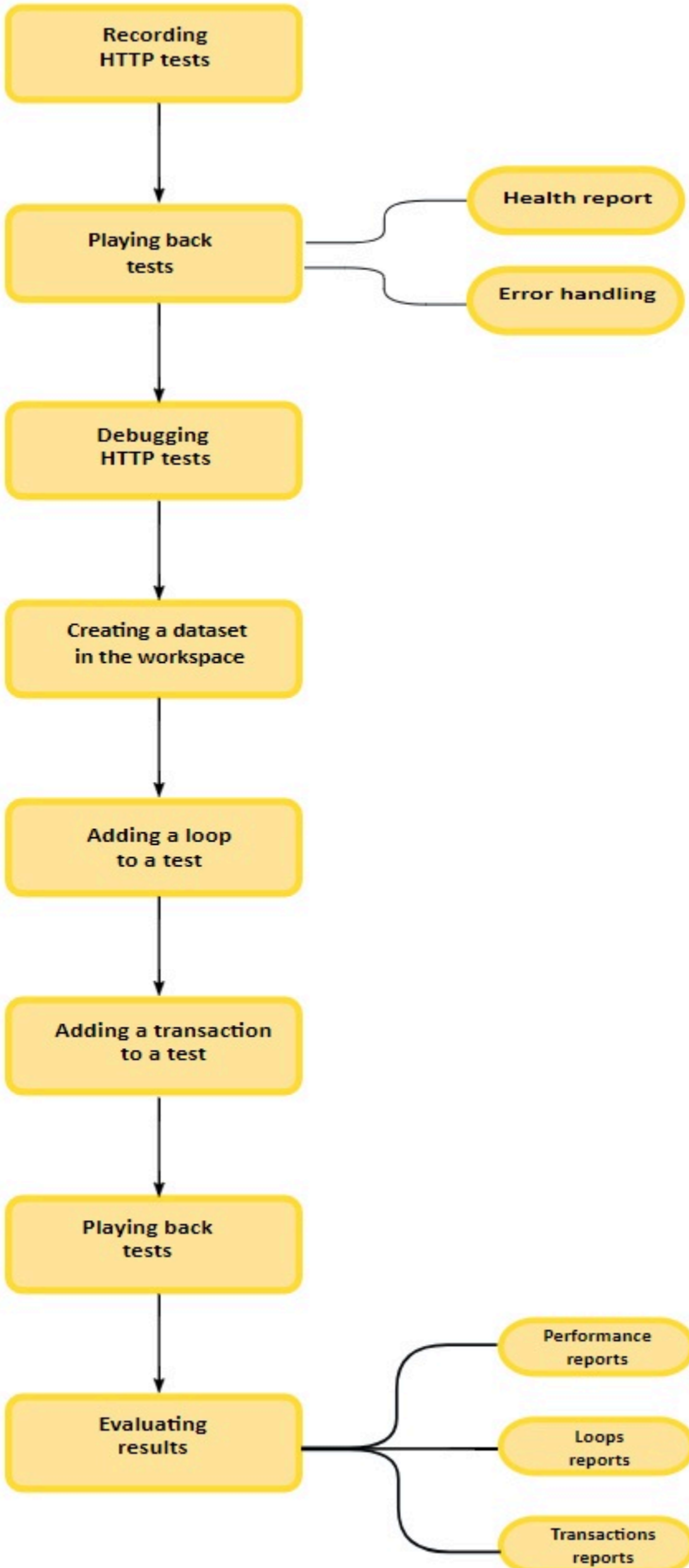
The task flows are designed to help you understand the end-to-end process of some of the capabilities of performance testing. For some of the technologies, there are specific task flows that you can follow to gain a clear understanding of the process.

Task flow: HTTP record

The diagram shows the task flow of HTTP recording to test the performance of an application using HCL OneTest™ Performance.

To test the performance of an application, you must first record the HTTP traffic that traverses between the client and the server.

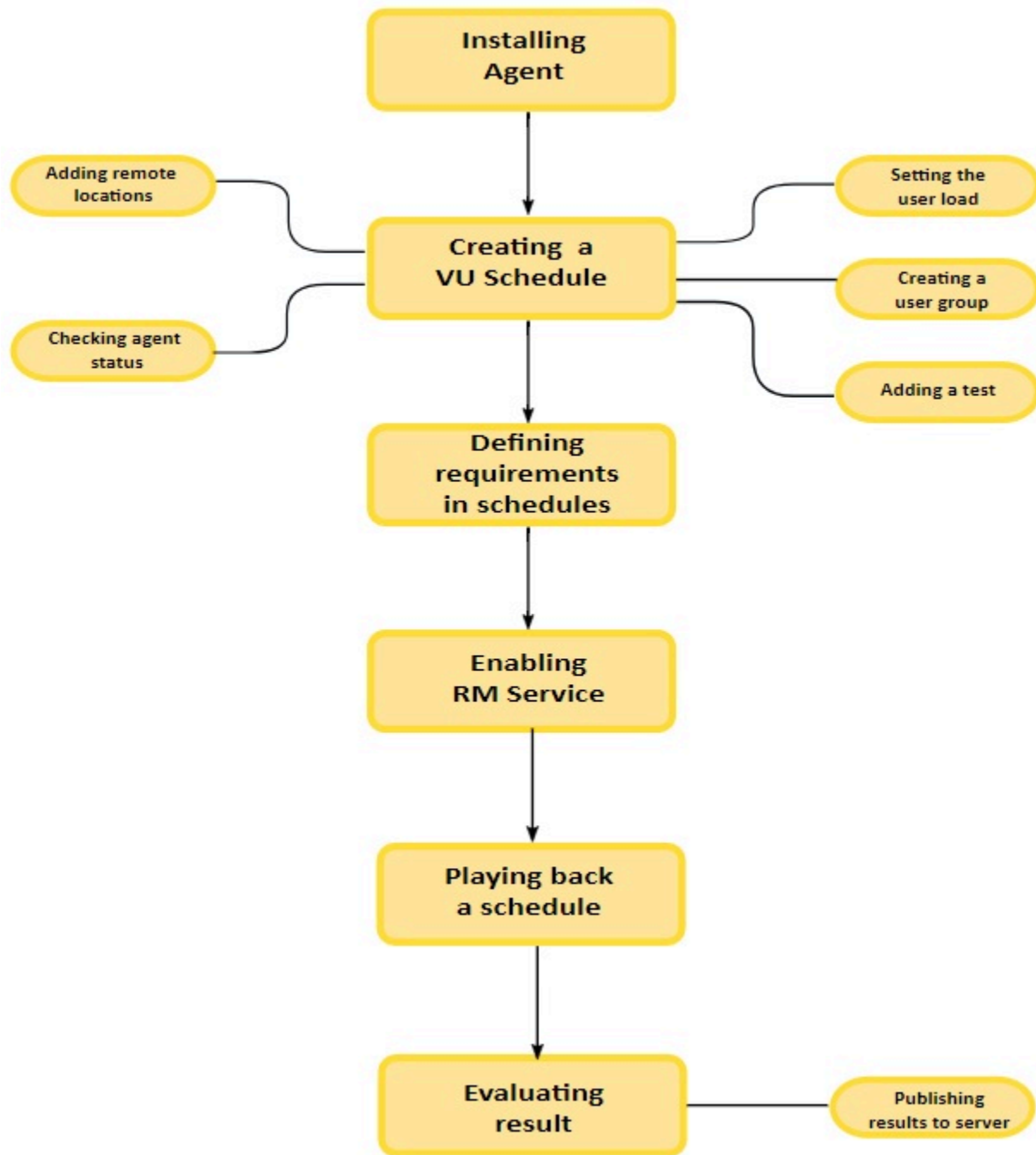
Once you complete the basic HTTP recording, play it back and fix the errors, if any. You can then customize your test by adding other elements such as datasets, loops, and transaction.



Task flow: Generating workload

The diagram shows the task flow of generating workload by creating user groups and dividing the load across different remote agents.

The agents generate load on the application under test. The number of agents required depends on various factors including the size of the load you want to put on the application under test, hardware capacity of systems that install the agents, and the application under test itself.




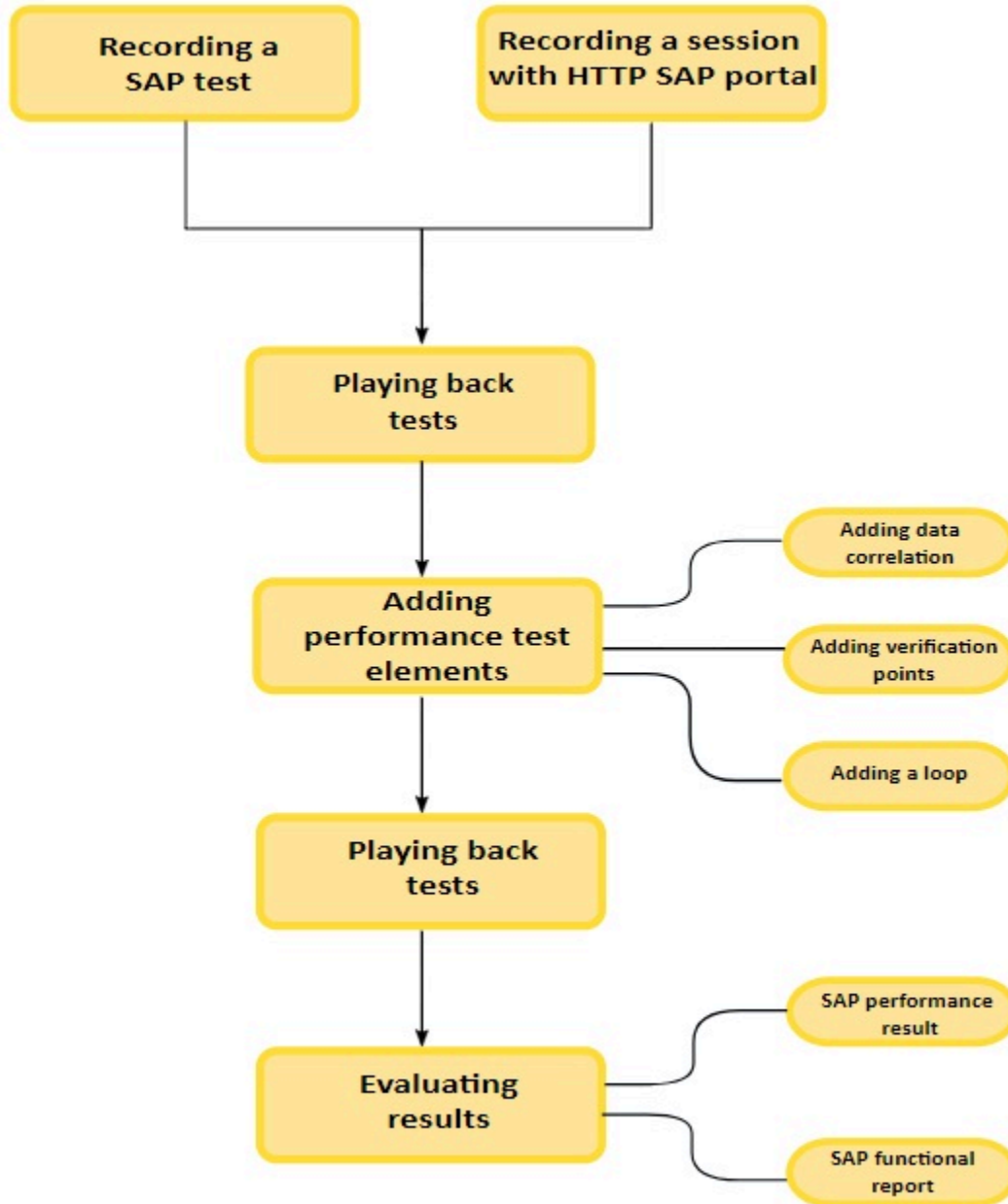
1. Installing HCL OneTest Performance on page 92
2. Creating a VU Schedule on page 524
3. Defining requirements in schedules on page 558
4. Enablement of Resource Monitoring services for a schedule on page 570
5. Running a local schedule or test on page 608
6. Evaluating results in web analytic reports on page 773
7. Publishing test results to the server on page 776
8. Running a user group at a remote location on page 538
9. Checking the status of agents on page 553
10. Setting user loads on page 527

Task flow: Recording a SAP test

The task flow shows the recording of interactions with the SAP GUI client to generate a SAP test.

The recording wizard opens the SAP GUI client and records all the interactions that occur between the client and the server. You can record a SAP test by using the HTTP SAP Portal option to measure the performance of a SAP Portal from a web interface. After you complete SAP recording or HTTP recording session with SAP portal, you must play it back and fix the errors, if any. You can add other HCL OneTest™ Performance elements such as data correlation, verification points, and loops and playback the test again to evaluate the results.

 **Note:** If a test is not behaving as expected during playback, you must ensure that the connection to the SAP server is available. For more information, see [SAP connection details on page 1206](#).

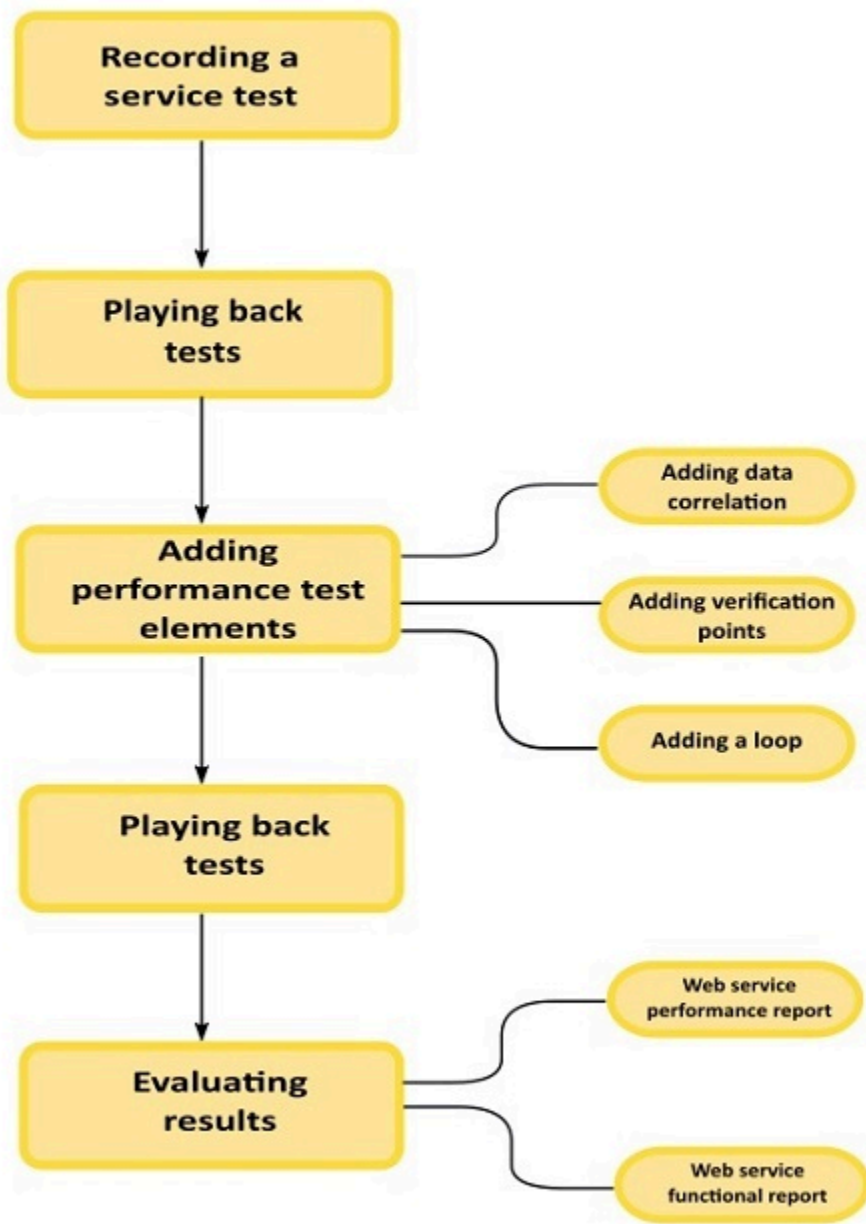


1. Recording a SAP test
2. Recording a session with HTTP SAP Portal
3. [Running a local schedule or test on page 608](#)
4. [Running a local schedule or test on page 608](#)
5. [Evaluating results in web analytic reports on page 773](#)
6. [Correlating response and request data on page 443](#)
7. [Verifying application behavior on page 339](#)
8. [Adding a loop to a test on page 387](#)
9. [SAP Performance report on page 829](#)

Task flow: Recording a service test to test an SOA application

The task flow shows the testing services in an SOA environment. You can record a test session by invoking service calls by using a generic service client or an existing client.

To test the performance of an SOA application, you must record a service call and create a service test. After you complete the service test recording, play it back and fix the errors, if any. You can edit your service test to include verification points, data correlations, loops, and play back the test again to evaluate the results.



1. [Recording service tests on page 225](#)
2. [Running a local schedule or test on page 608](#)
3. [Running a local schedule or test on page 608](#)
4. [Correlating response and request data on page 443](#)
5. [Verifying application behavior on page 339](#)
6. [Adding a loop to a test on page 387](#)
7. [Web service reports on page 837](#)
8. [Generating functional test reports on page 775](#)

Task flow: Recording a SAP batch input test

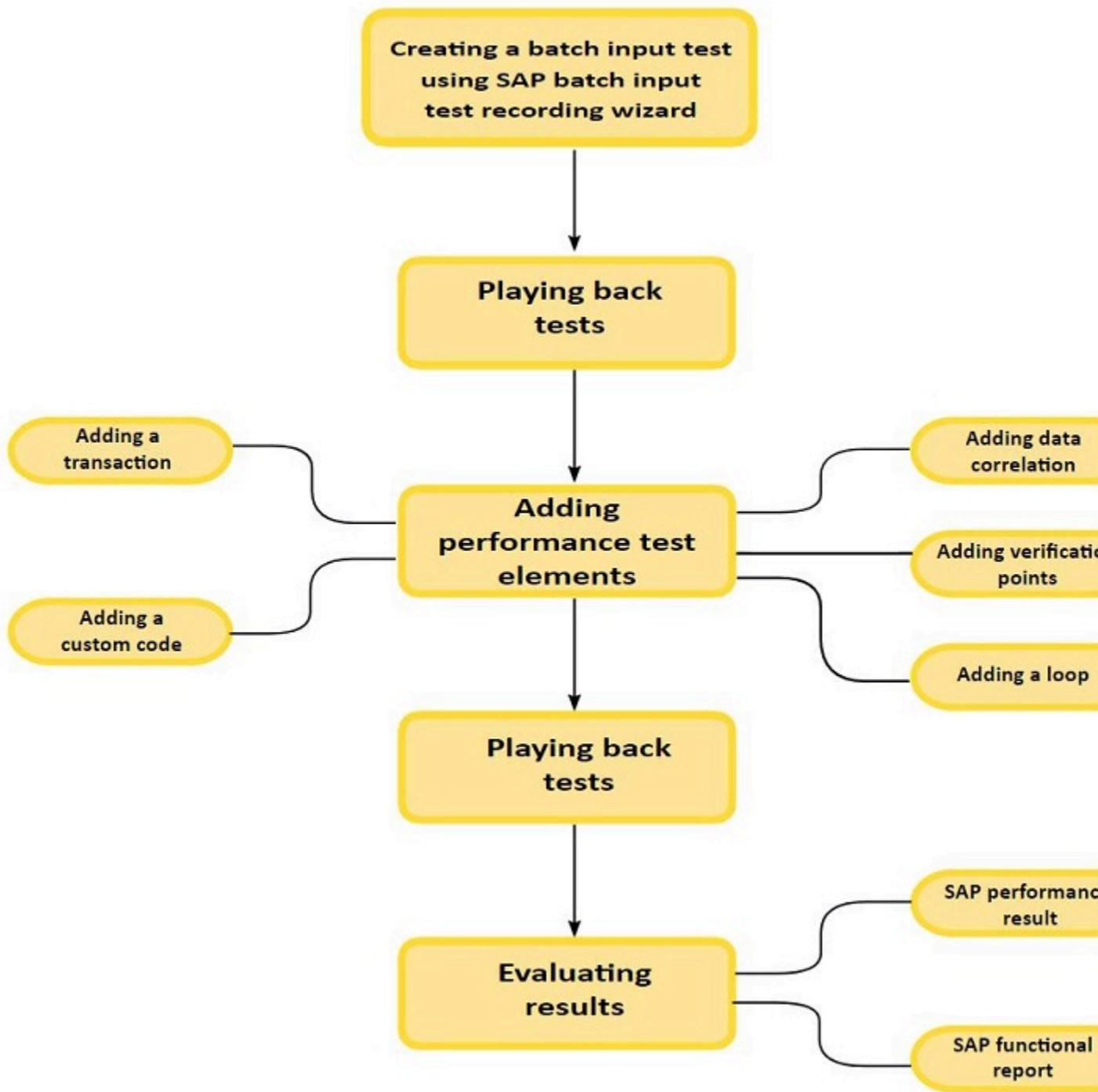
The task flow shows the recording of certain *SAP transaction* sessions from SAP GUI with SAP batch input tests recording wizard. When you record a session, the recording wizard automatically starts a SAP GUI interface and records the transaction that you specified. Typically, you use batch input tests in a schedule mixed with normal SAP performance tests to increase the load on the server.

After you complete the recording, the wizard generated a SAP batch input test in HCL OneTest™ Performance. You can then play it back again to fix the errors, if any.

**Note:**

- You must perform the recording of certain *SAP transaction* sessions from SAP GUI. For more information, see [Recording a SAP batch input test](#).
- If a test is not behaving as expected during playback, you must ensure that the connection to the SAP server is available. For more information, see [SAP connection details on page 1206](#).

Later, you can add the other HCL OneTest™ Performance elements such as data correlation, verification points, loops, transactions, and custom codes to test and playback the test to evaluate the results.



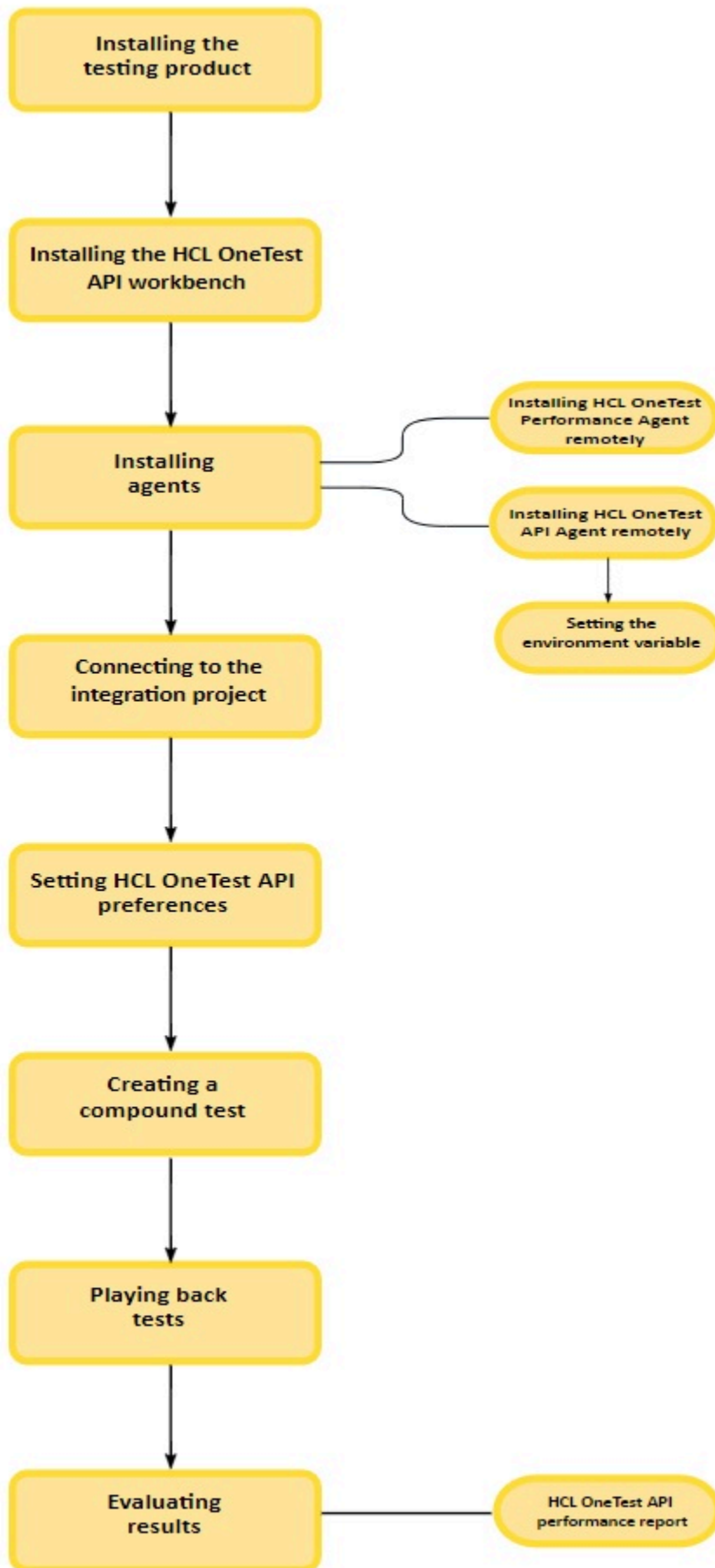
1. Recording a SAP batch input test
2. [Running a local schedule or test on page 608](#)
3. [Data correlation overview on page 444](#)
4. [Verifying application behavior on page 339](#)
5. [Adding a loop to a test on page 387](#)
6. [Adding a transaction to a test on page 383](#)
7. [Creating custom Java code on page 657](#)
8. [Running a local schedule or test on page 608](#)
9. [Evaluating results in web analytic reports on page 773](#)
10. [SAP Performance report on page 829](#)
11. [Generating functional test reports on page 775](#)

Task flow Integrating HCL OneTest™ Performance and HCL OneTest™ API

You can execute integration tests in HCL OneTest™ Performance by using HCL OneTest™ Performance Extension for HCL OneTest™ API. In HCL OneTest™ Performance, you can create a compound test to run the integration tests by using agents.

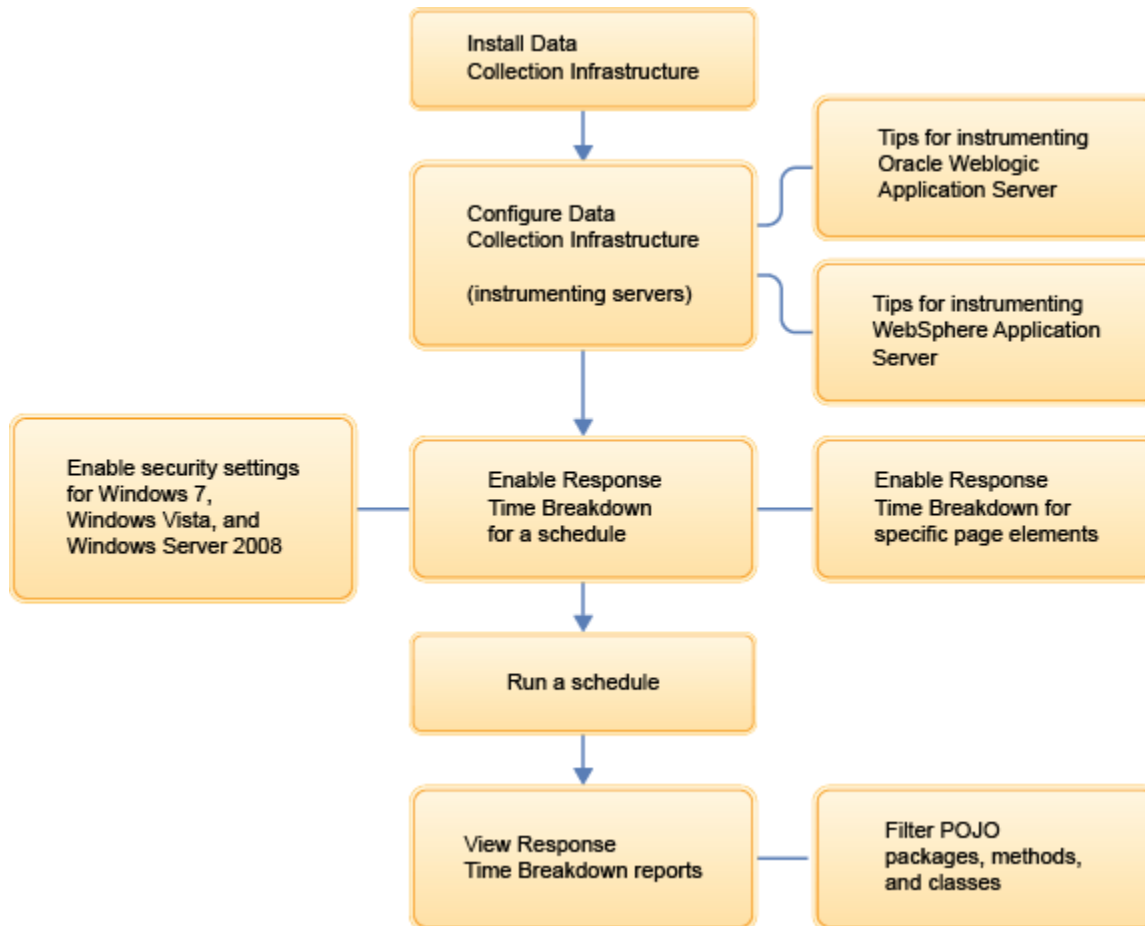
To integrate tests, you must install HCL OneTest™ Performance Extension for HCL OneTest™ API. Also, to execute the tests remotely, you must install HCL OneTest™ Performance Agent and HCL OneTest™ API Agent.

After installing all the required software, you must set the environment variable and connect to the integration project. To open the HCL OneTest™ API project from HCL OneTest™ Performance Test Navigator, you must set the path to the execution file in the **HOT-API Integration** preferences. Later, you must create a compound test and play back the test to evaluate the results.



Task flow: Response time breakdown

To collect Response Time Breakdown data, you must follow a workflow that includes installing and configuring Data Collection Infrastructure, enabling Response Time Breakdown for a schedule, running a schedule, and viewing Response Time Breakdown reports.



1. [../topics/t_start_install_launchpada.html](#)
2. [../topics/tinstrolocal.html](#)
3. [../topics/cdcibeatips.html](#)
4. [../topics/cdciwastips.html](#)
5. [../topics/tenablertbvista.html](#)
6. [../topics/tenableresponsetime.html](#)
7. [../topics/tenablingrespelem.html](#)
8. [Running a local schedule or test on page 608](#)
9. [Viewing response time breakdown on page 795](#)
10. [../topics/tfilterpojo.html](#)

Task flow: Rate Scheduler

The task flow shows running a rate scheduler to monitor the workload at the desired rate (transactions per second).

After installing HCL OneTest™ Performance agent, you must create a rate scheduler by adding elements such as rate runner group, tests, user load, and checking the agent status. Then, you must define the performance requirements for a schedule to specify the acceptable thresholds to validate the service-level agreements. Later, you must enable the resource monitoring service and play back the scheduler to evaluate the results.



1. Installing HCL OneTest Performance on page 92
2. Setting rate load on page 543
3. Rate Runner group overview on page 545
4. Creating a Rate Schedule on page 542
5. Checking the status of agents on page 552

Product overview

You can gain the conceptual understanding of the product and its test extensions with these topics.

HTTP performance testing overview

There are five stages when performance testing HTTP applications: test creation, test editing, workload emulation with schedules, schedule execution, and evaluation of results.

- **Test creation.** Although it is possible to write a test from scratch, you generally create HTTP performance tests by recording representative interactions with an application. These actions are saved and a test is generated from these recorded actions. Test recording and related tasks are explained in [Recording HTTP tests on page 182](#).
- **Test editing.** After recording a test, run it individually and inspect the results to make sure that the HTTP tests are doing what you expect. You might want to edit your tests. Typical changes that you might make are:
 - Replacing a value in a recorded test with values in a *dataset*. This produces more realistic test data. [Providing tests with variable data \(datasets\) on page](#) explains how to use datasets to replace data.
 - Adding dynamic data to a test (referred to as *data correlation*). Data correlation, including how to manually correlate test values, is explained in [Correlating response and request data on page](#).
 - Enabling verification points, so that you can determine whether an expected behavior occurred. Verification points are explained in [Verifying expected behavior on page 292](#).
- **Workload emulation with schedules.** After editing HTTP tests, you create a *schedule*. You add *user groups* to the schedule and add appropriate tests to each group to emulate a task. A typical schedule contains the following:
 - User groups and tests. A schedule requires at least these items to run.
 - User groups running from a remote location. This separates your workbench activity and your load-adding activity. For more information, see [Running a user group at a remote location on page 538](#).
 - Optional schedule items, such as loops, delays, and think time behavior settings, to further emulate a load. For more information, see [Emulating workloads on page 522](#).
- **Schedule execution.** You typically run a schedule as explained in [Running a local schedule or test on page 608](#).
- **Evaluation of results.** Reports are displayed during the schedule run. You can also regenerate reports after the run, customize reports, and export reports in HTML format, so that others who do not have the product can see them.

SAP performance testing overview

With HCL OneTest™ Performance Extension *for SAP Solutions*, you can test the performance of SAP applications.

Informative performance test results rely upon sound test development. Each of the following stages contributes to generating meaningful results when performance testing SAP applications:

- **Test creation.** You create your test by recording a session with the SAP GUI client. Typically, the recorded session starts when you log on to the SAP server. You then interact with the application in order to produce a relevant performance test, and the session ends when you log out. The recorded session is split into transactions and SAP screens. Response time measurements and verification points are automatically added to transactions and SAP screens.
- **Test editing.** After recording, you can edit the events in each transaction and SAP screen. With the **SAP Protocol Data** view, you can use snapshots of the SAP screen to edit the events. You can replace recorded test values with variable test data, or add dynamic data to SAP tests. You can also set verification points on field values or window titles to validate that the test behaved as expected.
- **Test validation.** Before deploying the test, you can run the test manually as a single virtual user to make sure that the test runs smoothly and produces the expected results in a nominal environment with minimal server load. You can experience multiple test editing and validation cycles before your test is robust.
- **Workload emulation with schedules.** When the test runs repeatedly as anticipated, you specify an execution schedule and user groups to emulate a workload that is generated by a large number virtual users. You can add SAP batch input tests to the schedule to simulate a heavy load on the servers while minimizing virtual tester resources.
- **Schedule execution.** You run the schedule, deploying test execution over virtual users that can be hosted on remote hosts. Each virtual user runs an instance of the SAP GUI client. Response time results are provided by the SAP server and recorded. Verification points are checked and results are recorded.
- **Evaluation of results.** You evaluate the results produced by the SAP performance tests through the various reports that are generated during execution. You can also design custom reports.

Citrix performance testing overview

HCL OneTest™ Performance Extension *for Citrix Presentation Server* allows you to test the performance of Citrix applications.

Informative performance test results rely upon sound test development. Each of the following stages contributes to generating meaningful test results:

- **Test creation.** You create your test by recording a session with the Citrix XenApp client. Typically, the recorded session starts when you log in to the Citrix server. You then interact with the application to produce a relevant performance test. The session ends when you log out. The recording is split into window events and contains keyboard and mouse interactions with the Citrix server. You can use the **Citrix Recording Control** window to add screen captures or comments. You can add image synchronizations to help the test remain synchronized with the server.
- **Test editing.** After recording, you can edit the events in each window element. Because the recorded input is primarily made of low-level keyboard and mouse input, you can streamline the test by, for example, replacing key-press events with string inputs. You can use the comments and recorded screen captures to make navigating through the test easier. You can replace recorded test values with variable test data, or add dynamic data to the test. You can also set verification points on window titles or image synchronizations to validate that the application behaves as expected.

- **Test validation.** Before deploying the test, you can run the test manually as a single virtual user to make sure that the test synchronizes user input actions and server output correctly in a nominal environment with minimal server load. You might experience multiple test editing and validation cycles before your test runs as expected.
- **Workload emulation with schedules.** When the test runs repeatedly as anticipated, you specify an execution schedule and user groups to emulate a workload that is generated by a large number of virtual users.
- **Schedule execution.** You run the schedule, deploying test execution over virtual users that can be hosted on remote hosts. Each virtual user runs an instance of the Citrix client.
- **Evaluation of results.** You evaluate the results produced by the tests through the various reports that are generated during execution. You can also design custom reports.

Prerequisites

Before you can test the performance of Citrix applications, a Citrix XenApp client must be installed on the same computer as HCL OneTest™ Performance. The Citrix XenApp client is required for recording and execution of performance tests.

Ensure that you have a working Citrix client environment and that you can connect to a Citrix server. .

If you are deploying tests over remote computers to emulate a large number of users, the following software must be installed on each remote computer:

- The Citrix XenApp client
- The HCL OneTest™ Performance Agent

Limitations

Citrix performance tests use window creation and change events, and optionally image recognition techniques, to synchronize user input with server output. Before you record a session with a Citrix application, the behavior of that application must be perfectly reproducible. Specifically, the application must always create windows and GUI elements at the same locations and in the same sequence. Mouse or keyboard events must always produce the same output. Consider these examples:

- If the application displays windows or dialog boxes on only the first execution of a particular program or feature, such as tips or security warnings, ensure that they are disabled when you record the test. Any windows or dialog boxes that were recorded but are not displayed on subsequent executions, or displayed at different coordinates on the screen, will fail the test and cause synchronization timeouts.
- If you save a file during a recorded session, the application might issue a warning for an existing filename when you replay the tests. If the warning was not in the recorded session, this will fail the test and cause errors.

It is essential to be aware of the context of user actions when you edit a test. Because the Citrix performance tests interact with the Citrix XenApp client at a very low level (mouse movements and key presses) any changes that you

make to the test after the recording, such as moving test elements, adding loops or conditions, or inserting new sequences, can alter the context of the emulated user actions and cause synchronization timeouts.

Service testing overview



The service testing capabilities of HCL OneTest™ Performance automate the creation, execution and analysis of functional, regression and performance tests for SOAP-based web services, including support for Java Message Service (JMS), Websphere MQ, WebSphere Java MQ, and Microsoft .NET Windows Communication Foundation (WCF), as well as any service that produces XML, plain text, or binary data.

Informative test results rely upon sound test development. Each of the following stages contributes to generating meaningful test results:

- **Preparation.** Set up your test environment with the libraries and configuration files required for SOAP-based web services or custom security algorithms. You can import Web Service Description Language (WSDL) definition files and digital certificates that are required by the web services to automatically generate your tests. You can create SOAP security profiles with security algorithms for the web service calls and message returns.
- **Test creation:** Create your test by recording the service requests and responses either with the **generic service client**, or with an existing client or a web browser through a recording proxy. When you start the recording, you interact with the service by performing service requests and receiving responses. You can also create service tests manually or from a synchronous Business Process Execution Language (BPEL) model.
- **Test editing:** After recording, you can edit the requests and responses in the test. You can use XML Schema Description (XSD) documents to facilitate XML edition. You can replace recorded test values with variable test data, or add dynamic data to the test.
- **Functional testing:** You can run the test to ensure that service matches the expected behavior defined in *verification points*. During the run, each verification point is checked and receives a *pass*, *fail* or *inconclusive* status.
- **Performance testing:** If you are using HCL OneTest™ Performance, you can specify an execution schedule and user groups to emulate a workload that is generated by a large number of virtual users. Then, you can run the schedule, deploying test execution on virtual users that can be hosted on remote computers. Each virtual user runs an instance of the test client. Response times are measured and recorded. Verification points are checked and recorded.
- **Stub simulation:** Service stubs are functional simulations of an existing service. Service stubs are useful for replacing a service that is unavailable or impractical to use in a test environment. They can also be used to input specific data into a service under test or for prototyping. You can deploy stubs onto a stub server, which can replace the actual server in your test or development environment.
- **Evaluation of results:** You evaluate the results that the tests produce through the performance and verification point reports that are generated during execution. You can also design custom reports by manipulating various counters. Functional reports provide a comprehensive view of the behavior of the service under test. Reports can be exported and archived for validation.

Service testing tools

The following tools are available in the product:

- The **generic service client** enables you to manually perform service requests for a wide variety of transport protocols, authentication configurations and security profiles, making it an extremely versatile service client. It effectively replaces a dedicated client and can be used to record service calls or for manual testing and debugging a service during development. To open the generic service client, click the **Generic Service Client**  toolbar button.
- The **WSDL security editor** allows you to set up sophisticated *algorithm stacks* for your service requests and responses. Algorithm stacks contain digital certificate information and the security algorithms that are applied to messages to perform secure communication with a web service. Algorithm stacks are made of blocks, which can be key definitions, encryption, time stamp, or signature operations which can be associated with any operation in the WSDL file. To open the WSDL security editor, right-click a WSDL file in your workspace and select **Edit WSDL Security** or click the **WSDL Security Editor**  button in the generic service client.
- The **test editor** is where you develop your test. After recording, you can modify the test to add data correlation or verification points. You can also add loops and conditions and you can edit every detail of the service requests.
- The **stub editor** enables you to create service stubs. With the stub editor, you can define multiple input conditions, which are similar to verification points. Each condition triggers a predefined simulated response, which is functionally identical to a response from the simulated service.
- In HCL OneTest™ Performance, the schedule editor lets you deploy multiple virtual users on local and remote computers to generate a heavy load for performance testing. A schedule typically contains multiple tests and multiple virtual users.

Generic service client overview

The purpose of the generic service client is to send requests to any service that uses an HTTP, JMS, WebSphere® MQ, or Microsoft™ .NET transport. The generic service client also displays the response returned by the service.

The generic service client is useful for debugging or testing a service when you do not have access to a dedicated client to send the request. You can set up a large variety of transport and security configurations for the service, edit the parameters of the request and send attachments.

When a request is successfully invoked, its message return is added to the **Request History**. You can use this feature to look back at results that were produced at different times.

If you are using HCL OneTest™ Performance, you can select requests in the **Request History** and click **Generate Test** to generate a test that will replay all the selected requests. You can edit the test to replace recorded test values with variable test data, or add dynamic data correlation to the test. You can also set verification points on the contents of the XML documents in the service response.

Supported services

The generic service client enables you to send requests for many types of services that use the following transport protocols:

- HTTP
- Java™ Message Service (JMS), including JBoss and WebSphere® implementations
- WebSphere® MQ
- Microsoft™ .NET Framework Windows™ Communication Foundation (WCF).



Note: If you are using IBM® Security AppScan®, only the HTTP transport protocol is supported.

Encryption and security

The Java™ Runtime Environment (JRE) that the product uses must support the level of encryption required by the digital certificate that you select. For example, you cannot use a digital certificate that requires 256-bit encryption with a JRE that supports only 128-bit encryption. By default, the product is configured with restricted or limited strength ciphers. To use less restricted encryption algorithms, you must download and apply the unlimited jurisdiction policy files (`local_policy.jar` and `US_export_policy.jar`).

For Oracle Java, download the files from this site: <http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>.

Before installing these policy files, back up the existing policy files in case you want to restore the original files later. Then overwrite the files in `/jre/lib/security/` directory with the unlimited jurisdiction policy files.

SSL Authentication

Service tests support simple or double SSL authentication mechanisms:

- Simple authentication (server authentication): In this case, the test client needs to determine whether the service can be trusted. You do not need to setup a key store. If you select the **Always trust** option, you do not need to provide a server certificate key store.

If you want to really authenticate the service, you can configure a certificate trust store, which contains the certificates of trusted services. In this case, the test will expect to receive a valid certificate.

- Double authentication (client and server authentication): In this case, the service needs to authenticate the test client according to its root authority. You must provide the client certificate keystore that needs to be produced to authenticate the test as a certified client.

When recording a service test through a proxy, the recording proxy sits between the service and the client. In this case, you must configure the SSL settings of the recording proxy to authenticate itself as the actual service to the client (for simple authentication), and as the client to the service (for double authentication). This means that you must supply the recording proxy with the adequate certificates.

When using stub services, you can also configure the SSL settings of the stub service to authenticate itself as the actual server. This means that you must supply the service stub with the adequate certificate.

NTLM and Kerberos Authentication

The product supports Microsoft™ NT LAN Manager (NTLMv1 and NTLMv2) and Kerberos authentication. The authentication information is recorded as part of the test during the recording phase.

To enable NTLMv2 support, you must add a third party library to the workbench. For more information, see [Configuring the workbench for NTLMv2 authentication on page 499](#).

Digital certificates

You can test services with digital certificates for both SSL and SOAP security protocol. Digital certificates must be contained in Java™ Key Store (JKS) keystore resources that are accessible in the workspace. When dealing with keystore files, you must set the password required to access the keys both in the security editor and the test editor. For SOAP security you might have to provide an explicit name for the key and provide a password to access the private keys in the keystore.

Limitations

Arrays are not supported.

Because of a lack of specification, attachments are not supported with the Java™ Message Service (JMS) transport. The envelope is directly sent using UTF-8 encoding.

All security algorithms are not always available for every Java™ Runtime Environment (JRE) implementation. If a particular security implementation is not available, add the required libraries to the class path of the JRE that this product uses.

The Microsoft™ .NET transport protocol does not support transactions, scopes, or duplex mode requests such as callbacks or two-way services based on the MS-MQ transport.

Socket API performance testing overview

With HCL OneTest™ Performance Extension *for Socket Protocols*, you can test the performance of any application that uses a TCP/IP socket-based protocol.

Informative performance test results rely upon sound test development. Each of the following stages contributes to the generation of meaningful test results:

- **Test creation.** You create your test by recording a session with a client application. Typically, the recorded session starts when you run the client application. You then interact with the application in order to produce relevant network traffic, and the session ends when you close the application or end the recording. The recording is used to generate a performance test that reproduces the behavior of the client application.
- **Test editing.** After recording, you can edit the events that were recorded. You can replace recorded test values with variable test data or add dynamic data to the test.

- **Test validation.** Before deploying the test, you can run the test manually as a single virtual user to make sure that the test runs smoothly and produces the expected results in a nominal environment with minimal server load. You might experience multiple test editing and validation cycles before your test runs as expected.
- **Workload emulation with schedules.** When the test runs repeatedly as anticipated, you specify an execution schedule and user groups to emulate a workload that a large number of virtual users generates.
- **Schedule execution.** You run the schedule, deploying test execution over virtual users that can be hosted on remote hosts. Each virtual user runs an instance of the test. Response time results are collected.
- **Evaluation of results.** You evaluate the results produced by the tests through the various reports that are generated during execution. You can also design custom reports.

TN3270 performance testing overview

With HCL OneTest™ Performance Extension for Socket Protocols, you can test the performance of TN3270 terminal server applications.

Informative performance test results rely on sound test development. Each of these stages contributes to the generation of meaningful test results:

- **Test creation.** You create a test by recording a session with a client application. Typically, the recorded session starts when you run the TN3270 terminal client. You then interact with the application in order to produce relevant network traffic. The session ends when you close the terminal client or end the recording. The recording is used to generate a performance test that reproduces the behavior of the client application.
- **Test editing.** After recording, you can edit the recorded events. You can replace recorded test values with variable test data or add dynamic data to the test.
- **Test validation.** Before deploying the test, you can run the test manually as a single virtual user to make sure that the test runs smoothly and produces the expected results in a nominal environment with minimal server load. You might complete multiple test editing and validation cycles before your test is robust.
- **Workload emulation with schedules.** When the test runs repeatedly as anticipated, you specify an execution schedule and user groups to emulate a workload that a large number of virtual users generates.
- **Schedule execution.** You run the schedule, deploying test execution over virtual users that can be hosted on remote hosts. Each virtual user runs an instance of the test. Response time results are collected.
- **Evaluation of results.** You evaluate the results that the tests produce through the various reports that are generated during execution. You can also design custom reports.

IBM® Engineering Test Management overview

IBM® Engineering Test Management is a collaborative, web-based, quality management solution that offers comprehensive test planning, manual testing, and integration with other test tools.

Quality Manager is based on the IBM® Rational® Jazz™ platform (<http://jazz.net> and <http://www.ibm.com/software/rational/jazz/>) and inherits many characteristics from that platform. Engineering Test Management is designed to be used by test teams of all sizes and supports a variety of user roles, such as test manager, test architect, test lead, tester, and lab manager, as well as roles outside the test organization.

Comprehensive test planning

A *test plan* that you define in Engineering Test Management drives activity for distributed teams through all phases of the project life cycle. The test plan defines the objectives and scope of the test effort and contains criteria to help teams determine the answer to the question "Are we ready to release?"

The test plan can be configured to meet the needs of your organization. You can use the test plan to do any and all of the following tasks:

- Define business and test objectives
- Establish a review and approval process for the test plan and for individual test cases
- Manage project requirements and test cases and establish the interdependencies between the two
- Estimate the size of the test effort
- Define the schedule for each test iteration and track the dates of other important test activities
- List the various environments to be tested and generate test configurations
- Create a read-only snapshot of the test plan at a particular point in time
- Define quality goals, entrance criteria, and exit criteria
- Create and manage test cases

Test script construction, execution, and reuse

Engineering Test Management provides a full-featured manual test editor. You can also import manual test scripts from IBM® Rational® Manual Tester. You can add reuse and automation capabilities to your manual tests by using keywords.

With Engineering Test Management, you can manage and execute test scripts that are created with tools such as HCL OneTest™ Performance, IBM® Rational® Service Tester for SOA Quality, and IBM® Security AppScan® Tester Edition.

You can also import test artifacts from external test management solutions, such as IBM® Rational® ClearQuest® Test Manager and IBM® Rational® Test Manager.

Test analysis and reporting

Engineering Test Management includes several standard test reports to help you evaluate test results. Reports are available during all phases of the test process.

You can use reports to perform these tasks:

- Determine the validity of a test run.
- Check feature coverage against test plans, test inputs, configurations, and so on. This can also be used to measure test progress and to analyze trends.
- Run a gap analysis to measure the resources needed to do your testing versus the resources that are available

Team collaboration

Engineering Test Management makes it easy to share information with other members of your team. With the Jazz-based work-item system, team members can assign tasks and defects to each other and to view everyone's status. Test plan authors and test case designers can distribute their work for review and track the status of each reviewer. New and changed requirements are visible to the team, as are the test cases that are needed to satisfy those requirements. Team members are notified automatically of any changes and milestones that impact their work.

Lab management

With Engineering Test Management lab management capabilities, you can create requests for the test environments that your test plan specifies. You can then work with the lab manager to ensure that lab resources and test environments are available when needed. Lab managers can track all lab resources from a centralized resource repository and fulfill requests from the test team.

Web application security

Engineering Test Management helps IT and security professionals protect against the threat of attacks and security breaches through its integration with IBM® Security AppScan® Tester Edition. Security testing for your web applications can result in higher-quality, more secure applications at a reasonable cost.

Governance

Engineering Test Management helps ensure that your business processes comply with industry, corporate, and departmental standards and regulations. Throughout the testing life cycle, Engineering Test Management provides you with the tools to obtain an up-to-the-minute measurement of software quality and project metrics. With its comprehensive test plan and integration with requirements management and defect tracking tools, Engineering Test Management helps streamline your test strategy and produce reliable records of test results and project history.

Streamlined Eclipse and full Eclipse overview

When you work in the streamlined Eclipse mode, only those functions that are directly related to the product are enabled in the workbench. When you install the product, by default, the check box to use the streamlined Eclipse mode is selected. With the full Eclipse mode, you have access to all Eclipse functions.

The streamlined Eclipse mode disables options from the menus that are not typically used during testing. Both the fully-enabled and streamlined Eclipse modes can operate using the same workspace, so if you start the product in the streamlined mode and discover that you cannot accomplish all of your tasks, you can close the workbench and restart it in the full Eclipse mode.

The choice of the mode in which to start the product depends on the user's activity and objectives. The streamlined mode is designed for straightforward testing and shows only those menu items that are related to testing. However, this restricts functions. The following list includes use cases where the full Eclipse mode might be preferred:

- You have multiple products installed and you want to use them in the same session.
- You are using the profiling and logging features. The profiling and logging view is not available in the streamlined mode.
- You are using advanced features of custom code including debugging custom code. For more information on custom code, see [Extending test execution with custom code on page](#) .

Starting HCL OneTest™ Performance in full Eclipse mode

You can start the product in the full Eclipse mode to continue to use native Eclipse features along with HCL OneTest™ Performance Rational® Service Tester for SOA Quality.

Before you begin

You must have installed HCL OneTest™ Performance.

1. Click **Start > HCL > HCL OneTest Performance - Full Eclipse - Full Eclipse**.
2. Perform the following steps to select a working directory, if you are starting the installation of HCL OneTest™ Performance for the first time.
 - a. Enter the path of a `working directory` in the **Workspace** field or click **Browse** to select the directory.
 - b. Select **Use this as the default and do not ask again** to make this your default workspace.
You can change your workspace from HCL OneTest™ Performance by clicking **File > Switch Workspace**.
3. Click **OK**.

Results

You have started HCL OneTest™ Performance in full Eclipse mode.

Starting HCL OneTest™ Performance in streamlined Eclipse mode

If you do not want to view native Eclipse UI, you can start HCL OneTest™ Performance in streamlined mode.

Before you begin

You must have installed HCL OneTest™ Performance.



Note: To start HCL OneTest™ Performance in the streamlined mode, the streamlined mode must be installed as an optional feature. It is automatically selected during the installation process.

1. Click **Start > HCLHCL OneTest Performance**.
2. Perform the following steps to select a working directory, if you are starting the installation of HCL OneTest™ Performance for the first time.

- a. Enter the path of a `working directory` in the **Workspace** field or click **Browse** to select the directory.
- b. Select **Use this as the default and do not ask again** to make this your default workspace.
You can change your workspace from HCL OneTest™ Performance by clicking **File > Switch Workspace**.

3. Click **OK**.

Results

You have started HCL OneTest™ Performance in streamlined Eclipse mode.

Chapter 4. Tutorials

This section contains the tutorials which explains the main features of HCL OneTest™ Performance.

You can find the following information:

- [Performance test a Web application on page 64](#)
- [Performance test an SAP application on page 72](#)
- [Performance test a Citrix application on page 78](#)

Performance test a Web application

The movies in this tutorial show you the main features of HCL OneTest™ Performance. The tutorial requires Flash Player to view.

Learning objectives

You will learn how to perform the following tasks:

- Record user actions to create a test
- Create a schedule, which enables you to run multiple tests and control how they run
- Run the schedule
- Analyze the results from the schedule run

Time required

45 minutes

Introduction: Test a Web application

This tutorial is designed to introduce you to testing.

Learning objectives

The tutorial is divided into six modules, each with its own learning objectives. You will learn to:

- Record a user actions to create a test
- Create a schedule, which enables you to run multiple tests and control how they run.
- Analyze the results from the schedule run

Time required

This tutorial should take approximately 45 minutes to finish. If you explore other concepts related to this tutorial, it could take longer to complete.

Skill level

Beginner

Prerequisites

This tutorial assumes that you are familiar with using the perspectives and views in the IBM® Rational® Software Delivery Platform.

Module 1: Creating projects and recording user actions

Learn how to create projects and record user actions for testing. The second part of this module will help you to build an understanding of a recorded test.

Learning objectives

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Create a project
- Record a test of user interactions
- View a test in the test editor

Time required

This module requires approximately 8 minutes to complete.

Lesson 1.1: Recording a test

In this lesson, you will learn how to create a project and record user actions. Note that the server under test in this Watch and Learn tutorial is not publicly available.

About this task

See video

Lesson 1.2: Understanding a recorded test

In this lesson, you will become familiar with your recorded test.

About this task

See video

Module 1 summary

In this tutorial module, you learned how to create projects and record user actions for testing

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Creating a project is the first step in testing
- Associating elements of the project, such as the tests, with the project
- Recording the actions of your users
- Identifying the components of a recorded test

Related information

[Recording HTTP tests on page 182](#)

[Editing HTTP tests on page 276](#)

Module 2: Editing a test

See how to edit your recorded test to include variable data using datasets and verification points.

Learning objectives

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Enable a verification point
- Create a dataset
- Use a dataset in a test

Time required

This module requires approximately 8 minutes to complete.

Lesson 2.1: Enabling a verification point

In this lesson, you will learn how to add verification points to a test. Verification points check whether the expected behavior actually occurs during a run.

About this task

See video

Lesson 2.2: Creating a dataset

In this lesson, you will learn how to create a dataset, which enables you to vary the values from what you recorded in the test.

About this task

See video

Lesson 2.3: Using a dataset in a test

In this lesson, you will learn how to enable your test to use a dataset.

About this task

See video

Module 2 summary

In this tutorial module, you learned how to use variable data to edit and manage testing situations.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Adding verification points to check whether an expected behavior occurs during a test run
- Enabling datasets to provide substitute variable data for the fixed values stored in the recorded test
- Importing data that is contained in a comma-separated-value (CSV) file
- Updating your test with a reference to the dataset so that the test can use variables from a dataset during a run

Related information

[Providing tests with variable data on page](#)

Module 3: Validating a test with a single user

Watch a demonstration on running a single-user test for debugging and getting dataset values to work properly.

Learning objectives

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Run a test with a single user
- View the test log

Time required

This module requires approximately 5 minutes to complete.

Lesson 3.1: Running a test with a single user

In this lesson, you will learn how to run a test with a single user, to ensure that your test and the dataset values are working properly.

About this task

See video

Lesson 3.2: Viewing the test log

In this lesson, you will learn how to open the test log and view a record of all the events that occurred during your test run or schedule run.

About this task

See video

Module 3 summary

In this tutorial module, you learned how to run a test with a single user and view the test log.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Viewing changes in the reports while the schedule is running
- Running a test locally with one user
- Running a schedule with a default launch configuration

Related information

[Viewing test logs on page](#)

Module 4: Representing workloads

View how to create schedules that represent a typical workload for your site.

Learning objectives

After completing the lessons in this module, you will understand how to do the following tasks and understand the associated concepts:

- Create a schedule and add user groups
- Add loops to a schedule
- Add tests to a schedule
- Display your assets by type or in alphabetical order.
- Enable resource monitoring

Time required

This module requires approximately 10 minutes to complete.

Lesson 4.1: Controlling test execution with a schedule

In this lesson, you will learn how to control test execution by creating a schedule and adding user groups, loops, and tests to the schedule. At the end of the lesson, you will learn how to display your assets in a logical order.

About this task

See video

Lesson 4.2: Enabling resource monitoring

In this lesson, you will learn how to enable resource monitoring in a schedule and to add data sources.

About this task

See video

Module 4 summary

In this tutorial module, you learned how to represent a workload accurately by creating a schedule and adding user groups, tests, and other elements to it.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Grouping tests in a logical order with user groups
- Adding a loop to a schedule to repeat a test for a number of iterations and set the test run rate
- Adding a test to a schedule to emulate accurately the actions of individual users
- Displaying your assets by type or in the default alphabetical order.
- Monitoring resources to capture system resource data, such as processor or memory usage

Related information

[Emulating workloads on page 522](#)

[Monitoring resource data on page 570](#)

Module 5: Running the test

Become familiar with configuring a schedule, running a schedule, and adding virtual users during a test run.

Learning objectives

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Configure and run the schedule
- View the report
- Add virtual users during a run

Time required

This module requires approximately 10 minutes to complete.

Lesson 5.1: Configuring the schedule

In this lesson, you will learn how to set up a schedule configuration, which controls where the execution results are stored.

About this task

See video

Lesson 5.2: Running the schedule and viewing the reports

In this lesson, you will learn how to run the schedule.

About this task

See video

Lesson 5.3: Adding virtual users during a run

In this lesson, you will learn how to increase the number of virtual users during a schedule run.

About this task

See video

Module 5 summary

In this tutorial module, you learned how to configure and run a schedule, and add virtual users to the schedule during a run.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Specifying the name and location for your execution results by configuring the schedule
- Running a schedule and viewing the reports
- Increasing the number of virtual users in a schedule during the test run

Related information

[Running schedules on page 608](#)

[Setting a launch configuration on page 621](#)

Module 6: Evaluating results

Learn how to retrieve stored test results in reports, customize a report, and get detailed information on an event that occurred in a test.

Learning objectives

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Review reports after the schedule has run
- View the verification report
- Customize and compare reports.

Time required

This module requires approximately 5 minutes to complete.

Lesson 6.1: Viewing verification point status

In this lesson, you will learn how to view the status of the verification points that you set in the test.

About this task

See video

Lesson 6.2: Customizing and comparing reports

In this lesson, you will learn how to customize reports to investigate a specific performance problem in more detail than that provided in the default reports. You will also learn how to compare two reports.

About this task

See video

Module 6 summary

In this tutorial module, you learned how to retrieve stored test results in reports, customize a report, and get detailed information on an event that occurred in a test.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Evaluating the results that are generated dynamically during a run
- Regenerating the results for viewing and analysis after a run
- Customizing and comparing reports

Related information

[Customizing reports on page 778](#)

Summary

This tutorial has taught you the basics of Rational® Performance Tester. You have learned how this product provides the following functions and features:

- Ease of recording
- A graphical test editor for programmers and nonprogrammers alike
- Datasets to supply realistic test data
- Real-time reporting of response time

Lessons learned

After completing all of the modules, you should now be able to perform the following tasks:

- Create projects and record user actions
- Edit and manage variable data using datasets
- Represent workloads
- Run schedules
- View reports
- Add virtual users
- Retrieve reports from previous runs and customize them

Performance test an SAP application

The movies in this tutorial show you the main features of IBM® Rational® Performance Tester for testing applications on SAP servers. The tutorial requires Flash Player to view.

Learning objectives

You will learn how to perform the following tasks:

- Record a test for SAP
- Create a multi-user schedule
- Analyze SAP test results

45 minutes

Introduction: Performance test SAP solutions

This tutorial introduces you to performance testing applications running on SAP servers.

Learning objectives

The tutorial is divided into six modules, each with its own learning objectives. You will learn to perform the following tasks:

- Record an SAP test
- Create a multi-user schedule
- Analyze SAP test results

This tutorial requires approximately 45 minutes to finish. If you explore other concepts related to this tutorial, it might take longer to complete.

Skill level

Beginner

Prerequisites

To complete this tutorial, you need to be familiar with using the perspectives and views in IBM® Rational® Software Development Platform, with SAP , and with the SAP GUI environment.

Module 1: Creating projects and recording user actions

In this module, you will learn how to create projects and record user actions for testing. The second part of this module will help you understand a recorded test.

Learning objectives

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Create a test project
- Record a test of user interactions
- View a test in the test editor

This module requires approximately 8 minutes to complete.

Lesson 1.1: Recording an SAP test

In this lesson, learn how to create a test project and record user actions in the SAP GUI client.

About this task

The most convenient way to create a test in IBM® Rational® Performance Tester is to record an SAP session. The recording contains SAP events that are transmitted between the SAP GUI client and SAP server.

For each SAP transaction, the recorder logs all the SAP events that occur in the SAP GUI and generates a test that can reproduce the sequence of events in the SAP GUI.

For SAP recording tips, see SAP performance testing guidelines.

See video

Lesson 1.2: Understanding a recorded test

In this lesson, become familiar with your recorded test.

About this task

With the test editor, you can edit and view the elements that constitute your SAP performance test.

The left pane of the editor displays the structure of the test as a sequence of SAP transactions, which contain the recorded SAP Set, Get, and Call events. You can add elements to this sequence, such as transactions, loops, conditions, synchronization points, or comments. However, if you change the structure of the test, you must ensure that the SAP Set events are always applicable to the situation on the SAP GUI screen.

The right pane displays details about the current test element.

See video

Lesson 1.3: Creating an SAP batch input test

In this lesson, learn how record a batch input test for generating a load on the SAP server.

About this task

With SAP batch input tests, you can generate a load on the server without requiring that each virtual tester run the SAP GUI client. For performance testing, this is particularly useful because the test computer can typically simulate a much larger number of users with batch input virtual testers than with SAP GUI virtual testers.

However, SAP batch input tests cannot contain verification points and do not produce performance results by themselves. Therefore, you typically use batch input tests in a schedule mixed with SAP GUI tests to increase the load on the server. The actual test results are produced by the SAP GUI tests.

See video

Module 1 summary

In this module, you learned how to create projects and record user actions for testing.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Creating a project is the first step in performance testing
- Associating elements of the project, such as tests, with the project
- Recording the actions of your users in a test and a batch input test
- Identifying the components of a recorded test

Module 2: Editing an SAP test

In this module, you will see how to edit your recorded test to include variable data using datasets and verification points.

Learning objectives

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Add a verification point
- Create a dataset
- Use a dataset in a test

This module requires approximately 8 minutes to complete.

Lesson 2.1: Adding verification points

In this lesson, learn how to add verification points to check whether an expected behavior occurs during a run.

About this task

With verification points, you can verify the behavior of the SAP application during a test. For example, you can use verification points to ensure that a particular transaction returns the expected result in any graphical element of the SAP GUI screen. You can also use regular expressions and you can search for the occurrence of an expected string in a list.

Each verification point returns a `Pass`, `Fail` or `Inconclusive` verdict in the test log. You can view a summary of verification point verdicts in the verification point report after running the test.

See video

Lesson 2.2: Creating and using datasets

In this lesson, learn how to create a dataset that can provide tests with variable data and how to enable your test to use a dataset during a run.

About this task

Datasets provide tests with variable data during a run. When you record a test, you perform a sequence of steps that you expect a typical user to perform. From the recording, a test is generated that exactly reproduces these interactions. During a run, this test uses the same data that you used during recording. To vary the data in the test, you use a dataset, which is typically a table that contains variable data. At run time, this variable data is substituted for the data in the recorded test.

See video

Module 2 summary

In this module, you learned how to use variable data to edit and manage testing situations.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Adding verification points to check whether an expected behavior occurs during a test run
- Enabling datasets to provide substitute variable data for the fixed values stored in the recorded test
- Importing data that is contained in a comma-separated-value (CSV) file
- Updating your test with a reference to the dataset so that the test can use variables from a dataset during a run

Module 3: Running tests

In this module, you will see how to create schedules that represent a typical workload for your SAP server.

Learning objectives

After completing the lessons in this module, you will understand how to do the following tasks and understand the associated concepts:

- Create a schedule and add user groups
- Add loops to a schedule
- Add tests to a schedule

This module requires approximately 10 minutes to complete.

Lesson 3.1: Running a single test and viewing the test log

In this lesson, learn how to validate a test by running it with a single user and viewing the results in the test log.

About this task

Before deploying a full emulated workload test with a schedule, ensure that your test runs adequately with a single user. Developing a performance test often requires several iterations of editing the test and validating changes with a single user run.

See video

Lesson 3.2: Creating a schedule and adding user groups

In this lesson, learn how to control test execution by creating a schedule and adding user groups.

About this task

A schedule can be as simple as one virtual user running one test, or as complicated as hundreds of virtual users in different groups, each running different tests at different times.

A schedule is the "engine" that runs a test. You add user groups, tests, and other items to the schedule to emulate a workload. However, schedules are much more than simple vehicles for running tests. For example, you can use a schedule to control tests in the following ways:

- Group tests under user groups to emulate the actions of different types of users.
- Set the order in which tests run: sequentially, randomly, or in a weighted order.
- Set the number of times that each test runs.
- Run tests at a certain rate.
- Run tests for a certain time and increase or decrease virtual users during the run.

See video

Lesson 3.3: Running the schedule and viewing the performance report

In this lesson, learn how to run a schedule in a custom run configuration and to view the results in the SAP performance report.

About this task

After you have validated your individual tests and added them to a schedule, you are ready to deploy the schedule and run the virtual users on remote computers. By using remote computers, you can multiply the number of virtual users to simulate a real work load. During the run, you can monitor the behavior of each virtual user.

When the test is finished, you can view the performance report to help locate bottlenecks in your SAP environment.

See video

Module 3 summary

In this module, you learned how represent a workload accurately by creating a schedule and adding user groups, tests, and other elements to it.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Grouping tests in a logical order with user groups
- Adding a test to a schedule to emulate accurately the actions of individual users
- Adding a loop to a schedule to repeat a test for a number of iterations and set the test run rate

Summary

This tutorial introduced you the basics of the Rational® Performance Tester Extension *for SAP Solutions*. You have learned how this product provides the following functions and features:

- Ease of recording
- A graphical test editor for programmers and nonprogrammers alike
- Datasets to supply realistic test data
- Real-time reporting of response time

Lessons learned

After completing all of the modules, you can perform the following tasks:

- Create projects and record user actions
- Edit and manage variable data using datasets
- Represent workloads
- Run schedules
- View reports
- Add virtual users
- Retrieve reports from previous runs

Resources

To learn more about using Rational® Performance Tester to test the performance of your SAP environment, visit developerWorks®, IBM's resource for developers, at: <http://www.ibm.com/developerworks/>

Performance test a Citrix application

The movies in this tutorial show you the main features of using IBM® Rational® Performance Tester for testing applications running in a Citrix XenApp environment. The tutorial requires Flash Player to view.

Learning objectives

You will learn how to perform the following tasks:

- Record a test for Citrix
- Create a multi-user schedule
- Analyze Citrix test results

45 minutes

Introduction: Performance test Citrix applications

This tutorial is designed to introduce you to performance testing applications running on Citrix XenApp.

Learning objectives

The tutorial is divided into five modules, each with its own learning objectives. You will learn to perform the following tasks:

- Record a Citrix test
- Create a multi-user schedule
- Analyze Citrix test results

This tutorial requires approximately 45 minutes to finish. If you explore other concepts related to this tutorial, it might take longer to complete.

Skill level

Beginner

Prerequisites

To complete this tutorial, you need to be familiar with using the perspectives and views in the IBM® Rational® Software Development Platform and with Citrix XenApp.

Module 1: Creating projects and recording user actions

In this module, you will learn how to create projects and record user actions for testing. The second part of this module will help you to build an understanding of a recorded test.

Learning objectives

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Create a project
- Record a test of user interactions
- View a test in the test editor

This module requires approximately 8 minutes to complete.

Lesson 1.1: Recording a Citrix performance test

In this lesson, you learn how to create a test project and record user actions in Citrix XenApp.

About this task

The most convenient way to create a new test in IBM® Rational® Performance Tester is to record a Citrix session. The recording contains mouse and key strokes as user input actions and window events as responses from the Citrix server.

Because of the way the recorder works, consider these issues during the recording:

- Set up your application so that all actions are repeatable. For example, turn off "tips of the day," ensure that any `file already exists` warnings are consistent, and avoid using dynamic menu items such as **recent documents**.
- Ensure that mouse movements are clearly decomposed. The recorder does not record mouse wheel scrolls.
- Use specific Microsoft® Windows® accounts for performance testing that have limited potential for data loss if there are errors or synchronization failures during the test run.

For more Citrix recording tips, see [Citrix performance testing guidelines on page 209](#).

The recorder synchronizes user actions and window events automatically so that the generated test waits for a window event before triggering the next user input. In some cases, however, you must manually add synchronizations. For example, add synchronizations when screen updates do not result directly from a user action. These synchronizations tell the test to wait for a specific window, text, or graphic to open on the screen before running the next user input.

See video

Lesson 1.2: Understanding a recorded test

In this lesson, you become familiar with your recorded test.

About this task

With the test editor, you can edit and view the elements that constitute your Citrix performance test.

The left pane of the editor displays the structure of the test as a sequence of input actions and window events. You can add elements to this sequence, such as transactions, loops, conditions, synchronization points, and comments. However, if you change the structure of the test, you must ensure that the input actions always apply to the context of the actual Citrix screen.

The right pane displays details about the current test element.

See video

Module 1 summary

In this module, you learned how to create projects and record user actions for testing.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Creating a project is the first step in performance testing.
- Associating elements of the project, such as tests, with the project.
- Recording the actions of your users in a test.
- Identifying the components of a recorded test.

Module 2: Editing a Citrix performance test

In this module, you will see how to edit your recorded test to include verification points and response time measurements.

Learning objectives

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Enable a verification point
- Create a response time measurement

This module requires approximately 8 minutes to complete.

Lesson 2.1: Enabling verification points

In this lesson, learn how to add verification points to check whether an expected behavior occurs during a run.

About this task

With verification points, you can test the behavior of the application during a test. For example, you can use verification points to ensure that a particular window is displayed after a specific action or that a portion of the screen graphically matches an expected bitmap.

Each verification point returns a `Pass`, `Fail`, or `Inconclusive` verdict in the test log. You can view a summary of verification point verdicts in the verification point report after running the test.

In Citrix tests you can also use optical character recognition (OCR) on a screen capture to check for text content.

See video

Lesson 2.2: Creating a response time measurement

In this lesson, learn how to create a response time measurement that can record the time that the application spends processing an event.

About this task

Response time measurements report how fast the application responds to a specific user input event. Typically, response time measurements are automatically created between a user input action and the window event that occurs immediately after.

In addition, you can manually add your own response time measurements, for example to record a global response time that covers multiple window events.

See video

Module 2 summary

In this module, you learned how to use verification points and response time measurements to evaluate server response.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Adding verification points to check whether an expected behavior occurs during a test run
- Setting up timers inside the test to measure response times

Module 3: Representing workloads

In this module, you will see how to create schedules that represent a typical workload for your Citrix server.

Learning objectives

After completing the lessons in this module, you will understand how to do the following tasks and understand the associated concepts:

- Run individual validation tests
- Create a schedule and add user groups
- Add loops to a schedule
- Add tests to a schedule

- Run schedules using a run configuration
- View results in the test log and the Citrix performance report

This module requires approximately 20 minutes to complete.

Lesson 3.1: Running a single test and viewing the test log

In this lesson, learn how to validate a test by running it with a single user and viewing the results in the test log.

About this task

Before you deploy a full emulated workload test with a schedule, ensure that your test runs adequately with a single user. Developing a performance test often requires several iterations of editing the test and validating changes with a single user run.

See video

Lesson 3.2: Creating a schedule and adding user groups

In this lesson, learn how to control test execution by creating a schedule and adding user groups.

About this task

A schedule can be as simple as one virtual user running one test, or as complicated as hundreds of virtual users in different groups, each running different tests at different times.

A schedule is the "engine" that runs a test. You add user groups, tests, and other items to the schedule to emulate a workload. However, schedules are much more than simple vehicles for running tests. For example, you can use a schedule to control tests in the following ways:

- Group tests under user groups to emulate the actions of different types of users.
- Set the order in which tests run: sequentially, randomly, or in a weighted order.
- Set the number of times that each test runs.
- Run tests at a certain rate.
- Run tests for a certain time, and increase or decrease virtual users during the run

See video

Lesson 3.3: Running the schedule and viewing the performance report

In this lesson, learn how to run a schedule in a custom run configuration and to view the results in the performance report.

About this task

After you have validated your individual tests and added them to a schedule, you are ready to deploy the schedule and run the virtual users on remote computers. By using remote computers, you can multiply the number of virtual users to simulate a real work load. During the run, you can monitor the behavior of each virtual user.

When the test is finished, you can view the performance report to help locate bottlenecks in your Citrix environment.

See video

Module 3 summary

In this module, you learned how represent a workload accurately by creating a schedule and adding user groups, tests, and other elements to it.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Running a test locally with one user
- Viewing changes in the reports while the schedule is running
- Grouping tests in a logical order with user groups
- Adding a test to a schedule to emulate accurately the actions of individual users
- Adding a loop to a schedule to repeat a test for a number of iterations and set the test run rate
- Running a schedule with a default run configuration

Summary

This tutorial has taught you the basics of the Rational® Performance Tester Extension for Citrix Presentation Server. You have learned how this product provides the following functions and features:

- Ease of recording
- A graphical test editor for programmers and nonprogrammers
- Real-time reporting of response time

Lessons learned

After completing all of the modules, you can now perform the following tasks:

- Create projects and record user actions
- Represent workloads
- Run schedules
- View reports
- Add virtual users
- Retrieve reports from previous runs

Resources

To learn more about using performance tests on your Citrix environment, visit developerWorks®, IBM's resource for developers, at: <http://www.ibm.com/developerworks/>

Chapter 5. Samples

This section describes about the sample project which can be used with HCL OneTest™ Performance to test the functionality of an application.

Installation tuning tests for WebSphere® Application Server

The installation tuning tests were created to quickly and conveniently apply load to an application server for the purpose of tuning the application server for maximum throughput. The three tests that are provided stress various parts of the application server.

Time required to import sample project: 5 minutes

The installation tuning tests are designed to test IBM® WebSphere® Application Server Version 7. Each test has different requirements that must be met before you run the test.

- The Schedule_Snoop test is ready to use with any WebSphere® Application Server installation.
- The "Schedule_Plants: PlantsByWebSphere" sample application must be installed. The sample application is provided with WebSphere® Application Server, but its installation is optional.
- The Schedule_Daytrader test application is freely available from the Apache project Geronimo. Daytrader requires access to an IBM DB2 database.

Best results can be achieved for all tests by using one or more separate, dedicated computers to provide the greatest possible loads on the application server. When testing a large WebSphere® Application Server application, a single computer that runs HCL OneTest™ Performance and provides a load on the application might not provide sufficient stress to test the application.

An installation tuning test is designed to stress your application server to achieve maximum throughput from basic tuning. Tuning is a complex procedure with many possible adjustments available according to the varying requirements of the application that you plan to run. For more information on tuning WebSphere® Application Server, see http://www.ibm.com/developerworks/websphere/techjournal/0909_blythe/0909_blythe.html.

A simple goal when you run the installation tuning tests is to achieve the highest percentage of CPU utilization on the application server while the test runs. Follow these guidelines as you run each test:

- Use Task Manager, vmstat, or a similar tool to observe CPU utilization on the computer that runs the application server. With maximum throughput as a goal, you work to achieve the highest CPU percentage possible. To achieve maximum throughput, this number must increase as you make tuning adjustments.
- Monitor the CPU utilization of the computer that runs HCL OneTest™ Performance. If CPU utilization exceeds 70%, you might be approaching the limits of what that computer can provide. However, HCL OneTest™ Performance supports using multiple computers to achieve an even greater load.
- Because these tests provide high throughput, the tests might exceed the capabilities of your network. A tool such as Task Manager can show network utilization. If you are limited by a 100 Mbps network, access to a 1000 Mbps network might be required to fully stress the application server.

Related information

[Installing the assets for tuning tests on page 85](#)

[Testing with the Snoop test on page 85](#)

[Testing the Daytrader application on page 86](#)

[Testing the PlantsByWebSphere application on page 85](#)

Installing the assets for tuning tests

HCL OneTest™ Performance provides Installation Tuning Tests for WebSphere Application Server as a sample project that you can import into your workspace.

1. Right-click on [Installation Tuning Tests for WebSphere Application Server](#) and save the sample project.



Note: After saving, import the sample project in Eclipse. To import the sample, click **File > Import > Existing projects into workspace > Select archive file** and select the downloaded sample project file.

2. Click **Finish**. If you are asked to switch to the **Performance Test Perspective**, click **Yes**.

What to do next

Expand the sample in the **Test Navigator** view to open the project and explore the resources and reports.

Testing with the Snoop test

The Snoop test stresses the servlet engine of IBM® WebSphere® Application Server.

Snoop is an application that is provided with all WebSphere® Application Server installations. This exercises only a small, direct portion of the application server. The Plants test is more complex than the Snoop test.

1. Select **Snoop** in the Test Project to open it in the editor.
2. Open **Test Variables**.
3. In **hostname** enter the name of the WebSphere® Application Server to stress test.
4. Select **Schedule_Snoop** in the Test Project to open the test in the editor.
5. Click **Run** to start the test. The test runs for approximately 5 minutes.

Testing the PlantsByWebSphere application

The Plants test provides load using the PlantsByWebSphere application.

This application is provided with IBM® WebSphere® Application Server, but it must be installed. Use the WebSphere Administrative console to ensure that the PlantsByWebSphere application is installed and running.

The Plants test is more complex than the Snoop test, because the test exercises more parts of the application server.

1. Select **Plants** in the Test Project to open the test in the editor.
2. Open **Test Variables**.
3. In **hostname** enter the name of the WebSphere® Application Server to stress test.
4. Select **Schedule_Plants** in the Test Project to open it in the editor.
5. Click **Run** to start the test. The test runs for approximately 5 minutes.

Testing the Daytrader application

The Daytrader test provides a load using the Daytrader application.

This application is available at <http://geronimo.apache.org/GMOxDOC30/daytrader-a-more-complex-application.html>.

1. Click **Daytrader** in the Test Project to open the application in the editor.
2. Open the **Test Variables**.
3. Change the **hostname** variable to the name of the IBM® WebSphere® Application Server to stress test.
4. Select **Schedule_Daytrader** in the Test Project to open the schedule in the editor.
5. Click **Run** to start the test. The test runs for approximately 5 minutes.

On the **Daytrader application configuration** page, you can change the configuration from Enterprise JavaBeans (EJB) to Java Database Connectivity (JDBC). Run the Daytrader test in both modes to obtain the widest possible coverage.

Chapter 6. Administrator Guide

This guide describes how to install HCL OneTest™ Performance. After you install the software, you can perform administration tasks such as license configuration and integration with other products. This guide is intended for administrators.

Installation of HCL OneTest™ Performance

Installing the product involves verifying requirements, planning, managing licenses, and configuring web-based help. This section lists all such topics.

This installation guide covers two independent products: HCL OneTest™ Performance and the HCL OneTest™ Performance Agent. The HCL OneTest™ Performance Agent is a tool that you use with HCL OneTest™ Performance. It is included as part of the HCL OneTest™ Performance product kit.

HCL OneTest™ Performance Agent consists of two capabilities:

- Generate load for the application under test by using the virtual users. You can increase the load generation capacity by installing additional agents on remote computers.
- Gather data for the Response Time Breakdown feature and in support of the startup and control of web services stubs in the SOA protocol.



Note: The HCL OneTest™ Performance and HCL OneTest™ Performance Agent are separate offerings and must be installed separately.

Installation requirements

Installation requires the correct hardware, software, server environment, operating systems, and user privileges for installing and running your software.

Hardware and Software requirements

Before you install the product, verify that your system meets the hardware and software requirements.

For information about hardware and software compatibility, see [System Requirements on page 16](#).

User privileges requirements

You must have a user ID that meets the following requirements to install HCL OneTest™ Performance Rational® Service Tester for SOA Quality and HCL OneTest™ Performance Agent.



Notes:



- Your user ID must not contain double-byte characters.
- For Windows operating system, you must have a user ID that belongs to the Administrators group.
- For Linux operating system, you must be able to log in as root.

Installation conventions and terminology

Understanding these terms and conventions can help you take full advantage of the installation information and your product.

The following conventions are used in this installation information:

- The default installation directory is written as `C:\installation_directory\product\inst.file`.
- The default log location for installation information is `C:\log_file_dir\log.txt`.

These terms are used in the installation topics.

Installation directory

The location of product artifacts after the package is installed.

Package

An installable unit of a software product. Software product packages are separately installable units that can operate independently from other packages of that software product.

Package group

A package group is a directory in which different product packages share resources with other packages in the same group. When you install a package using Installation Manager, you can create a new package group or install the packages into an existing package group. Eclipse-based packages installed in the same package group are able to use the shell-sharing features of Eclipse. Some packages cannot share a package group, in which case the option to use an existing package group is unavailable.

Repository

A storage area for installable software packages. A repository can be disc media, a folder on a local hard disk, or a server or web location.

Shared directory

In some instances, product packages can share resources. These resources are located in a directory that the packages share.

Installation Manager overview

Installation Manager is a program for installing, updating, and modifying packages. It helps you to manage the applications or packages that it installs on your computer. Installation Manager also helps you to keep track of what you have installed, determine what is available for you to install, and to organize installation directories.

Installation Manager provides features that help you keep packages up to date, modify packages, manage the licenses for your packages, and uninstall packages.

Installation Manager includes six wizards that make it easy to maintain packages:

- The **Install** wizard walks you through the installation process. You can install a package by simply accepting the defaults or you can modify the default settings to create a custom installation. Before you install, you get a complete summary of your selections throughout the wizard. Using the wizard you can install one or more packages at one time.
- The **Update** wizard searches for available updates to packages that you have installed. An update might be a released fix, a new feature, or a new version of the product. Details of the contents of the update are provided in the wizard. You can choose whether to apply an update. The **Update** wizard searches connected repositories for updates. If you are not connected to the Internet, you may not see newly available updates for your installed products. To apply an update to a computer that is not connected to the Internet, you must download the update and extract it to a local repository.
- The **Modify** wizard helps you modify certain elements of a package that you have already installed. During the first installation of the package, you select the features that you want to install. Later, if you require other features, you can use the modify packages wizard to add them to your package. You can also remove features and add or remove languages.
- The **Manage Licenses** wizard helps you set up the licenses for your packages. Use this wizard to change your trial license to a full license, to set up your servers for floating licenses, and to select which type of license to use for each package. HCL OneTest™ Performance requires runtime floating license keys to run tests with multiple virtual users and to use product extensions such as protocols. Runtime floating license keys are not managed using Installation Manager. Use the License Key Administrator program, installed with the Rational® License Server, to manage runtime floating license keys.
- The **Roll Back** wizard helps you to revert to a previous version of a package.
- The **Uninstall** wizard removes a package from your computer. You can uninstall more than one package at a time.

Installation locations

Installation Manager retrieves product packages from specified repositories and installs the products into selected locations, which are referred to as package groups.

Package groups

During installation, you specify a *package group* into which to install a product.

- A package group represents a directory in which products share resources.
- When you install a product by using Installation Manager, you either create a package group or install the product into an existing package group. A new package group is assigned a name automatically; however, you choose the installation directory for the package group.
- After you create a package group you cannot change the installation directory. The installation directory contains files and resources that are shared by the products that are installed into that package group.

- Product resources that are designed to be shared with other packages are installed in the shared resources directory. Not all products can share a package group, in which case the option to use an existing package group is disabled.
- When you install multiple products at the same time, all products are installed into the same package group.

Shared resources directory

The *shared resources directory* is where product resources are installed so that they can be used by multiple product package groups. You define the shared resources directory the first time that you install the first product package. For best results, use your largest disk drive for shared resources directories. You cannot change the directory location unless you uninstall all product packages.

Coexistence

Some products are designed to coexist and share functions when they are installed in the same package group. A package group is a location where you can install one or more software product packages.

Offering coexistence considerations

When you install each product package, you select whether to install the product package into an existing package group or to create a package group. Installation Manager prevents you from installing products into package groups products that are not designed to share or do not meet version compatibility and other requirements. To install more than one product at a time, the products must be able to share a package group.

Any number of eligible products can be installed to a package group. When a product is installed, the product functions are shared with all the other products in the package group. If you install a development product and a testing product into one package group, when you start either of the products, you have both the development and testing functions available to you in your user interface. If you add a product with modeling tools, all the products in the package group have the development, testing, and modeling functions available.

Eclipse instance overview

The product package that you install using Installation Manager comes with a version of Eclipse, which is the base platform of this product package. If you already have Eclipse installed on your workstation, you can add your product package directly to that Eclipse installation and extend the functions of the Eclipse integrated development environment (IDE).


Extending an Eclipse IDE adds the functions of the newly installed product, but maintains your IDE preferences and settings. Previously installed plug-ins are also still available.

In most cases, your current Eclipse IDE must be the same version as the Eclipse that the product you are installing uses. Installation Manager checks that the Eclipse instance that you specify meets the requirements for the installation package and helps you install the latest updates from eclipse.org, if required.

Increasing the number of file handles on Linux™ workstations

For best product performance, increase the number of file handles above the default setting of 1024 handles.

About this task


 **Important:** Before you work with your product, increase the number of file handles. Most of the products use more than the default limit of 1024 file handles per process. A system administrator might need to make this change.

Exercise caution when using the following steps to increase your file descriptors on Linux™. If the instructions are not followed correctly, the computer might not start correctly.

1. Log in as root.


If you do not have root access, you must obtain it before continuing.

2. Change to the `etc` directory.


 **Attention:** If you decide to increase the number of file handles in the next step, *do not* leave an empty `initscript` file on your computer. If you do so, your computer will not start up the next time that you turn it on or restart.

3. Use the `vi` editor to edit the `initscript` file in the `etc` directory. If this file does not exist, type `vi initscript` to create it.
4. On the first line, type `ulimit -n 30000`.

The point is that 30000 is significantly larger than 1024, the default value on most Linux™ computers.

 **Important:** Do not set the number of handles too high, because doing so can negatively impact system-wide performance.

5. On the second line, type `eval exec "$4"`.
6. Save and close the file after making sure that you have completed steps 4 and 5.

 **Note:** Ensure that you follow the steps correctly. If this procedure is not completed correctly, your computer will not start.

7. **Optional:** Restrict the number of handles available to users or groups by modifying the `limits.conf` file in the `etc/security` directory.

If you do not have this file, consider using a smaller number in step 4 in the previous procedure (for example, 2048). Do this so that most users have a reasonably low limit on the number of open files that are allowed per process. If you use a relatively low number in step 4, it is less important to do this. However, if you set a high number in step 4 earlier and you do not establish limits in the `limits.conf` file, computer performance can be significantly reduced.

The following sample `limits.conf` file restricts all users, and then sets different limits for others afterwards. This sample assumes that you set handles to 8192 in step 4 earlier.

```
*      soft nofile 1024
*      hard nofile 2048
root   soft nofile 4096
root   hard nofile 8192
user1  soft nofile 2048
user1  hard nofile 2048
```

Note that the `*` in the preceding example sets the limits for all users first. These limits are lower than the limits that follow. The root user has a higher number of allowable handles open, while the number that is available to user1 is between the two. Make sure that you read and understand the documentation that the `limits.conf` file contains before making changes.

What to do next

For more information on the `ulimit` command, see the main page for `ulimit` in the Linux™ documentation.

Installation of the product by using Installation Manager

You must download the product bits and then from the Installation Manager point to the Setup disk.



Note: For Linux™ computers, you must log in as the root user before you begin installation process.

Installing HCL OneTest™ Performance

To test the performance of an application, you must install HCL OneTest™ Performance.

Before you begin

- Download and install Installation Manager from <https://jazz.net/downloads/ibm-installation-manager/>.
- If you were using the IBM testing product and want to reuse the test assets in the HCL testing product, follow these instructions:
 - From the IBM testing product, export all the test assets with dependencies. Follow instructions till step 5 in [Copying test assets with dependencies on page 396](#).
 - Uninstall the IBM testing product.
 - Install the HCL testing product by following the instructions.
 - Import the test assets with dependencies to the HCL workspace. Follow instructions from step 6 in [Copying test assets with dependencies on page 396](#).

About this task

For more information about installing the product from the command prompt in the silent mode, see [IBM Installation Manager Knowledge Centre](#).

**Notes:**

- You cannot upgrade to the latest version of the product. You must uninstall the existing version of the product before installing the latest version of the product.
- Currently, collection of response time breakdown data from macOS system is not supported.

1. In the Installation Manager, click **File > Preferences > Repositories**, and add a repository link to the product's setup disk and click **OK**.
2. Click **Install**.
3. Click a product package to highlight it.

Result

The description of the package is displayed in the **Details** pane at the end of the screen.

4. To search for updates to the product packages, click **Check for Other Versions, Fixes, and Extensions**. If updates for a product package are found, then they are displayed in the **Installation Packages** list on the **Install Packages** page under their corresponding products. Only recommended updates are displayed by default.

Choose from:

- To view all updates that are found for the available packages, click **Show all versions**.
- To display a package description in the **Details** pane, click the package name. If additional information about the package is available, such as a `readme` file or release notes, a **More info** link is included at the end of the description text. Click the link to display the additional information in a browser. To fully understand the package that you are installing, review all information.



Note: For Installation Manager to search the predefined IBM® update repository locations for the installed packages, the **Search the linked repositories during installation and updates** preference on the **Repositories** preference page must be selected. This preference is selected by default. Internet access is also required. A progress indicator shows that the search is taking place. You can install updates at the same time that you install the base product package.

5. Select the product package and any updates to the package to install. Updates that have dependencies are automatically selected and cleared together. Click **Next** to continue.



Note: If you install multiple packages at the same time, then all the packages are installed into the same package group.

6. On the **Licenses** page, read the license agreement for the selected package.
 - a. If you agree to the terms of all of the license agreements, click **I accept the terms of the license agreements**.
 - b. Click **Next** to continue.

7. On the **Location** page, type the path for the *shared resources directory* in the **Shared Resources Directory** field, or accept the default path. The shared resources directory contains resources that can be shared by one or more package groups. Click **Next** to continue.

The default path to use follows:

- **Windows** C:\Program Files\HCL\HCLIMShared
- **Linux** /opt/HCL/HCLIMShared
- **Mac OS X** /Applications/HCL/HCLIMShared



Important: You can specify the shared resources directory only the first time that you install a package. Use your largest disk for this to help ensure adequate space for the shared resources of future packages. You cannot change the directory location unless you uninstall all packages.

8. On the **Location** page, create a *package group* to install the product package into or if this is an update, use the existing package group. A package group represents a directory in which packages share resources with other packages in the same group. To create a package group:
 - a. Click **Create a new package group**.
 - b. Type the path for the installation directory for the package group.
The name for the package group is created automatically.

The default path follows:

- **Windows** C:\Program Files\HCL\HCLOneTest
- **Linux** /opt/HCL/HCLOneTest
- **Mac OS X** /Applications/HCL/HCLOneTest

- c. Click **Next** to continue.

9. **Optional:** On the next **Location** page, you can choose to extend an existing Eclipse IDE that is installed on your computer, which adds the functions in the packages that you are installing. You must have Eclipse Version 3.6 with the latest updates from eclipse.org to select this option. Click **Next** to continue.



Note: HCL OneTest™ Performance does not support extending an existing Eclipse IDE.

10. On the **Features** page under **Translations**, select the languages for the package group. The corresponding translations for the user interface and documentation for the product package will be installed.
11. On the next **Features** page, select the package features to install.
 - a. **Optional:** To see the dependency relationships between features, select **Show Dependencies**.
 - b. **Optional:** Click a feature to view its brief description under **Details**.

- c. Select or clear features in the packages. Installation Manager automatically enforces any dependencies with other features and displays updated download sizes and disk space requirements for the installation.
- d. When you integrate HCL OneTest™ Performance along with Rational® Application Developer or Rational® Software Architect in a shell-shared mode, you must clear **Java 8 OpenJDK with Eclipse OpenJ9** check box during the installation. If you not clear this check box, multiple instances of OpenJDK are installed as Rational® Application Developer and Rational® Software Architect already have an existing OpenJDK.



Note: If you install Rational® Application Developer or Rational® Software Architect in a separate package, you can select the **Java 8 OpenJDK with Eclipse OpenJ9** check box.

- e. When you are finished selecting features, click **Next** to continue.
12. On the **Summary** page, review your choices before installing the product package. To change the choices that you made on previous pages, click **Back**, and make your changes. When you are satisfied with your installation choices, click **Install** to install the package.

Result

A progress indicator shows the percentage of the installation that is completed.

13. When the installation process is complete, a message confirms the completion of the process.
- a. Click **View log file** to open the installation log file for the current session in a new window. You must close the **Installation Log** window to continue.
 - b. In the Install Package wizard, select whether to start the product when you exit.
 - c. Click **Finish** to start installing the selected package.
14. License the product.

See the [Applying the license on page 101](#) topic.

Installing HCL OneTest™ Performance Agent

You must install HCL OneTest™ Performance Agent on different computers to apply load on the server that hosts the application under test.

Before you begin

- To install the software using Installation Manager, download and install Installation Manager from <https://jazz.net/downloads/ibm-installation-manager/>.

About this task

For more information about installing the product from the command prompt in the silent mode, see [IBM Installation Manager Knowledge Centre](#).

**Notes:**

- To use the response time breakdown feature in HCL OneTest™ Performance Agent 8.7 or later, uninstall the existing version of the agent and install it afresh.
- Currently, collection of response time breakdown data from macOS system is not supported.
- You cannot upgrade to the latest version of the product. You must uninstall the existing version of the product before installing the latest version of the product.

1. In the Installation Manager, click **File > Preferences > Repositories**, and add a repository link to the product's setup disk and click **OK**.
2. Click **Install**.
3. Click a product package. Its description is displayed in the **Details** pane at the end of the screen.
4. To search for updates to the product packages, click **Check for Other Versions, Fixes, and Extensions**. Make sure you are connected to the Internet. If updates for a product package are found, they are displayed in the **Installation Packages** list on the **Install Packages** page under their corresponding product. Only recommended updates are displayed by default.

Choose from:

- To view all of the updates that are found for the available packages, click **Show all versions**.
 - To display a package description under **Details**, click the package name. If additional information about the package is available, such as a `readme` file or release notes, a **More info** link is included at the end of the description text. Click the link to display the additional information in a browser. To fully understand the package that you are installing, review all of the information.
5. Select the product package and updates to the package to install. Updates that have dependencies are automatically selected and cleared together. Click **Next** to continue.




Note: If you install multiple packages at the same time, all the packages are installed into the same package group.

6. On the **Licenses** page, read the license agreement for the selected package:
 - a. Read and understand the terms of all of the license agreements before clicking **I accept the terms of the license agreements**.
 - b. Click **Next** to continue.
7. On the **Location** page, in the **Shared Resources Directory** field, type the path for the *shared resources directory* or accept the default path. The shared resources directory contains resources that one or more package groups can share. Click **Next** to continue.

The following default paths are provided:

- **Windows** C:\Program Files\HCL\HCLIMShared
- **Linux** /opt/HCL/HCLIMShared
- **Mac OS X** /Applications/HCL/HCLIMShared

 **Important:** You can specify the shared resources directory only for the first time when you install a package. You must use your largest disk for this directory to help ensure adequate space for the shared resources of future packages. You cannot change the directory location unless you uninstall all packages.

8. On the **Location** page, create a package group to install the product. If you are updating the product, use the existing package group. A package group represents a directory in which packages share resources with other packages in the same group. To create a package group, complete these steps:

- a. Click **Create a new package group**.
- b. Type the path for the installation directory for the package group.
The name for the package group is created automatically.

The following default paths are provided:

- **Windows** C:\Program Files\HCL\HCLOneTest
- **Linux** /opt/HCL/HCLOneTest
- **Mac OS X** /Applications/HCL/HCLOneTest

- c. Click **Next** to continue.

9. HCL OneTest™ Performance Agent does not support extending an existing Eclipse IDE, so you can ignore this page.

10. On the **Features** page, under **Translations**, select the languages for the package group. The corresponding translations for the user interface and documentation for the product package will be installed.

11. On the next **Features** page, select the features to install.



Note:

- The Load Generation Agent is used to generate a load on the system under test and gather data for the Response Time Breakdown feature.
- Installation Manager automatically enforces any dependencies with other features and displays updated download sizes and disk space requirements for the installation.

- a. **Optional:** To see the dependency relationships between features, select **Show Dependencies**.
- b. **Optional:** Click a feature to view its brief description under **Details**.
- c. After you are finished selecting features, click **Next** to continue.

12. On the next **Features** page, configure the agent:

- a. **Windows**: The Web UI test panel is applicable only if you are shell-sharing HCL OneTest™ Performance and HCL OneTest™ UI and executing a Web UI test on the remote agent computers. If you are not running a Web UI test, do not select the **The agent will be used primarily to support remote execution of Web UI tests** check box so that Majordomo runs as a service. When Majordomo runs as a service, it starts automatically after the computer reboots. However, if you must run the Web UI test on remote agent computers, select the check box so that, after the installation of the agent, Majordomo runs as a batch file instead of a service. When Majordomo runs as a batch file, it stops after the machine reboots. You must restart Majordomo by double-clicking `AgentInstallDir/Majordomo/Majordomo.bat`.

On Linux, this check box is not available, and by default, the agent runs as a service and can run a Web UI test.

Click **Next**.

- b. For Load Generation Agent, type the host name of the workbench . If needed, change the port number. You can also change the port number after you install HCL OneTest™ Performance Agent. For more information, see [Configuring ports for agents on page 554](#).
- c. To connect the agent with HCL OneTest™ Server, specify the fully qualified domain name in **Server Hostname**, the port number, and the offline user token. For information about offline user token, see [Managing access to HCL OneTest™ Server](#).



Note: Use the default port number 443 to connect to the server.

- d. Click **Next**.

13. On the **Summary** page, review your choices before installing the product package. To change the choices that you made on previous pages, click **Back**, and make your changes. When you are satisfied with your installation choices, click **Install** to install the package.

Result

A progress indicator shows the percentage of the installation that is complete.

14. When the installation process is complete, a message confirms the completion of the process.
 - a. Click the **View log file** button to open the installation log file for the current session in a new window. You must close the **Installation Log** window to continue.
 - b. Click **Finish** to install the selected package.

What to do next

After the installation, the agent starts polling the workbench based on the IP address or the host name provided by you. You can check the status of agents from the workbench. See [Checking the status of agents on page 553](#).

Uninstalling the product by using Installation Manager

When you no longer require HCL OneTest™ Performance, you can use Installation Manager to uninstall HCL OneTest™ Performance that you have installed.

Before you begin

You must have completed the following tasks:

- Installed Installation Manager.
- Closed any open windows of HCL OneTest™ Performance.
- Closed any open web browsers.
- Closed all the other applications that are enabled by HCL OneTest™ Performance.

1. Open Installation Manager.
2. Click **Uninstall**.
3. Select the HCL OneTest™ Performance package checkbox on the **Uninstall Packages** window, and then click **Next**.
4. Review the list of packages that are ready to uninstall, and then click **Uninstall**.

Result

The **Complete** page is displayed after the uninstallation process is complete.

5. Click **Finish** to exit the Installation Manager wizard.

Results

You have uninstalled HCL OneTest™ Performance from your computer.

Installing the software by using stand-alone installer

This guide intended for an administrator, describes how to install the product by using stand-alone installer on different operating systems.

Installing HCL OneTest™ Performance Agent

To get started with HCL OneTest™ Performance Agent, you must install the product by using the stand-alone installer.

License management

Licensing for your HCL software is administered through [HCL® License & Delivery portal](#). This portal is FlexNet-based web application to manage software entitlements and licenses.

You must have ordered software. When a software order is placed and acknowledged, a software entitlement is created. Then, you have to follow the instructions in Software Order Acknowledgment document that you received to activate your entitlement, create devices, and download the software from the Portal.

If you do not have access to the Internet, you can install and configure a local license server.

License descriptions

The usage of base product and test extensions are enabled by floating licenses, whereas schedule runs can be enabled by either floating or consumption-based licenses. With floating licenses, multiple users can use the product; however, the total number of concurrent users cannot exceed the number of floating licenses you purchase.

When a software order is placed and acknowledged, a software entitlement is created for the user. You can then create devices and map the software entitlement with the devices. Every device is associated to a server ID. This server ID is applied in the product. Multiple software entitlements can be created based on the requirements.

You can use any of these licenses according to your requirements:

Product licenses

- **Floating license to use the product:**

To use the product, you need a floating license that is mapped to the product or you can also use a HCL OneTest™ Studio floating license. A floating license is checked out when you use the product and is returned to HCL® License & Delivery portal when the license is not used for 15 minutes.

- **Floating licenses for test extensions:**

For HCL OneTest™ Performance, in addition to the license for product usage, if you use SOA, SAP, Citrix, and Siebel test extensions, you need separate software entitlements. You can map all the entitlements to one server ID. If you have an HCL OneTest™ Studio license, it is enough to start using the product as well as use the above test extensions. A license is checked out from the time the product is opened by the user. The license remains checked out until the user closes the product. When the user closes the product, after 15 minutes, the license is returned to the HCL® License & Delivery portal.

-

Licenses for A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule. runs

- **HCL® OneTest™ Studio - Virtual Users:**

These are floating licenses that are purchased in blocks of 100 virtual users each. After the run, the licenses are checked back in to the server. For example, if you have 200 HCL® OneTest™ Studio - Virtual Users available, you can run schedule A with 100 virtual users and schedule B with 20 virtual users, leaving 80 virtual users for other team members to use for schedule execution using the same license server. After the schedule run completes, the all the virtual users are returned to the server for others to use.

- **HCL® OneTest™ Studio - Virtual User Execution Capacity:**

These licenses are referred as consumption-based license. They are purchased in blocks, for example, 1000 hours. Every time you run a schedule or compound test, the number of hours are deducted from the block.

Prior to schedule run, the product provides an estimate of the VU-Hours required to run the schedule. At the end of successful schedule run, the product sends a request to the HCL® License & Delivery portal to deduct the number of VU-Hours consumed. For **Run Until Finished** schedule run, the estimate warns that the actual number of VU-Hours to be consumed cannot be determined.

When you run a schedule, the product first searches HCL® OneTest™ Studio - Virtual Users licenses. If they are unavailable or insufficient, the schedule looks for HCL® OneTest™ Studio - Virtual User Execution Capacity licenses. For example, suppose you are entitled to 500 HCL® OneTest™ Studio - Virtual Users and 1000 hours HCL® OneTest™ Studio - Virtual User Execution Capacity license. If a team member starts schedule A that requires 400 virtual users and another team member starts schedule B that requires 200 virtual users, schedule A picks up 400 virtual users and runs as expected. Schedule B searches 200 virtual users but does not find them. So, the duration of schedule B will be calculated and the estimated number of hours is displayed to the user. If the user clicks **OK**, HCL® OneTest™ Studio - Virtual User Execution Capacity licenses are consumed. The calculation of Rate Schedule VU-Hours is based on 1 transaction per second being the equivalent of one virtual tester.

•

You can add more virtual users or change the rate of execution during the test or schedule run only if there are sufficient number of Virtual User licenses or if a sufficient number of VU-Hours are on balance in the customer's account.

Applying the license

To start using a product, you must first apply a license.

About this task

When you start the product for the first time, a licensing dialog box is displayed. Specify the server ID that was provided to you or copy the ID from the HCL® License & Delivery portal. When you submit the server ID, the product connects with the HCL® License & Delivery portal to verify it and if there is a license available, it is checked out so that you can use the product. If the license is not available, a message is displayed about it. In most cases, you should not change the server URL in the licensing dialog.



Note:

- If the license is not used for 15 minutes, the license is returned to the server for others to consume it. If the workbench loses connectivity to the HCL® License & Delivery portal, you can use the product for two hours only if the server ID that you entered last time is correct.
- After the product installation, when you apply the license to the product, the license information is cached in a system directory. If there are any permission issues accessing the directory or the directory is deleted, there would be licensing error when starting the product. The workaround is to



create an environment variable `HCL_ONETEST_LICENSING_STORAGE` and specify a directory path. The licensing information will now be stored in this new directory. For example,

The screenshot shows a dialog box titled "New User Variable" with a close button (X) in the top right corner. It contains two text input fields: "Variable name:" with the value "HCL_ONETEST_LICENSING_STORAGE" and "Variable value:" with the value "C:/Users/<username>/HCLOneTest". Below the fields are four buttons: "Browse Directory...", "Browse File...", "OK", and "Cancel".

You can also apply the license after starting the product. To apply the license from the product, click **Window > Preferences > Test > Licensing** and specify the server ID. You can also use this page to check the balance Event-hours on the server.

To apply license without opening the product, add the licensing parameters as environment variables:

Variable	Value
<code>HCL_ONETEST_LICENSING_URL</code>	https://hclsoftware.compliance.flexnetoperations.com
<code>HCL_ONETEST_LICENSING_ID</code>	Enter your server ID. The ID is a 12-character alphanumeric identifier.

When running the tests from command line, you do not have to specify any licensing argument. However, when running a schedule for HCL OneTest™ Performance or accelerated/distributed tests for HCL OneTest™ UI in an uninterrupted mode, you must specify `vmargs -Dhptcostconfirm` argument in the command. Uninterrupted mode refers to any of the non-GUI based test execution scenarios such as the following use cases:

- Running a test from the command line
- Running a test with Jenkins
- Running a test with Ant
- Running a test with UrbanCode™ Deploy
- Running a test with IBM® Rational® Quality Manager

What to do next

You can now work with the product.

Using a local license server

In some cases, your computer might not be connected to the open Internet. In these cases, you can set up a local license server behind your company firewall. You must install the server on a physical computer and not on a Virtual Machine.

Before you begin

As part of configuration, the local license server must already have mapped your entitlements and be able to serve your requests.

About this task

The local license server is supported on 64-bit Windows™ and Linux® platforms. For more information about installing and configuring the local license server, see the documentation for the local license server on the HCL® License & Delivery portal product download site.

The local license server will be available soon after the 9.2.0.1 release.

1. After the license server is installed and configured, start the testing product.
2. Within the test product, click **Windows > Preferences > Test > Licensing**.
3. When the licensing dialog opens, select the **Server Type** as **Local** and replace the URL of the cloud-based license server with the URL of your own local license server. Enter the Server URL of the local license server in this format - `http://myserver:portNumber/request`. Do not specify the server ID.
4. To test the connection to the server, click **Test Connection**.

What to do next

You can now work with the product.

Configuring product licensing to use a proxy server

In most cases, you connect directly to a cloud-based license server when you start the testing product for the first time. As an alternative, you can configure the testing product to use a proxy server.

1. Within the test product, click **Windows > Preferences > Test > Licensing**.
2. Select the **Server Type** as **Cloud** or **Local**.
3. Enter the Server URL and Server ID of the license server (if applicable)
4. Select the **Use Proxy Server** check box and specify the **Host** and **Port** of the proxy server.
5. If login credentials are required to access the server, select the **Provide Credentials** check box and specify the **User Name** and **Password**.
6. To test the connection to the server, click **Test Connection**.

Collecting usage metrics data

To provide an insight into the usage patterns of the product itself and not necessarily the application that it is testing, the usage metrics are collected. The product can collect the metrics about the usage of the tool including and not limited to the number of tests executed, the number of actions performed against an application under test, the number of successful versus failed actions, verification points, and so on.

Before you begin

- Ensure that HCL® Quality Server is installed. See the HCL® Quality Server documentation for the installation instructions.
- Ensure that the license key is applied. See [Applying the license on page 101](#)

About this task

When you collect usage metrics, the data is collected for internal use to provide you better services. For example, in a Web UI test, all user actions such as click, enter text, and so on are counted and collected. Similarly, for an HTTP test, the number of HTTP Pages and Requests are counted and logged. No information about the application under test is collected.

1. From the product, click **Window > Preferences > Test >** .
2. Specify the IP address or host name of the computer where HCL® Quality Server server is installed.



Note: If the IP address or host name is not specified, depending on the license key setting, a warning message might be logged every time you run a test, the test execution itself might be blocked, or the tests can be executed without any warning messages.

3. Click **Test Connection** to check whether the connection is established. Click **OK** if the connection is established.

Product upgrade and migration

When you want to use the enhanced functionalities of HCL OneTest™ Performance Rational® Service Tester for SOA Quality, you must upgrade to the latest version of the product software.

You can upgrade HCL OneTest™ Performance to the latest version by uninstalling the existing version of HCL OneTest™ Performance. You can then install the latest version of HCL OneTest™ Performance.

Migrating test assets to new version of the product

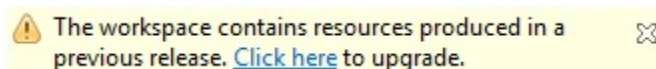
After you install a later version of the product and you choose to open the product from an old workspace, you are prompted to migrate test projects, tests, schedules, rules, and reports. Tests and schedules are migrated automatically when you modify and save them.


You cannot have two versions of the products installed on your computer at one time. Before you install a new version, uninstall the previous version of the product. If you update the product with IBM® Installation Manager, you do not have to uninstall the previous version. Uninstalling a previous version does not delete your test assets.



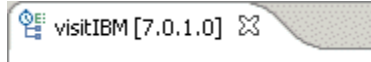
Note: When you want to uninstall the previous version of the product and then install the latest version, you must install the latest version in the same folder that contained the previous version. Thereby, you can avoid compilation errors in the project in the latest version.


When you open a project that contains an older test asset, a message is displayed in the **Test Navigator** view. Typically, you upgrade your tests, schedules, and rules.



 **Important:** Ensure that you back up the test assets before upgrading them for the new version of the product. Do not open a migrated test project with a previous version of the product.

If you leave tests, schedules, and rules unchanged, they will not have the new functions that current release adds. You can always save a modified test asset under a new name, which preserves the older asset. You can identify an older asset by its version, which is listed in brackets:



 **Note:** A new release might include enhancements to the default reports. When you run a test or schedule or open a report, you are prompted to upgrade reports to the latest version. If you upgrade the default reports to the latest version, you lose customizations that you have made to the reports.

If you encounter errors when you open a workspace from a different version of the product, reset the perspective. To reset the perspective, click **Window > Reset Perspective**.

Configuration of the product

You must configure HCL OneTest™ Performance when you want to run SAP tests, collect response time breakdown data, or use the product in the Docker container.

The following topics provide more information about the configuration of HCL OneTest™ Performance:

Configuring the environment for SAP tests


Some of the capabilities of service tests require that you manually install additional libraries and files or that you configure some elements of the test environment.

Configuring SAP for performance testing


Performance test recording and execution requires scripting to be enabled on the SAP application server and on all SAP GUI clients that are installed on remote computers. The following instructions are for SAP version 6.40 and might vary with other versions. Refer to SAP documentation for further information.

Before you begin

Performance testing relies on the SAP Scripting API and ActiveX. Make sure that Active X is installed when installing the SAP GUI client and enable scripting on the SAP server.

 **Note:** You only need to perform the following actions once. Scripting should remain enabled on the SAP server after a restart.

To enable scripting on the SAP server:

1. Check that there is a `Scripting` directory located in the SAP GUI installation directory.
If this directory does not exist, then the SAP Scripting API is not installed and you must reinstall SAP GUI with the SAP Scripting API option.
2. Run the SAP GUI client and logon to SAP with your user name and password.
Administrator privileges might be required to enable scripting on the server.
3. In SAP, run the transaction `rz11`, type the parameter name `sapgui/user_scripting`, and then click **Display**.
If the parameter is not found, then make sure you have the correct support package level from SAP. Contact your SAP representative for guidance.
4. If the **Current value** is `FALSE`, click the **Change value** button, and then set the **New value** to `TRUE` in uppercase characters.
5. Click **Save**, and then end the transaction.
Scripting will be enabled the next time you log on.
6. In the SAP GUI client toolbar, click the **Customizing of Local Layout** toolbar button, and then select **Options**.
7. Select the **Scripting** page.
8. Select **Enable Scripting**, and then disable both **Notify When a Script Attaches to Running GUI** and **Notify When a Script Opens a Connection**.
9. Click **OK**.
10. In the **Help** menu, select **Settings**, and then select the **F4 Help** page.
11. In **Display**, select **Dialog (modal)** and then click the **Enter**  button.

What to do next

Repeat steps 5 through 9 for SAP GUI clients on all remote computers. For more information about enabling scripting on the SAP application server, refer to the following SAP notes, available from SAP:

- 480149: ABAP and patch level requirements
- 587202: Limitations of SAP GUI Scripting
- 527737: Composite SAP Note on SAP GUI Scripting
- 612454: SAP GUI Scripting status and lifetime
- 619459: SAP GUI Scripting support of SAP applications

Configuring the environment for batch input tests

To play back SAP batch input tests, you must install the SAP Java™ Connector (JCo) libraries provided by SAP on local and remote computers running the HCL OneTest™ Performance agent.

Before you begin

The following files are required:

- `sapjco3.jar`
- `sapjco3.dll`

Contact SAP for instructions on obtaining these files.

On each local and remote computer used for running batch input tests and schedules, you must prepare an environment with these libraries, set the class path of the Java™ Runtime Environment (JRE) that the HCL OneTest™ Performance Agent uses.

To configure a computer for SAP batch input tests:

1. Copy `sapjco3.jar` into the `jre/lib/ext` directory of the JRE.

By default, this is the following directory: `C:\Program Files\HCL\HCLOneTest\jdk\jre\lib\ext`.

2. Copy `sapco3.dll` into the `C:\Windows\System32` directory. If you are using a 64 bit machine, you must copy the file to `C:\Windows\SysWOW64`.
3. Click **Start > Control Panel > Administrative Tools > Services**, and then stop and restart the Majordomo service.

What to do next

When running a simple SAP batch input test, results should appear in the **Batch Input Transaction Rate** page of the SAP performance report. If not, look at the test log for connection and transaction execution information.

Configuring the data collection infrastructure

If you want to collect response time breakdown data from remote computers used in distributed applications, you must install and configure the data collection infrastructure on the remote computers.

Data collection infrastructure overview

The data collection infrastructure collects performance profiling data for distributed applications and sends the data to the workbench computer, where you can view and analyze the collected data. The data collection infrastructure must be installed, configured, and running on all remote computers that will be used in your distributed application. The data collection infrastructure is installed when you install HCL OneTest™ Performance Agent.

You can monitor a live application in a development or testing environment and collect data from the application in real time. Data collection is supported on the following web servers:

- IBM® WebSphere® Application Server, Version 6.1, 7.0, 8.0, or 8.5 running on Microsoft™ Windows™, AIX®, and Linux™
- Oracle WebLogic Application Server, Version 9 or 10 running on Microsoft™ Windows™ and Linux™

You can also query IBM® Tivoli® Monitoring for Transaction Performance, IBM® Tivoli® Composite Application Manager for Response Time Tracking, IBM® Tivoli® Composite Application Manager for WebSphere® or IBM® Tivoli® Composite Application Manager for Application Diagnostics management server databases to collect past performance data for an application that is deployed in a production environment. Data collection is supported on the following software versions:

- IBM® Tivoli® Monitoring for Transaction Performance, Version 5.3 fix pack 1 (5.3.0.1)
- IBM® Tivoli® Composite Application Manager for Response Time Tracking, Versions 6.0 and 6.1

- IBM® Tivoli® Composite Application Manager for WebSphere®, Versions 6.0 and 6.1
- IBM® Tivoli® Composite Application Manager for Application Diagnostics, Version 7.1

Instrumenting local servers

You must instrument application servers to collect response time breakdown data. You can use either a command-line or graphical user interface to instrument application servers to work with the data collection infrastructure.

Before you begin

- The data collection infrastructure must be installed on any computer from which you want to collect performance data.
- If you are instrumenting a WebSphere® application server, the application server must be running.
- The Application Server Instrumenter prompts you to restart BEA WebLogic application servers.
- You must have root or administrator user privileges to instrument servers.

1. In Windows™, click **Start > Programs > __BRAND_NAME__ Data Collection Infrastructure > Application Server Instrumenter**.

In Linux™, use the start menu to open the Application Server Instrumenter. In the AIX® operating system, type `/opt/HCL/HCLOneTest/DCI/rapa_prod/instrument_comp/ASI` at a command prompt to start the Application Server Instrumenter.

Result

The Application Server Instrumenter starts.

2. Use the Application Server Instrumenter to instrument the server. See the examples that follow.
3. If you are instrumenting a BEA WebLogic server, after you have instrumented the server you must restart the application server.


The Application Server Instrumenter automatically restarts WebSphere® application servers.




Note: All HCL OneTest™ Performance systems that are involved with the data collection infrastructure must have the data collection software running for the transaction breakdown function to be available within the results of a HCL OneTest™ Performance test schedule.



Note: The application server instrumenter or the `instrumentServer.bat` (or `instrumentServer.sh`) batch file can fail with a generic error message (`Error during install/uninstall`) when instrumenting or uninstrumenting a server. If this error occurs, you can find more information to help troubleshoot the error in the log files in the IBM® Tivoli® common directory. On Windows™, the default location of this directory is `C:\Program Files\IBM\tivoli\common`. On Linux™, the default location of this directory is `/var/ibm/tivoli/common`. If the IBM® Tivoli®

 common directory is not in the default location, search for a path that contains `tivoli/common`, or for any of these log files: `trace-install.log`, `trace-ma.log`, or `trace-tapmagent.log`.

 **Note:** For IBM® WebSphere® Application Server 6.0 or later, if you create a new profile and then use the application server instrumenter to instrument this profile without first starting WebSphere® Application Server, the application server instrumenter reports that the server is instrumented and prompts you to restart the server manually. This message is incorrect; the server is not actually instrumented. To work around this problem, complete these steps:

- a. Close and then restart the application server instrumenter.
- b. Select the entry that you just added from the list of instrumented servers, and click **Remove**.
- c. Restart WebSphere® Application Server.
- d. Restart the application server instrumenter, and use it to instrument the server.

To avoid this problem, after you create a new profile, start the WebSphere® Application Server profile manually. Then, use the application server instrumenter to instrument the server.

Example

To instrument a Linux™ computer that is equipped with the 64-bit version of IBM® WebSphere® Application Server, Version 6.1, where the server is named `server2`, the server is installed in the `/opt/WebSphere/AppServer` directory, with the profile name set to the default value, and security enabled:

1. Click **Instrument Local Server**.
2. From the **Type** list, select **IBM WebSphere Application Server v6.1**.
3. Type `/opt/WebSphere/AppServer` in the **Server home** field.
4. Type `server2` in the **Server name** field.
5. Type `default` in the **Profile name** field.
6. Under **Server JVM**, click **64-bit**.
7. Select the **Requires global security** check box.
8. Type the WebSphere® user ID in the **User** field.
9. Type the WebSphere® password in the **Password** field.
10. Click **OK**.

To instrument a Microsoft™ Windows™ computer that is equipped with a BEA WebLogic 10 application server named `server1` that is installed in the `C:\bea\weblogic10` directory, using the 32-bit version of JRockit JVM, and the `C:\bea\weblogic10\mydomain\startWebLogic.cmd` start script file :

1. Click **Instrument Local Server**.
2. Select **BEA WebLogic Application Server v10.x** from the **Type** list.
3. Type `server1` in the **Server name** field.
4. Type `C:\bea\weblogic10` in the **Server home** field.
5. Type `C:\bea\weblogic10\mydomain\startWebLogic.cmd` in the **Start script** field.
6. Under **Server JVM**, select **Oracle JRockit 32-bit**.

7. Click **OK**.
8. Stop and restart the server.

What to do next

Repeat the instrumentation steps for every application server that is involved in data collection for the applications that you will profile. Typically, there will be only one application server. You can instrument only one local application server per computer.

Tips for instrumenting Oracle WebLogic Application Server

Tips when instrumenting Oracle WebLogic Application Server to collect response time breakdown data.

- You must instrument using an account with root or administrator privileges.

You must provide correct information in the Application Server Instrumenter.

Server name is the name of the server to instrument. For example, for Oracle WebLogic Application Server 10.3 running on Microsoft™ Windows™ with the default sample application MedRec installed, the server name is `MedRecServer`.

Server home is the path to the server bin directory. For example, for Oracle WebLogic Application Server 10.3 running on Microsoft™ Windows™ with the default sample application MedRec installed, the server home is: `C:\Oracle\Middleware\wlserver_10.3\samples\domains\medrec`.

Start script is the script, a .cmd or .sh file, used to start the Oracle WebLogic Application Server. For example, for Oracle WebLogic Application Server 10.3 running on Microsoft™ Windows™ with the default sample application MedRec installed, the start script is `C:\Oracle\Middleware\wlserver_10.3\samples\domains\medrec\bin\startWebLogic.cmd`.

Server JVM is the type of Java™ Virtual Machine (JVM) used by Oracle WebLogic Application Server. To determine the server JVM, examine the system processes while the server is running and see whether the JRockit JVM or the Oracle JVM is active.

Tips for instrumenting WebSphere® Application Server

Follow these tips when you instrument WebSphere® Application Server to collect response time breakdown data.

- The instance of WebSphere® Application Server to instrument must be running.
- You must instrument with an account with root or administrator privileges.
- If security is enabled for WebSphere® Application Server, you must know the WebSphere® Application Server administrator ID and password before you can instrument the server.
- Vertical clusters are not supported. You can instrument a horizontal cluster by instrumenting each physical server separately.

You must provide correct information in the Application Server Instrumenter.

Server home is the complete path to the WebSphere® Application Server installation directory. By default, on Microsoft™ Windows™ the server home is `C:\Program Files\IBM\WebSphere\AppServer`. In Linux™ and AIX®, the default server home is `/opt/IBM/WebSphere/AppServer`.

Server name is the name of the WebSphere® Application Server instance to instrument. To see a list of server names, change to the `bin` directory, and type this command at a command prompt: `wsadmin -conntype none -c "puts stdout [$AdminConfig list Server]"`.

An example of output from this command is as follows:

```
WASX7357I: By request, this scripting client is not connected to any server process. Certain configuration
and application operations will be available in local mode. server1(cells/MachinenameNode01Cell/nodes/
MachinenameNode01/servers/server1|server.xml#Server_1183122130078)
```

In this case, **server1** is the server name.

Profile name is the name of the profile that is associated with the server to instrument.

To see a list of profiles from WebSphere® Application Server, change to the `bin` directory, and type this command at a command prompt: `manageprofiles -listProfiles`. By default, in Microsoft™ Windows™ the `bin` directory is `C:\Program Files\IBM\WebSphere\AppServer\bin`. In Linux™ and AIX®, the default `bin` directory is `/opt/IBM/WebSphere/AppServer/bin`.

Instrumenting servers by using the command prompt

You must instrument application servers to collect response time breakdown data. You can use either a command prompt or graphical user interface to instrument application servers to work with the data collection infrastructure.

Before you begin

The data collection infrastructure must be installed on each computer from which you want to collect performance data. The application server must be running. You must have administrator or root user privileges to instrument servers.

1. Open a command prompt, and go to the `instrument_comp` folder in the directory where the data collection infrastructure is installed.
By default, in Microsoft™ Windows™, this directory is `C:\Program Files\HCL\HCLOneTest\DCI\rapa_prod\instrument_comp`. In Linux™ and IBM® AIX®, this directory is `/opt/HCL/HCLOneTest/DCI/rapa_prod/instrument_comp`.
2. Type the command name with the arguments to use to instrument a server. See the examples that follow.
The instrumentation utility is `instrumentServer.sh` on AIX® and Linux™ systems and `instrumentServer.bat` on Windows™ systems. Enter the command name with no arguments to see the syntax details for the command.
3. After you have instrumented the application server, you might have to restart the application server.
Instrumentation changes take effect after the application server is restarted.

Example

Assume that you must instrument a Windows™ computer that is configured as follows:

- IBM® WebSphere® Application Server, Version 7.0.
- The server is named **my_Server2**.
- The application server is installed in the C:\Program Files\was7.0 directory.
- The profile name is **default**.
- Security is enabled.
- The server Java™ Virtual Machine (JVM) is a 64-bit JVM.

Type the following command and arguments:

```
instrumentServer -install -type was7 -serverName my_Server2 -serverHome "C:\Program Files\was7.0" -serverVMArch
64 -user my_WAS_userId -password my_WAS_password -profileName default
```

To instrument a Linux™ computer that is equipped with a BEA WebLogic 10 application server, with configuration details as indicated, using the 32-bit version of JRockit JVM, type the following command and arguments:

```
./instrumentServer.sh -install -type wls10 -serverName server1 -serverHome /opt/BEA/weblogic10 -serverVM oracle
-serverVMArch 32 -startScript /opt/BEA/weblogic10/mydomain/startWebLogic.sh
```

To instrument a Liberty Profile, for `-type` use `liberty85`, and for `-serverHome` use the path to the Liberty server installation directory until `wlp`.

What to do next

Repeat the instrumentation steps for each application server that is involved in data collection for the applications that you will profile. Typically, there will be only one application server. You can instrument only one local application server per computer.

Removing instrumentation

Before you uninstall the data collection infrastructure, you must remove instrumentation from all application servers that were instrumented to work with the data collection infrastructure.

To remove the instrumentation from an application server:

1. Click **Start > Programs > __BRAND_NAME__ Data Collection Infrastructure > Application Server Instrumenter**.
On the IBM® AIX® operating system, type `/DCI/rapa_prod/instrument_comp/ASI` at a command prompt to start the Application Server Instrumenter.
2. Select the server to remove the instrumentation from.
3. Click **Uninstrument Server**.
4. Restart the server if prompted to do so.

What to do next

You can uninstall the data collection infrastructure.

Removing instrumentation using the command prompt

Before you uninstall the data collection infrastructure, you must remove instrumentation from all application servers that were instrumented to work with the data collection infrastructure.

Before you begin

If you have uninstalled the application server or removed an instance of the application server, you cannot use the instrumentation utility to remove instrumentation from the server. This situation will block the data collection infrastructure uninstall process. Do not uninstall an application server before you have removed instrumentation from the application server.

Instrumented servers are listed in the file `InstrumentationRegistry.xml`. After you have removed instrumentation from a server, the `InstrumentationRegistry.xml` file will be empty of references to any application server. If you uninstall an application server before you have removed instrumentation, to uninstall the data collection infrastructure you must edit `InstrumentationRegistry.xml` to remove the `applicationServer` element for the application server that you uninstalled.

To remove instrumentation from application servers:

1. Open a command prompt, and change directories to the `instrument_comp` folder in the data collection infrastructure installation directory. By default, on Microsoft™ Windows™ this directory is `C:\Program Files\HCL\HCLOneTest\DCI\rapa_prod\instrument_comp`. On Linux™ and IBM® AIX®, this directory is `/opt/HCL/HCLOneTest/DCI/rapa_prod/instrument_comp`.
2. Type the command name with the `-uninstall` argument and all of the other arguments that you used when instrumenting the server.
The instrumentation utility, which is also used to remove instrumentation from servers, is `instrumentServer.sh` on AIX® and Linux™ systems and `instrumentServer.bat` on Windows™ systems. Enter the command name with no arguments to see the syntax details for the command.
3. Restart the server if prompted to do so.

What to do next

You can uninstall the data collection infrastructure.


Configuring Docker containers

Instead of reading through system requirements list and installing the products, you can now deploy the Docker containers on any computer and get started with testing. To automate playing back tests, you can push the product images to the Docker container.

1. Download and install Docker-CE. For more information, see the following links:
 - Windows - <https://store.docker.com/editions/community/docker-ce-desktop-windows>.
 - Ubuntu - <https://docs.docker.com/install/linux/docker-ce/ubuntu/>.
 - Others - <https://store.docker.com/search?type=edition&offering=community>.
2. To verify whether your Docker installation was successful, open PowerShell or a terminal of your choice and run:

```
$ docker run hello-world
```

- Download the container image for the agents from the same location you downloaded the product bits and extract the compressed files.

 **Important:** The version of the workbench and agents must match.

- Load the agent image into the Docker repository:

```
$ docker load -i "imageFileName.tar.gz"
```

Result

When the image is loaded, the following message is displayed - Loaded image: *imageFileName:versionNumber*

What to do next

You can now set up the playback environment on Docker by following instructions in [Running tests with containerized agents on page 611](#) and [Running automated tests with containerized workbench and agents from Docker on page 613](#).

Integration with other products

You can integrate HCL OneTest™ Performance Rational® Service Tester for SOA Quality with certain products to run tests, manage test assets, and create defects.

The following topics provides more information about the integration of HCL OneTest™ Performance Rational® Service Tester for SOA Quality with other products:

Integration plugin compatibility matrix

You can find information about the versions of the integration plugin that are compatible with HCL OneTest™ Performance.

The following table lists the versions of the integration plugin that are required to integrate Jenkins, Ant, and UrbanCode™ Deploy with HCL OneTest™ Performance.



Note: You must download the required version of the integration plugin from the [HCL® License & Delivery portal](#) based on the existing version of HCL OneTest™ Performance. You can then integrate Jenkins, Ant, and UrbanCode™ Deploy with HCL OneTest™ Performance.

HCL OneTest™ Performance	Ant plugin	Jenkins plugin	UrbanCode™ Deploy plugin	HCL™ Launch plugin
10.1.0	HOT-PERF-Ant-5.0	HOT-PERF-Jenkins-6.0	HOT_PERF_1010_-UCD-5.0	N/A

HCL OneTest™ Performance	Ant plugin	Jenkins plugin	UrbanCode™ Deploy plugin	HCL™ Launch plugin
10.1.1	HOT-PERF-Ant-6.0	HOT-PERF-Jenkins-7.0	HOT-PERF-UCD-6.0	HOT-PERF-LAUNCH-6.0

Testing with Ant

You can use `ant` to run performance and compound tests from the command line. Starting from V2.0.0 of the `ant` plugin, you can run multiple tests simultaneously. V5.0 of the `ant` plugin is supported in HCL OneTest™ Performance V10.0.2 of the testing products.

Before you begin

You must have completed the following tasks:

- Installed Installation Manager.
- Installed HCL OneTest™ Performance.
- Verified that you have test assets residing within HCL OneTest™ Performance.
- Downloaded the HCL OneTest™ Performance Ant plugin from the [HCL® License & Delivery portal](#) on to the computer where you install the product.

For more information about specific versions of plugin, see [Integration plugin compatibility matrix on page 114](#).

- Added `ant` to the `PATH` environment variable.

About this task

To run performance tests on Mac OS, you must add an environment variable that points to the installation directory of HCL OneTest™ Performance.

For example, `export TEST_WORKBENCH_HOME=/opt/HCL/HCLOneTest`.



Note: For Windows™ and Linux®, the environment variable is set when you install the product.

1. Extract the following files from the downloaded plugin:

- `HOT-PERF-Ant-x.0.jar`

Where, `x` is the version number of the `ant` plugin.

- `ExecutePerformanceTest.xml`
- `README.txt`

2. Open the `ExecutePerformanceTest.xml` file and provide required parameter values.


For example,




```
<pt name="test1" workspace="C:\workspace" projectname="TestProject" suite="Tests/test1.testsuite"
results="Results/test1_on_anttask" />
```



Note: You can add an additional `<pt>` task and provide the details for each test to run multiple tests simultaneously.

The following table explains each field.

Field	Description
- name	Required. The name of the test for the particular test product.
- work-space	Required. The complete path to the Eclipse workspace.
- project-name	Required. The path, including the file name of the project relative to the workspace.
- suite	Required. The path, including the file name of the test to run relative to the project. A test can be a performance schedule or a compound test.
- varfile	Optional. The complete path to the XML file that contains the variable name and value pairs.
- configfile	Optional. The complete path to a file that contains the parameters for a test or schedule run.
- results	Optional. The name of the results file. The default result file is the test or schedule name with a time stamp appended.
- overwrite	Optional. Determines whether a results file with the same name is overwritten. The default value is true, which means the results file can be overwritten.
- quiet	Optional. Turns off any message output from the launcher and returns to the command shell when the run or the attempt is complete.
- vmargs	Optional. Java™ virtual machine arguments to pass in.
- swap-datasets	Optional. For a test or schedule, the default value is the dataset specified in the test editor or schedule editor. Overrides the default dataset value to run if required. <p> Note: You must use the swapdatasets option to replace dataset values during a test or schedule run. You must ensure that both original and new datasets are in the same workspace and have the same column names. You must also include the path to the dataset. For example: <code>/project_name/ds_path/ds_filename.csv:/project_name/ds_path/new_ds_filename.csv</code>. You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semi-colon. For example: <code>/project_name1/ds_path/ds_filename.csv:/project_</code></p>

Field	Description
	 <code>name1/ds_path/new_ds_filename.csv;/project_name2/ds_path/ds_filename.csv:/project_name2/ds_path/new_ds_filename.csv</code>
-override-ermlabels	<p>Optional. For a schedule (Rate schedule or VU schedule), use -overrideermlabels to perform any of the following actions:</p> <ul style="list-style-type: none"> ◦ To enable the Resource Monitoring from Service option for a performance schedule if the Resource Monitoring from Service option is not enabled from the schedule editor in HCL OneTest™ Performance. ◦ To ignore Resource Monitoring sources that were set in the performance schedule and to change for a label matching mode. ◦ To replace an existing set of Resource Monitoring labels that were set in the performance schedule and run the schedule with a new set of Resource Monitoring labels. <p> Note: You can add multiple Resource Monitoring labels separated by a <code>comma</code>.</p> <p> Important: You must add the Resource Monitoring labels to the Resource Monitoring sources on the Resource Monitoring page in your HCL OneTest™ Server project.</p>
-export-stats	Optional. The complete path to a directory that can be used to store exported statistical report data.
-export-statreportlist	Optional. A comma-separated list of absolute paths to custom report format files (.view files) to use when exporting statistical report data with the <code>-exportstats</code> option.
-export-statshhtml	Optional. The complete path to a directory that can be used to export web analytic results. The results are exported in the specified directory. Analyze the results on a web browser without using the test workbench.
-user-comments	Optional. Add text within the double quotation mark to display it in the User Comments row of the report.
-users	Optional. For a schedule, the default value is the number of users specified in the schedule editor. For a test, the default value is one user. Overrides the default number of users, if required.
-imshared-loc	Optional. The complete path to HCLIMShared location, if it is not at the default location.

3. Open a command prompt and navigate to the directory where you downloaded the `Ant` plugin.
4. Enter `ant -f ExecutePerformanceTest.xml` to run the test.

Results

You have run the test by using the `Ant` plugin.

What to do next

You can view that the `ant` execution output is logged into the `logfile.txt` file, and a test log is created in a temp directory called `HOT-PERF-Ant-x.0`.

If you have configured the URL of HCL OneTest Server in the preferences (**Window > Preferences > Test > HCL OneTest Server**) of HCL OneTest™ Performance and set **Publish result after execution** as **Always**, then the **Reports information** section on the **Log** file displays the names of the report along with its corresponding URLs.

Integration with Azure DevOps

When you use Azure DevOps for continuous integration and continuous deployment of your application, you can create tests for your application in HCL OneTest™ Performance and run those tests in Azure DevOps pipelines. You can integrate Azure DevOps with HCL OneTest™ Performance by using the *HCL OneTest Studio* extension that is available in the **Visual Studio Marketplace** portal.

Prerequisites

Before you integrate Azure DevOps with HCL OneTest™ Performance, you must have completed certain tasks. See [Prerequisites for Azure DevOps Integration](#).

Overview

You can use the *HCL OneTest Studio* extension that enables you to select any type of test created in HCL OneTest™ Performance that you can add to your task for the job in the Azure DevOps pipelines.

Running HCL OneTest™ Performance tests in an Azure DevOps Pipeline

After you create the tests in HCL OneTest™ Performance for the application that you are testing, and after you install the *HCL OneTest Studio* extension in your organization, you can run the tests in Azure DevOps pipelines.


Before you begin

You must have completed the following tasks:

- Installed the *HCL OneTest Studio* extension in your organization. See [Installing the HCL OneTest Studio extension](#).
- Installed an agent in your pipeline. See [Azure Pipelines agents](#).

About this task


After you add the *HCL OneTest Studio* extension in your Azure DevOps organization, you can use an existing pipeline or create a new one to add HCL OneTest™ Performance test tasks. You can install an agent or use the one that you installed in your default agent pool. You can add the HCL OneTest™ Performance tests to your task for the agent job, configure the task, and then run the task in the Azure DevOps pipeline.

1. Open your **Organization** page in Azure DevOps and perform the following steps:
 - a. Click the project you want to use.
 - b. Initialize the repository by performing the following steps:
 - i. Click **Repos** from the left pane.
 - ii. Click **Initialize** from the **Initialize with a README or gitignore** section.
-  **Note:** Select the **Add a README** check box if it is not selected.
- c. Click **Pipelines** from the left pane.
 - d. Click **Create Pipeline**.
 - e. Click **Use the classic editor** to create a pipeline without YAML.
 - f. Verify the project, repository, and branch for manual and scheduled builds, and then click **Continue**.
 - g. Click **Empty job**.

2. Select **Pipeline** and complete the following steps:
 - a. Change the name for the build pipeline if required.
 - b. Select the **Agent pool** for your build pipeline.

You can use the agent from the default agent pool or use the one you have installed.

- c. Select the **Agent Specification** for the agent if required.
3. Add a task to the agent job by completing the following steps:

- a. Click the **Add Task** icon  for the agent job.

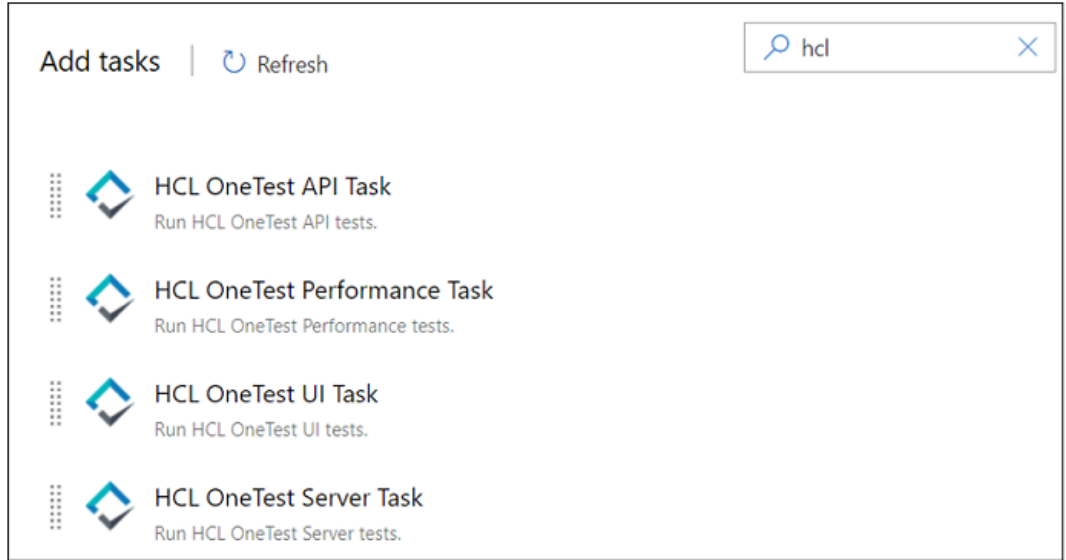
Result

The **Add tasks** pane is displayed.

- b. Search for the HCL tasks defined in the *HCL OneTest Studio* extension.

Result

The tasks that you can select are displayed.



Depending on the type of test that you have created in HCL OneTest™ Performance, you can select the type of task. You must use the following table to identify the task you must select:

Type of test	Task to select
<ul style="list-style-type: none"> ▪ Compound tests ▪ Performance tests ▪ Schedules 	HCL OneTest Performance Task

c. Select the **HCL OneTest Performance** option, and then click **Add** to add the task to the agent job.

Result

The selected task is added to the agent job and it is displayed with a warning that some settings require attention. You must configure the settings mentioned in [step 4 on page 120](#).



You can also remove the tasks that are not required in your job. Select the tasks in the list that you want to remove. You can then right-click the tasks, and click **Remove selected task(s)** to remove them.




4. Configure the settings by performing the following steps:

- a. Select the task version from the list if required.
- b. Follow the action for the performance task by referring to the following table:



Note: All mandatory fields are marked with an asterisk (*) in the UI.

Field	Action
Display name	Displays the name of the selected task. Enter the name of the task.
Product Path	The path where the desktop client is installed on your computer. Enter the complete path of the installation directory of the desktop client. For example, C:\Program Files\HCL\
IMShared Path	The path to the IMShared folder on your local computer. Enter the complete path to the location of the HCLIMShared folder. For example, C:\Program Files\HCL\HCLIMShared
Work-space Location	The complete path to the Eclipse workspace. Enter the complete path of the Eclipse workspace.
Project Name	The name of the project containing the test. Enter the name of the project containing the test.
VM Arguments	<p>Java™ virtual machine arguments to pass in.</p> <p>Enter the Java™ virtual machine arguments.</p> <p> Note: You can add multiple virtual machine arguments files separated by a comma.</p>
Test Suite Name	The name of a test within the project to use. A test can be a performance test, schedule, or compound test. Enter the name of the test that you want to run.
Var File	The complete path to the XML file that contains the variable name and value pairs. Enter the complete path to the location of the variable file.
Dataset Override	<p>For a test or schedule, the default value is the dataset specified in the test editor or schedule editor. Overrides the default dataset value to run if required. Use the <code>Dataset Override</code> option to replace dataset values during a test or schedule run. If a test or schedule is associated with a dataset, you can replace the dataset at run time while initiating the run from the command line.</p> <p> Note:</p> <p>You must use the <code>Dataset Override</code> option to replace the dataset values during a test or schedule run. You must ensure that both original and new datasets are in</p>

Field	Action
	<p> the same workspace and have the same column names. You must also include the path to the dataset.</p> <p>For example,</p> <pre data-bbox="548 426 1377 489">/project_name/ds_path/ds_filename.csv:/project_name/ds_path/new_ds_filename.csv.</pre> <p>You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon (;).</p> <p>For example,</p> <pre data-bbox="548 688 1377 793">/project_name1/ds_path/ds_filename.csv:/project_name1/ds_path/new_ds_filename.csv;/project_name2/ds_path/ds_filename.csv:/project_name2/ds_path/new_ds_filename.csv</pre>
<p>Resource Monitoring Labels Override</p>	<p>For a schedule (Rate schedule or VU schedule), use Resource Monitoring Labels Override to perform any of the following actions:</p> <ul style="list-style-type: none"> ▪ To enable the Resource Monitoring from Service option for a performance schedule if the Resource Monitoring from Service option is not enabled from the schedule editor in HCL OneTest™ Performance. ▪ To ignore Resource Monitoring sources that were set in the performance schedule and to change for a label matching mode. ▪ To replace an existing set of Resource Monitoring labels that were set in the performance schedule and run the schedule with a new set of Resource Monitoring labels. <p> Note: You can add multiple Resource Monitoring labels separated by a comma.</p> <p> Important: You must add the Resource Monitoring labels to the Resource Monitoring sources on the Resource Monitoring page in your HCL OneTest™ Server project.</p>

c. Expand **Control Options** and configure the settings for your task if required.

d. Expand **Output Variables** and configure the settings for your task if required.

5. Select the following options:

- a. Click **Save** to save the configured settings for the task.



Note: The task is not queued for a run.

You can save the task to a build pipeline and opt to run the build at a later time.

- b. Click **Save & queue** to save the configurations and queue the run in the pipeline.

Result

The **Run pipeline** dialog box is displayed.

6. Complete the following steps:

- a. Enter a comment for the test in the **Save comment** field.
- b. Select the agent that you configured for the test from the **Agent pool** list.
- c. Select the agent specification from the **Agent Specification** list for the agent if required.
- d. Select the branch from the **Branch/tag** list.
- e. Add the variables and demands for the task run from the **Advanced Options** pane if required.
- f. Select the **Enable system diagnostics** check box for a detailed log view.
- g. Click **Save and run**.

Result

The `pipeline summary` page displays the progress of the job run.

Results

You have run the tests for the application by using the *HCL OneTest Studio* extension in the Azure DevOps pipeline.

What to do next

You can open the job to view the task logs from the `pipeline summary` page.

You must click the task to open the **Task** page to view the test results.

In HCL OneTest™ Performance, if the HCL OneTest™ Server URL is configured in **Window > Preferences > Test > HCL OneTest™ Server** and **Publish result after execution** is set as **Always** in **Window > Preferences > Test > HCL OneTest™ Server****Results**, then the **Reports information** section on the **Task** page displays the names of the report along with its corresponding URLs. The report URLs are the HCL OneTest™ Server URLs where the reports are stored. You can access the report URLs to view the test execution information at any point of time.

Related information

[Running a test or schedule from a command line on page 627](#)

Integrating with Apache JMeter

Starting from 9.5, you can use JMeter tests extension with HCL OneTest™ Performance to execute JMeter tests.

In HCL OneTest™ Performance, you have the option to import JMeter tests, add tests to a schedule or compound test to run them. Additionally, JMeter test helps to simulate a heavy load on a server, group of servers, or to investigate overall sample response time under different load types.

JMeter samples are terminal elements in JMeter tests that inform JMeter to send requests to a server and wait for a response. When you execute a JMeter test, a JMeter performance report is generated during a run and saved after a run. This report contains the data most significant to the run, shows the response trend of the lowest 25 samples in the test, and graphs the response trend of each sample for a specified interval.

With JMeter test, you can load and test the performance of an application that uses the following protocols:

- HTTP/HTTPS
- SOAP/REST
- FTP
- LDAP
- MOM
- SMTP/POP3/IMAP
- TCP

Installing the JMeter

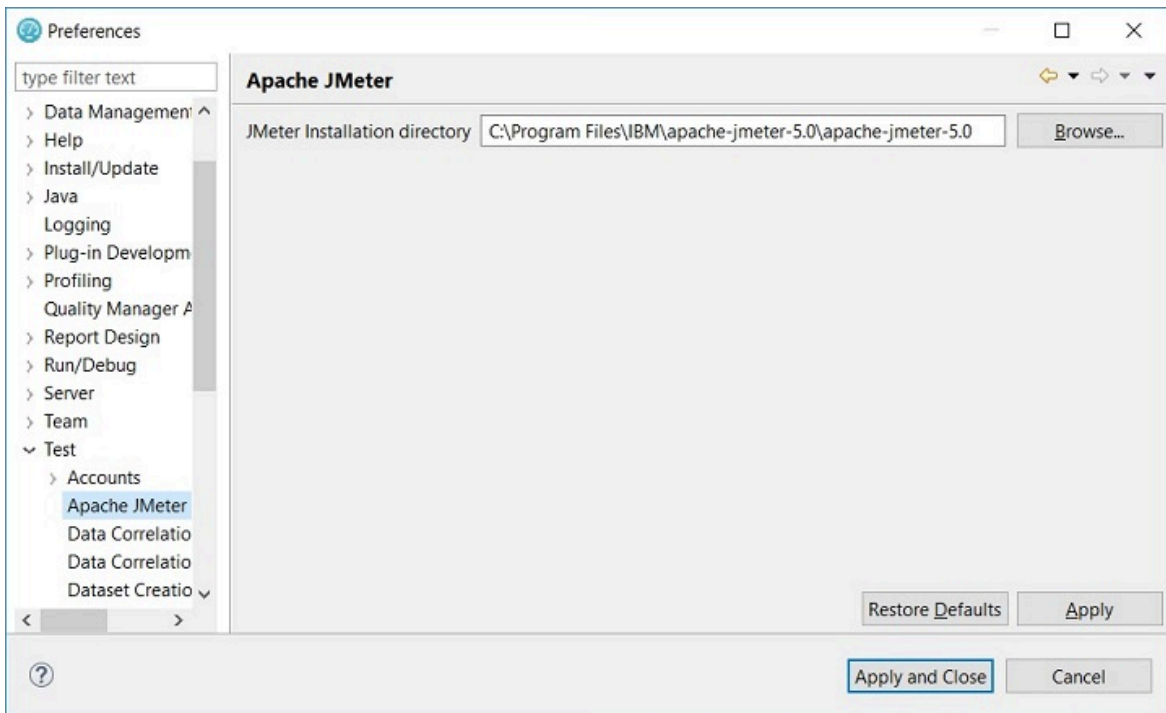
To work with JMeter tests, you must download the JMeter executable from https://jmeter.apache.org/download_jmeter.cgi and unzip it. To run the JMeter test as part of schedule, you must install JMeter on the remote agent machine and set the JMETER_HOME environment variable to the root installation folder.

To run the JMeter tests, you must either specify the Apache JMeter path in **Preferences** or set the environment variable.

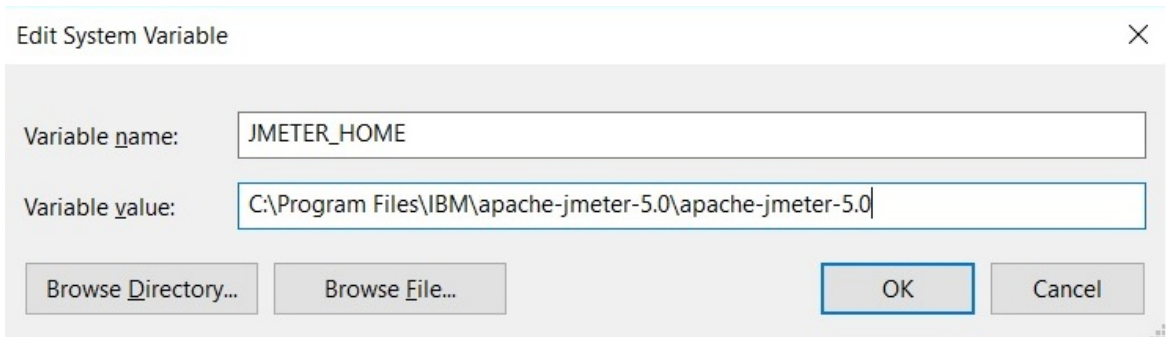


Note: When you run the compound test or schedule, an error message is displayed, if you have not specified the preferences or set the environment variable.

- To access the preference settings for Apache JMeter, click **Window > Preferences > Test > Apache JMeter** and point it to the apache-jmeter-5.0 directory.



- Set the environment variable JMETER_HOME and point it to the apache-jmeter-5.0 directory.



Importing a JMeter test to a Test Workbench project

You can import your JMeter tests to HCL OneTest™ Performance to run them as part of VU schedule.

About this task

If you have an existing JMeter test, you can import the test by dragging and dropping a JMeter file (JMX file) into the project in the test navigator. Alternatively, you can use the following procedure to import the tests.

1. In the Test Navigator, right-click and click **Import**.
2. In the Import dialog box, expand **General** in the source list, select **File System** and then click **Next**.
3. Specify the directory where the JMeter test resides.
Click **Browse** to select a directory from where you can import the JMeter test. The JMeter test assets in the folder you selected are displayed.
4. Select the JMeter test you want to import.
5. Click **Browse** to choose the location to import JMeter test.
6. Click **Finish**. The imported JMeter test is displayed in the **JMeter Tests** folder.

Adding a JMeter test to an existing VU schedule

You can add a JMeter test to an existing VU schedule to test the performance of both static and dynamic resources and web applications.

About this task

When you add a JMeter test to a VU schedule, a user group with a loop is created and this loop contains the JMeter test invocation.

1. In the Test Navigator, browse to the schedule and double-click it. The schedule editor opens.
2. In the VU schedule editor, right-click the Schedule, and then click **Add > JMeter Test**.
3. In the Select Tests window, expand the project name and **JMeter Tests** folder and choose the test that you want to add.
4. Click **OK**. A new User Group with a loop that contains the JMeter test invocation is created.

What to do next

You must run the schedule or compound test, to view the statistics on the executed sequences.

- To install the JMeter, see [Installing the JMeter on page 124](#).
- To run the schedule or compound test, see [Running a local schedule or test on page 608](#) or [Running compound tests on page 486](#).

Converting JMeter tests to VU schedule

You can convert a JMeter test to a VU schedule to load and test the performance of an application under test.

About this task

HCL OneTest™ Performance analyzes the selected JMeter test to add the number of users and loop iteration count in the VU schedule. When you convert the JMeter test to a VU schedule, the following events occur:

- The load information identified within the **Thread Group** nodes from the original JMeter test is examined to build a new VU schedule.
- The content of each **Thread Group** node is extracted from the original JMeter test and copied into a new JMeter test.
- The new JMeter test is then invoked by the VU schedule as an external test.



Note: The extracted JMeter test does not contain any load information such as the number of users and loop count, because the VU schedule manages all the information.



Remember:

- The content of the original **Thread Group** is not considered during the conversion process. Therefore, if there are any loops in the JMeter **Thread Group**, those cannot convert into a loop element in the new VU schedule.
 - If you have a JMeter test with more than one **Thread Group** node, each **Thread Group** is extracted to separate the JMeter test.
 - If you have a complex JMeter test, you must extract the functionalities that are included in the **Modules** or **Include controllers** into another JMeter tests. You must then add those JMeter tests to a VU schedule to run it.
1. Browse and select the JMeter test from the **Test Navigator**.
 2. Right-click the selected test, and then click **Convert to VU Schedule**.
 3. Verify that the name of the schedule is same as name of the JMeter test.
 4. Click **Finish**.

Result

The schedule editor opens.

What to do next

You must perform the following tasks:

- Install JMeter on the remote agent machine and set the JMeter_HOME environment variable to the root installation folder to run the JMeter test as part of schedule. See [Installing the JMeter on page 124](#).
- Run the converted test assets against successive builds of the application under test. See [Running a local schedule or test on page 608](#).
- Analyze the test results that are recorded. See [Running compound tests on page 486](#).

JMeter Performance report

The JMeter performance report summarizes the validity of the run, shows the average sample response time for the requests in the test, and graphs the sample response time of each sample for a specified interval.

Overall page

The Overall page provides a progress indicator that shows the status of the run and a bar chart that shows percentage of passed JMeter samples.

Summary page

The Summary page provides important information about the run. This page shows the following Run Summary information:

- The name of the test.
- The number of active users and the number of users who completed testing. This number is updated during the run.
- The elapsed time is the run duration, which is displayed in hours, minutes, and seconds.
- The status of the run. For example, the status can be Initializing Computers, Adding Users, Running, Transferring data to test log, Stopped, or Complete.

JMeter Samples page

JMeter samples are terminal elements in JMeter tests that informs JMeter to send requests to a server and wait for a response. The JMeter Samples page shows the average sample response time for all the requests in the test. The bar chart shows the average sample response time for all the requests. Each bar represents a sampler of the JMeter test. The corresponding table provides the following additional information:

- The minimum, average, and maximum duration for each sample in the run.
- The standard deviation of the duration.
- The completed sample rate and total number of completed samples per request.

JMeter Transaction page

The JMeter Transaction page shows the average transaction response time for all the requests in the test. The bar chart shows the average transaction response time for all the requests. Each bar represents a page that you visited during recording. The corresponding table provides the following additional information:

- The minimum, average, and maximum duration for each transaction in the run.
- The standard deviation of the duration.
- The completed transaction rate and total number of completed transaction per request.

Samples versus Time Summary page

The Samples versus Time Summary page shows the sample response trend as graphed for a specific interval. The Sample Response versus Time graph shows the sample response time for all the requests during the run. Each point on the graph is an average of what has occurred during that sample interval. The table after the graph lists the total average duration for all requests in the run and the standard deviation. To set the sample interval value, open the schedule, choose the **Statistics** tab from the drop-down menu, and then view or modify **Statistics sample interval**.

Samples versus Time Detail page

The Samples versus Time Detail page shows sample response trend for each of the request in the test. The line graph shows the average sample response time of each requests for a specific interval. The table after the graph provides the minimum, average, and maximum duration for the run and the standard deviation in the average sample response time.

Sample Throughput page

The Sample Throughput page summarizes the frequency of requests that are transferred per sample interval. The line graph on the left side shows the sample rate and passed sample rate per interval for all samples. The summary table after the graph lists the passed rate of total samples and counts for each passed samples. The line graph on the right side shows active users and the users who completed testing, per interval, over the course of a run. You can set the Statistics sample interval value in the schedule, as a schedule property. As the run nears completion, the number of active users decreases and the number of completed users increases. The summary table after the graph lists the active and completed users for the entire run.

To set the sample interval value, open the schedule, choose the **Statistics** tab from the drop-down menu, and then view or modify **Statistics sample interval**.

Server Throughput page

The Server Throughput page lists the rate and number of bytes that are transferred per interval and for the entire run. The page also lists the status of the virtual users for each interval and for the entire run. The line graph on the left side shows the rate of bytes sent and received per interval for all intervals in the run. The summary table after the graph lists the total number of bytes sent and received and bytes sent and received throughput rate for the entire run.

The line graph on the right side shows active users and users who are completed testing, per interval, over the course of a run. You set the Statistics sample interval value in the schedule, as a schedule property. As the run nears completion, the number of active users decreases and the number of completed users increases. The summary table after the graph lists the active and completed users for the entire run.

Server Health Summary page

The Server Health Summary page gives an overall indication of how well the server is responding to the load. The bar chart shows the total number of samples and total number of passed samples for the run. The table under the bar chart lists the same information.

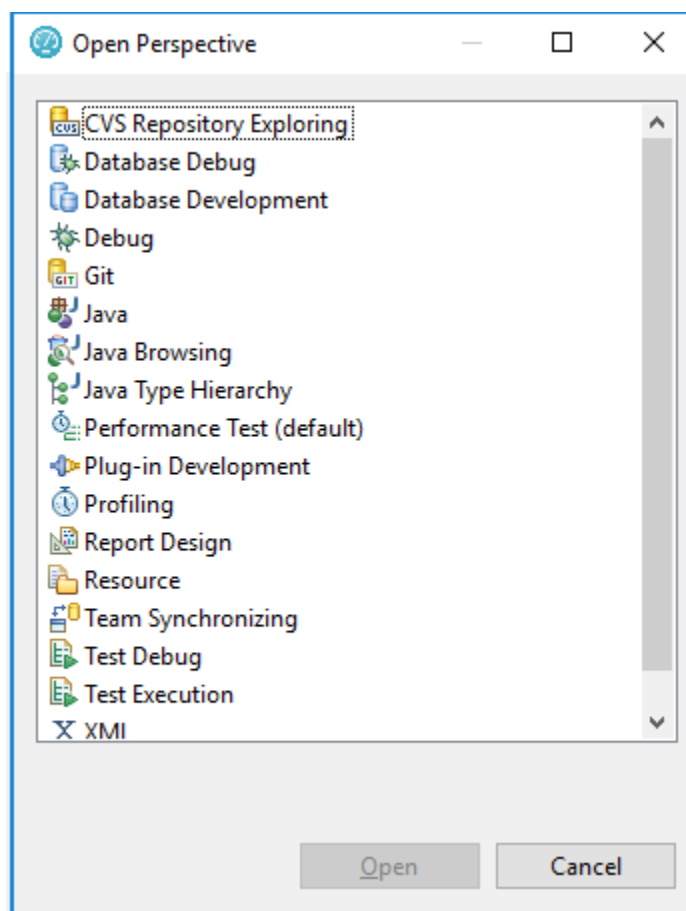
Server Health Detail page

The Server Health Detail page provides specific details for 25 samples with the lowest success rates. The summary table lists the number of samples completed and passed in the run, and the passed sample percent and completion rate.

EGit integration

EGit is an Eclipse plugin for the Git version control system. You can store your test assets in the Git repository and use EGit for the daily version control operations.

When you install the product, by default EGit is part of the product. To avoid the clutter, EGit is a separate Eclipse perspective in the product. For more information about EGit, read its [documentation](#).



In Eclipse perspective, when you initialize a new Git repository for a project, a `.gitignore` file is created in the project folder, by default. While committing the contents of a project to Git, the `.gitignore` file ignores the report files.



Note:

- After you pull a project in Git perspective, 'Project is missing required Library' error appears in the **Verify Problems** tab. This error occurs due to the `.classpath` file which is specific to a workspace or system and needs to be pointed to the newly imported location. To resolve this issue:
 1. Under **Package Explorer**, from the required project, navigate to **Java Build Path > Libraries**.
 2. Delete all the jar files that are missing after the pull. These files are marked with a red cross.
 3. Playback the project. The required jar files are added to the project.
- You can also specify additional file types in the `.gitignore` file so that these file types are ignored while committing the project contents to Git.

Integration with Engineering Test Management

You can integrate IBM® Engineering Test Management (formerly known as IBM® Rational® Quality Manager) with HCL OneTest™ Performance to initiate test runs from Engineering Test Management.

To run tests from Engineering Test Management, you must configure the default adapter that is installed when you install HCL OneTest™ Performance.

You can run the adapter in the following modes:

- GUI
- CLI
- Windows™ service

Engineering Test Management reports

When you run a test script from Engineering Test Management, the default report that is displayed during a test run is attached to the results of Engineering Test Management. You can customize the reports based on your requirements. See [Customizing reports on page 778](#).

If you use Engineering Test Management 4.0 or later, you can view and analyze the test reports in Engineering Test Management. You can analyze the test reports while the test is in running state and after the test run is complete. You can click the **Analyze Results Interactively using HCL OneTest™ Performance** option from the **Execution Results** dialog box to view the test reports in Engineering Test Management.



Note: To access reports from outside of HCL OneTest™ Performance, you must enable the remote access from HCL OneTest™ Performance. See [Access reports remotely on page 797](#).

The result completion state that is reported to Engineering Test Management reflects the overall verdict of the test log that is associated with the run. See [Logs overview on page 797](#). In many cases, a test might contain a failed verification point, but still is considered as passed. You can view the attached report in the execution result of Engineering Test Management, and then set the execution results status accordingly.

You can view the full run results from within HCL OneTest™ Performance by opening HCL OneTest™ Performance in the workspace that is configured to be used by the adapter.

If the adapter is running from the command line or as a Windows™ service, you must stop the adapter before opening HCL OneTest™ Performance. When HCL OneTest™ Performance is opened, you can access the full test reporting and test log capabilities. The test results for the runs that are initiated from Engineering Test Management are under the Engineering Test Management **Results** page.

For HCL OneTest™ Performance schedules, the result completion state that is reported to Engineering Test Management is based on the overall **Requirements** status. Only performance requirements for the last user stage that is defined in the schedule are covered by the report. If no requirements are specified, the result completion state in Engineering Test Management is set to *inconclusive*. In this case, you can view the attached performance reports and manually set the completion state in Engineering Test Management. See [Defining requirements in schedules on page 558](#).

Known limitations

- You cannot run tests from Engineering Test Management with encrypted datasets. When using such datasets, a password prompt is not displayed in the adapter service or in the command-line interface. The use of encrypted datasets are not recommended in the GUI mode, because it requires user interaction with HCL OneTest™ Performance to initiate test runs from Engineering Test Management.
- You can start only one adapter per product installation on a given computer. If you use multiple adapters on the same computer, it requires you to install each product as its own software package in its own directory. If you want to run multiple adapters on the same computer, you must ensure that adapters are using different workspaces.

For information about using Engineering Test Management, refer to the [IBM Engineering Lifecycle Management](#) documentation.

Refer to the following topics to learn more about integrating Engineering Test Management with HCL OneTest™ Performance.

Configuring the Engineering Test Management adapter

You must configure the Engineering Test Management adapter to establish a successful connection between HCL OneTest™ Performance and Engineering Test Management.


Before you begin

- You must have the following information:
 - The URL of the Engineering Test Management server.
 - A user credential and valid license to access Engineering Test Management.
 - The user account must be added to the project area that is being accessed by the adapter with write permissions to the project.
- You must have added `-Dhptcostconfirm` to the `eclipse.ini` file before you run a test from Engineering Test Management.



The `eclipse.ini` file must be available at the installation directory of HCL OneTest™ Performance.

For more information about Engineering Test Management, refer to the [IBM Engineering Lifecycle Management](#) documentation.

1. Open HCL OneTest™ Performance.
2. Click **Window > Preferences > Quality Manager Adapter**.
3. Enter the following information of the Engineering Test Management:

Fields	Actions
Server URL	Enter the URL of Engineering Test Management. For example, <code>https://<hostname>:<portnumber>/qm</code>  Note: If you rename the Engineering Test Management server, you must perform the following tasks: <ol style="list-style-type: none"> a. Update the Engineering Test Management server name in the hosts file with a new name. b. Update the Server URL field with the new name. c. Configure the adapter to point to the new URL.
Adapter name	Enter a unique name to identify the Engineering Test Management adapter. The Engineering Test Management adapter uses the name of the computer as the default name of the adapter.
Project area	Enter the name of the project area in Engineering Test Management.

4. Select one of the following **Authentication type** from the drop-down list to connect to Engineering Test Management:

Authentication type	Actions
Username and Password	<p>Perform the following steps:</p> <ol style="list-style-type: none"> Enter the username associated with Engineering Test Management in the User ID field. Enter the password associated with the username of Engineering Test Management in the Password field.
KERBEROS	<p>Click Browse to locate and select the <code>kerberos.ini</code> file in the Configuration File field.</p> <p> Note: The <code>kerberos.ini</code> file is automatically created when you set up Kerberos.</p> <p>For example, on Windows systems, you can locate the file in the <code>c:\windows\krb5.ini</code>. The file name and the location might change based on the operating systems.</p>
SSLCERT	<p>Perform the following steps:</p> <ol style="list-style-type: none"> Enter the location of the SSL certificate keystore in the Certificate Location field. Enter the keystore password in the Password field. <p> Note: The expected format of the keystore is p12. The keystore must contain the client certificate that the adapter uses when you authenticate with Engineering Test Management.</p>
SMARTCARD	Select a certificate from the drop-down list from the Certificate Selection field.

5. **Optional:** Select the **Enable Proxy** checkbox to connect through a proxy computer and perform the following steps to enter the **Proxy Details** of the computer:
- Enter the hostname of the proxy computer in the **Host** field.
 - Enter the port number of the proxy computer in the **Port** field.
 - Enter the username and password of the proxy computer in the **User** and **Password** fields.
6. Click **Apply and Close** to save and close the configuration.

Results

You have configured the details of Engineering Test Management on HCL OneTest™ Performance.

What to do next

You must start the adapter from HCL OneTest™ Performance, command-line interface, or as a Windows service.

Related information

[Connecting and disconnecting the Engineering Test Management adapter from the GUI mode on page 136](#)

[Starting and stopping the Engineering Test Management adapter from the command line on page 137](#)

[Starting and stopping the Engineering Test Management adapter as a Windows service on page 138](#)

[Importing test assets into Engineering Test Management on page 140](#)

Configuring the workspace directory of the adapter

You must configure the workspace directory of the adapter to start or stop the Engineering Test Management adapter either from command-line interface or as a Windows service.

About this task

If the **Use resources that are local to a test machine** option is set in Engineering Test Management, then the `WORKSPACE_DIR` must be set to the same workspace where your test assets are located.

1. Locate the `adapter.config` file in the `product_install_dir \HOT-PERF_RQMAadapter\config\` directory.

Where `product_install_dir` is the directory where HCL OneTest™ Performance is installed.

For example, `C:\Program Files\HCL\HCLOneTest`.

2. Edit the `WORKSPACE_DIR` variable in the `adapter.config` file to point to the same test workspace that you want the adapter to use.

For example, `WORKSPACE_DIR= C:\Users\username\HCL\HCLOneTest\my_adapter_workspace`.

Results

You have configured the workspace directory of the adapter.

What to do next

You can start or stop the Engineering Test Management adapter either from command-line interface or as a Windows service.

Connecting and disconnecting the Engineering Test Management adapter from the GUI mode

You can use the **Quality Manager Adapter** view to connect, disconnect, and view adapter activities from HCL OneTest™ Performance.

Before you begin

You must have configured the Engineering Test Management adapter. See [Configuring the Engineering Test Management adapter on page 132](#).

About this task

In the GUI mode, when a script is run from Engineering Test Management, you can see the test run in progress inside HCL OneTest™ Performance as though the test were run manually in HCL OneTest™ Performance.

Push buttons to connect and disconnect to the Engineering Test Management server are located in the upper-right corner of **Quality Manager Adapter** view. This view also has a local preferences menu that you can use to control some behavior of the GUI mode adapter. If you see errors or warnings, use the **Error Log** view for further investigation.



Note: You must not use HCL OneTest™ Performance while the adapter is running. If you do so, you might interfere with the ability of the adapter to run test scripts. You must stop the adapter before you open HCL OneTest™ Performance.

The following image displays the activities of the adapter in the **Quality Manager Adapter** view:

Problems Recording Control Protocol Data Quality Manager Adapter Git Staging

Connection status: Communication error with server.



Test script: -

Start time: -

End time: -

Messages	Date
i Adapter is successfully connected to the server.	26-Nov-2021, 11:43:23 am
w Communication error with server.	26-Nov-2021, 11:43:22 am

1. Open HCL OneTest™ Performance.
2. Click **Window > Show View > Quality Manager Adapter**.
3. Perform the following actions either to connect or disconnect the adapter:

- Click the **Connect to RQM** icon  to connect the adapter.
- Click the **Disconnect from RQM** icon  to disconnect the adapter.

Results

You have connected or disconnected the Engineering Test Management adapter from HCL OneTest™ Performance.

Related information

[IBM Engineering Test Management overview on page 59](#)

[Configuring the Engineering Test Management adapter on page 132](#)

[Starting and stopping the Engineering Test Management adapter from the command line on page 137](#)

[Starting and stopping the Engineering Test Management adapter as a Windows service on page 138](#)

[Importing test assets into Engineering Test Management on page 140](#)

Starting and stopping the Engineering Test Management adapter from the command line

You can use the command-line interface to start, stop, and view activities of the Engineering Test Management adapter that you configured in HCL OneTest™ Performance.

Before you begin

You must have performed the following tasks:

- Configured the adapter in HCL OneTest™ Performance. See [Configuring the Engineering Test Management adapter on page 132](#).
- Configured the workspace directory of the adapter. See [Configuring the workspace directory of the adapter on page 135](#).

About this task

When you run test assets from the command-line interface, the adapter activities are printed to the `adapter.log` file that can be accessed from `product_install_dir\HOT-PERF_RQMAdapter\logs`.

To print the current status of the adapter, you must navigate to the `product_install_dir\HOT-PERF_RQMAdapter\bin` directory, and then you can run the `RQMAdapter.bat STATUS` command.

Where, `product_install_dir` is the installation directory of HCL OneTest™ Performance.



Warning: You must not use HCL OneTest™ Performance while the adapter is running. You must stop the adapter before you open HCL OneTest™ Performance for any reason.

1. Open a command-line interface.
2. Navigate to the `product_install_dir\HOT-PERF_RQMAdapter\bin\` directory.
3. Perform the following step either to start or stop the adapter:

- Run the following command to start the adapter from the command line:

Operating system	Command to be run
Windows™	RQMAdapter.bat START
Linux™	RQMAdapter.sh START

- Run the following command to stop the adapter from the command line:

Operating system	Command to be run
Windows™	RQMAdapter.bat STOP
Linux™	RQMAdapter.sh STOP

Results

You have started or stopped the Engineering Test Management adapter from the command-line interface.

Related information

[IBM Engineering Test Management overview on page 59](#)

[Configuring the Engineering Test Management adapter on page 132](#)

[Connecting and disconnecting the Engineering Test Management adapter from the GUI mode on page 136](#)

[Starting and stopping the Engineering Test Management adapter as a Windows service on page 138](#)

[Importing test assets into Engineering Test Management on page 140](#)

Starting and stopping the Engineering Test Management adapter as a Windows™ service

You can use the Windows™ service to start, stop, and view adapter activities.

Before you begin

You must have completed the following tasks:

- Configured the adapter in HCL OneTest™ Performance. See [Configuring the Engineering Test Management adapter on page 132](#).
- Installed Microsoft™ .NET Framework 3.5.x in Windows systems for the adapter service.
- Configured the workspace directory of the adapter. See [Configuring the workspace directory of the adapter on page 135](#).

About this task

When you install HCL OneTest™ Performance, you must install the adapter as a Windows™ service. The default status of **Startup Type** is set to **Manual**.

Group Policy Client	The service is responsible for applying setti...	Running	Automatic (Tri...	Local System
HCL OneTest Performance adapter for RQM			Manual	Local System
Human Interface Device Service	Activates and maintains the use of hot butt...		Manual (Trigg...	Local System
HV Host Service	Provides an interface for the Hyper-V hyper...		Manual (Trinn...	Local System

Optionally, to configure the service to start automatically, you can right-click the adapter listing, and then select **Properties**. You can then select **Automatic** from the drop-down list in the **Startup Type** field. Therefore, the adapter can start automatically when you restart your computer and does not require you to log in.



Notes:

- When you start the adapter as a service, you cannot run the Web UI tests of HCL OneTest™ Studio from IBM® Engineering Test Management.
- You must not open HCL OneTest™ Performance in the same workspace while the adapter is running as a Windows™ service. You must stop the adapter before you open HCL OneTest™ Performance in the configured workspace.

When you run the adapter as a service, the status of the adapter is printed to the `adapter.log` file that can be accessed from `product_install_dir\HOT-PERF_RQMAdapter\logs`.

You can also print the current status of the adapter by navigating to the `product_install_dir\HOT-PERF_RQMAdapter\bin\` directory, and then run the `RQMAdapter.bat STATUS` command.

1. Open the Windows™ services.
2. Perform the following step either to start or stop the adapter:
 - Right-click **adapter for RQM** and, then click **Start** to start the service.
 - Right-click **adapter for RQM** and, then click **Stop** to stop the service.

Results

You have started or stopped the Engineering Test Management adapter as a Windows service.

Related information

[IBM Engineering Test Management overview on page 59](#)

[Configuring the Engineering Test Management adapter on page 132](#)

[Connecting and disconnecting the Engineering Test Management adapter from the GUI mode on page 136](#)

[Starting and stopping the Engineering Test Management adapter from the command line on page 137](#)

[Importing test assets into Engineering Test Management on page 140](#)

Importing test assets into Engineering Test Management

You can import the performance and service tests into Engineering Test Management by using an adapter.

Before you begin

The adapter must be running on a computer where the test assets are located.

About this task

The names of script types, HCL OneTest™ Performance, HCL OneTest™ UI, and HCL OneTest™ Studio, are compatible with HCL OneTest™ Performance, HCL OneTest™ UI, and HCL OneTest™ Studio products.

1. Log in to Engineering Test Management.
2. Click **Construction > Import Test Scripts**.
3. Select one of the following test scripts in the **Script Type** field:
 - a. **HCL OneTest Performance** to import a performance test or schedule from HCL OneTest™ Performance.
 - b. **HCL OneTest UI** to import a functional test from HCL OneTest™ UI.
 - c. **HCL OneTest Studio** to import a Web UI test from HCL OneTest™ UI or import a test from HCL OneTest™ Studio.
4. Select **Use test resources that are local to a test machine**, and click **Select Adapter**.
5. Select the computer on which the adapter is running, and click **Next**.
6. Enter the name of the project in the **Project Path** field, and then click **Go**.



Note: You must specify only the project name and not the entire path to the project.

7. Select the test assets that you want to import, and then click **Finish**.
8. Select those test assets to import again, and then click **Import**.

Results

You have imported the test assets to Engineering Test Management by using the adapter.

Related information

[IBM Engineering Test Management overview on page 59](#)

[Configuring the Engineering Test Management adapter on page 132](#)

[Connecting and disconnecting the Engineering Test Management adapter from the GUI mode on page 136](#)

[Starting and stopping the Engineering Test Management adapter from the command line on page 137](#)

[Starting and stopping the Engineering Test Management adapter as a Windows service on page 138](#)

[Testing shared assets with Engineering Test Management on page 141](#)

Testing shared assets with Engineering Test Management

You can make test projects and assets shareable in Engineering Test Management. By sharing assets, any computer with your product, that is connected to Engineering Test Management can execute a test or schedule.

Before you begin

When you are working with tests or schedules from a remote shared location, HCL OneTest™ Performance uses a local workspace for the Engineering Test Management adapter. This adapter workspace is different from normal workspaces because the test assets are stored remotely. This means that every asset that is related to the test or schedule is downloaded from the shared location into the local workspace before execution. The following limitations apply:

- Assets in the adapter workspace might be deleted or overwritten with newer versions when updates are made to the shared location.
- If you change the shared location in the adapter workspace, the entire project is removed from the adapter workspace.
- Test results are stored in a different project, called `RQM_Results`, and are never deleted. The Engineering Test Management test result page links to the correct location.



Note: Do not edit test assets in the adapter workspace because you might lose your work. You must use these assets only for running tests and schedules.

If you are using source control and want to include only the minimum required assets, then include the following files:

- All `*.testsuite` tests and schedule files
- The `/src` directory if you use custom code
- All `*.dtp` dataset files
- All `*.location` location files

- All digital certificates
- All WSDL and SOA security files



Note: All other assets, such as test results, are not required.

Custom code Java™ classes in the shared assets cannot use libraries that are outside the workspace. If your custom code must use such a library, then copy the library into the project, and update the classpath to use the local copy.

1. Create a shared directory on the computer that hosts the UNC file system that contains the test projects to share.

Example

For example, create a directory called: `C:\MyRemoteWorkspace`.

2. Copy the test projects to share into the shared directory.

If a project is stored in source control software, then copy it from there.

3. Check that the Engineering Test Management server can access the shared directory by using UNC paths.

Example

For example, the `\\MyServer\RPTRemoteAssets\` path must be mapped to the `C:\MyRemoteWorkspace` directory.

4. In Engineering Test Management, specify the directory that contains the actual test projects that are located in the shared directory.
5. Verify that you have correctly specified the UNC shared directory by browsing for the shared resource. Ensure that the first dialog box contains the projects at the first level.

You must not have intermediate directories between the UNC shared directory and the project directory.

Related information

[IBM Engineering Test Management overview on page 59](#)

[Configuring the Engineering Test Management adapter on page 132](#)

[Importing test assets into Engineering Test Management on page 140](#)


Integration with IBM® Engineering Workflow Management

You can integrate Engineering Workflow Management (formerly known as Rational® Team Concert™) to create and track defects (bugs) or other work items, as a defect tracking tool in HCL OneTest™ Performance.

You can use HCL OneTest™ Performance to record and play back tests for the application that you develop and view their results. When you discover that you might want to raise defects, issues, or other types of work items for the test assets, you can create defects, issues, or other work items without the need to open Engineering Workflow Management.

For more information about Engineering Workflow Management, refer to the [IBM® Engineering Workflow Management](#) documentation.

The following table lists the tasks that you must perform to integrate HCL OneTest™ Performance with Engineering Workflow Management:

Tasks	Go to...
Install HCL OneTest™ Performance.	Installing the software by using stand-alone installer on page 99
Create any or all of the following types of test assets in HCL OneTest™ Performance to test your application: <ul style="list-style-type: none"> • Compound tests • Performance tests • Schedules (Rate or VU Schedules) 	Test Author Guide on page 180
Install Engineering Workflow Management and gain access to it.	IBM® Engineering Workflow Management documentation  Note: The System Requirements on page 31 provide more information about specific versions of Engineering Workflow Management requirements.
Run the test assets.	Running a local schedule or test on page 608
Configure the Engineering Workflow Management server URL in HCL OneTest™ Performance.	Configuring the URL of Engineering Workflow Management on page 143
Create defects from HCL OneTest™ Performance.	Tracking defects with Engineering Workflow Management on page 144

Configuring the URL of Engineering Workflow Management

You must configure the URL of the Engineering Workflow Management server to use it as defect tracking tool in HCL OneTest™ Performance.

Before you begin

You must have the URL of the Engineering Workflow Management server.

About this task

Bugzilla is configured as the default defect tracking tool in the **Preferences** window of HCL OneTest™ Performance. If you are using Engineering Workflow Management to create and track defects (bugs), you can provide the URL of the

Engineering Workflow Management server to search, submit, or add work items to test results from HCL OneTest™ Performance.

1. Open HCL OneTest™ Performance.
2. Click **Window > Preferences > Test > Test Log Editor**.
3. Enter the URL of the Engineering Workflow Management server in the following fields:

Fields	Format of the URL
Submit URL	<code>https://ewm.example.com:9443/ccm/web/projects/projectname#action=com.ibm.team.workitem.newWorkItem</code>
Search URL	<code>https://ewm.example.com:9443/ccm/web/projects/projectname#action=com.ibm.team.workitem.newWorkItem</code>
Open URL	<code>https://ewm.example.com:9443/ccm/web/projects/projectname#action=com.ibm.team.workitem.newWorkItem&id=</code>

Where,

- `ewm.example.com:9443` is the URL of the Engineering Workflow Management server.
- `projectname` is the name of the project in the Engineering Workflow Management server.



Note: You must update the URL, if there is a change in the name of the project in Engineering Workflow Management.

4. Click **Apply and Close** to save the configuration and close the **Preferences** window.

Results

You have configured the URL of Engineering Workflow Management in HCL OneTest™ Performance.

What to do next

You can create defects for the test results that are available in your project in HCL OneTest™ Performance. See [Tracking defects with Engineering Workflow Management on page 144](#).

Tracking defects with Engineering Workflow Management

You can submit defects to Engineering Workflow Management from HCL OneTest™ Performance. By default, the test log editor uses Bugzilla as the defect tracking site. You must configure the product to use Engineering Workflow Management for defect tracking.

1. Click **Window > Preferences > Test > Test Log Editor**.

Result

The **Test Log Editor** preferences window opens.

- Specify the URL of Engineering Workflow Management server in the **Submit URL**, **Search URL**, and **Open URL** fields.

Contact the administrator of the Engineering Workflow Management server for more information.

Example

The following are the example of URLs, where the name of the Engineering Workflow Management server is *evm.example.com* and the name of the project is *projectname*:

Table 2. Example of EVM Server URL

Field names	Field values
Submit URL	<code>https://evm.example.com:9443/jazz/web/projects/projectname#action=com.ibm.team.workitem.newWorkItem</code>
Search URL	<code>https://evm.example.com:9443/jazz/web/projects/projectname#action=jazz.viewPage&id=com.ibm.team.workitem</code>
Open URL	<code>https://evm.example.com:9443/jazz/web/projects/projectname#action=com.ibm.team.workitem.viewWorkItem&id=</code>



Note: You must change the URLs, if there is change in name of the Engineering Workflow Management server.

Related information

[Logs overview on page 797](#)

Integration with HCL Launch

When you use HCL Launch for automating application deployments of your application, you can create tests for your application in HCL OneTest™ Performance and run those tests in HCL Launch by using the HCL OneTest™ Performance Launch plugin.

Overview

You can use the HCL OneTest™ Performance Launch plugin to integrate HCL Launch with HCL OneTest™ Performance. Integrating HCL Launch with HCL OneTest™ Performance automates the test execution process. If you have many tests to run at regular intervals automatically, you can use HCL Launch to initiate test execution.

Installing the HCL OneTest™ Performance Launch plugin

You must install the HCL OneTest™ Performance Launch plugin to run performance test assets from HCL Launch.

Before you begin

You must have downloaded the latest version of the HCL OneTest™ Performance Launch plugin from [HCL® License & Delivery portal](#).

For more information about specific versions of plugin, see [Integration plugin compatibility matrix on page 114](#).

1. Open the HCL Launch dashboard.
2. Click **Settings**.
3. Click **Automation Plugins** from the **Automation** pane.
4. Click **Load Plugin**.
5. Click **Choose File** to locate and **Open** the compressed file.



Note: Do not extract the compressed file contents.

6. Click **Submit**.

Result

The HCL OneTest™ Performance Launch plugin is displayed in the **Automation Plugins** tab.

What to do next

You can add tests that you created in HCL OneTest™ Performance to your task and then run the tests on the HCL Launch server. See [Running HCL OneTest Performance tests on the HCL Launch server on page 146](#).

Running HCL OneTest™ Performance tests on the HCL Launch server

After you install the HCL OneTest™ Performance Launch plugin on the HCL Launch server, you can create a `process request` that contains the test for your application, and then run the test on the HCL Launch server.

Before you begin

You must have installed the latest version of the HCL OneTest™ Performance Launch plugin. See [Installing the HCL OneTest Performance Launch plugin on page 145](#).

About this task

After you have installed the HCL OneTest™ Performance Launch plugin on the HCL Launch server, you can either use an existing component in your project or create a component. You can create a component process and select the HCL OneTest™ Performance test step to edit the *step properties* for the test you want to run. After selecting the agent, you can create an application. You can then create an application process for the application, and then submit the application process for a run.

1. Log in to the HCL Launch server, if you are not already logged in.
2. Click **Components** from the HCL Launch dashboard, and then click **Create Component** to create a component.



Note: You can either use an existing component or create a component.

3. Create a component process in the component by performing the following steps:

- a. Open the component that you created.
- b. Click the **Processes** tab from the component dashboard, and then click **Create Process**.

Result

The **Create Process** dialog box is displayed.

- c. Enter the required values to create a component process and click **Save**.



Note: All mandatory fields are marked with an asterisk (*) in the UI.

- i. Enter the process name in the **Name** field.
- ii. Select **Operational (No Version Needed)** from the **Process Type** list.



Note: The **Default Working Directory** field displays the folder path where the agent can download the artifacts and create temporary files.

The process that you created is listed in the **Processes** list and the **Design** tab for the process is displayed.



Note: The process opens in the process editor. The process editor lists the plugins and steps. The required **Start** and **Finish** steps represent the beginning and the end of the process and are automatically placed on the design area.

4. Select the process step you want to run by completing the following steps:

- a. Search for the HCL OneTest™ Performance test process step from the left design pane.
- b. Select the HCL OneTest™ Performance test process step and drag the test into the design area.

Result

The selected test is placed in between the **Start** and **Finish** steps.

5. Specify the properties for the selected test by performing the following steps:

- a. Click the **Edit** icon .

Result


The **Edit Properties for Run an HCL OneTest™ Performance test** dialog box of the selected test is displayed.



b. Specify the properties for the selected test step by following the action in the table that follows.



Note: All mandatory fields are marked with an asterisk (*) in the UI.

Field	Action required for a HCL OneTest™ Performance test
Name	Enter the name of the step.
Workspace	The complete path to the Eclipse workspace.
Project	The path, including the file name of the project relative to the workspace.
Test Suite Name	The path, including the file name of the test to run relative to the project.
IMShared Location	The complete path to HCLIMShared location.
Var File	The complete path to the XML file that contains the variable name and value pairs.
Config File	The complete path to a file that contains the parameters for a test or schedule run.
Results File	The name of the results file. The default result file is the test or schedule name with a time stamp appended.
Overwrite Results file	Determines whether a results file with the same name is overwritten. The default value is true, which means that the results file can be overwritten.
Quiet	Turns off any message output from the launcher and returns to the command shell when the run or the attempt is complete.
Number of Virtual Users	For a schedule, the default value is the number of users specified in the schedule editor. For a test, the default value is one user. Overrides the default number of users, if required. This option creates a new copy of the schedule that contains the specified number of users.
VM Args	Java™ virtual machine arguments to pass in.

Field	Action required for a HCL OneTest™ Performance test
Dataset Override	<p>For a test or schedule, the default value is the dataset specified in the test editor or schedule editor. Overrides the default dataset value to run if required.</p> <p> Note:</p> <p>You must use the <code>Dataset Override</code> option to replace the dataset values during a test or schedule run. You must ensure that both original and new datasets are in the same workspace and have the same column names. You must also include the path to the dataset.</p> <p>For example,</p> <pre data-bbox="971 940 1398 1052">/project_name/ds_path/ds_file- name.csv:/project_name/ds_path/new_d- s_filename.csv.</pre> <p>You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon (;).</p> <p>For example,</p> <pre data-bbox="971 1329 1398 1514">/project_name1/ds_path/ds_file- name.csv:/project_name1/ds_path/new_- ds_filename.csv;/project_name2/ds_- path/ds_filename.csv:/project_- name2/ds_path/new_ds_filename.csv</pre>
Resource Monitoring Labels Override	<p>For a schedule (Rate schedule or VU schedule), use Resource Monitoring Labels Override to perform any of the following actions:</p> <ul style="list-style-type: none"> ▪ To enable the Resource Monitoring from Service option for a performance schedule if the Resource Monitoring from Service option is not enabled from the

Field	Action required for a HCL OneTest™ Performance test
	<p>schedule editor in HCL OneTest™ Performance.</p> <ul style="list-style-type: none"> ▪ To ignore Resource Monitoring sources that were set in the performance schedule and to change for a label matching mode. ▪ To replace an existing set of Resource Monitoring labels that were set in the performance schedule and run the schedule with a new set of Resource Monitoring labels. <p> Note: You can add multiple Resource Monitoring labels separated by a comma.</p> <p> Important: You must add the Resource Monitoring labels to the Resource Monitoring sources on the Resource Monitoring page in your HCL OneTest™ Server project.</p>
Exported HTTP Test log file	The complete path to a file to store the exported HTTP test log.
Exported Statistical Report Data File	The complete path to the directory to store exported statistical report data.
Custom Report Format Files	A comma-separated list of absolute paths to custom report format files (.view files) to use when exporting statistical report data with the -export-stats option.
User Comments	Add text within the double quotation mark to display it in the User Comments row of the report.
Field	Optional action for a HCL OneTest™ Performance test
Working Directory	Specify an alternative path to the working directory for this step. ¹

1. You need not set this property for a step when you are running an HCL OneTest™ Performance test.

Field	Action required for a HCL OneTest™ Performance test
Post Processing Script	Specify if you want to run any scripts after the completion of the test run. ¹ The Step Default is selected by default. You can click New to add new scripts.
Precondition	Specify any conditions that are to be completed before the test runs. You can edit the script by clicking the script displayed. ¹
Use Impersonation check box	Select this check box to run the test as a different user. ¹
Auth Token Restriction	Set the authentication token actions by applying token restrictions. ¹ The System Default is selected by default. You can add a new token restriction or edit the one already added.

c. Click **OK** to save the properties for the test.

6. Click **Save** in the design area.

7. Click the **Resources** tab from the HCL Launch dashboard and create a resource by clicking **Create Top-Level Group**.

Result

The created resource is displayed on the **Resource Tree** tab page.

8. Select the agent that runs the test by completing the following steps:



Important: You must have already installed the agent on the HCL Launch server that you want to use.

a. Select the resource displayed on the **Resource Tree** tab page.

b. Click the **Horizontal ellipsis** icon  for the selected resource.

c. Click **Add Agent**.

d. Select the agent to add to the resource, and then click **Save**.


Result

The selected agent is added to the resource in the **Resource Tree** pane and the status of the agent can also be viewed.



Important: The agent must be **Online** for the test to run.

9. Add the component to the agent by performing the following steps:

- a. Click the **Horizontal ellipsis** icon  for the agent.
- b. Click **Add Component** on the list.
- c. Select the component to add to the resource, and then click **Save**.

Result

The selected component is added to the agent for the resource in the **Resource Tree** pane.

10. Create an application by completing the following steps:

- a. Click the **Applications** tab from the HCL Launch dashboard.
- b. Click **Create Application**.
- c. Complete the details in the **Create Application** dialog box, and then click **Save**.

Result

The **Environments** tab page is displayed for the created application.

- d. Click **Create Environment** to create an environment for the application that you created.
- e. Complete the details in the **Create Environment** dialog box, and then click **Save**.

Result

The environment that you created is displayed.

- f. Click the environment to open, and then click **Add Base Resources**.
- g. Select the resource from the list in the **Add Resource to Environment** dialog box, and then click **Save** to add the resource to the environment.

Result

The resource added to the environment is displayed.



Note: When you add a resource to an environment, the corresponding agent and the component are displayed for the resource.

- h. Click the application from the `breadcrumbs`.

Result

The **Environments** tab page is displayed for your application.

- i. Add the component to the application by performing the following steps:
 - i. Click the **Components** tab.
 - ii. Click **Add Component**.
 - iii. Select the component from the list in the **Add a Component** dialog box, and then click **Save**.

Result

The selected component is displayed on the **Components** tab page.

- j. Create a process for the application by performing the following steps:
 - i. Click the **Processes** tab.
 - ii. Click **Create Process**.
 - iii. Complete the details in the **Create an Application Process** dialog box, and then click **Save**.

Result

The **Design** tab page for the application process that you created is displayed.

- k. Select the component process from the left pane and drag it into the design area.



Note: You can click the **Edit** icon  to add the properties, if required.

- l. Click **Save** in the design area.

11. Select the application process to run the test by completing the following steps:

- a. Click **Applications** from the HCL Launch dashboard.
- b. Click the application that you configured for a test run.
- c. Click **Request Process**.

Result

The **Create Deployment** page is displayed.

- d. Specify the process request by performing the following steps:
 - i. Select the application from the **Application** list.
 - ii. Select the environment from the **Environment** list.
 - iii. Select the application process from the **Process** list.
 - iv. Optionally, enter a description in the **Description** field.
 - v. Click **Next**.
 - vi. Select the component versions as required, and then click **Next**.
 - vii. Enable **Run Now** to run the `application process request`.



Note: You can also schedule the `application process request` at a later point of time. To do this, you can disable **Run Now**, and then specify the date, time, and the recurrence pattern to run the `application process request` at a stipulated time.

viii. Click **Next**.

ix. Verify the process request details, and then click **Submit Deployment**.

Result

The HCL Launch dashboard shows the progress of the application process request.


Results

You have used the HCL OneTest™ Performance Launch plugin to integrate HCL Launch with HCL OneTest™ Performance and run the test from your project on the HCL Launch server.

After the HCL Launch process request runs successfully, you can view the status of the completed process request displayed as follows:

- **Success:** When the test run is successful
- **Failed:** When the test run is failed

What to do next

- You can view the details of the test run as a process from the HCL Launch dashboard. Expand the `step`. You can then expand the application process. You can then hover over the process, and then click the **Output Log** icon . The output log is displayed. You can verify the log details.



Note: In HCL OneTest™ Performance, if the HCL OneTest™ Server URL is configured in **Window > Preferences > Test >** and **Publish result after execution** is set as **Always** in **Window > Preferences > Test > > Results**, then the **Reports information** section on the **Log** file displays the names of the report along with its corresponding URLs. The report URLs are the HCL OneTest™ Server URLs where the reports are stored. You can access the report URLs to view the test execution information at any point of time.

Testing with HCL OneTest™ API

Starting from V9.1.1, you can use HCL OneTest™ API extension to execute API tests. In HCL OneTest™ Performance, you can either import the projects from HCL OneTest™ API or manage them from HCL OneTest™ Performance by establishing the connection between the products. In HCL OneTest™ Performance, you can create a schedule or compound test to run the IntegrationAPI tests by using the Agents.

Before you begin

To be able to work with API tests, you must install HCL OneTest™ Performance Extension for HCL OneTest™ API.

You also need a HCL OneTest™ Performance agent or a HCL OneTest™ API agent to execute the tests remotely. When installing HCL OneTest™ API agent, it is recommended to choose **This Agent will only run probes** option.

Moreover, if you update API tests in HCL OneTest™ Performance and want to apply the updates back to HCL OneTest™ API, you must install HCL OneTest™ API and define the path to its installation directory to set the connection.

Following are the use cases to work with API tests in the HCL OneTest™ Performance:

- Both the products are installed and you connect to the API project, or you open the API resource directly from Test Navigator view and you work directly with the sources files.
- HCL OneTest™ API is not installed and you import the projects in the HCL OneTest™ Performance workspace.



Note: The imported tests must be edited in HCL OneTest™ API. Similarly, the schedules and compounds tests must be edited in HCL OneTest™ Performance.

- To [execute Performance tests with imported tests on page 162](#), the HCL OneTest™ API agent must be installed along with the HCL OneTest™ Performance agent. The environment variable `INTEGRATION_TESTER_AGENT_HOME` must be defined on each location where the agent is installed, and must point to the root directory of the HCL OneTest™ API agent installation.

The following environment variables must be set to apply licenses:

```
HCL_LICENSING_ID = ServerID
```

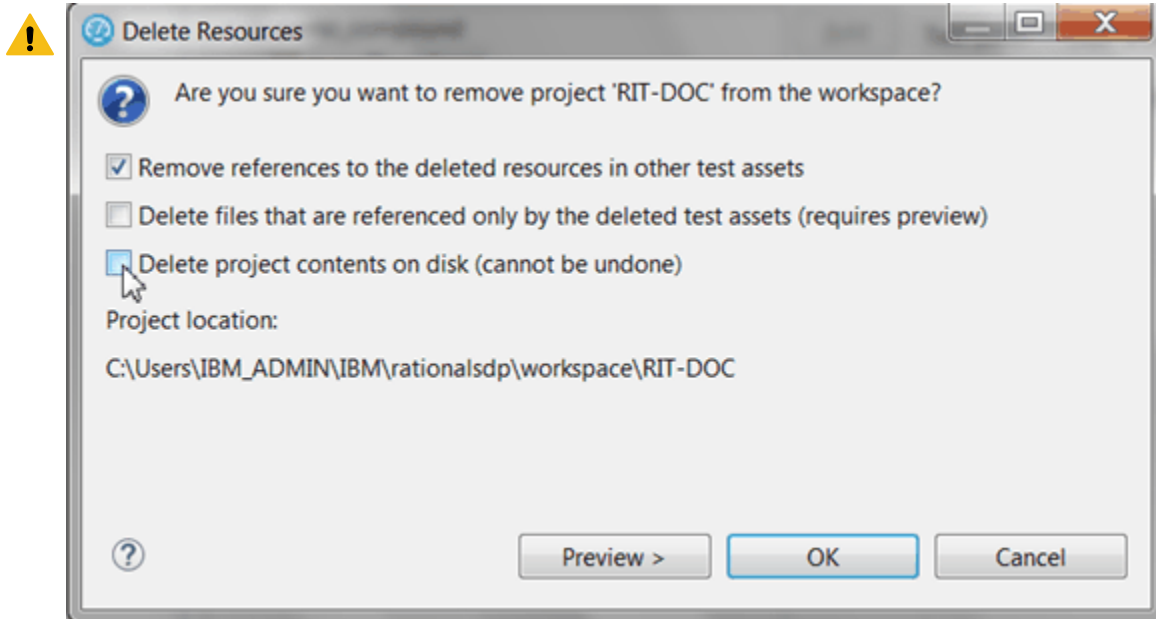
```
HCL_LICENSING_URL = Server url
```

Connecting to an existing API project

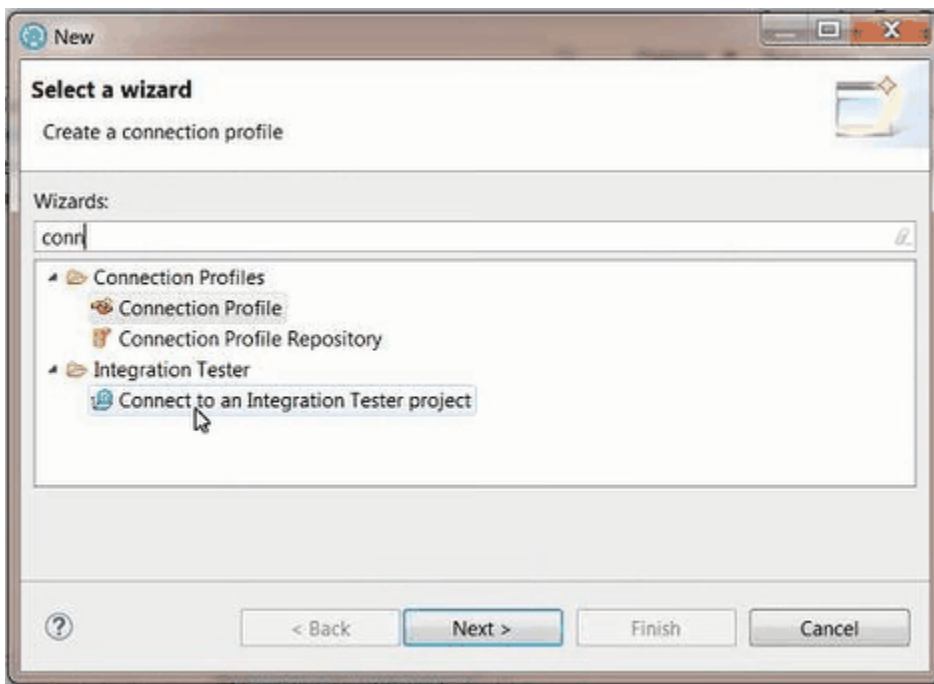
When you connect both the products any change or delete action made in one product workspace is reflected on the other product workspace, if both the products are installed on your machine.



Warning: If you delete a project from the Test Navigator, be sure that the option **Delete project contents on disk** is not selected in the **Delete Resources** dialog, otherwise the project would be deleted in HCL OneTest™ API if it is connected.



- In HCL OneTest™ Performance, right-click on the **Test Navigator**, select **New > Other>HCL OneTest™ API Connect to an HCL OneTest™ APIProject** and click **Next**.

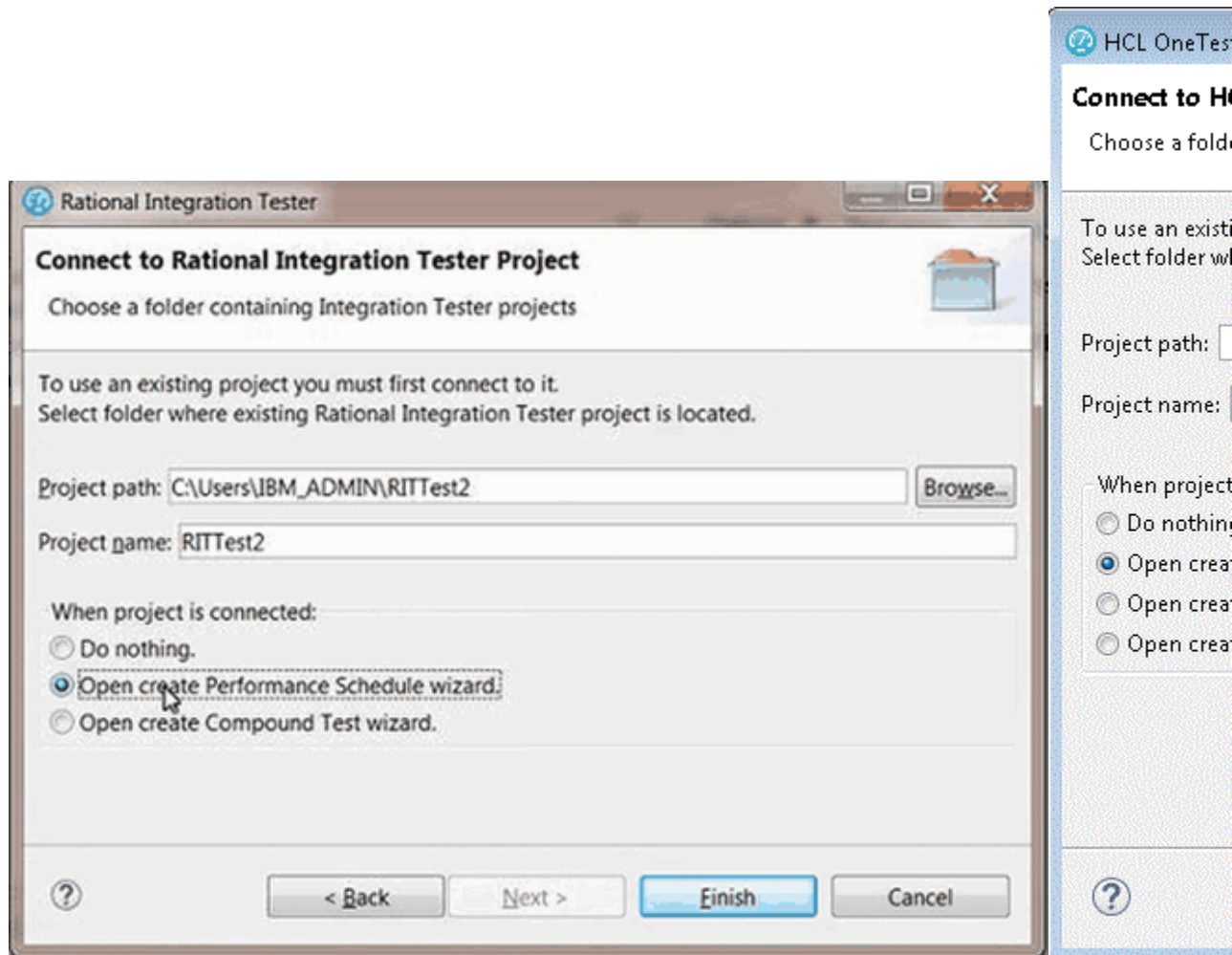


- In the wizard page, click **Browse** and select the root folder that contains the project.

If the path contains a project, its name should automatically appear in **Project Name** and the **Finish** button should be enabled.

- In **When project is connected**, you have to perform one of the following actions:
 - Click **Do nothing** to only import the project.
 - Click **Open Create Rate Schedule wizard** to select the test, create a Rate Schedule, and add the test to it.
 - Click **Open Create VU Schedule wizard** to select the test, create a VU Schedule, and add the test to it.
 - Click **Open Create Compound Test Wizard** to select the test, create a Compound test, and add the test to the Compound test. For more details, see [Creating a compound test on page 483](#) and [Adding tests into a compound test on page 485](#).

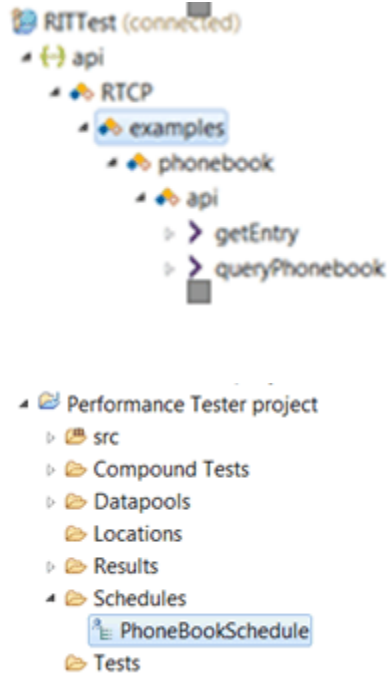
For example: select the **Open Create VU Schedule wizard** VU Schedule action and click **Next**.



The **Create Rate Schedule for Integration Tester** or **Create VU Schedule for HCL OneTest API wizard** displays the list of tests contained in your project.

- Select one or more tests and click **Next**.
- In **Schedule File Name and Location** wizard, select a schedule in an existing project, or create it from this window. You just need to enter a name for the new schedule and click **Finish**. The procedure is the same for the compound test.

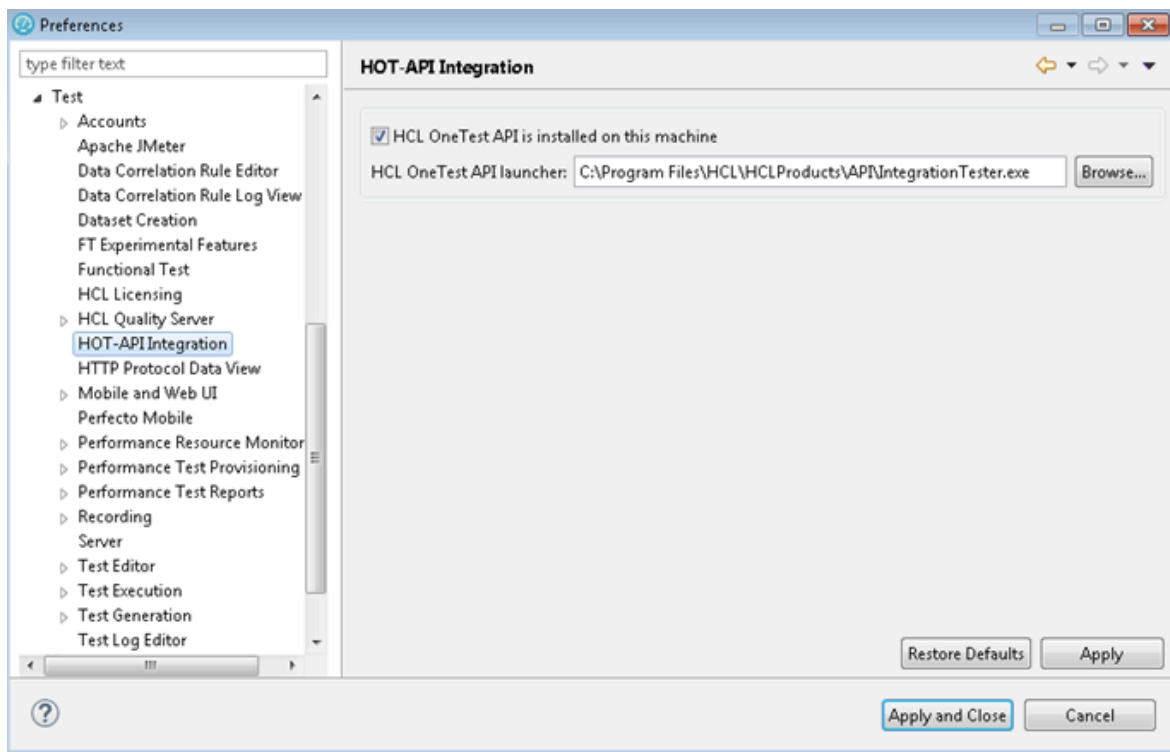
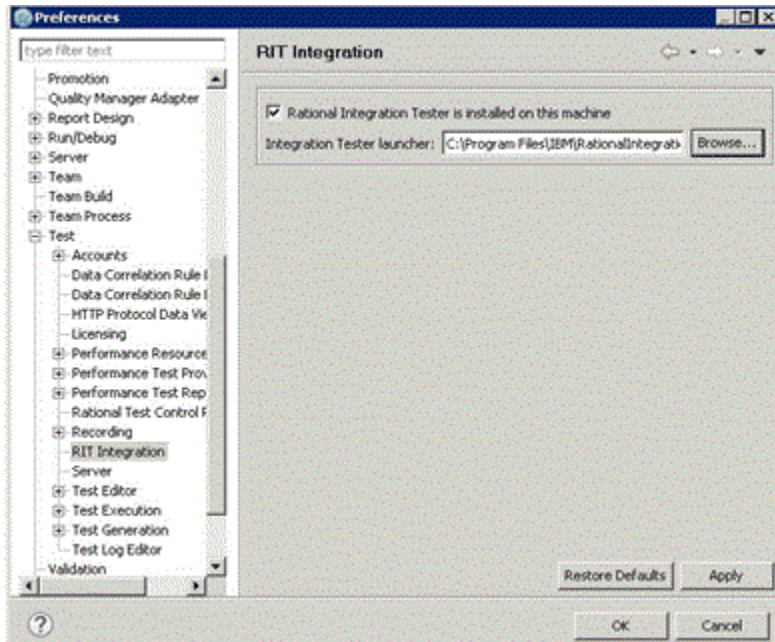
- The **Test Navigator** displays the projects:
 - The HCL OneTest™ API project you are connected to, or that you imported, with the tests.
 - The HCL OneTest™ Performance project containing the schedule or compound test. If HCL OneTest™ API is installed, **Connected** is indicated near the name of the project in the Test Navigator. The Schedule or compound test automatically opens in the dedicated editor.



Setting HCL OneTest™ API preferences

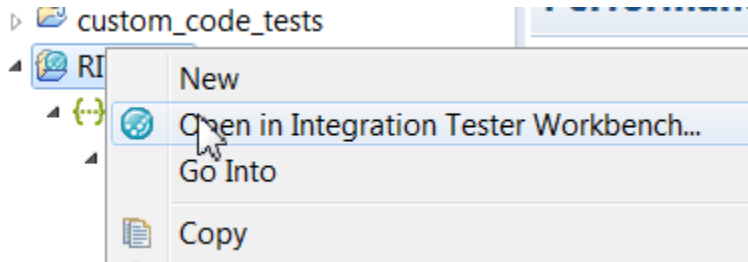
To be able to open an HCL OneTest™ API project from HCL OneTest™ Performance Test Navigator, you need to have both the products installed on the same computer, and you must set the path to the execution file in the Preferences.

- In HCL OneTest™ Performance, click **Window > Preferences > Test > HIT Integration**.
- Click **Browse** and set the installation path to HCL OneTest™ API execution file. On Windows, the default location would be `C:\Program Files\HCL\IntegrationTester.exe`.
- Click **Apply** and **OK**.



Opening HCL OneTest™ API resources from the Test Navigator

- Once the preferences are set, you can open an HCL OneTest™ API project.
- In the **Test Navigator**, open the project root node and children nodes, and at any level, right-click and select **Open in HCL OneTest™ API Workbench**.



If HCL OneTest™ API is automatically detected, the workspace opens for the selected resources.

If HCL OneTest™ API is not detected, a dialog opens on a Preference page where you need to verify the path to the execution file.

- **Warning:** HCL OneTest™ API cannot open more than one project at a time. If you have another project open, you will get an error. In that case, close HCL OneTest™ API and try to open the project again.

Importing HCL OneTest™ API project

If both the products are not installed on the same machine, you can import an HCL OneTest™ API project in your workspace. Another reason for the import is when you have HCL OneTest™ API installed but you do not want to connect to the HCL OneTest™ API project. In that case, the project is duplicated, any updates in one product workspace will not be reflected in the other product's workspace.

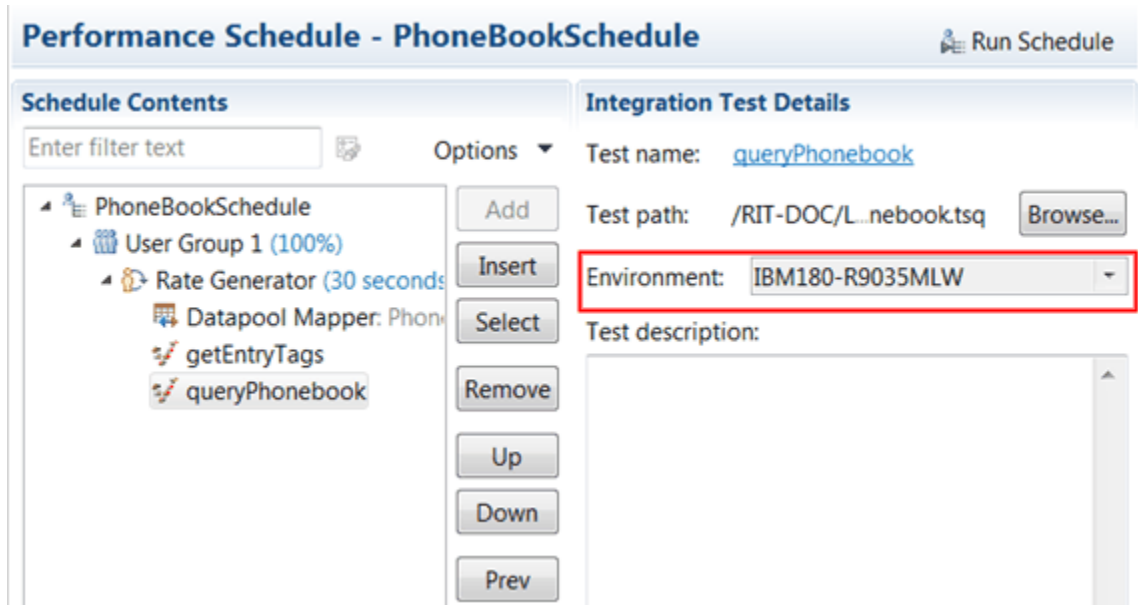
- To import an HCL OneTest™ API project:
- Right-click on the **Test Navigator**, choose **Import** and select **Existing project into workspace**.
- Choose **Select root directory** or **Select archive file**; select a project to import and click **Finish**.

The selected project appears in the **Test Navigator** and the compound test or schedule editor automatically opens.

Modifying HCL OneTest™ API environments in HCL OneTest™ Performance

In the schedule or compound test, you can select HCL OneTest™ API tests and change the environment of each test. The environments are set in HCL OneTest™ API, you can only change the selection from the edited schedule or compound test.

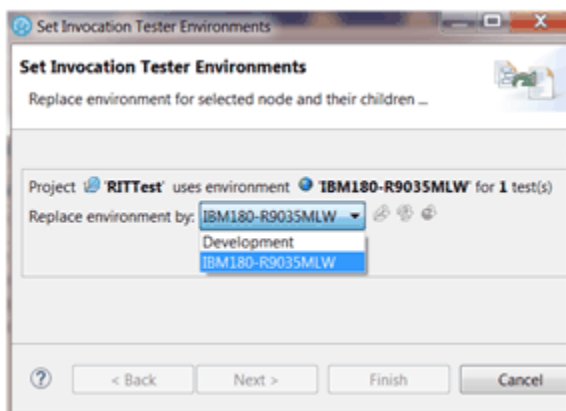
- Open the schedule or compound editor and select a test.
- In the HCL OneTest™ API details, you can browse and change the properties of the selected test. The **Test path**, the **Environments** and **Description** are automatically updated accordingly.



- To select another environment for the Integration Tester test, use the dropdown menu.

Alternatively, you can change the environment selection for a test for a collection of tests:

- Right-click on the tree at any level under a node in the schedule or compound test and select **Replace HCL OneTest™ API Environments**.
- In the **Set Invocation Tester Environments** wizard, the first page displays the list of projects that use the selected environment and the number of tests from project that use this environment in the schedule or compound test.



- Select another used environment in the dropdown list. Click **Finish**. The new choice applies to the selected node and its children.

Next step is to create a compound test or schedule in HCL OneTest™ Performance to [run the Integration tests on page 162](#), see [Creating a VU Schedule on page 524](#) or [Creating a compound test on page 483](#).

You can add a dataset mapper in the compound test or schedule for tests that are using multiple tags. See [Adding Dataset Mapper on page 389](#) to map tags in the HCL OneTest™ API tests with the variable values of HCL OneTest™ Performance.

Running HCL OneTest™ API tests

You can use HCL OneTest™ Performance Extension for HCL OneTest™ API to run HCL OneTest™ API tests.

You can install both the products on the same machine, establish the connection, and run HCL OneTest™ API tests. See [Testing with HCL OneTest™ API on page 154](#).

You also have the option to just import the projects to HCL OneTest™ Performance HCL OneTest™ API, add the tests to a schedule or compound test to run them. You can either use HCL OneTest™ Performance Agent or HCL OneTest™ API Agent to generate the load. You need a compound test or schedule that contains the HCL OneTest™ API tests.

Setting environment variable

To run the tests with HCL OneTest™ API Agent, set the environment variable *INTEGRATION_TESTER_AGENT_HOME* and point it to the HCL OneTest™ API Agent installation directory.

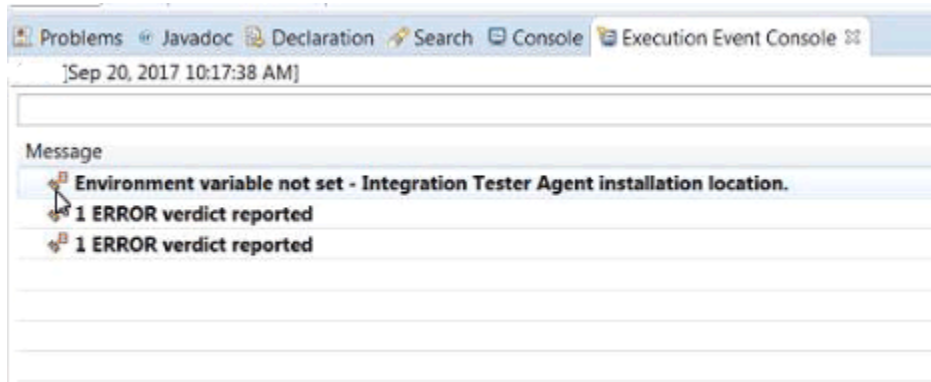
On Windows, set:

```
INTEGRATION_TESTER_AGENT_HOME = C:\Program Files\HCL\HCLProducts\Agent
```

On Linux, set:

```
INTEGRATION_TESTER_AGENT_HOME=/opt/HCL/HOT-API-Agent
export INTEGRATION_TESTER_AGENT_HOME
echo $INTEGRATION_TESTER_AGENT_HOME
```

If the variable is not set, you will get a test log with error message when the compound or schedule test is run.



Running the compound test or schedule

- Click **Run Compound Test** or **Run VU Schedule** or **Run Rate Schedule**.

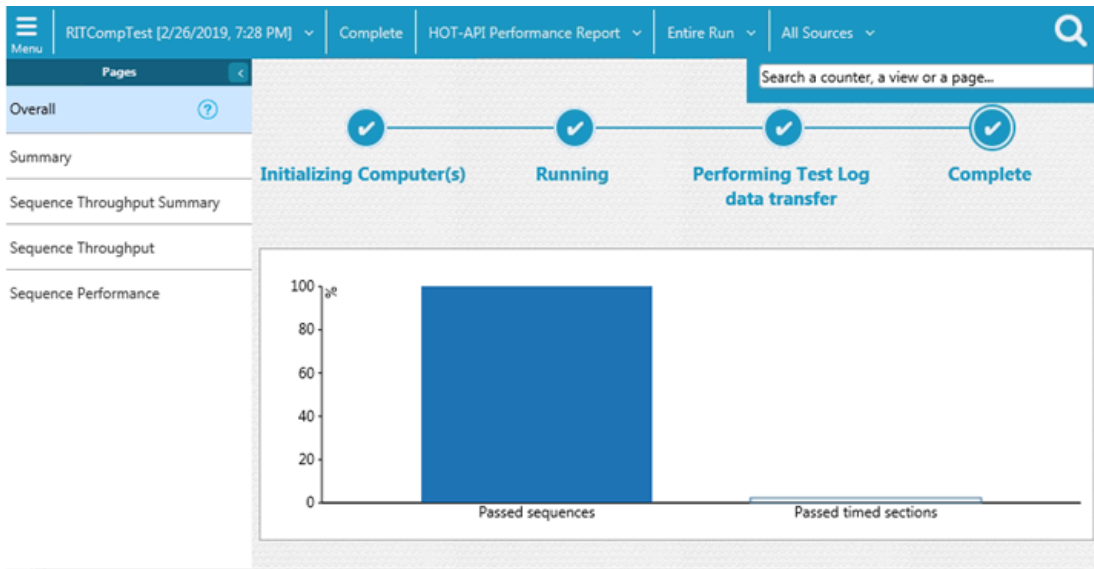
HCL OneTest™ API Performance Report

The report contains the following pages.

OVERALL

The **Overall** page provides this information:

- A progress indicator that shows the state of the run.
- A bar chart on the left shows the overall percentage of passed sequences for the entire run. A sequence corresponds to a test run.
- A bar chart on the right shows the overall percentage of passed timed sections, which are corresponding to individual sections within the steps of the tests.

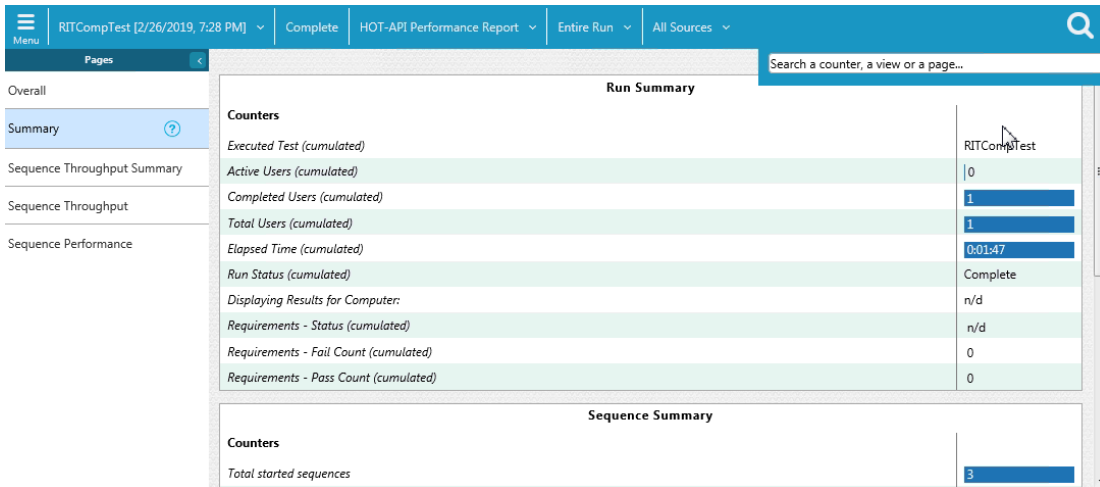
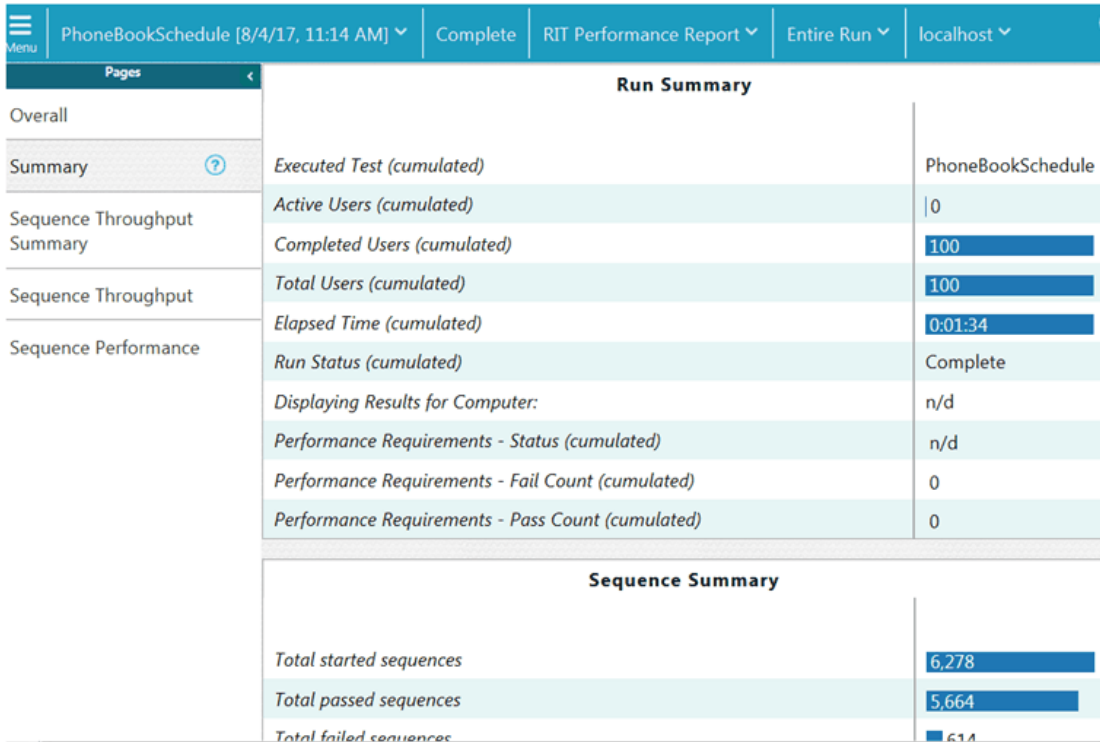


SUMMARY

This page displays information about the run:

- The number of users that are active and the number of users that have completed testing. This number is updated during the run.
- The elapsed time (run duration)
- The status of the run.
- The Performance Requirements: validates the performance requirements that you set in a schedule.

The summary page also summarizes the data about the sequence run and timed sections: Total number of sequences that were started and the total number that have been completed, or failed, with a minimum and maximum rate.

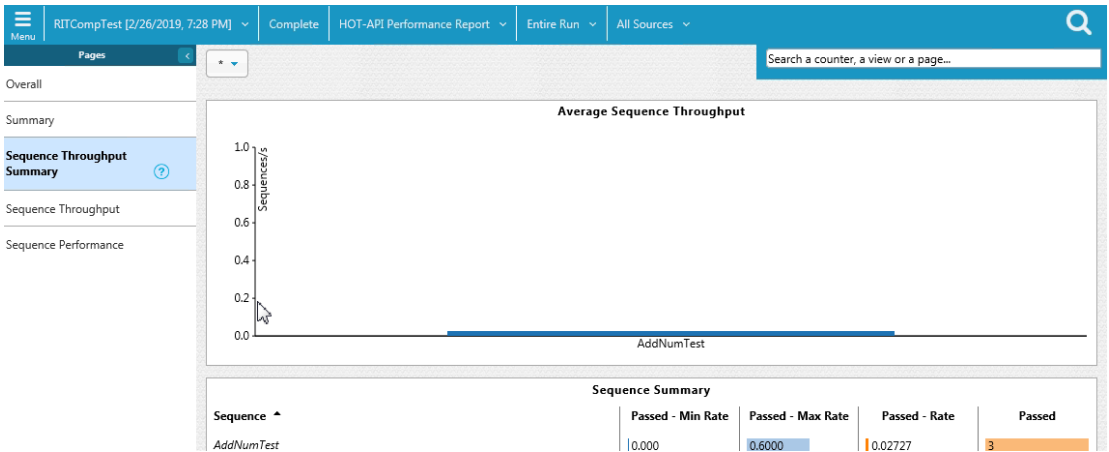
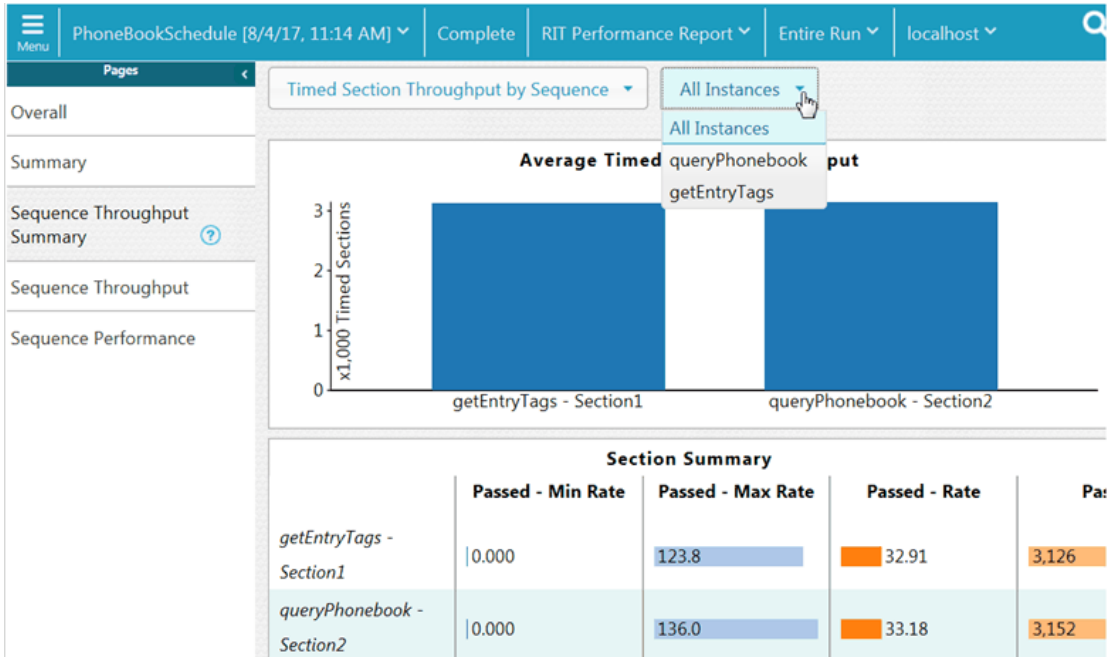


SEQUENCE THROUGHPUT SUMMARY

This page shows a bar chart of average throughput in seconds for each sequence.

It provides times section throughput for each sequence. You can use the filters to have displayed results for one sequence only.

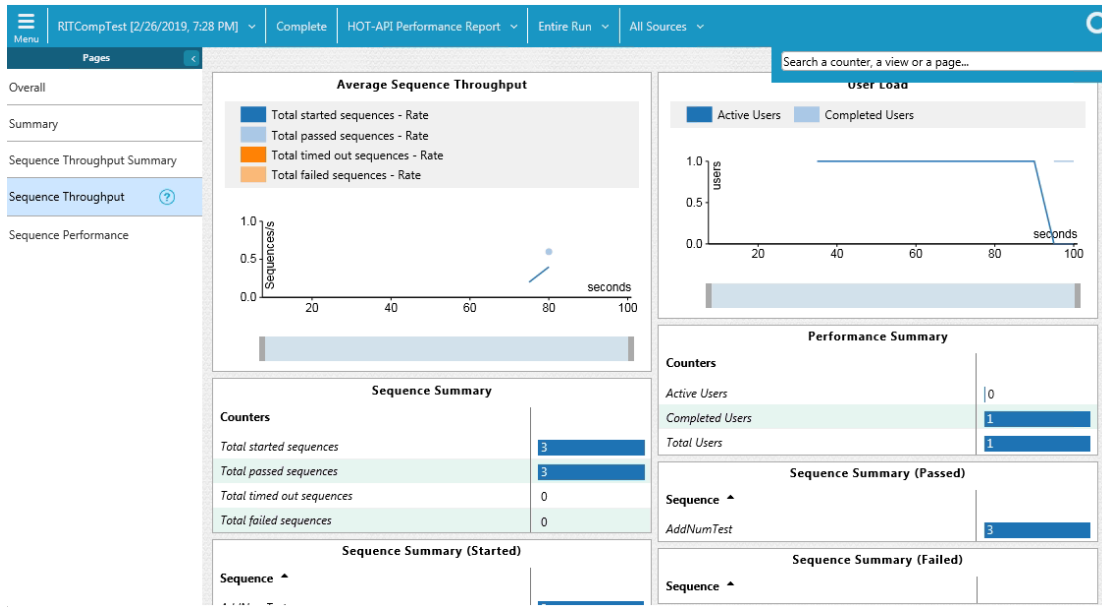
And if you add filters, you can see the timed sections throughput for each sequence.



SEQUENCE THROUGHPUT

This page shows the average throughput for all combined sequences during the last recorded interval over the time period. Click on one of the Total rate boxed to have one graph displayed at a time or click Select All to see all rates on the same graph.

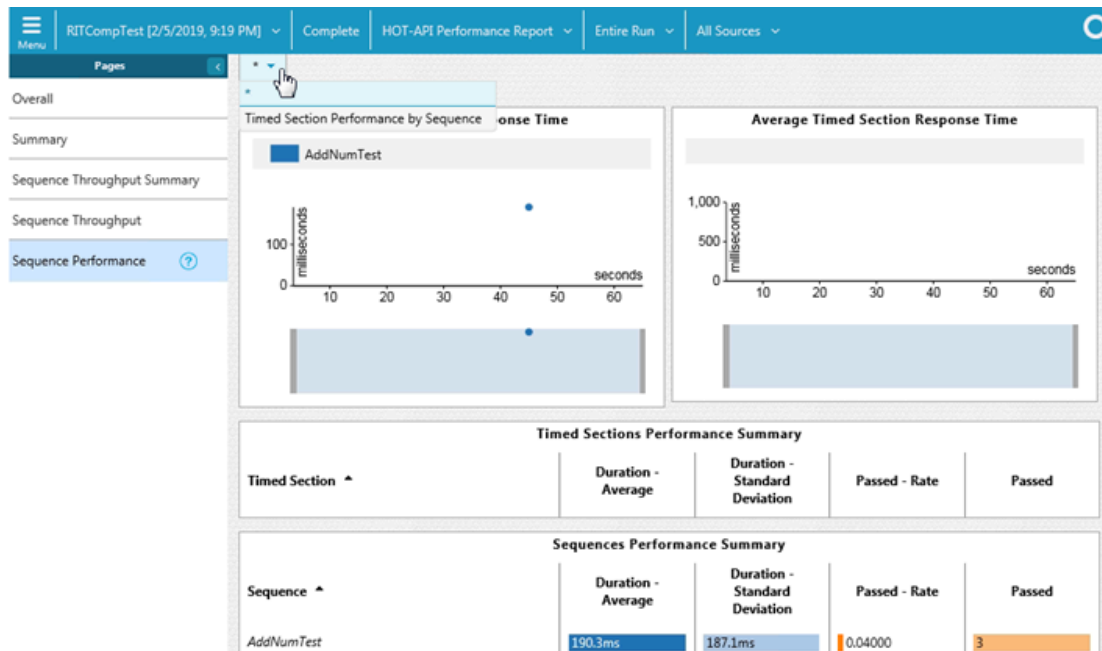
The User Load graph shows active users compared to users that have completed testing. The table after the graph lists the number of active users, the number of users that have completed testing, and the total user count for the entire run.



SEQUENCE PERFORMANCE

The chart displays the response time of the sequences as the test progresses. You can apply a filter to see the timed sections of all sequences or the timed sections of a particular sequence.

The table under the chart shows the average duration for each section of a sequence, the standard deviation of the average response time, the passed rate, and the number of passed sequences.



Integration of Jaeger with the product

Jaeger is software for tracing transactions between distributed services. You can use Jaeger to monitor and troubleshoot complex microservices environments.

You can set up the Jaeger UI in your local environment by using one of the following methods:

- One Jaeger agent shared by all HCL OneTest™ Performance agents
- One Jaeger agent for each HCL OneTest™ Performance agent

One Jaeger agent shared by all HCL OneTest™ Performance agents

When you use this method, ensure that the Jaeger agent is accessible by HCL OneTest™ Performance and all the HCL OneTest™ Performance agents. You must set the `JAEGER_AGENT_HOST` property as an environment variable by using the command line before running the schedule.

You must also ensure that the Jaeger agent ports 6831, 6832, and 5778 are accessible from other computers to communicate with the HCL OneTest™ Performance agent via the User Datagram Protocol (UDP). If you want to define any other Jaeger environment variables, set those environment variables only on HCL OneTest™ Performance.

One Jaeger agent for each HCL OneTest™ Performance agent

When you use this method, you must install the Jaeger agent in the same location where you installed HCL OneTest™ Performance and on all the HCL OneTest™ Performance agents.

You must also ensure that the Jaeger agent ports 6831, 6832, and 5778 are accessible from other processes on the same computer to communicate with the HCL OneTest™ Performance agent via the UDP. If you want to define any other Jaeger environment variables, set those environment variables on all the computers where the Jaeger agent is installed.

For more information about Jaeger, refer to [Jaeger documentation](#).

Related information

[Viewing test logs in Jaeger on page 167](#)

[Running a test or schedule from a command line on page 627](#)

Viewing test logs in Jaeger

You can use the Jaeger UI to view the test logs of the *tests* or *schedules* that you run from the command-line interface to analyze traces of transactions between distributed services.

Before you begin

You must have completed the following tasks:

- Downloaded Jaeger components from the [Jaeger](#) website.
- Created a test or schedule to run it from the command-line interface. See [Creating tests on page 180](#) or [Creating a VU Schedule on page 524](#).

About this task

While running *tests* or *schedules* by using the command-line interface, you must include the command **-history jaeger** in your *test* or *schedule* run. Adding the **-history jaeger** enables you to view the test log of the completed *test* or *schedule* from the Jaeger UI in a web browser.

1. Run a *test* or *schedule* from the command-line interface by adding the **-history jaeger** option.
For example: **cmdline.bat -workspace workspace_full_path -project proj_rel_path -suite suite_rel_path -stdout -history jaeger**

Result

The *test* or *schedule* runs and the result of the run is displayed.

2. Open the Jaeger UI in a browser.
For example: `http://<host IP>:<port>`.
3. Select **HCL OneTest Product** from the **Service** list.
4. Click **Find Traces**.

In the Jaeger UI, you can view the entire test log of the *test* or *schedule* that you ran from the command-line.

What to do next

- You can use the Jaeger traces to analyze test results.
- You can compare the traces in the Jaeger UI with test logs in HCL OneTest™ Performance to confirm that they are the same.

Related information

[Running a test or schedule from a command line on page 627](#)

[Viewing test logs on page 799](#)

[Integration of Jaeger with the product on page 167](#)

Testing with Jenkins

You can use the HCL OneTest™ Performance Jenkins plugin for performance testing to run tests on a Jenkins server by using a Jenkins build step.

Introduction

To automate testing with Jenkins, you must configure two computers: A Jenkins master and a Jenkins slave. This master-slave configuration allows a single Jenkins installation on the master computer to host multiple slave environments for building and running tests. You must install the latest version of the HCL OneTest™ Performance

Jenkins plugin on the master computer, and install the products themselves on the slave computer, where you create the tests. For detailed information about the Jenkins master-slave relationship, see the [Distributed Builds](#) section on the Jenkins site.



Note: You can download the HCL OneTest™ Performance Jenkins plugin from the [HCL® License & Delivery portal](#).

For more information about specific versions of plugin, see [Integration plugin compatibility matrix on page 114](#).

You can then install the plugin on the Jenkins server. After you create the tests in HCL OneTest™ Performance, you can run the tests on the Jenkins server by using this plugin. If you have earlier version of the HCL OneTest™ Performance Jenkins plugin, you must uninstall it before installing the latest version of the plugin on the Jenkins server.

Before you begin

You must have completed the following tasks:

- Installed Installation Manager, which is required for installing the products.
- Installed the products themselves.
- Verified that you have a Jenkins master computer where Jenkins and the HCL OneTest™ Performance Jenkins plugin is installed.
- Verified that you have a Jenkins slave computer where the products are installed.
- Verified that you have a test residing within an Eclipse workspace on the Jenkins slave where the products are installed.



Note:

To run performance test on Mac OS, you must add an environment variable that points to the installation directory of the product, for example, `export TEST_WORKBENCH_HOME=/opt/HCL/HCLOneTest`. For Windows™ and Linux®, this environment variable is set when you install the product.

Installing the HCL OneTest™ Performance Jenkins plugin on the Jenkins master computer

1. Download the HCL OneTest™ Performance Jenkins plugin 7.0 from the [HCL® License & Delivery portal](#).
2. From the Jenkins dashboard, perform the following tasks:
 - a. Click **Manage Jenkins > Manage Plugins**.
 - b. Click **Advanced**.
 - c. From the **Upload Plugin** section, click **Choose File** to locate and **Open** the HCL OneTest™ Performance Jenkins plugin.
 - d. Click **Upload**.

The HCL OneTest™ Performance Jenkins plugin is displayed in the **Installed** tab.

3. Configure **Global Security** to allow *Random TCP Ports* for Java™ Network Launch Protocol (JNLP) agents.


Job Configuration



1. Create a new Jenkins free-style software project. For more information to create a free-style Jenkins project, see [Building a software project in Jenkins](#).
2. Optionally, set an environment variable in **Manage Jenkins > Configure System > Global properties** from the Jenkins dashboard.

Select the **Environment variables** check box if it is not selected. Add the variable and save the details.

3. From the Jenkins dashboard, perform the following tasks:
 - a. Open the Jenkins free-style software project, and then click **Configure**.
 - b. Click **Build > Add build step**.
 - c. Click **Run HCL OneTest™ Performance test**.
4. Provide the details about the test run as shown. The following table explains each field.

Field	Description
Name	Required. The name of the test.
Work-space	Required. The complete path to the Eclipse workspace.
Project	Required. The path, including the file name of the project relative to the workspace.
Test Suite Name	Required. The path, including the file name of the test to run related to the project.
IMShared Location	Optional. The complete path to HCLIMShared location,if it is not the default location.
Var File	Optional. The complete path to the XML file that contains the variable name and value pairs.
Config File	Optional. The complete path to a file that contains the parameters for a test or schedule run.
Results File	Optional. The name of the results file. The default result file is the test or schedule name with a time stamp appended. The results file is stored in the Results directory. If you are running multiple tests, do not provide a name for the results file.
Overwrite Results File	Optional. Determines whether a results file with the same name is overwritten. The default value is true, which means the results file can be overwritten.
Quiet	Optional. Turns off any message output from the launcher and returns to the command shell when the run or the attempt is complete.

Field	Description
Number of Virtual Users	Optional. For a schedule, the default value is the number of users specified in the schedule editor. For a test, the default value is one user. Overrides the default number of users, if required.
VM Args	Optional. Java™ virtual machine arguments to pass in.
Dataset Override	<p>Optional. For a test or schedule, the default value is the dataset specified in the test editor or schedule editor. Overrides the default dataset value to run if required.</p> <p> Note:</p> <p>You must use the <code>Dataset Override</code> option to replace the dataset values during a test or schedule run. You must ensure that both original and new datasets are in the same workspace and have the same column names. You must also include the path to the dataset.</p> <p>For example,</p> <pre data-bbox="483 894 1398 919">/project_name/ds_path/ds_filename.csv:/project_name/ds_path/new_ds_filename.csv.</pre> <p>You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon (<code>:</code>).</p> <p>For example,</p> <pre data-bbox="483 1119 1390 1220">/project_name1/ds_path/ds_filename.csv:/project_name1/ds_path/new_ds_filename.csv;/project_name2/ds_path/ds_filename.csv:/project_name2/ds_path/new_ds_filename.csv</pre>
Exported Statistical Report Data File	Optional. The complete path to a directory to store exported statistical report data.
Custom Report Format Files	Optional. A comma-separated list of absolute paths to custom report format files (.view files) to use when exporting statistical report data with the Export Statistical Report Data File option.
Exported Statistical Report in html	Optional. The complete path to a directory to export web analytic results. Analyze the results on a web browser without using the test workbench. If you run multiple tests, do not provide a value in this field. The web analytic results will be exported to Jenkins workspace.
Resource Monitor-	Optional. For a schedule (Rate schedule or VU schedule), use Resource Monitoring Labels Override to perform any of the following actions:

Field	Description
ing Labels Override	<ul style="list-style-type: none"> ◦ To enable the Resource Monitoring from Service option for a performance schedule if the Resource Monitoring from Service option is not enabled from the schedule editor in HCL OneTest™ Performance. ◦ To ignore Resource Monitoring sources that were set in the performance schedule and to change for a label matching mode. ◦ To replace an existing set of Resource Monitoring labels that were set in the performance schedule and run the schedule with a new set of Resource Monitoring labels. <p> Note: You can add multiple Resource Monitoring labels separated by a <code>comma</code>.</p> <p> Important: You must add the Resource Monitoring labels to the Resource Monitoring sources on the Resource Monitoring page in your HCL OneTest™ Server project.</p>
User Comments	Optional. Add text within the double quotation mark to display it in the User Comments row of the report.

If you do not supply a value for `Exported Statistical Report Data File`, these logs will be saved in Jenkins workspace/temp directory.

5. Click **Save**.
6. Optionally, to run multiple tests under the same job, click **Add build step** again, and provide details for the next test.

Environment variable configuration

After you set the environment variable in **Manage Jenkins > Configure System > Global properties** on the Jenkins server, you can enter the variable name by using any of the following methods for the corresponding text fields in the **Run HCL OneTest™ Performance test** step:

- Use the dollar sign (\$) followed by the variable name.

For example, `$workspace`

- Use the dollar sign (\$) followed by the variable name between braces.

For example, `${workspace}`

You can then run the Jenkins build by using environment variables in the **Run HCL OneTest™ Performance test** step. The environment variable that you added to your test is substituted with the actual value associated with the variable that you set in **Manage Jenkins > Configure System > Global properties** for the job. The HCL OneTest™ Performance Jenkins plugin uses the actual value while running the job.

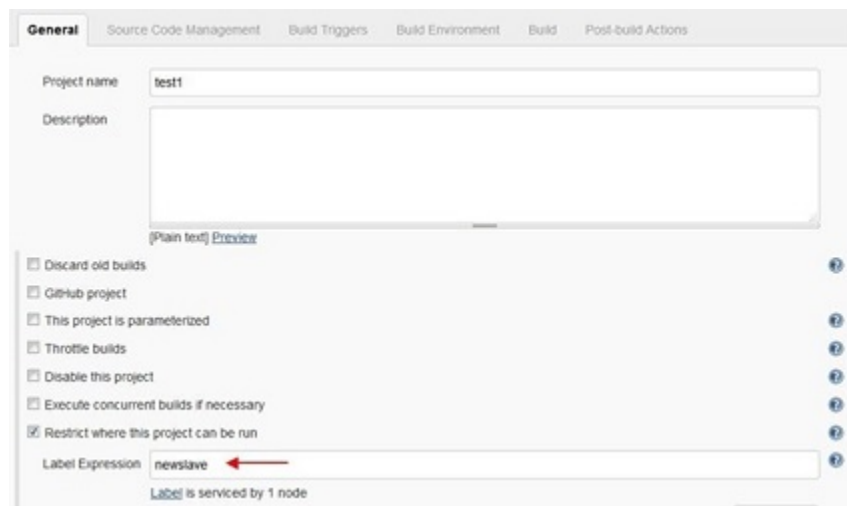
For example,

If you set the environment variable named `workspace` with the value `C:\Users\HCL\workspace1` in **Manage Jenkins > Configure System > Global properties**, then you can use `workspace` or `workspace` as input to the corresponding text field of your choice for the **Run HCL OneTest™ Performance test** step in the Jenkins build step. When you run the build, `workspace` or `workspace` is substituted with its corresponding value `C:\Users\HCL\workspace1`.

Master/Slave Configuration

Master and slave configurations are supported by HCL OneTest™ Performance Jenkins plugin . Refer to [Distributed builds](#) for more information.

While creating the job configuration, in addition to the preceding steps, you must select the **Restrict where this project can be run** check box and provide the name of the slave node in the **Label Expression** field. This is the location where the products are installed and where tests can be run.



Running tests

After you save the project, you must open the project, and then click **Build Now**. This action starts the test run on the slave computer. You must specify the relative path from the project to the test including the file name of the test. To run multiple tests from different projects, you must add new build step for each project.

Building result and logs

1. After the build completes, click the build number and open the console for the project. Locate the `Test Result` to check the test execution status.



Note: If you add multiple build steps to run multiple tests, multiple `Test Result` instances are displayed.

In HCL OneTest™ Performance, if the HCL OneTest™ Server URL is configured in **Window > Preferences > Test >** and **Publish result after execution** is set as **Always** in **Window > Preferences > Test > > Results**, then the **Reports information** section on the **Console** page displays the names of the report along with its corresponding URLs. The report URLs are the HCL OneTest™ Server URLs where the reports are stored. You can access the report URLs to view the test execution information at any point of time.

2. For product logs, log in to the slave computer and search in the Jenkins slave workspace/temp directory.

Testing with Maven

Starting from V9.2.0, you can use the Maven plug-in that is provided with the testing product to run tests as part of your Maven build. Apache Maven is a software build tool based on the concept of a project object model (POM).

Before you begin

- You must have installed the testing product from V9.2.0 and set an environment variable that points to the installation location.

For Mac OS, add an environment variable that points to the installation directory of the product: `export TEST_WORKBENCH_HOME=/opt/HCL/HCLOneTest`

For Windows™ and Linux®, this environment variable is set when you install the product.

- You must have installed Maven from V3.2.0 and set up an environment variable that points to the `M2_HOME` installation directory.

Introduction

To automate testing with Maven, you must configure a `pom.xml` file and launch your tests from the command line using Maven command. You can either use your own `pom.xml` file, or one that is delivered with the product.

Three files are delivered with the product installation in the `<product install location>\HCLOneTest\maven2\` folder:

- `pomCustomSurefireSample.xml` for Windows, Linux and macOS.
- `pomMojoExecPluginSample_Linux.xml` for Linux and MacOS.
- `pomMojoExecPluginSample_Windows.xml` for Windows.

The files contain all types of dependencies as well as arguments required to execute the test scripts. There are two methods to run tests with Maven.

Method 1

With this method, you can run one or several tests. If you use your own pom.xml file, edit it with the following lines and indicate which test(s) must be executed, otherwise, use the pomCustomSurefireSample.xml file as follows:

- Copy the pomCustomSurefireSample.xml to a directory.
- Edit the file and update the lines, enter the name and location of the test(s) that must be run. If the product is installed on a different drive or a different location, or if IBMIMSharedHCLIMShared location has been changed, enter the correct path to the HCLIMShared plug-in folder. For aftsuite attribute, you can input aft xml file as the parameter value.

```
<!--test suite="testSources/Test1.testsuite"/-->
<!--test suite="Test2.testsuite"/-->
<test suite="C:/Runtimes/runtime-RptMvn/AA/testSources/Test2.testsuite" plugins="C:/Program
Files/HCL/HCLIMShared/plugins"/>
<!--test    schedule="Schedule.testsuite" project="AA" workspace="C:/Runtimes/runtime-RptMvn"
plugins="C:/Program Files/HCL/HCLIMShared/plugins"/-->
<!--test    suite="Test2.testsuite" project="AA" workspace="C:/Runtimes/runtime-RptMvn"/-->
<!--test aftsuite="Test1.xml" project="AA" workspace="C:/Runtimes/runtime-RptMvn"
plugins="C:/Program Files/IBM/IBMIMShared/plugins"/-->
```

- Run Maven to update the pom file version command and use the plug-in version currently available on delivered repositories.

```
mvn versions:update-properties -Dincludes=com.hcl.products.test.it -f pomCustomSurefireSample.xml
```

- Run the test(s).

```
mvn clean verify -f pomCustomSurefireSample.xml
```

Fail safe reports are generated in the target directory, especially in target/failsafe-reports/ <ProjectName>/<TestName>_<timestamp>.txt that will contain the screen capture of the execution.

In HCL OneTest™ Performance, if the HCL OneTest™ Server URL is configured in **Window > Preferences > Test >** and **Publish result after execution** is set as **Always** in **Window > Preferences > Test > > Results**, then the **Reports information** section on the **Console** page displays the names of the report along with its corresponding URLs. The report URLs are the HCL OneTest™ Server URLs where the reports are stored. You can access the report URLs to view the test execution information at any point of time.

Method 2

With this method, no Maven report is generated. If you use your own pom.xml file, copy the following lines and provide your parameter values. Otherwise, you can use the pomMojoExecPluginSample_Linux.xml or pomMojoExecPluginSample_Windows.xml sample file. Example with the pomMojoExecPluginSample_Windows.xml sample file:

- Copy `pomMojoExecPluginSample_Windows.xml` to a directory.
- Edit the file and update the arguments to reflect which test to execute. If the product is installed on a different drive or a different location, or if `IBMIMShared` location has been changed, update the two last lines with the path to the `IBMIMShared` plug-in folder.

```
<argument>/C</argument>
    <argument>${pt-plugin-cmdline}</argument>
    <argument>-workspace</argument>
    <argument>C:\Runtimes\runtime-RptMvn</argument>
    <argument>-project</argument>
    <argument>AA</argument>
    <argument>-suite</argument>
    <argument>Test1.testsuite</argument>
    <argument>-plugins</argument>
    <argument>C:/Program Files/HCL/HCLIMShared/plugins</argument>
```

- In the argument tags, instead of the `-suite` option, you can use the `-aftsuite` option and input the aft xml file as the parameter value in the subsequent argument tag to run the AFT test. For example, in the preceding template, `<argument>-suite</argument> <argument>Test1.testsuite</argument>` can be replaced with `<argument>-aftsuite</argument> <argument>aftfile.xml</argument>`.
- Run the test.

For Windows:

```
mvn clean verify -f pomMojoExecPluginSample_Windows.xml
```

For Linux or MacOS:

```
mvn clean verify -f pomMojoExecPluginSample_Linux.xml
```

Related information

<https://maven.apache.org/index.html>

Integrating and running performance test scripts in Micro Focus ALM

To obtain test result details, you must integrate and run performance test scripts in Micro Focus Application Lifecycle Management (ALM) by using a readily available template available in HCL OneTest™ Performance installation directory.

About this task

The performance test template is available in the installation directory of HCL OneTest™ Performance. You must copy the contents of the template to a new VAPI-XP VBScript test script in Micro Focus ALM, add your test script details into the VAPI-XP VBScript test script, and then run the test script.



Note: You can use Micro Focus ALM client only on Microsoft™ Internet Explorer. For more information, see [Micro Focus ALM system requirements](#).

1. Navigate to the directory `HCL\HCLOneTest\alm` in HCL OneTest™ Performance installation directory.

You can use `PT_ALM_Windows.txt` file for performance test scripts.

2. Copy the contents of the template.
3. Log in to the **Micro Focus ALM** portal, if you are not already logged in.

Result

The Micro Focus ALM dashboard is displayed.

4. From Micro Focus ALM, create a new integration test by performing the following actions:
 - a. Expand **Testing** from the left pane and then click **Test Plan**.
 - b. Expand **Integration**.
 - c. Right-click **Integration** and click **New Test** to create a new test.
 - d. Enter a test name in the **Test Name** field.
 - e. Select **VAPI-XP-TEST** as test type from the **Type** drop-down list.
 - f. Click **OK**.

The **VAPI-XP Wizard** is displayed.

- g. Select a test script language by performing the following steps:
 - i. Select **VBScript** from the **Script Language** drop-down list.
 - ii. Enter a script name (for example, script) in the **Script Name** field.
 - iii. Click **Next** and select a test type if required.




Note: The **COM/DCOM Server Test** test type is already selected as a test type.



- iv. Click **Finish**.

The test is created.

5. Click the **Test Script** tab.
6. Paste the content of the template that you copied in [step 2 on page 177](#) to the test script.
7. Enter the test details in the VAPI-XP Vbscript test script by referring to the following table:

Parameter	Description
Workspace	Required. The complete path to the Eclipse workspace.
Project	Required. The path, including the file name of the project relative to the workspace.
TestsuiteName	Required. The path, including the file name of the test to run relative to the project.
IMSharedLocation	Optional. The complete path to HCLIMShared location.

Parameter	Description
Varfile	Optional. The complete path to the XML file that contains the variable name and value pairs.
Configfile	Optional. The complete path to a file that contains the parameters for a test or schedule run.
ResultsFile	Optional. The name of the result file. The default result file is the test or schedule name with a time stamp appended.
Overwrite-ResultsFile	Optional. Determines whether a results file with the same name is overwritten. The default value is true, which means the results file can be overwritten.
Quiet	Optional. Turns off any message output from the launcher and returns to the command shell when the run or the attempt is complete.
Users	Optional. For a schedule, the default value is the number of users specified in the schedule editor. For a test, the default value is one user. Overrides the default number of users, if required.
VMArgs	Optional. You can use this parameter to pass Java™ virtual machine arguments.
Dataset-Override	<p>Optional. For a test or schedule, the default value is the dataset specified in the test editor or schedule editor. Overrides the default dataset value to run if required.</p> <p> Note:</p> <p>You must use the <code>Dataset Override</code> option to replace the dataset values during a test or schedule run. You must ensure that both original and new datasets are in the same workspace and have the same column names. You must also include the path to the dataset.</p> <p>For example,</p> <pre>/project_name/ds_path/ds_filename.csv:/project_name/ds_path/new_ds_filename.csv.</pre> <p>You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon (;).</p> <p>For example,</p> <pre>/project_name1/ds_path/ds_filename.csv:/project_name1/ds_path/new_ds_file- name.csv;/project_name2/ds_path/ds_filename.csv:/project_name2/ds_path/new_ds_- filename.csv</pre>
Resource-MonitoringLabels-Override	Optional. For a schedule (Rate schedule or VU schedule), use ResourceMonitoringLabelsOverride to perform any of the following actions:

Parameter	Description
	<ul style="list-style-type: none"> ◦ To enable the Resource Monitoring from Service option for a performance schedule if the Resource Monitoring from Service option is not enabled from the schedule editor in HCL OneTest™ Performance. ◦ To ignore Resource Monitoring sources that were set in the performance schedule and to change for a label matching mode. ◦ To replace an existing set of Resource Monitoring labels that were set in the performance schedule and run the schedule with a new set of Resource Monitoring labels. <p> Note: You can add multiple Resource Monitoring labels separated by a comma.</p> <p> Important: You must add the Resource Monitoring labels to the Resource Monitoring sources on the Resource Monitoring page in your HCL OneTest™ Server project.</p>
ExportStatsFile	Optional. The complete path to a directory that you can use to store exported statistical report data.
ExportStatReportlist	Optional. A comma-separated list of absolute paths to custom report format files (.view files) that you can use to export statistical report data with ExportStatsFile.
ExportStatsHtml	Optional. The complete path to a directory that you can use to export web analytic results. The results are exported to the specified directory. You can analyze the results on a web browser without using the test workbench.
UserComments	Optional. You can add text within the double quotation mark to display it in the User Comments row of the report.

8. Run the VAPI-XP Vbscript test script.

Results

You have integrated and run performance test scripts in Micro Focus ALM.

What to do next

The test result details are displayed in the **Output** window of Micro Focus ALM.

In HCL OneTest™ Performance, if the HCL OneTest™ Server URL is configured in **Window > Preferences > Test > Publish result after execution** is set as **Always** in **Window > Preferences > Test > Results**, then the **Reports information** section on the **Output** window displays the names of the report along with its corresponding URLs. The report URLs are the HCL OneTest™ Server URLs where the reports are stored. You can access the report URLs to view the test execution information at any point of time.

Chapter 7. Test Author Guide

This guide describes how to create test scripts in HCL OneTest™ Performance and enhances tests by applying different test elements such as dataset, variables, and verification points. This guide is intended for testers.

Creating tests

To create a test, you record representative interactions with an application.

About this task

After you record a test, you can play it back to confirm that the recorded actions do what you expect.



Note: When you record a test that includes a file download, the file is not physically saved to disk. However, you can confirm that the file was retrieved from the server by looking in the response of the request that asked for the file. One method to locate the request for large downloaded files is to look for a request with a large response size.

1. Click the **test name** in test editor.
2. Click **Select**.
3. Select **HTTP Request**.

Performance testing tips

Use these tips to make HCL OneTest™ Performance run faster and more efficiently.

The following suggestions can help you to get the best performance from HCL OneTest™ Performance:

- **Number of computers.** Have at least two computers for a test. The user interface consumes significant resources; therefore play back a test or schedule on a computer (agent) that is separate from the computer that is running the workbench (UI).
- **Number of virtual users at remote locations.** When you assign a user group to a remote location, do not overload the remote computer (agent). If you exceed the number of virtual users that the remote computer can run, the performance measurements of the server will be skewed because they will be affected by the performance of the computer. The test results will reflect the load of the computer more than the load of the server. For best results on a computer with a 1 GHz processor and 1 GB of RAM, do not exceed 1000 concurrent virtual users.
- **TCP/IP ports.** Your computer must have a sufficient number of TCP/IP ports. On computers with Microsoft Windows™, the typical limit is 5000. Issue the **netstat -a** command to observe port use. If the largest number you see is 5000, then you need to increase the number. To increase it, open the registry. Under `HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/Tcpip/Parameters`, create a new `dWord` named `MaxUserPort`, and set its value up to 65000. Restart the computer.
- **Open file limit for Linux™.** Computers that are running Linux™ need a per-process open file limit higher than 1024. As root, enter `ulimit -n 30000` (or another appropriate value) before starting Agent Controller.

- **Looping within tests.** If you are stress testing a server, your test typically contains a loop. Your connection behavior differs depending upon whether the loop is set at the schedule level or at the test level. Setting a loop at the test, rather than the schedule, level gives you a performance advantage, because the connections are reused during the looping process. For more information, see [Add a loop on page 637](#).
- **Logging levels.** After the test is stable, for maximum performance, reduce the test log level and problem determination log level and sample a small number of users. Increase the statistics sample interval to 30 or 60 seconds for long-running tests.
- **Workbench heap size.** The Java™ Virtual Machine (JVM) heap size on the workbench is based on the available physical memory. Do not run the workbench on a computer with less than 768 MB of physical memory. The maximum workbench heap size depends on your JVM. Although the heap size is not strictly necessary for playback performance, you can increase the workbench heap size. To increase the heap size, set the `-Xmx` parameter in the `eclipse.ini` file, which is located in the product installation directory. For Windows™, if your physical memory is 3 GB or more, then the maximum heap size must not exceed 1200 MB. For Linux™, the maximum heap size is approximately 3000 MB. If the workbench is sluggish or fails to start after you increase the heap size, reset the heap size to the default by removing the `VMARGS=-Xmx` line from the `eclipse.ini` file.
- **Location (agent) heap size.** To access maximum heap, after one successful test of any size, search for a location (agent) attribute called `RPT_DEFAULT_MEMORY_SIZE`. If you cannot find this attribute, you can specify a maximum heap by creating a new attribute: `RPT_VMARGS=-Xmx1500m` (for example, max heap 1.5 GB). For more information, see [Increasing memory allocation on page 638](#).

HCL OneTest™ Performance sets heap size for `RPT_DEFAULT_MEMORY_SIZE` based on the bit-type of the JRE:

- For 32-bit Java Runtime Environment (JREs), HCL OneTest™ Performance sets 70% of the size of physical memory to `RPT_DEFAULT_MEMORY_SIZE`. Typically, the maximum limit is set to 1200m.
 - For 64-bit JREs, some workloads might perform better with a lesser heap size than 70% of physical memory up to a maximum of 12000m.
- **Disk space.** Verify that there is sufficient free disk space on the workbench and agent computers. Also, verify that there is sufficient free disk space on the drive that contains the system temporary directory.
 - **Recording length.** If you record for a relatively long time, test generation also takes a long time. If test generation is taking a relatively long time, try shorter recording scenarios.

Creating a project

The tests that you create, and the assets associated with the tests, reside in a project on your desktop. You can create the project separately, or you can simply record a test, which automatically creates a project named `testproj`.

1. Select **File > New > Performance Test Project**.

Result

The **Create a Project** window opens.

2. In the **Project Name** field, type a name for the project.
If you plan to collect response time breakdown data, do not use a project name that contains spaces.
3. Select **Use default location**.

4. Optional: Click **Next** and select the folders to create in the new project. These folders organize your files by asset (Tests, Results, and so on).
5. Click **Finish**.

Result

After you click finish, you are prompted to record a test. You can create a test from a new recording or from an existing recording, or just click **Cancel** to create a test project without recording a test.

Recording HTTP tests

When you record a test, the test creation wizard records your interactions with a web-based application, generates a test from the recording, and opens the test for editing. You can record tests from Internet Explorer (which is the default on Windows™) or from another browser.

Recording reliable HTTP tests

You use a web browser to capture the HTTP test. To record reliable HTTP tests, certain configuration of the web browsers are required. Read the guidelines in this topic for robust HTTP tests.

Remove temporary files

To ensure that your recording accurately captures HTTP traffic, remove temporary files from the web browser's cache before you record a test.

To remove temporary files from the Microsoft Internet Explorer cache:

1. Open Internet Explorer, and click **Tools > Options**.
2. On the **General** tab, click **Delete**.
3. In the **Delete Browsing History** window, click **Delete**, and then click **OK**.

To remove temporary files from the Mozilla FireFox cache:

1. Open Mozilla Firefox and click **Tools > Options**.
2. Click the **Advanced** tab and then the **Network** tab.
3. Click **Clear Now**.
4. Click **OK**.

Allow recorded pages to load completely

When recording, wait for each page to load completely. This wait does not affect performance results, because you can remove extra think time when you play back the test.

Recording with IP addresses

When recording, you can enter the IP address of the website to connect to instead of entering the host name. When you play back the test, however, make sure that the IP address can resolve to a host name. Typically, this resolution occurs via a reverse DNS lookup.



Note: While recording a test in Internet Explorer, if you use `localhost` in the URL, Internet Explorer does not route the traffic through HCL OneTest™ Performance Agent. The test will not contain any recorded data. You must use a host name or IP address to record the traffic.

To resolve the IP address, map each IP address to a host name in the `/etc/hosts` file. On a Windows system, the file is typically stored in `C:\windows\system32\drivers\etc\hosts`.

If an address cannot be resolved, your test shows a slower connection time. This time might be delayed by as much as 8 seconds, depending on the network configuration of the computer.

Configuring Internet Explorer for recording from a secure web site

You can suppress the security warning that Internet Explorer displays when you are recording from a secure web site.

About this task

HCL OneTest™ Performance uses a proxy recording program that intercepts all traffic between the browser and the web server. During recording at a secure web site (with a web address that starts with `https://`), by default you see a security warning before every action and must confirm your acceptance of a security risk to continue.

If you do nothing, this warning is displayed with every browser action, and you must repeatedly click **Yes** to continue. Performing the following procedure installs the recorder certificate on the local host as a trusted authority and thereby disables warnings from Internet Explorer during recording at secure web sites. This procedure does not affect other browsers that record from secure web sites—they will display warnings before every action.

To disable security warnings when using Internet Explorer to record from a secure web site:

1. During test recording, the first time the warning is displayed, click **View Certificate**.
2. In the **Certificate** window, click **Install Certificate**.
3. In the **Certificate Import Wizard** window, click **Next**.
4. On the second page of the wizard, click **Next**.
5. On the last page of the wizard, click **Finish**.
6. In the **Root Certificate Store** confirmation window, click **Yes**.
7. In the window that reports that the import was successful, click **OK**.
8. In the **Certificate** window, click **OK**.
9. In the **Security Alert** window, click **OK**.

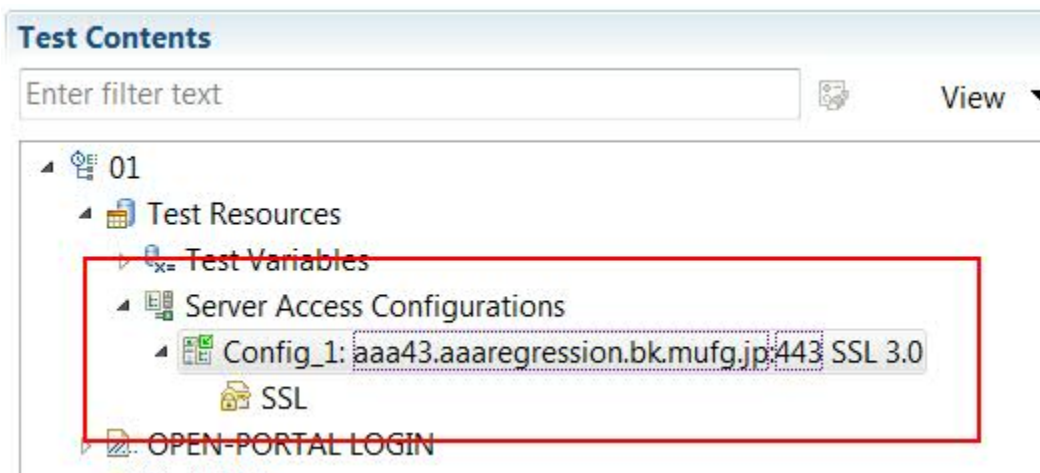
Recording an HTTP test

To test the performance of an application, you must first record the HTTP traffic that traverses between the client and the server. You record the HTTP traffic of the application by initiating the recording from the product. When you record the test, a proxy recorder intercepts the HTTP traffic between the browser and the web server.

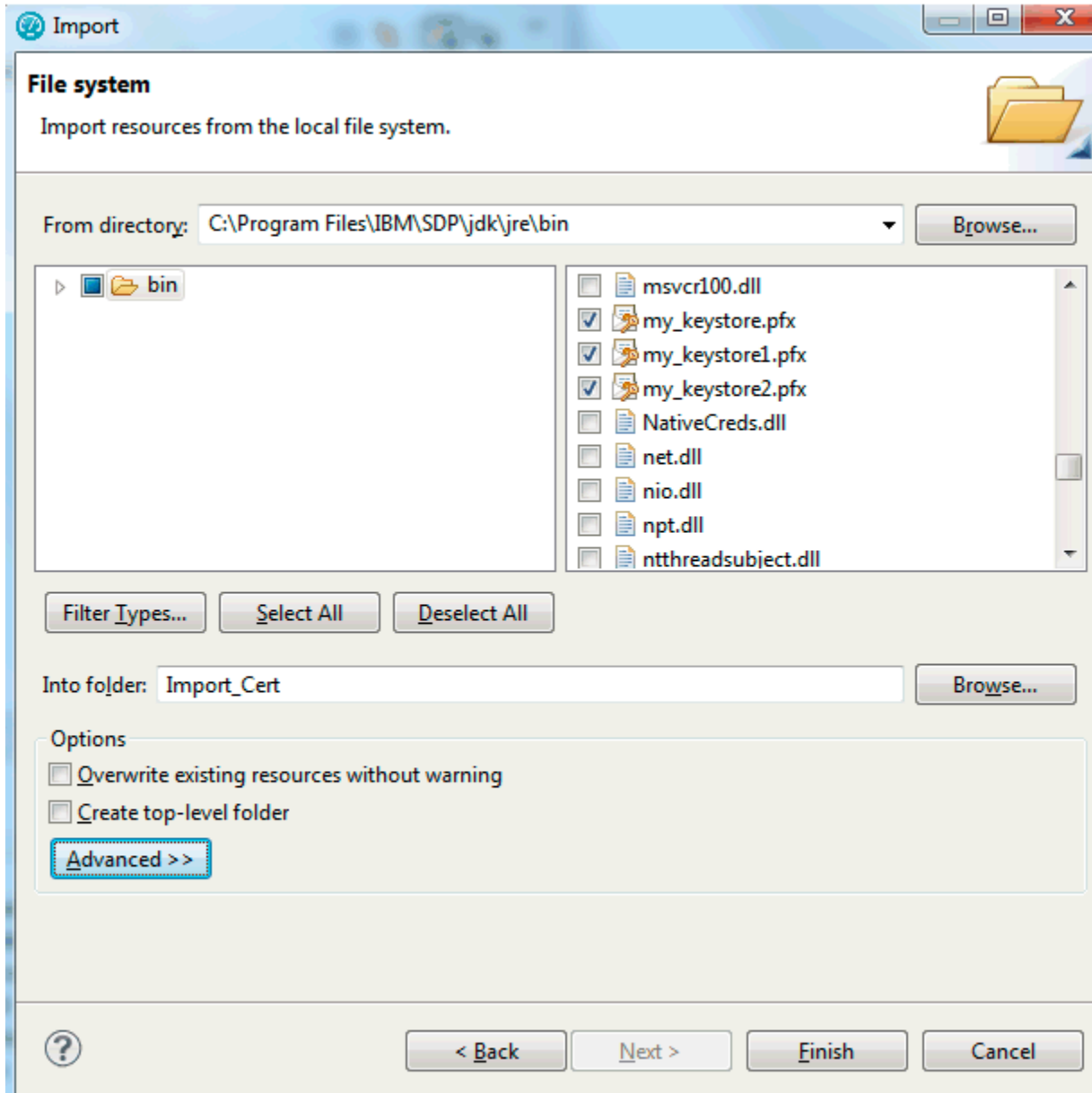
Before you begin

Certain websites require appropriate certificates to use a proxy recorder to record the site. The recorder certificate is required to record all the secured sites. The client certificate is different and it serves as an additional layer of security that is required by the web server to authenticate the client/browser. If some applications use Secure Sockets Layer (SSL), the proxy recorder can cause authentication problems because SSL relays traffic between the client and the server. Depending on the authentication method in place, the client might require the proxy recorder to authenticate itself as the server, and the server might require the proxy recorder to authenticate as the client. If the client program requires an authenticated server, you must either have access to the server certificate keystore and provide it to the proxy recorder or configure the client to accept the default certificate from the proxy recorder instead of the certificate from the actual server.

If you have recorded a test that does not use SSL, you can convert that test to be secure by adding an SSL object to the corresponding Server Access Configuration in the test.



To record an application that requires a client-side certificate, import the client certificate to the HCL OneTest™ Performance project. To import the certificate, click **File > Import > General > File System**, and navigate to the folder that contains the certificates and click **Finish**.



About this task


The following recorders are available for recording HTTP traffic from a browser:

- **SOCKS proxy recorder:** Use this recorder when no proxy connections are required.
- **HTTP proxy recorder:** Use this recorder when proxy connections are required to connect to the network or when the client program does not support SOCKS.
- **Socket recorder:** Use this recorder for low-level network traffic when the client does not support proxies.

You can also record and generate a test by using REST APIs. The API documentation to record a test is located at `C:\Program Files\HCL\HCLIMSHARED\plugins\com.ibm.rational.test.lt.server.recorder.jar`.

The API documentation to generate a test after the recording completes is located at `C:\Program Files\HCL\HCLIMSHARED\plugins\com.ibm.rational.test.lt.server.testgen.jar`.

To record an HTTP performance test with a browser:

1. In the Performance Test perspective, on the toolbar, click the **New Test From Recording** icon  or click **File > New > Test From Recording**.
2. In the **New Test From Recording** wizard, click **Create a test from a new recording**, select **HTTP Test**, and click **Next**.

If you are recording sensitive data, click **Recording encryption level** and select the encryption level to record.

3. On the **Select Location** page, select the project and folder to create the test in, type a name for the test, and click **Next**.

If necessary, click the **Create the parent folder** icon  to create a performance test project or folder.

4. On the **Select Client Application** page, select the web browser to use.

The type of application defines the recorder that can be used. The following client application types are supported for recording a service test:

Choose from:

- **Microsoft Internet Explorer:** This option records traffic that is sent and received with Internet Explorer.
 - **Mozilla Firefox:** This option records traffic that is sent and received with Firefox.
 - **Google Chrome:** This option records traffic that is sent and received with Chrome.
 - **Apple Safari:** This option records traffic that is sent and received with Safari.
 - **Opera:** This option records traffic that is sent and received with Opera.
 - **Managed Application:** This option starts a browser that is not on the list.
 - **Unmanaged Application:** This option records HTTP traffic from one or multiple client programs that use a proxy. You must manually start the client programs, and the proxy recorder records all traffic that is sent and received through the specified network port.
5. Depending on your selection in Step 4, perform one of the following steps: On the **Recorder Settings** page, depending on the browser that you selected, specify the following details:
 - a. If you selected a browser, specify the recording method:
 - To record HTTP or SOCKS traffic through a proxy, click **Record traffic with the proxy recorder**. To record low-level network traffic for applications where a proxy cannot be used, click **Record traffic with the socket recorder**. Use this option when you are not using Mozilla Firefox or Microsoft™ Internet Explorer.



Note: When using proxy recording, you can filter out HTTP or HTTPS requests to a specific endpoints so that any requests to those endpoints are not recorded. See [Proxy recording preferences on page 1190](#).

- If HCL OneTest™ UI is installed and shell-shared with HCL OneTest™ Performance, for the Web UI tests, you can use the **Record user actions** option to record the functional aspects of the

application in the same HTTP recording session. Thus, both the functional and performance aspects of the application are recorded.

- If the server requires client SSL authentication, provide the client certificate for the proxy recorder to be authenticated by the server as though the proxy recorder were the client. Select **The server requires a specific client certificate**.

To provide single certificate keystore, specify the file name and password of the server certificate keystore. If multiple certificates are required, click **Multiple certificates**, and click **Add** to specify a certificate keystore file name and password for each host name and port.

- To record a secured site using Internet Explorer or Google Chrome on Windows, install the recorder certificate by selecting **Register the recorder root certificate authority**. Before the recording starts, the browser prompts you to install the certificate. After the recording is stopped, the browser prompts you to uninstall the certificate. To avoid multiple prompts for each recording, select **Keep the recorder root certificate authority after recording**.



Note: If you already had the certificate from a version prior to 9.2.1 and then install the latest version of the product, you might have to install the certificate again.

This option is not available when you record by using the Firefox or Safari browser. To record a secured site on these browsers, manually import the certificate in the browser from the default location `C:\Program Files (x86)__VENDOR_NAME____VENDOR_NAME__IMShared\plugins\com.ibm.rational.test.lt.recorder.proxy_version\SSLCertificate`. For information about how to import the certificates, see the browser's documentation.

- If you selected **Mozilla Firefox**, you can choose to use a temporary Firefox profile. This option starts the Firefox browser without any bookmarks, plug-ins, or toolbars that might be associated with your usual profile. Select **Use an alternate Firefox profile**, and then select **Use a temporary Firefox profile**.
- If you clicked **Record traffic with the proxy recorder**, click **Advanced** to specify whether to use an HTTP or SOCKS proxy recorder to review and edit network connection settings that the browser uses or to specify advanced SSL authentication settings. If you clicked **Record traffic with the socket recorder**, specify the advanced SSL authentication settings.

- b. If you selected **Managed Application**, complete the following steps:
 - i. On the **Recording Method** page, click a recorder to record HTTP traffic and click **Next**.
 - ii. On the **Managed Application Options** page, for **Program path**, click **Browse** to select the program. If necessary, specify the Working directory, and in the **Arguments** field, type the command-line arguments that the program requires. Click **Next**.
 - iii. If the program requires user input from a command-line interface, select the **Open console for user input** check box. Click **Next**.
 - iv. Depending on the recording method that you selected, complete one of the following steps:

- On the **Proxy Recorder Settings** page, specify whether to use an HTTP or SOCKS proxy recorder to review and edit network connection settings that the browser uses or to specify the advanced SSL authentication settings.
 - On the **Socket I/O Recorder Secured Settings** page, specify the advanced SSL authentication settings.
- c. If you selected **Unmanaged Application**, on the **Proxy Recorder Settings** page, specify whether to use an HTTP or SOCKS proxy recorder to review and edit network connection settings that the browser uses or to specify the advanced SSL authentication settings.
- To record an HTTP test from a mobile device, see [Recording an HTTP test for mobile applications on page 190](#).

6. Click **Finish**.

Result

A progress window opens while your browser starts.








7. In the browser address field, type the address of the web application to test, and activate the link.




Note: If you enter the address of a secure website (one that starts with `https:`), your browser might display a security alert. Depending on the security certificate for the site, you might be required to accept a security risk to proceed with the recording.

8. Complete the user tasks to test. While you are recording, adhere to the following guidelines:
- Wait for each page to load completely. This wait does not affect performance results, because you can remove extra waiting time (think time) when you play back the test.
 - Do not change any browser preferences.

You can use the **Recorder Test Annotations** toolbar to add comments, record synchronizations, or take screen captures during the recording.


- To change the page name, click the **Change page name** icon . In the resulting test, the page element in the test editor uses the new name; however, the original name is preserved in the **Page Title Verification Point** area so that page title verification points still work correctly.
- To add a comment to the recorded test, click the **Insert comment** icon . Add a comment when you get a prompt.
- To add a screen capture to the recorded test, click the **Capture screen** icon . The screen and window captures make your tests easier to read and help you visualize the recorded test. You can change the settings for screen captures and add a comment to the image.
- To manually add a synchronization point to the recording, click the **Insert synchronization** icon .
- To manually add a transaction folder to the recording, click the **Start Transaction** icon  and **Stop Transaction** icons  to start and stop the transaction. Transactions can be nested.
- To insert a split point into the recorded test, click the **Split point** icon . With split points, you can generate multiple tests from a single recording, which you can replay in a different order with a

schedule. See [Splitting an HTTP test during recording on page 194](#) for more information about splitting a test.

- To filter packets that display during recording sessions, click the **Filter packets**  icon. You can specify the filter criteria for the following elements:

- SAP packet type
- SAP packet attribute
- Socket packet type
- Socket packet attribute
- Packet type
- Proxy connection IDs
- Citrix events
- A group of conditions
- Remote host

You can add and remove packet filters as needed.

9. After you finish the user tasks in the browser, stop the recorder. You can stop the browser by closing the client program or by clicking the **Stop** icon  in the **Recording Control** view.
10. Select the domains to include in the test and click **Finish**. The domains that are not selected are not included in the test. You can add them back by generating the test again from the recording.

To include all the domains for all of the recordings, click the **Select all and remember my decision** check box.

To enable the filter again for HTTP tests, click **Window > Preferences > Test > Test Generation > HTTP Test Generation**, and, for Service tests, click **Service Test Generation** and then click the **Enable domain review before test generation** check box.

Results

A progress window is displayed while the test is generated. After the test is complete, the **Recording Control** view displays the `Test generation completed` message, the test navigator lists your test, and the test opens in the test editor.

Related information

[Creating tests on page 180](#)

[Recording reliable HTTP tests on page 182](#)

[Recording sensitive session data on page 272](#)

Recording a WebSocket test

If your application is based on the WebSocket protocol, you can check the performance of the application by creating a recording of the application and playing it back.

About this task

WebSocket is not a separate test extension in the product. To test a WebSocket application, you have to record the **HTTP** test. After the recording completes, you must not delete or change the content of the requests, response, and connection objects in the test script.

To record the HTTP test, see [Recording an HTTP test on page 183](#). The requests and responses contain text or binary data. You can do data correlation on the text data.

Starting from 9.1.1.1, the WebSocket test generation automatically does data correlation. See [Correlating request and response data on page 443](#).

Starting from 9.1.1.1, you can specify error handling behavior for WebSocket tests. See [Specifying error handling behavior on page 297](#).

Starting from 9.2, you can add verification points to the requests and responses just as you would do to HTTP tests. See [Verifying expected behavior on page 292](#).

After the test run completes, view the test results by selecting the **HTTP WebSocket Report**. The report displays the number of requests attempted, succeeded, and the rate of success.

Recording an HTTP test for mobile applications


You can use HCL OneTest™ Performance to test a native or web application from the mobile device by using an HTTP protocol.

Before you begin

- You must have installed HCL OneTest™ Performance on your computer.
- You must have a mobile device.
- HCL OneTest™ Performance and the mobile device must be connected to the same wireless network.

About this task

To record the HTTP traffic to and from the mobile applications, you must download a digital certificate to connect to a wireless network, which is same as that hosts HCL OneTest™ Performance. To record communication between the mobile device and the internet, the computer on which HCL OneTest™ Performance is installed must be set up as a proxy server. This setup communicates all mobile network traffic through HCL OneTest™ Performance's recording proxy.

1. To download the certificate from HCL OneTest™ Performance to your computer, perform the following sub-steps:
 - a. In the Performance Test perspective, on the toolbar, click the New Test From Recording icon  or click **File > New > Test From Recording**.
 - b. In the **New Test From Recording** wizard, select **HTTP Test**, and click **Next**.
 - c. On the **Select Location** page, select the project and folder, type a name for the test, and click **Next**.

- d. On the **Select Client Application** page, select the **Unmanaged Application**.
 - e. On the **Proxy Recorder Settings** page, click hyperlinked **save** option to save the recorder root certificate on your local desktop.
2. Share the downloaded certificate with a mobile device that is under test.
 3. To download and install the certificate on a mobile device, perform the following sub-steps depending on your mobile device:

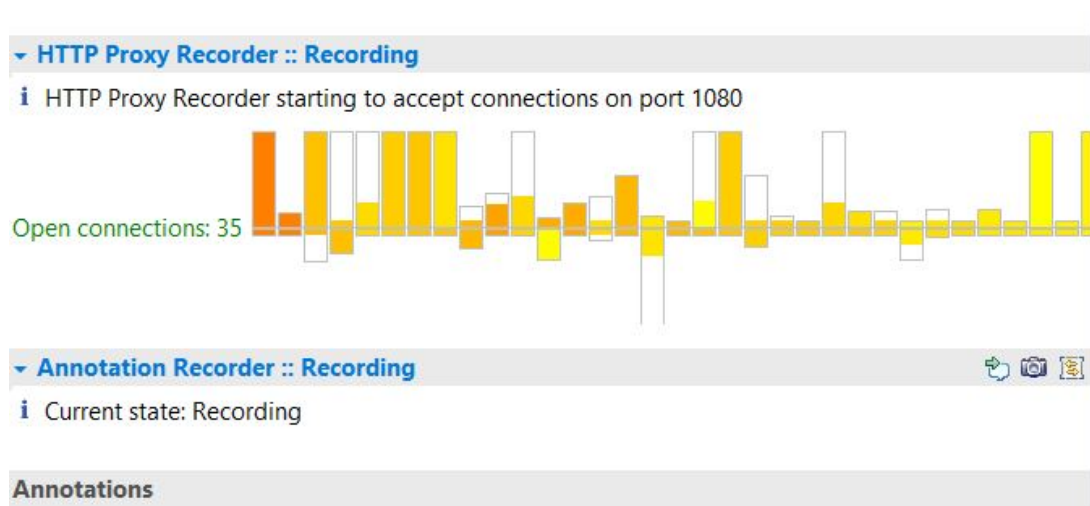
iPhone device	Android device
<ol style="list-style-type: none"> a. Open the email that contains certificate on the mobile device. b. Select the certificate. c. Choose iPhone as the device. d. Go to Settings > General > Profile > HCL OneTest Performance Recorder. e. From Install Profile screen, tap Install. f. If you have a PIN or passcode set on your iPhone, you must enter it. After you enter the PIN or passcode, ignore the warning and tap Install. g. Tap Done to exit the screen. 	<ol style="list-style-type: none"> a. Open the email that contains certificate on the mobile device. b. Download the certificate. c. Go to Settings > Additional settings > Privacy > Encryption & credentials > Install from storage. d. Select the certificate to install. e. If you have a PIN or password set on your Android device, you must enter it and tap OK. f. Specify a name for the certificate and click OK. g. Go to Settings > Additional settings > Privacy > Encryption & credentials > User credentials to view the installed certificate.

4. If you are using an iPhone, go to **Settings > General > About > Certificate Trust Settings** to manually enable the installed certificate.
5. To configure the mobile device proxy, perform the following sub-steps depending on your mobile device:

iPhone device	Android device
<ol style="list-style-type: none"> a. From iPhone, go to Settings > Wi-Fi and tap the connected network. b. In the HTTP PROXY section, select Manual for Configure Proxy and specify the following settings: <ul style="list-style-type: none"> ▪ Server - The IP address or the host name of the computer that hosts HCL OneTest™ Performance. ▪ Port - Enter <code>1080</code> as HCL OneTest™ Performance listening port. ▪ Authentication - Do not enable. 	<ol style="list-style-type: none"> a. From your Android device, go to Settings > Wi-Fi and tap the connected network. b. In the PROXY section, select Manual and specify the following settings: <ul style="list-style-type: none"> ▪ Hostname - The IP address or the host name of the computer that hosts HCL OneTest™ Performance. ▪ Port - Enter <code>1080</code> as HCL OneTest™ Performance listening port. ▪ Bypass for - Leave this field blank. c. In the IP SETTINGS section, select DHCP.

6. To record an HTTP performance test:

- a. From HCL OneTest™ Performance project, right-click the folder where you want your recording to be placed, and select **New > Test From Recording**.
- b. Select **HTTP Test** in the **Recording Session** window and click **Next**.
- c. On the **Select Location** page, select the project and folder, specify a name for the test, and click **Next**.
- d. Select **Unmanaged Application** in the **Select Client Application** window. This option records HTTP traffic from a mobile device that uses a proxy.
- e. Click **Next**.
- f. Select **Proxy Type** as HTTP and **Proxy port** as 1080 on the **Proxy Recorder Settings** page and click **Finish**.
- g. From your mobile device, navigate through your installed application to start the recording. If all settings are configured correctly, you can see some activities on the **HTTP Proxy Recorder** tab during recording.



- h. To pause your recording, and if you want to navigate to other section of mobile application that needs no recording, click the **Pause/Resume Recording** button. Click the button again to resume recording.
- i. When you finish recording all the required transactions from your mobile application, click the **Stop** button to stop the recording.
- j. Select the domains to include in the test and click **Finish**.

Results

A progress window is displayed while the test is generated. After the test is complete, the Recording Control view displays the Test generation completed message, the test navigator lists your test, and the test opens in the test editor.

Generating HTTP tests from a Web UI test

The performance tests record the HTTP traffic against the HTTP server that hosts the application under test. You would have recorded many scenarios against the server. If a new version of the server is about to be used, there can be changes in the HTTP traffic for all of the existing tests. You might have to re-record all of the tests to bring them back to the state where they can be run successfully. This is a time-consuming process.

Before you begin

You must have installed HCL OneTest™ PerformanceHCL OneTest™ UI in the same package group, also referred to as shell-share mode.

About this task

When you know that a new version of the HTTP server is going to be installed and it would impact your performance tests, you can record the scenarios with the Web UI Test recorder of HCL OneTest™ UI and then just generate the HTTP tests. When you have all of the HTTP tests generated, you can add the required test elements and run them to view the performance results. When the new version of the server is installed, you can start refreshing the generated HTTP tests. This action records the traffic from the Web UI test and uses the test elements such as loops and dataset that were edited in the HTTP test with the new HTTP traffic. The new HTTP tests are re-recorded with new HTTP traffic without manual intervention.

1. From the Web UI perspective, create a Web UI test.
For information about creating a Web UI test, see [Recording a Web UI test](#) .
2. In the Test Navigator view of Performance Test perspective, right-click the Web UI test and click **Generate HTTP Test**.

You can have only one Web UI test as the *master* test.

Result

The HTTP Test recorder opens the web browser and automatically records the HTTP traffic for the UI actions that were recorded in the Web UI test and generates the HTTP test.

3. To refresh the HTTP test, right-click the generated HTTP test and select **Refresh HTTP Test**.
This option is available only for the HTTP tests that were generated out of Web UI tests.
4. Specify a name for the test, select a project, and click **Next**.
5. Click **Finish**.

The HTTP Test recorder opens the web browser and automatically records the HTTP traffic from the Web UI test and generates the HTTP test against the new version of the server.

Preparing to record a test for the HTTP/2 protocol

To collect the performance data of an application that supports HTTP/2 protocol, record a test against the HTTP/2 application. Before recording the HTTP/2 application, follow the procedure in this topic to configure your computer.

About this task

This configuration is required because this feature is released as Beta and is intended for the non-production use.

1. Download the following Application Layer Protocol Negotiation (ALPN) boot jar file <https://mvnrepository.com/artifact/org.mortbay.jetty.alpn/alpn-boot/8.1.8.v20160420>
2. Create or rename the `productInstallDir\jdk` folder to `..\jdk.abc`. You can rename the folder back to `jdk` later to test with Java.
3. Download Oracle Java 1.8.0u92 from <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>. You can then either extract the compressed file or install Java at `productInstallDir\jdk`.
4. Copy the ALPN jar file to `productInstallDir\majordomo\lib`.
5. From `productInstallDir`, open `eclipse.ini` and add the following flags:

```
-Xbootclasspath/p:<productInstallDir>\majordomo\lib\alpn-boot-8.1.8.v20160420.jar
```



Note: If there are any other flag starting with `-X`, delete those flags.

6. Configure HCL OneTest™ Performance Agent to use Oracle Java.

- a. Stop the Majordomo process.

On Windows systems, run the following command: `cd "c:\program files\hcl\hclonetest\majordomo"`
`ngastop`

On Linux systems, run the following command: `cd /opt/HCL/HCLOneTest/Majordomo ./MDStop.sh`

- b. Set the environment variable `RPT_JAVA` to the Oracle Java binary or executable.

On Windows systems, run the following command: `set RPT_JAVA=c:\program files\java`
`\jdk1.8.0_92\bin\java.exe`

On Linux systems, run the following command: `export RPT_JAVA=/root/jdk1.8.0_92/jre/bin/java`

- c. Start the Majordomo process.

What to do next

You can now record a test for the HTTP/2 application. After the recording, in the Version field of request details, the requests are marked with HTTP/2 indicating that the HTTP/2 traffic is captured. If the test playback fails, check if all the steps are correctly followed.

Splitting an HTTP test during recording


You can insert split points when you record a test. With split points, you can generate multiple tests from a single recording that you can replay in a different order with a schedule. You can also create a schedule that contains all of the tests that are generated from the split points.

About this task

During the recording process, you can select the option to create a schedule for the tests that are generated from the split points. The schedule will contain these attributes:


- One user
- One user group for the local computer
- All of the tests from the recording, in serial order
- One stage: Run until finished
- Recorded think times, with the maximum think time set to 2 seconds
- Statistics:
 - Statistics log level: All
 - Statistics sample interval: 5 Seconds
 - Only store All Hosts statistics
- Test Log:
 - Show errors and failures: All
 - Also show warnings: All
 - And also show all other types: All
- Problem Determination log level: Warning

To insert split points when you record a test:

1. Start recording the test. The **Recorder Test Annotations** toolbar opens near the top of the screen.
2. To insert a split point into the recorded test, click the **Split point** icon . The **Insert Split Point** window is displayed.

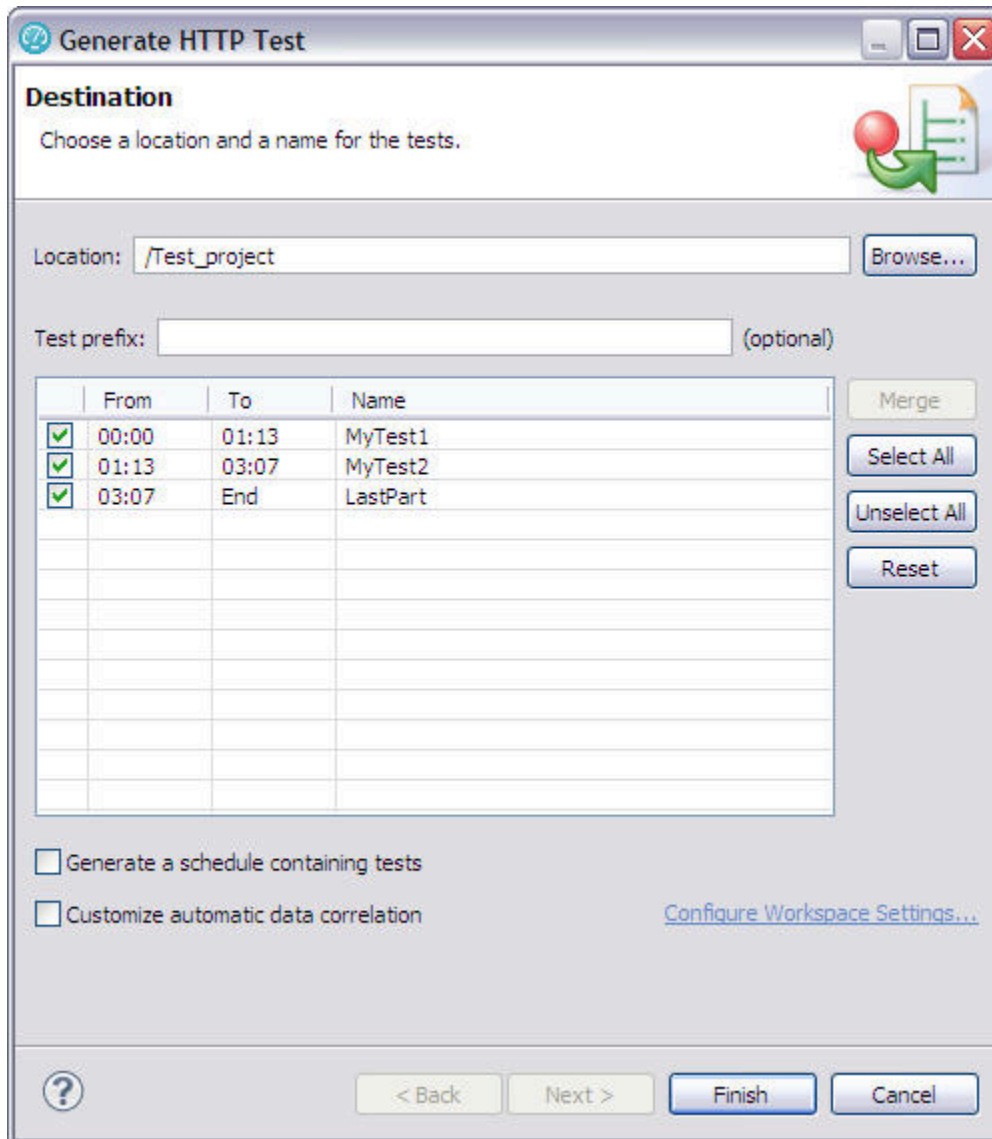
Choose from:

- Click **Test name**, and then type a name for this section of the test.

 **Tip:** You are naming the previous section of the test, not the upcoming section of the test.

Repeat this step between recorded user actions as needed to split tests.

3. Close the client program to stop the recording. The **Generate HTTP Test** window is displayed.



Choose from:

- Type a **Test prefix** for all the tests in the schedule.
 - Select **Generate a schedule containing tests** to create a schedule for the tests. When you select this option, you can modify the name of the schedule.
 - Select **Customize automatic data correlation** to choose automatic data correlation or rule-based data correlation. If you select this option and choose rule-based data correlation, you can specify which data correlation rule sets to use.
4. Click **Finish**.

Result

The **Test Generation** window displays the status of generating the tests and schedule and the data correlation. You can view the test generation log from this window.

- When test generation is complete, you can select the test to open and then click **Open Selected Tests**, or you can click **Close** to finish this process.

Results

The schedule and tests are generated using the names that you specified in the wizard.

Returning a browser to its initial state if recording is interrupted

Browser options are changed during recording and are reset after recording is complete. If you interrupt recording, the browser stays in its changed state, and you may receive `The page cannot be displayed` messages. To fix this, reset the browser to its initial state.

- Right-click the Internet Explorer icon, and select **Properties**.
- On the Connections page, click **LAN Settings**.
 - If you do not use a proxy, in the **Local Area Network (LAN) Settings** window, clear **Use a proxy server for your LAN**.

- If you use a proxy:

In the **Local Area Network (LAN) Settings** window, select **Use a proxy server for your LAN**, and then click **Advanced**.

In the **Proxy settings** window:

- Add the proxy address and port number to the **HTTP** and the **Secure** fields.
 - Remove the proxy address and port number from the **Socks** field.
 - In the **Proxy Settings** window, click **OK**.
- In the **Local Area Network (LAN) Settings** window, click **OK**.
 - In the **Internet Properties** window, click **OK**.

Creating HTTP tests manually

The typical—and simplest—way to create a test is by recording it. However, you can also write a test from scratch.

Creating an empty test

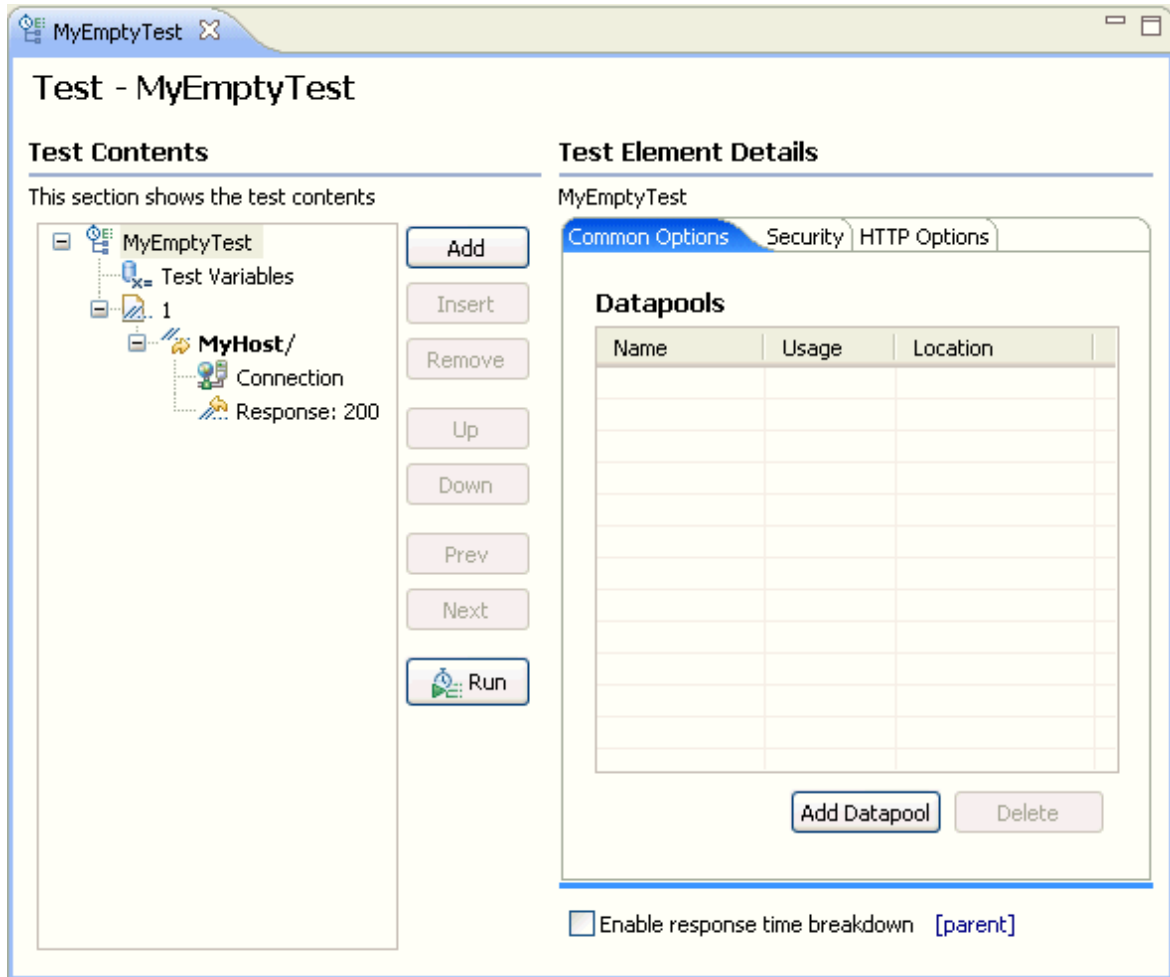
You can create an empty template for a test that you plan to write manually.

- In the Test Navigator, right-click a project and click **New > Other**.
- In the **New** window, expand **Test**, expand **Test Assets**, click **New Test**, and then click **Next**.
- In the **Enter, create, or select the parent folder** field, click the test project in which to store the test.
- In the **Name** field, type a name for the test, and then click **Next**.
- Optionally, in the **Test Attributes** window, type a description for the test, and then click **Next**.
- In the **Protocol and Features** window, select **HTTP Protocol**, and then click **Next**.

7. In the **HTTP Extension** window, enter the name of the host, and set the options for the test.
8. In the **New test summary** window, inspect your selections, and then click **Finish**.
9. Confirm that you want to open the editor.

Result

A test template opens for you to edit. The following figure shows HTTP test template with the default options (one page, one request per page, and a response for the request).



Adding templates for new elements

You can add a template for a new test element that you plan to write by hand.

To add a template for a page, page request, or basic authentication block to a test:

1. Open the test. If you are writing a test completely by hand, you can create an empty test as explained in [Creating an empty test on page 197](#).
2. Click the parent or sibling element.
3. Click **Add** or **Insert**.

Result

The editor enforces the test hierarchy: tests contain pages, pages contain requests, and requests contain authentication blocks.

- **Add** adds the template to the bottom of the selected element: a page is added to the bottom of the test, a request is added to the bottom of the selected page, and an authentication block is added to the end of the selected page request.
 - **Insert** inserts the template at the point of the cursor: a page is inserted before the selected page, a page request is inserted before the selected request, and an authentication block is inserted at the top of the selected page request.
4. Select one of the following options. The options that are displayed depend on the element that you select.

Option	Description
HTTP Page	Adds a page to the test.
HTTP Request	Adds a request to the test.
Basic Authentication	Adds an authentication block to a request. A folder named Authentication is added, and the Test Element Details area displays the User id , Password , and Realm fields.
Custom Code	Adds a block of custom code to the test. For more information on custom code, see Extending test execution with custom code on page .
Delay	Adds a delay to the test.
Loop	Runs part of the test a specific number of times. In addition to setting the number of iterations, you can control the rate of iterations and randomly vary the delay between iterations. For more information, see Extending test execution with custom code on page .
Condition	Adds a conditional block to the test. In most cases, a conditional block issues HTTP requests depending on the value of a reference or field reference. The reference or field reference must exist in the test and precede the conditional block. For more information, see Adding conditional logic on page .
Transaction	Enables you to view performance data about the transactions that you have added. In the Test Element Details area, give the transaction a meaningful name. This is useful in the Transactions report, which

Option	Description
	lists transactions by name. For more information, see Adding a transaction to a test on page .
Random Selector	Enables you to run test elements in random order. For more information, see Running test elements in random order on page .
Comment	Adds a comment that appears in the Test Element Details area and in the actual test.

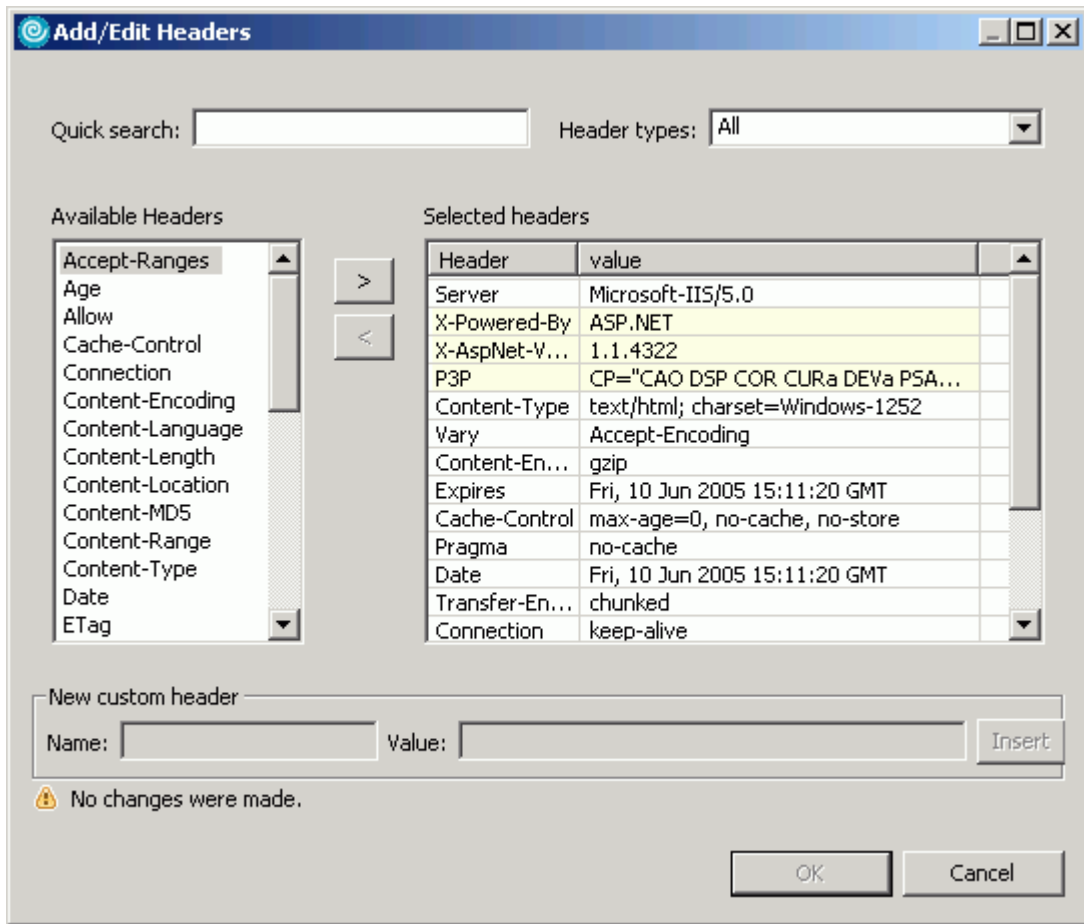
Adding a header

You can add a standard header or a custom header to a request or response.

1. Open the test.
2. In the test hierarchy, click a request (to add a request header) or the **Response Data** folder for a request (to add a response header).
3. In the **Test Element Details** area, locate the **Request Headers** table, and then click **Add**.

Result

The **Add/Edit Headers** window



opens.

4. To add a standard header:

- a. In the **Available Headers** list, locate the header to add and click it.

Use the **Quick search** field (start typing the name of a header) and **Header types** list (select the type of header you are looking for) to quickly locate a header in the **Available Headers** list.

- b. Click the right angle bracket (>).

Result

The selected header moves into the **Selected headers** list and your cursor is placed in the value column.

- c. Type the value for the header.

5. To add a custom header:

- a. In the **Header types** list, select **Custom**.

- b. At the bottom of the window, in the **New custom header** area, type the header information in the **Name** field and the **Value** field, and then click **Insert**.

Result

The custom header is added to the **Selected headers** list.

- When you have finished adding headers, click **OK**.

Editing header contents

You can modify the contents of the headers that are contained in requests and responses.

- Open the test.
- In the test hierarchy, click a request (to edit a request header) or the **Response Data** folder for a request (to edit a response header).
- In the **Test Element Details** area, locate the **Request Headers** table, and double-click the cell containing a value that you want to edit.

Result

An ellipsis button is displayed in the Value column.

Request Headers		
Header Name	Value	
Accept	image/gif, image/x-xbitmap, ima...	Add
Accept-Lan...	en-us,ja;q=0.8,fr;q=0.5,zh-tw;...	Modify
Accept-Enc...	gzip, deflate	Remove
User-Agent	Mozilla/4.0 (compatible; MSIE 6...	
Host	ibm.com	
Connection	Keep-Alive	
Cookie	IBMISP=477cb9a8b19511d9bb9...	

- Click **Modify**.
- In the **Edit Header** window, edit the header value as desired.
You can create a reference or field reference in the header value.
- Click **Table View** to return to the **Request Headers** table.

Creating tests from HTTP Archive files

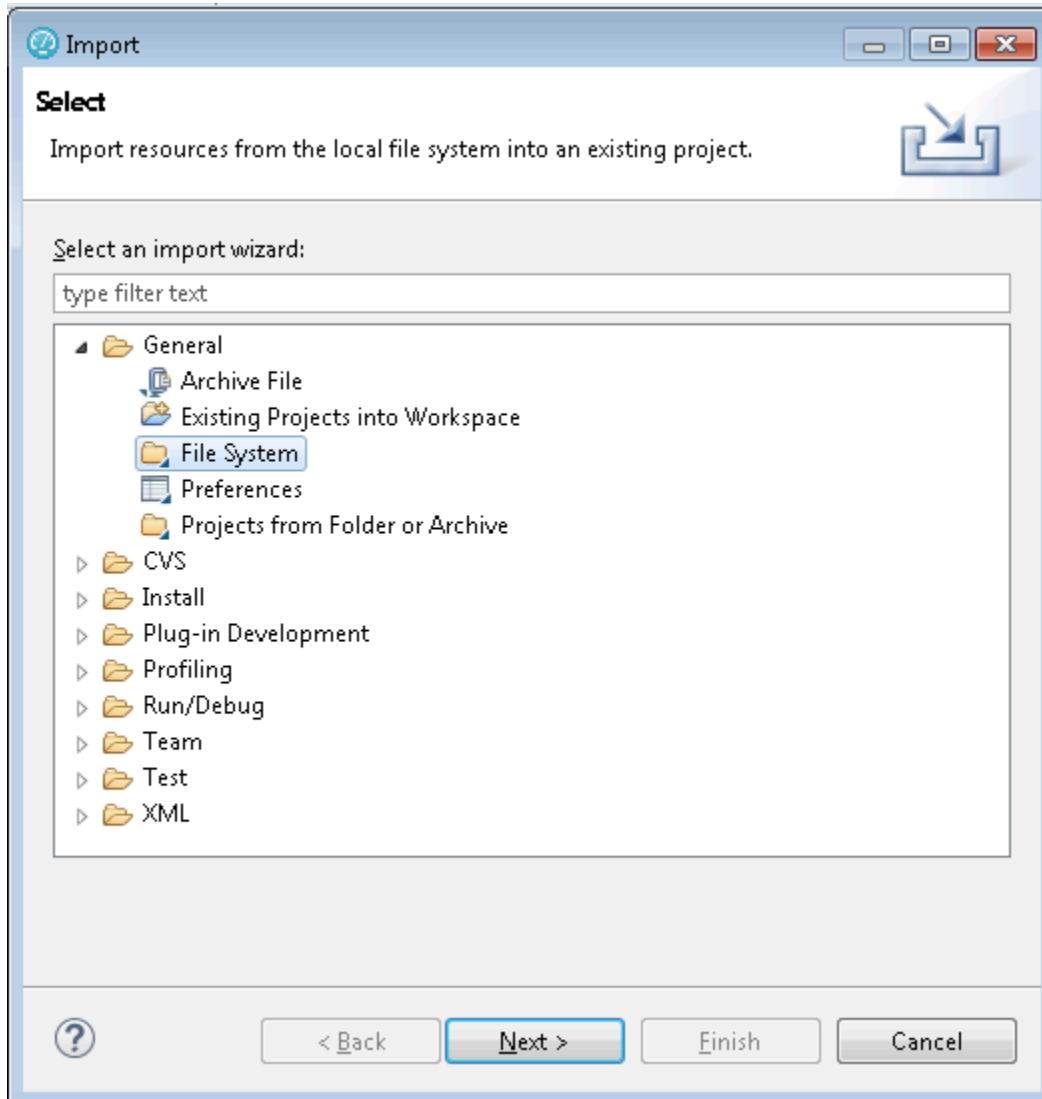
To migrate your tests from another tool to HCL OneTest™ Performance, you can export tests in the HTTP Archive (.har) file format and then import or copy the HTTP Archive files to HCL OneTest™ Performance. You can then create test scripts from the HTTP Archive files.

About this task

Depending on the tool that you use to export tests to the HTTP Archive format, Google Chrome and Mozilla Firefox, for example, do not save enough information to reflect exactly what happened on each connections. The tool exports the contents, HTTP requests and responses, timings, but connections and SSL information is not exported. Use a tool such as Fiddler2 that exports enough information about tests.

When you import or copy an HTTP Archive file, the product displays a message if the Archive file does not contain enough information to generate the test.

1. To create tests from the HTTP Archive file, you must first copy or import the archive file to the Test Navigator view of the product. To import the HTTP Archive files in the product, initiate the Import action from the Test Navigator view and choose the **File System** option in the wizard.



After the import is finished, a new category **HTTP Archive** is created in the Test Navigator view.

2. Generate the test from one of the following methods:
 - a. In the Test Navigator view:
 - i. Right-click the HTTP Archive file, click **Generate Test**.
 - ii. Specify a name for the test, and click **Finish**.
 - b. In the product:

- i. Click **File > New > Test from Recording > Create a test from an existing recording**
- ii. Select the HTTP Archive file and follow the wizard to create the test.



Note: If the HTTP Archive file cannot generate a test, a message is displayed when you select it.

3. Select the type of test that you want to create and click **Next**.
4. Select the domains that you want to include in the test and click **Finish**.

Result

A new test is generated.

What to do next

You can now run the test. See [Running a local schedule or test on page 608](#).

Recording Adobe™ Flex content

You can record an HTTP session that contains traffic to and from a `.swf` Adobe™ Flex application.

About this task

The Adobe™ Flash plugin must be installed in your web browser to run flex applications (`.swf` files) and to record the HTTP session.

Adobe™ Flex technology allows communication through XML, SOAP, or AMF. AMF is compressed binary data that must be transformed into XML to be used in an HTTP performance test.

1. Record the HTTP as described in [Recording an HTTP test on page 183](#).
By default, HCL OneTest™ Performance transforms the AMF format into the XML format.
2. **Optional:** If you configured the default setting of HCL OneTest™ Performance to not transform after the recording, follow the steps in the [Transforming binary data in tests on page 478](#) topic.

Results

After the transformation, the XML data is generated in the test with "amf" prefixes in the names, as in the following example:

```
<amf3Property amfName="zipcode" amfType="amf3String" zipcode="94103"/>
```

The generated XML can be used for data correlation. Only the `amfValue` attributes or attributes without the `amf` prefix can be used for data correlation.



Note: Do not modify or use data correlation on any XML attributes that start with the `amf` prefix except for `amfValue`. If these attributes are modified, the replay of the test will cause errors.

Related information

[Recording an HTTP test on page 183](#)

Recording Microsoft Silverlight applications

To test the performance of an application that was developed by using Microsoft™ Silverlight, you can record the application.

Before you begin

HCL OneTest™ Performance supports Microsoft Silverlight 5.

About this task

Microsoft™ Silverlight communicates with the server in the Windows™ Communication Foundation (WCF) Binary format. After recording the Microsoft Silverlight application, you can transform the format to XML for better readability and to apply data correlation.

1. Record the HTTP application as described in [Recording an HTTP Test on page 183](#).
By default, HCL OneTest™ Performance transforms the WCF Binary format into XML format.
2. **Optional:** If you changed the default setting of HCL OneTest™ Performance to not transform after the recording, follow the steps in the [Transforming binary data in tests on page 478](#) topic.

Changing HTTP test generation preferences

You can change how performance tests are generated, such as how tests process verification points, data correlation, and pages.

1. Click **Window > Preferences > Test > Test Generation > HTTP Test Generation**.
2. Select the preference to change.

The test generation preferences are as follows:

Do not generate a new page if think time is less than

Enter the shortest time, in milliseconds, that the generator uses as a delay to emulate user think time for an HTTP page. If your tests contain fewer pages than expected, try a shorter interval.

Create a new page if delay between requests is greater than

Enter the longest delay, in milliseconds, that the generator allows between page requests. If this time is exceeded, a new page is generated. If your tests contain more pages than expected, try a longer interval.

Maximum request delay

Enter the longest delay, in milliseconds, that the generator allows before truncating HTTP requests. The requests are truncated on the generated test. The recorded test still contains the original values, and you can get them back by generating a new test.

Save only the first 4KB of responses larger than

Enter the limit of response data, in KB, that the generator saves. If a response is larger than the specified limit, only the first 4 KB of data is saved.

Suppress NSLookup() and use numeric IPs

Select this option to shorten test generation time. The disadvantage is that IP addresses in a test are less user-friendly than web page format (www.example.com).

Disable Page Cache Emulation during test generation

Select this option to disable page cache emulation. When page cache emulation is enabled, caching information in server response headers is honored. Additionally, requests are not submitted to the server for content that is confirmed by the client as fresh in the local cache. Page cache emulation is enabled by default.

Enable domain review before test generation

Clear the check box to not show the test generation page to select specific domains to be added to the test. By default, in addition to the domain that you intend to record, other domains linked to the original domain are also recorded.

Remove HTTP request delays from page response times

To not include the client delays in the page response times for the test or A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule., select this check box. By default, the page response times include delays to represent processing time caused by clients such as a web browser. Sometimes this delay could exceed the logical limit causing page response times to increase drastically.

Use Legacy Test Generator

Select this option if you have been instructed to use the legacy HTTP test generator.

Automatically include verification point of

Click to specify the types of verification points to be automatically included. If a check box for a verification point is selected, the code and edit controls for this type of verification point are generated in all tests. Verification points can also be enabled or disabled within specific tests.

Relaxed

Response codes that are in the same category (for example, 200, 201, 203, 209) are considered equivalent. An error is reported if the response code is not in the same category.

Exact

An error is reported if the response code does not match the recorded value exactly.

Accept sizes for primary request within

If you are automatically generating response size verification points, click to specify the acceptable size range for primary requests. No error is reported if a response is within the specified percentage above or below the expected size. By default, for primary requests, **HTTP response size** verification points use range matching.

The data correlation preferences are as follows:

Automatically correlate host and port data

By default, host and port data is correlated automatically. If tests in a previous release have significant manual correlations, or you are using proxies, the migration of the replace-host functionality feature is likely to fail during playback. In this situation, clear the check box. When you reopen your tests, they will not have the automatic correlation feature in them.

Automatically correlate URL pathname if redirected by response

Specifies whether URL path names are correlated if they are redirected by a selected response code. If a check box for a response code is selected, the test generator performs correlations for that response code. This option applies only to responses that are redirects, with a status code between 300 and 399.

Automatically correlate Referers

By default, the Referer field in an HTTP request header is correlated automatically. Clear the check box if you plan to correlate Referers manually. If you run tests against servers that do not require a Referer field, clearing this check box reduces the number of correlations performed when the test runs, and can increase user throughput.

Enable all other data correlation

By default, request and response data is correlated automatically. Clear the check box to disable automatic data correlation of request and response data. Consider clearing the check box if you create your own data correlation rules in the rules editor.

Create substitutions for empty strings

Select this check box to correlate empty strings. For example, strings such as spouse name or middle initial sometimes become important to correlate. However, correlating empty strings increases the time to generate a test.

Optimize automatic data correlation for execution

Specifies the characteristic that tests are automated for.

- With the **Accuracy** setting (the default), many references with an identical session ID value are created and the value of each session ID is substituted from the nearest previous reference.
- To make a test run faster by reducing the number of references that are created during automatic data correlation, change the optimization to **Efficiency**. For example, consider a test where a session ID, which is assigned when a user logs in, is included in every

subsequent request in the test. With the **Efficiency** setting, all session IDs are substituted from a single previous reference. The downside of this setting is that it can result in incorrect correlations. For example, a request that contains the `Joe Smith` string might be incorrectly correlated with a request that contains the `Joe Brown` string.

URL rewriting for execution

Specifies how web addresses (URLs) are rewritten during test execution. When correlating data, the test generator replaces part of a URL request string with a value that the server returned in response to a previous request.

- **Automatic** (default): The test generator automatically determines when rewriting the entire URL during substitution will facilitate test execution.
- **On**: Select to rewrite URLs in every instance of data correlation. This produces larger tests that take longer to run. Try this setting if your tests fail unexpectedly.
- **Off**: Select to manually correlate the instances where URL rewriting is needed. This setting might cause execution errors.

URL encoding for execution

With this option, you can control the encoding of the URLs. If you set it to **Automatic**, the tool detects the encoding that already exists in the test and applies it to the substitution site. If you set it to **ON**, the tool always encodes the substitutions according to the encoding standards. If you set it to **OFF**, no encoding occurs.



Note: To turn data correlation off entirely or to set whether names are automatically generated for data correlation references, click **Window > Preferences > Test > Test Generation > HTTP Test Generation**, and click the **Data Correlation** tab.

The data correlation type preferences are as follows:

Data Correlation Types

Specify when to generate data correlation constructs. With the **Automatic** setting, the test generator creates the required constructs where needed. If the test does not contain the required constructs, change the setting to **On**, which will always perform data correlation. If tests do not require a specific construct, select **Off**, which has the additional benefit of improving performance on subsequent test generation.

Jazz Foundation Services

The **On** and **Automatic** options enable data correlation for Jazz applications that use REST storage or query APIs from Jazz Foundation Services. An example of such an application is Rational DOORS Next Generation. Although data correlation does not typically apply to browser-based Jazz web clients, it may be useful for other HTTP client-server applications that use REST services and the Atom Publishing Protocol for updating web resources.

Jazz Web Applications

The **On** and **Automatic** options enable data correlation for Jazz web applications that use the Jazz Foundation web UI framework. Examples of these web applications are the web interfaces for Rational Quality Manager and Rational Team Concert. Data correlation can also be useful for other web applications that contain javascript that employs JSON for client-server data exchange. This is a common practice with DOJO- and AJAX-based applications.

JSON

To perform data correlation on web applications that uses JSON framework, ensure that Automatic or ON is set to the JSON entry.

Prioritize correlation based on ID

Select **On** to correlate HTML response code based on its ID attribute. Generally, the HTML response code after the recording would appear as `<input type="username" name="User" id="aaa" value="John"/>`. Some applications dynamically update the *name* attribute. Therefore, when you play back the test, the HTML response code would appear as `<input type="username" name="idt020" id="aaa" value="John"/>`. Because the *name* attribute is changes dynamically, data correlation does not occur and the playback fails. When this option is turned on, the ID attribute is considered as the basis to correlate the *name* attribute in the request and to locate the *value* attribute.

3. After changing a setting, click **Apply**.

Recording Citrix tests

When you record a test, the test creation wizard records your interactions with the Citrix server, generates a test from the recording, and opens the test for editing. You can record a test session in the Citrix XenApp client.

Citrix performance testing guidelines

Citrix performance tests use synchronization mechanisms to replay the tests on multiple Citrix sessions independently of server performance.

These guidelines will help you record a reliable test and avoid synchronization timeouts during test execution.

Ensure that you have a working Citrix client environment and that you can connect to a Citrix server. .

The most efficient recording method is to specify a published application or a server in the recording wizard. It is preferable to specify your login credentials in the wizard instead of recording the login sequence as part of the test.

Ensure that the session that you are recording will be reproducible. To record tests that can be reliably replayed, follow these guidelines:

- If you save a file during a recorded session, when replaying the tests, some applications might produce a warning for an existing filename. If the warning was not in the recorded session, this might break the test and cause synchronization timeouts.
- Do not use the mouse wheel to scroll when recording Citrix tests. Mouse wheel events are not recorded and will result in synchronization timeouts on execution.
- Anticipate and make provisions for avoidable warnings and unrecorded windows and dialog boxes. For example, if you save a file while recording, the test will try to save the same file during the replay. Although the file might not exist during the initial recording, it does after recording. Attempting to save will generate an overwrite warning and cause synchronization timeouts.
- Disable or turn off warnings, windows, and dialog boxes that are displayed during the first start of an application. These items will be captured during the recording, but they might not be displayed in subsequent application starts. This will cause synchronization timeouts.
- Use dedicated test user accounts for performance tests. Ensure that the user accounts have minimal potential to cause problems if unpredictable mouse events occur outside of the application window after a synchronization timeout.
- Set up the test accounts and applications to minimize unpredictable window events, such as new mail notifications, automatic updates, or daily tips. Disable extensible menus and hover text tooltips when possible.
- Ensure that all computers used for recording and playback of Citrix tests use the same international settings and character sets. Different locales can cause some characters to be unavailable or keyboard inputs to be incompatible.
- To launch applications from the desktop, use the Quick Launch bar, desktop shortcuts, or select **Start > Run** and enter the name of the application. Do not launch applications or open files from locations that are likely to change, such as **Favorites**, **Recent Files**, or other dynamic menus.
- When using cascading menus like the **Start** menu, always wait for a moment for the submenu to display. After the recording, when editing the test, look at the mouse sequences that were generated to ensure that they follow the correct path to display the submenu.
- When recording your tests, before interacting with a window or dialog box, click the element to ensure that it gets focus, then provide input.
- When an application is busy, for example when the mouse cursor is a sand glass, avoid using the mouse or attempting to perform other operations.
- After recording a session, some applications require user input before quitting (for example, to record any changes). This can cause discrepancies between the state of the application at the end of a session and at the beginning of a test execution. To avoid problems, at the end of a recording session, close all applications manually and cleanly end the session by clicking **Stop** or **Close** on the **Citrix Recorder** window, rather than from the **Start > Log Off** menu.
- To ensure long duration schedules are run without issues such as test log transfer not completed, Citrix Online Plugin or Receiver crashes when virtual users are ramping down, and Citrix processes remaining, you must select the **Uninterruptible iteration** check box for the loops. When you use this option, you must increase the value of **Time limit for a user to respond to a stop request** in the **User Load** tab of the schedule as per the requirement.

After recording, and while you edit the test, it is important to perform regular verification runs in order to validate the test with a single user. After each run, open the test log to make sure that the test synchronizes correctly. If necessary, change the synchronization level from **Mandatory** to **Conditional** or **Conditional** to **Optional** on window events or image synchronizations that produce unnecessary timeouts. Only deploy the test on virtual users or run it in a schedule when the test is robust enough to run flawlessly with a single user.

HCL OneTest™ Performance Agent requirements

When you install HCL OneTest™ Performance Agent, by default, the agent starts as a service. To use the agent for Citrix recording, it must run as a process. For Windows, open Windows Services. If the MajordomoService is set to **Automatic** start up, set it to **Manual** and then stop the MajordomoService. You can now open the Majordomo folder from the HCL OneTest™ Performance Agent installation path and double-click `Majordomo.bat`.

Optical Character Recognition

You can use optical character recognition when performing image synchronizations. This allows the test to synchronize itself by recognizing the contents of a screen area. In some cases, the software can fail to correctly recognize portions of text or letters.

To improve results of the optical character recognition, follow these guidelines:

- Use preferably high contrast screen areas where the text is clearly separated from the background.
- Select text areas with a homogeneous font size, style, and color. Mixing text styles will produce poor results.
- Try changing the **OCR language**, **OCR zoom factor**, **OCR brightness**, **OCR recognition rate** or settings in the image synchronization test element. You can change the default values for these settings in the **Preferences > Test > Test Generation > Citrix Test Generation** window.
- If the test produces image synchronization timeouts because of inconsistent text recognition, open the Image Synchronization view in the test log, and add click the **Add value** button to add the unrecognized text as an alternate synchronization value.
- In some cases, it might be more efficient to use the bitmap hash code method for image synchronization, instead of optical character recognition.

When recording image synchronizations with optical character recognition, accuracy of the recognized text is not essential. It is only important that the recognized text is consistent each time the test is executed for the test to synchronize. For example, if a portion of text is displayed as "Hello" on the screen, and recognized as "He110" in the recorded test, you should not attempt to correct the value in the image synchronization test element, because the same result should occur when the test is executed for the test to synchronize.

Long duration test runs

When running long duration test schedules that exceed 24 hours, use the long run mode to reduce resource consumption with the Citrix client. This mode increases the reliability of long duration test runs by running the tests in multiple process. You must enable this option for each user group in the schedule.

In long run mode, a new process is created for each virtual tester.

Related information

[Running long duration Citrix tests on page 610](#)

Recording a Citrix test

You can record a Citrix session with the Citrix XenApp or XenDesktop. When you record, the recording wizard automatically starts the client and configures it for recording. When you have finished recording the session, the wizard generates a Citrix performance test.

Before you begin

- Install Citrix Receiver and connect to Citrix XenDesktop.
- If you use Microsoft Windows 2008, you must have Citrix Receiver 3.3 or 4.1. To run a Citrix test on Microsoft Windows 2008, you must install Citrix Receiver 3.3 or 4.1 and .Net Framework 3.5. To install the .Net Framework 3.5 feature, open **Control Panel** and click **Turn Windows features on or off**. In the Add Roles and Features Wizard, on the Features page, select the **.Net Framework 3.5 Features** check box, and click **Next**. To specify the source files path of the feature, click **Specify an alternate source path**, specify the path, click **OK**, and click **Install**.
- The behavior of the recording wizard is controlled by recorder preferences. To inspect the current settings, click **Window > Preferences**, expand **Test**, and click **Citrix Recording**. This procedure assumes that default settings are used.
- After you record a test or when you run a test on Windows 2008, the **Citrix Image Synchronization** tab does not display the captured screenshot for **Actual Image**. The bitmap hash code also varies for the actual and expected screenshot. You must use the bitmap hash code of the **Expected Image** for **Actual Image**. To do that, in the **Citrix Image Synchronization**, click the **Add Value** icon and click **Yes** in the confirmation message. The Image Synchronization entry is added to the test. Save the test.
- By default, HCL OneTest™ Performance Agent starts as a service. If you run a Citrix test on an agent, you must start HCL OneTest™ Performance Agent as a process.


To start an agent as a process:

1. Open the Windows Services window.
2. For the **Majordomo** service, change the **Startup Type** to **Manual** and then change the **Status** to **Stop**.
3. From the agent's installation directory, open the Majordomo folder, and double-click the Majordomo.exe file.

About this task

You can also record and generate a test by using REST APIs. The API documentation to record a test is located at `C:\Program Files\HCL\HCLIMSHARED\plugins\com.ibm.rational.test.lt.server.recorder.jar`. The API documentation to generate a test after the recording completes is located at `C:\Program Files\HCL\HCLIMSHARED\plugins\com.ibm.rational.test.lt.server.testgen.jar`.

To record a Citrix test:

1. In the Performance Test perspective, click the **New Test from Recording** toolbar button  or click **File > New > Test from Recording**.

2. In the **New Test from Recording** wizard, select **Create a test from a new recording**, select **Citrix Test**, and click **Next**.

If you are recording sensitive data, you can select a **Recording encryption level**.

3. On the **Select Location** page, select the project and folder locations to contain the new test, type a name for the test, and click **Next**.

If necessary, click the **Create Parent Folder**  push button to create a new project or folder.

4. On the **Citrix Connection Settings** page, specify how to connect to the Citrix server. Complete one of these tasks:

Choose from:

- If your Citrix administrator has provided you with an ICA file, complete these steps:
 - a. Select **With ICA file** to use its settings to connect to the server.
 - b. Click **Browse** to locate and select the ICA file on your computer.
 - c. Click **Next** to continue.

Choose from:

- If you want to manually specify the Citrix server to use for the session, complete these steps:
 - a. Select **On server** to connect directly to the server.
 - b. Specify the name or IP address of the server or click **Browse** to locate a server or server farm on your local network.
 - c. **Optional:** If you need to change the Citrix farm parameters, click **Farm Settings**.
 - d. To record a Windows™ desktop session, leave **Initial program** blank.

Choose from:

- If your Citrix administrator has published applications on the network, complete these steps:
 - a. Select **On published application**, and click **Browse** to choose the application from the list of published applications on the server or server farm.
 - b. **Optional:** If you need to change the Citrix farm parameters, click **Farm Settings**.

In this case, the server farm performs the load balancing and selects the server automatically.

5. To make the logon sequence part of the session parameters, select **Logon with user-specified credentials**, type your credentials, and specify the domain name. If this option is disabled, your credentials are recorded as part of the logon sequence during the recording.



Note: If you use the **With ICA file** option and the ICA file contains the LogonTicket command, even after you type your credentials as part of this step, you must type your credentials again on the Citrix server to start a Citrix test. To avoid entering credentials twice, you can manually remove the LogonTicket command from the ICA file.

6. Click **Next** to continue.

7. On the **Citrix Session Preferences** page, you can provide a description for the test, change the video settings, encryption mode, and advanced options for the Citrix XenApp client. Click **Next** to continue.



Note: Because Citrix performance tests are based on low-level interactions with the server, including mouse and window coordinates, the Citrix desktop must be large enough to support the application under test. You must particularly avoid scrolling windows during the recording.







8. If this is the first time you record a Citrix performance test, read the Privacy Warning, and then select **Accept** to proceed.
9. To start the recording, click **Finish**.

Result

The **Citrix Recording** window opens, displaying the Citrix XenApp session and a set of recorder controls.

10. In the **Citrix Recording** window, complete the tasks to test.

You can use the recorder controls in the toolbar to add comments, record synchronizations, or take screen captures during the recording.

- a. To add a user comment to the recorded test, click the **Insert user comment** icon . Because Citrix tests can be long and difficult to read, meaningful comments can help you locate important elements.
- b. To add an image synchronization to the recorded test, click the **Insert image synchronization** icon , select an area of the screen to be used for synchronization, and then click the **Insert image synchronization** icon again. Image synchronizations enable the test to keep track of the contents of a screen area during the replay instead of focusing only on window events. You can use these screen captures to maintain synchronization of a test in applications that do not create or modify many windows, but update the contents of a window regularly. The contents of an image can be evaluated either as a bitmap hashcode or as a text value obtained by optical character recognition.
- c. To insert a response time measurement during the recording, click the **Insert response time** icon  to create a start for the measurement in the recording, and click the icon again to create a stop.
- d. To add a screen or window capture to the recorded test, click the **Capture screen** icon  or **Capture window** icon . Screen and window captures make your tests easier to read and help you visualize the recorded test. To change the settings for screen and window captures, click the **Screen capture preferences** icon , and then select one of these options:

No automatic screen capture

Select this option if you do not want the test recorder to record screen captures automatically. When this option is selected, you can still record screen captures manually. This option is selected by default.

Capture screen every


Select this option to automatically record a periodic screen capture and specify the time between captures.


Capture screen on window creation

Select this option to record a screen capture each time a window object is created in Citrix.

Exclude tooltips

When **Capture screen on window creation** is selected, enable this option to prevent creating a screen capture each time a tooltip event is displayed during the recording. If this option is disabled, screen captures are recorded when tooltips are displayed.

To preview all screen captures, window captures, and image synchronizations on the side of the Citrix Recording window, click the **Screen capture preview button** icon .

- When you have completed the sequence of actions to be tested, close the session, and stop the recorder by clicking the **Stop recording** icon .

Result

A progress window opens while the test is generated. On completion, the **Recorder Control** view displays the `Test generation completed` message, the Test Navigator lists your test, and the test opens in the test editor.

Related information

[Creating tests on page](#)

[Recording sensitive session data on page 272](#)

Recording a test with the Citrix Web Interface

You can record a Citrix session from the Citrix Web Interface to use the load balancing feature provided by this connection method. When you have finished recording the session, the wizard generates a Citrix test.

Before you begin

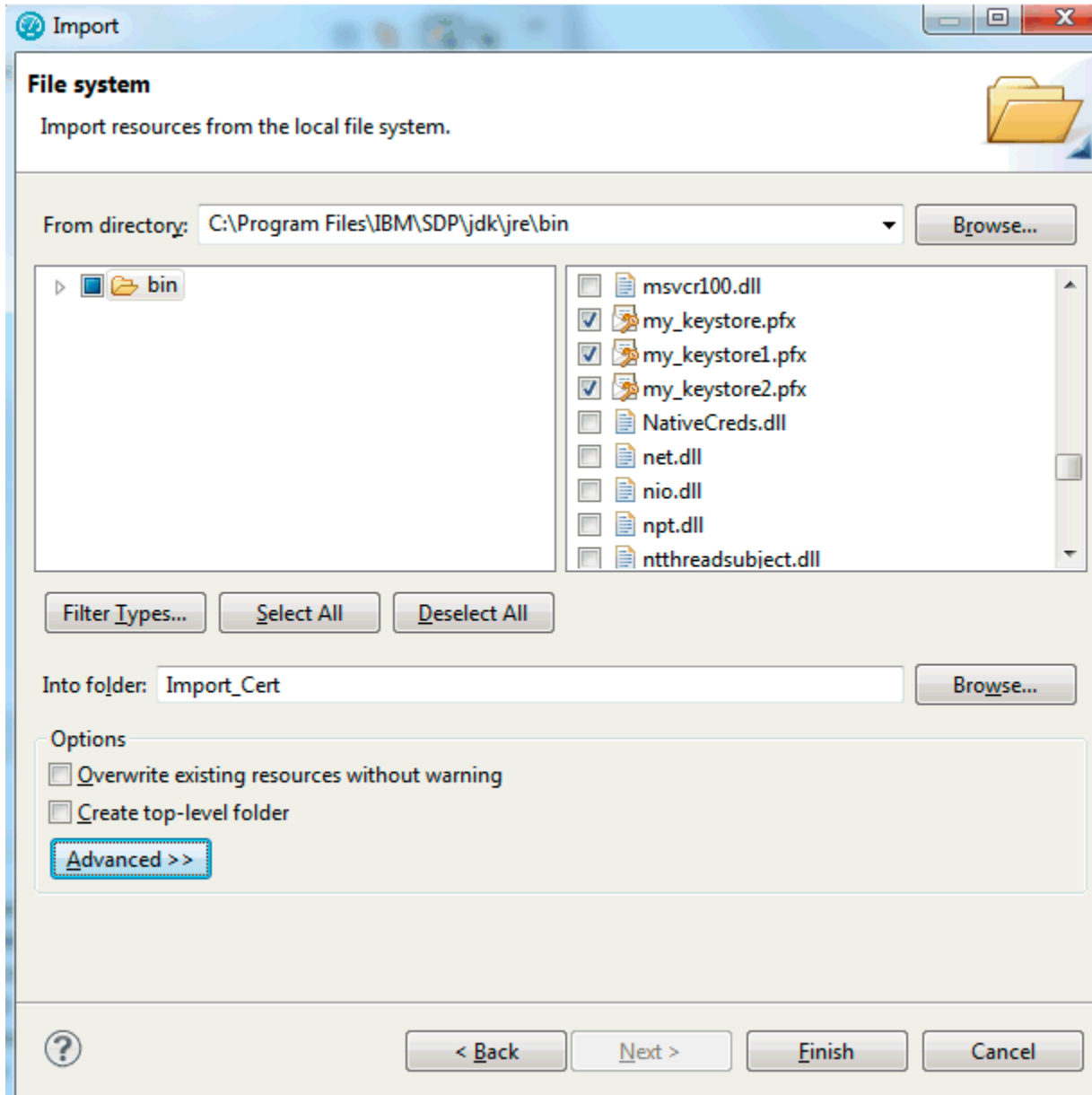
- By default, HCL OneTest™ Performance Agent starts as a service. If you run a Citrix test on an agent, you must start HCL OneTest™ Performance Agent as a process.

To start an agent as a process:

- Open the Windows Services window.
- For the **Majordomo** service, change the **Startup Type** to **Manual** and then change the **Status** to **Stop**.
- From the agent's installation directory, open the Majordomo folder, and double-click the Majordomo.exe file.

Certain websites require appropriate certificates to use a proxy recorder to record the site. The recorder certificate is required to record all the secured sites. The client certificate is different and it serves as an additional layer of security that is required by the web server to authenticate the client/browser. If some applications use Secure Sockets Layer (SSL), the proxy recorder can cause authentication problems because SSL relays traffic between the client and the server. Depending on the authentication method in place, the client might require the proxy recorder to authenticate itself as the server, and the server might require the proxy recorder to authenticate as the client. If the client program requires an authenticated server, you must either have access to the server certificate keystore and provide it to the proxy recorder or configure the client to accept the default certificate from the proxy recorder instead of the certificate from the actual server.

To record an application that requires a client-side certificate, import the client certificate to the HCL OneTest™ Performance project. To import the certificate, click **File > Import > General > File System**, and navigate to the folder that contains the certificates and click **Finish**.



1. Click **File > New > Test from Recording**, and then select **Citrix Recording**, and click **Next**.
2. On the **Select Location for Test Suite** page, select a project.
3. In **Test file name**, type a name for the Citrix test, and then click **Next**.
4. On the **Citrix Connection Settings** page, select **Through a Web interface**, and click **Next**.
5. On the **Select Client Applications** page, select the web browser to use and click **Next**.
6. Depending on your selection, take one of the following steps. On the **Recorder Settings** page, depending on the browser that you selected, specify these details:

- To record HTTP or SOCKS traffic through a proxy, click **Record traffic with the proxy recorder**. To record low-level network traffic for applications where a proxy cannot be used, click **Record traffic with the socket recorder**. Use this option when you are not using Mozilla Firefox or Microsoft™ Internet Explorer.



Note: When using proxy recording, you can filter out HTTP or HTTPS requests to a specific endpoints so that any requests to those endpoints are not recorded. See [Proxy recording preferences on page 1190](#).

- If HCL OneTest™ UI is installed and shell-shared with HCL OneTest™ Performance, for the Web UI tests, you can use the **Record user actions** option to record the functional aspects of the application in the same HTTP recording session. Thus, both the functional and performance aspects of the application are recorded.
- If the server requires client SSL authentication, provide the client certificate for the proxy recorder to be authenticated by the server as though the proxy recorder were the client. Select **The server requires a specific client certificate**.

To provide single certificate keystore, specify the file name and password of the server certificate keystore. If multiple certificates are required, click **Multiple certificates**, and click **Add** to specify a certificate keystore file name and password for each host name and port.

- To record a secured site using Internet Explorer or Google Chrome on Windows, install the recorder certificate by selecting **Register the recorder root certificate authority**. Before the recording starts, the browser prompts you to install the certificate. After the recording is stopped, the browser prompts you to uninstall the certificate. To avoid multiple prompts for each recording, select **Keep the recorder root certificate authority after recording**.



Note: If you already had the certificate from a version prior to 9.2.1 and then install the latest version of the product, you might have to install the certificate again.

This option is not available when you record by using the Firefox or Safari browser. To record a secured site on these browsers, manually import the certificate in the browser from the default location `C:\Program Files (x86)__VENDOR_NAME____VENDOR_NAME__IMShared\plugins\com.ibm.rational.test.lt.recorder.proxy_version\SSLCertificate`. For information about how to import the certificates, see the browser's documentation.

- If you selected **Mozilla Firefox**, you can choose to use a temporary Firefox profile. This option starts the Firefox browser without any bookmarks, plug-ins, or toolbars that might be associated with your usual profile. Select **Use an alternate Firefox profile**, and then select **Use a temporary Firefox profile**.
- If you clicked **Record traffic with the proxy recorder**, click **Advanced** to specify whether to use an HTTP or SOCKS proxy recorder to review and edit network connection settings that the browser uses or to specify advanced SSL authentication settings. If you clicked **Record traffic with the socket recorder**, specify the advanced SSL authentication settings.


7. To start the recording, click **Finish**.

Result

The web browser opens.

8. In the web browser, specify the Citrix Web Interface server URL
9. On the Citrix Web Interface page, type your user name, password, and domain to open a session. The Citrix Web Interface displays the list of applications published on a server and available to the user. Select the application to test.

The Citrix Recording window opens. You can use the recorder controls in the toolbar to add comments, record synchronizations, or take screen captures during the recording.

- a. To add a user comment to the recorded test, click the **Insert user comment** icon .




Because Citrix tests can be long and difficult to read, meaningful comments can help you locate important elements.
- b. To add an image synchronization to the recorded test, click the **Insert image synchronization** icon , select an area of the screen that will be used for synchronization, and then click the **Insert image synchronization** icon again.

Image synchronizations enable the test to keep track of the contents of a screen area during the replay instead of focusing only on window events. You can use image synchronizations to maintain synchronization of a test in applications that do not create or modify many windows, but update the contents of a window regularly. The contents of an image can be evaluated either as a bitmap hashcode or as a text value obtained by optical character recognition. You can also add verification points to image synchronizations in the test editor.

- c. To add a screen capture to the recorded test, click the **Capture screen** icon .

Screen captures make your tests easier to read and help you visualize the recorded test.
- d. To change the settings for screen captures, click **Screen capture preferences** icon , and then select one of these options:

No automatic screen capture

Select this option if you do not want the test recorder to record screen captures automatically. When this option is selected, you can still record screen captures manually. This option is selected by default.

Capture screen every


Select this option to automatically record a periodic screen capture and specify the time between captures.

Capture screen on window creation

Select this option to record a screen capture each time a window object is created in Citrix.

Exclude tooltips

When **Capture screen on window creation** is selected, enable this option to prevent creating a screen capture each time a tooltip event is displayed during the recording. If this option is disabled, screen captures are recorded when tooltips are displayed.

- When you complete the sequence of actions to be tested, close the session and stop the recorder by clicking the **Stop recording** () or close the Citrix application and web browser..

Result

A progress window opens while the test is generated. On completion, the **Recorder Control** view displays the `Test generation completed` message, the Test Navigator lists your test, and the test opens in the test editor.

Results

The Test editor displays both the HTTP pages and the Citrix user actions.

What to do next

To run the Citrix Web Interface test, click **Run Test** in the Test editor.

Inserting a new recording into a Citrix test

You can insert a new recording into a Citrix test. Use this feature to add or replace a part of a recorded session.

Before you begin

Inserting a new sequence into a test requires that the Citrix session reaches the same state as is expected at the point where the new sequence is inserted. For example, if your new sequence must interact with a particular window that was created earlier in the test, you must advance to the point where that window is in the expected state before the recording can start. To do this, the Citrix test recorder can either automatically replay the existing scenario up to the insertion point, or you can manually advance the session to the expected state.

When inserting a new sequence into a test, it is important that the context of user actions is preserved to ensure that the test synchronizes properly during the replay. The resulting test will probably require manual editing to make sure that test can replay smoothly.





Note: Because Citrix tests contain low level user input and synchronizations, minor changes can prevent the test from working. When editing these tests, you must ensure that they are functionally identical.

To insert a recording into a test:

- In the test editor, select the element before which you want to insert the new recording.
It is easier to manage the new test sequence when the insertion point is at the window event level of the test.


2. Click **Insert**, and then **Record scenario**.
3. In the **Update Recording** window, specify how you want the session to reach the state expected at the insertion point, and then click **OK**.

Choose from:

- Select **Automatically replay the scenario** to replay the test up to the insertion point. With this option, the test replays and stops when it reaches the insertion point. If the replay fails to synchronize, you will be asked to manually bring the session to the expected insertion point state.
 - Select **Manually advance to the expected state** to manually put the Citrix session into the state that will be expected at the insertion point.
4. After the Citrix session is in the expected state and you are ready to start the new recording, engage the **Enable/Disable recording**  button in the toolbar.
If you chose the automatic replay option, the recording is enabled automatically.
 5. Perform the sequence of actions that you want to add to the existing test. When you have finished, click the **Enable/Disable recording**  again to stop the recording.



Note: You can restart the recording several times. However, you must be sure that the Citrix session resumes from the same state that it was in when it was stopped. If actions are missing from the recorded sequence the test is unlikely to synchronize correctly during the replay.

6. When you have completed the sequence of actions to be inserted into the test, close the session and stop the recorder by clicking **Stop recording**  .

Result

A progress window opens while the test is generated. On completion, the **Recorder Control** view displays the message `Test generation completed`, the Test Navigator lists your test, and the test opens in the test editor.

7. After the test has been updated in the Test Navigator, check that the new sequence was properly inserted into the test before saving the new test.
Check that the context of the Citrix session is compatible with the user actions at the beginning and at the end of the of the inserted sequence. If the results of the insertion were not what you expected, you can revert to the previously saved version of the test by clicking **File > Revert** or try to correct any problems manually.

Changing Citrix recording preferences

You can change the behavior of the recorder by changing the preference settings.

1. Click **Window > Preferences**.
2. Expand **Test > Recording**, and then click **Citrix Recording**.
3. Select the setting you want to change.

Screen capture options

These settings specify how the test recorder performs screen captures of the Citrix desktop during recording.

No automatic screen capture

Select this option if you do not want the test recorder to record screen captures automatically. When this option is selected, you can still record screen captures manually. This option is selected by default.

Capture screen every

Select this option to automatically record a periodic screen capture and specify the time between captures.

Capture screen on window creation

Select this option to record a screen capture each time a window object is created in Citrix.

Exclude tooltips

When **Capture screen on window creation** is selected, enable this option to prevent creating a screen capture each time a tooltip event is displayed during the recording. If this option is disabled, screen captures are recorded when tooltips are displayed.

Capture screen on image synchronization

Select this option to ensure that a screen capture is recorded each time an image synchronization is recorded.

4. After changing a setting, click **Apply**.

Changing Citrix test generation preferences

You can change how Citrix events are converted into performance test elements.

1. Click **Window > Preferences**.
2. Expand **Test > Test Generation**, and then click **Citrix Test Generation**.
3. Select the setting you want to change.

Recording Optimization Options

These settings specify how mouse and window events are interpreted in the generated test.

Window activate recording

Specify whether to record no, last, or all window-activate actions when a sequence of similar actions is detected.

- **none** disables recording of window-activate events.
- **last** records only the last of an uninterrupted sequence of window events. This eliminates redundant window-activate actions from the recording.
- **all** records all events of the sequence.

Mouse move recording

This setting specifies which mouse move events are recorded. **Relevant** is the default setting.

- **All** records an uninterrupted sequence of mouse movements in the generated test.
- **Relevant** records only the mouse movements that generate a response, such as hover text.
- **First and last** records a simplified mouse-move action.

Automatic Generation

These settings specify test elements that are automatically generated after recording the test.

Verification point on every window title change

When enabled, this option generates a window title verification point whenever the caption changes. If this option is disabled, the window title is verified only when a new window is created. This option is disabled by default.

Response times for main windows

When enabled, this option generates response time measurements for all recorded main window-create events. A main window is a window that is created at the top level of the test contents tree and contains user actions. The generated response time measurement starts with the keyboard or mouse action that immediately precedes the window-create event. This option is enabled by default.

Window event synchronization criteria

Use this option to disable window recognition on the window position, size, or title. Disable any of these options if the test produces synchronization timeouts because a window changes its position, size, or title between or during test runs.

Default Test Execution Delays

This page specifies the default keyboard and mouse delays for the test client. Do not change these settings unless you are experiencing problems with events that do not run correctly.

Synchronization timeout delay

This is the delay after which a timeout error is produced when a window event or an image synchronization element is not recognized during test runs. The default value is 15000 milliseconds. The specified delay is for synchronizations that are set as conditional. Mandatory synchronizations use a delay of three times the specified delay. Optional synchronizations use a fixed delay of 2 seconds.



Note: In the generated test, the **Override synchronization timeout** for a particular window creation event will be enabled with the corresponding recorded time only if it is greater than what is specified in this preference.

If think time is under x ms, then replace with

If the delay between two events is above the specified limit, then it is handled as a think time. If the delay is below the limit, then the test generator replaces the think time with one of the following delays. The think time is the delay spent by a virtual user before performing an action. The default limit is 20000 milliseconds.



Note: In the generated test, the think time for a particular user action will be enabled only when the recorded think time is greater than the value specified for this preference.

Delay between mouse down and mouse up in a click

This is the default delay used to generate a mouse click action using a mouse down and a mouse up action. The default value is 20 milliseconds.

Delay between two mouse clicks in a double click

This is the default delay used to generate a double-click action using two mouse clicks. The default value is 50 milliseconds.

Delay between key down and a key up in a stroke

This is the default delay used to generate a key-stroke action using a key-down and a key-up action. The default value is 20 milliseconds.

Delay between two keyboard strokes in a text input

This is the default delay used to generate a text input action using multiple key stroke actions. The default value is 50 milliseconds.

Default OCR settings

This page specifies the settings for text extraction by optical character recognition in image synchronizations. You might need to experiment with various settings to obtain good results. These settings define the default behavior for new image synchronizations. You can change the behavior for individual image synchronization elements by changing the **OCR settings** in the test editor.

OCR default language

This is the language of the dictionary that is used to recognize words for the application that you are testing. This setting defines the subset of languages that will be available in image synchronization elements in the test editor.

OCR default zoom factor

This is the enlargement factor that is applied to the image. The default setting is medium for standard font sizes. Increase the zoom factor to improve recognition of smaller fonts or decrease for larger fonts.

OCR default brightness

This is the brightness level from 0 to 250 that is applied to the image. The default setting is 70 for text with normal contrast. Increase the brightness setting to improve recognition of darker images or decrease for lighter images.

OCR default recognition rate

This is the rate of recognition that is required for the extracted string to match the expected text. Decrease the recognition rate to tolerate a proportion of mismatching characters in the recognized text. The default is 100%, which means that an exact match is required.

4. After changing a setting, click **Apply**.

Recording service tests

When you record a test, the test creation wizard records your interactions with the service, generates a test from the recording, and opens the test for editing. You can record a test session by invoking service calls with the generic service client or by using an existing client. You can also create a service test manually or from a Business Process Execution Language (BPEL) model.

Service testing guidelines

Before you can test a service, you must set up your test environment and incorporate these guidelines in order to produce reliable tests.

Test prerequisites

Before creating service tests, you might need to perform some initial tasks. These tasks depend on the transport and security protocols that are implemented by the web service under test.

- **HTTP:** This transport method is supported by default; no additional configuration is required.
- **SSL:** The workspace must contain the certificate keystore (JKS) files that are required for single or double authentication.
- **Java™ Message Service (JMS):** The Web Services Description Language (WSDL) syntax must be compatible with the requirements of the product. Refer to [Verifying WSDL syntax compliance for JMS services on page 227](#).

Test generation

When the test is generated, message call envelopes are created according to the XML schema definition (XSD). During this process, mandatory fields are created, and default choices are assumed. You can modify these elements in the test editor.



Note: During recording, you might supply authentication details which are not relevant for the actual application under test. To exclude such actions from the generated test, in **Window > Preferences > Test > Test editor > Service test** ensure that the **Display the 'Skip if Empty' column in XML tree viewer** check box is selected. To select the empty XML elements that you want to skip, in the test editor, select the elements in the **Skip if empty** column.

Encryption and security

The Java™ Runtime Environment (JRE) that the product uses must support the level of encryption required by the digital certificate that you select. For example, you cannot use a digital certificate that requires 256-bit encryption with a JRE that supports only 128-bit encryption. By default, the product is configured with restricted or limited strength ciphers. To use less restricted encryption algorithms, you must download and apply the unlimited jurisdiction policy files (`local_policy.jar` and `US_export_policy.jar`).

For Oracle Java, download the files from this site: <http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>.

Before installing these policy files, back up the existing policy files in case you want to restore the original files later. Then overwrite the files in `/jre/lib/security/` directory with the unlimited jurisdiction policy files.

SSL Authentication

Service tests support simple or double SSL authentication mechanisms:

- Simple authentication (server authentication): In this case, the test client needs to determine whether the service can be trusted. You do not need to setup a key store. If you select the **Always trust** option, you do not need to provide a server certificate key store.

If you want to really authenticate the service, you can configure an certificate trust store, which contains the certificates of trusted services. In this case, the test will expect to receive a valid certificate.

- Double authentication (client and server authentication): In this case, the service needs to authenticate the test client according to its root authority. You must provide the client certificate keystore that needs to be produced to authenticate the test as a certified client.

When recording a service test through a proxy, the recording proxy sits between the service and the client. In this case, you must configure the SSL settings of the recording proxy to authenticate itself as the actual service to the client (for simple authentication), and as the client to the service (for double authentication). This means that you must supply the recording proxy with the adequate certificates.

When using stub services, you can also configure the SSL settings of the stub service to authenticate itself as the actual server. This means that you must supply the service stub with the adequate certificate.

NTLM and Kerberos Authentication

The product supports Microsoft™ NT LAN Manager (NTLMv1 and NTLMv2) and Kerberos authentication. The authentication information is recorded as part of the test during the recording phase.

To enable NTLMv2 support, you must add a third party library to the workbench. For more information, see [Configuring the workbench for NTLMv2 authentication on page 499](#).

Digital certificates

You can test services with digital certificates for both SSL and SOAP security protocol. Digital certificates must be contained in Java™ Key Store (JKS) keystore resources that are accessible in the workspace. When dealing with keystore files, you must set the password required to access the keys both in the security editor and the test editor. For SOAP security you might have to provide an explicit name for the key and provide a password to access the private keys in the keystore.

Limitations

Arrays are not supported.

Because of a lack of specification, attachments are not supported with the Java™ Message Service (JMS) transport. The envelope is directly sent using UTF-8 encoding.

All security algorithms are not always available for every Java™ Runtime Environment (JRE) implementation. If a particular security implementation is not available, add the required libraries to the class path of the JRE that this product uses.

The generic service tester displays the envelope as reflected in the XML document. However, security algorithms consider the envelope as a binary. Therefore, you must set up the SOAP security configuration so that incoming and outgoing messages are correctly encrypted but remain decrypted inside the test.

Performance

Virtual user performance depends on the implementation of the container application. For an HTTP transport, the product has been tested with a maximum of 900 concurrent virtual users under Windows™ and 600 under Linux™. For JMS, the maximum is 100 concurrent virtual users, although this number can vary due to the asynchronous implementation of JMS. Beyond these values, connection errors might occur and the transaction rate will decrease.

Verifying WSDL syntax compliance for JMS services

Various Java™ Message Service (JMS) providers vary in the syntax used for describing services. Before testing JMS services, you must ensure that Web Services Description Language (WSDL) files comply with the requirements of the tool.

1. In the project explorer or test explorer, locate and open the WSDL file for the JMS service that you want to test. If necessary, you can import a WSDL file from the file system by clicking **File > Import > File System**.
2. Ensure that the following criteria are met in the syntax of the WSDL file that you use.

- **Namespace:** `xmlns:jms="http://schemas.xmlsoap.org/wsdl/jms/"`
- **SOAP bindings are set to:** `transport="http://schemas.xmlsoap.org/soap/jms"`
- **JMS transports are defined either as a URL or as `jms:address` element**

3. If the WSDL file is not compliant, edit the file so that it meets the criteria, and then save and close the file.

Exemple

For example, a JMS defined as a URL looks like this:

```
<soap:address location="jms:/queue?jndiConnectionFactoryName=UIL2ConnectionFactory;
    jndiDestinationName=queue/testQueue;
    initialContextFactory=org.jnp.interfaces.NamingContextFactory;
    jndiProviderURL=9.143.104.47"/>
```

A JMS defined as an address looks like this:

```
<jms:address destinationStyle="queue"
    jndiConnectionFactoryName="myQCF"
    jndiDestinationName="myQ"
    initialContextFactory="com.ibm.NamingFactory"
    jndiProviderURL="iiop://something:900/">
</jms:address>
```

Configuring the environment for SOAP security

SOAP security profiles require access to the libraries that implement encryption, signature, and other security algorithms that transform the XML messages before sending and after receiving them. You must prepare an environment with these libraries to use SOAP security, set the class path of the Java™ Runtime Environment (JRE) that Eclipse uses, and set the class path of the virtual machine that the Agent Controller uses.

Before you begin

Before you can test SOAP-based services that use security algorithms, you must obtain a set of security libraries and configuration files for SOAP.

BouncyCastle (<http://www.bouncycastle.org>) is a provider of such security libraries. Use of these security libraries is optional for the Rational® test product.

1. Copy the library files into the `jre/lib/ext` of the JRE installation.

By default, this is the following directory: `C:\Program Files\HCL\HCLOneTest\jdk\jre\lib\ext`

2. Add the following VM argument either to the Eclipse launch command line or to the `eclipse.ini` file:

```
-vmargs-Dosgi.parentClassLoader=ext
```

The `eclipse.ini` file is located in the same directory as the `eclipse.exe` launcher binary that is used to run the product.

What to do next

To configure a remote computer that uses only the Agent Controller and does not require access to the workbench, perform only step 1 and restart the Agent Controller service.

After configuring the environment, you must import a Web Services Description Language (WSDL) file and use the **WSDL security editor** to set up a security profile for the WSDL file.

Recording a service test with the generic service client

You can record a service test by invoking service requests with the generic service client. After you have sent the requests and received the responses from the service, select the results in the History section of the generic service client to generate a test. If you do not have access to a dedicated client for the service calls, the generic service client is the easiest way to generate the calls and to record a test.

Before you begin

If you are testing a SOAP-based web service, ensure that you have access to a valid Web Services Description Language (WSDL) file. The wizard can import WSDL files from the workspace, the file system, a remote repository, or from a URL. Ensure that the WSDL files use the correct syntax for the test environment. The generic service client might not work with some WSDL files.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have the required key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files. See [Configuring the environment for SOAP security on page 228](#) for more information.

If the response in a recording or test generation is in XML and the size of the XML data is higher than the value set in the **XML Message Received maximum length** field, the response is automatically converted to text to avoid any memory issues. To convert the full response to text, the tool checks the value set for **Text Message Received maximum length**. If the value is lesser than the size of the response, the response is truncated. If you want the response to be in XML when the response size exceeds the value set in **XML Message Received maximum length**, you can manually increase the value for both recording and test generation. To change the value for recording, click **Window > Preferences > Generic Service Client > Message Edition**. To change the value for test generation, click **Window > Preferences > Test > Test Generation > Service Test Generation**.

About this task



You can also record and generate a test by using REST APIs. The API documentation to record a test is located at `C:\Program Files\HCL\HCLIMSHARED\plugins\com.ibm.rational.test.lt.server.recorder.jar`. The API documentation to generate a test after the recording completes is located at `C:\Program Files\HCL\HCLIMSHARED\plugins\com.ibm.rational.test.lt.server.testgen.jar`.

To use a WS-SecurityPolicy that is included in a WSDL or an external XML file, you need to configure the security policy as described in [Using a security policy on page](#) . If a recording contains the Security Assertion Markup

Language (SAML) token, the WS Security policy file must rely on the Service Token Service (STS) that produces the token. This token can then be used for encryption or other purposes as was designed.

Sample policy file that relies on SAML token:

```
<sp:SupportingTokens xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
  <wsp:Policy>
    <sp:IssuedToken
      sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
    <sp:Issuer>
    <Address
      xmlns="http://www.w3.org/2005/08/addressing">http://9.143.105.204:8080/axis2/services/STS</Address>
    </sp:Issuer>
    <sp:RequestSecurityTokenTemplate>
    <t:TokenType
      xmlns:t="http://schemas.xmlsoap.org/ws/2005/02/trust">http://
docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</t:TokenType>
    <t:KeyType
      xmlns:t="http://schemas.xmlsoap.org/ws/2005/02/trust">http://
schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey</t:KeyType>
    <t:KeySize xmlns:t="http://schemas.xmlsoap.org/ws/2005/02/trust">256</t:KeySize>
    </sp:RequestSecurityTokenTemplate>
    <wsp:Policy>
    <sp:RequireInternalReference/>
    </wsp:Policy>
    </sp:IssuedToken>
  </wsp:Policy>
</sp:SupportingTokens>
```




1. In the Performance Test perspective, click the **New Test from Recording** toolbar button  or click **File > New > Test from Recording**.
2. In the **New Test from Recording** wizard, click **Create a test from a new recording**, select **Service Test**, and click **Next**.
If you are recording sensitive data, you can select a **Recording encryption level**.
3. On the **Select Location** page, select the project and folder where you want to create the test, type a name for the test, and click **Next**.
If necessary, click **Create Parent Folder**  to create a project or folder.
4. On the **Select Location** page, select **Generic Service Client**.
This option uses the generic service client if you do not have access to a dedicated client for the service calls. See [Recording a service test through a client program on page 231](#) for information about using other client programs to record the test.
5. Click **Next**. If this is the first time you are recording a web service test, read the Privacy Warning, select **Accept**, and click **Finish** to proceed.

Result

The generic service client opens.

6. If your service uses a transport or authentication protocol that requires overriding the default settings, then click the **Transport** tab and create an HTTP, Java™ Message Service (JMS), IBM® WebSphere® MQ, IBM® WebSphere® Java MQ, or Java MQ transport.
7. Click the **Requests** tab.

Choose from:


- Right-click **WSDLs**  and select one of the options to get the WSDL file.
- Right-click **WADLs**  and select one of the options to get the WADL file.
- Right-click **Endpoints**  and select one of the options to send the request.

See [Sending service requests with the generic service client on page 495](#) for more information about using the generic service client.

8. After creating the call, click the **Edit Data** arrow to change the details of the call if necessary.
9. Click the **Invoke** arrow to invoke the service call.

Result

If the call was successful, the response is displayed under the **View Response** arrow.


10. To record a test with multiple calls, repeat steps 6 through 9.
11. When you have finished sending service requests, stop the recorder. You can do this by closing the generic service client or by clicking the **Stop** push button  in the **Recorder Control** view.
If you changed the network settings of the client program as described in step 8, you can revert to the default settings before closing the program.

Result

The **Generate Service Test** wizard opens.

12. Click **Finish**.

What to do next

Alternatively, you can use the generic service client to create, edit, and invoke the calls without recording. Successful responses are added to the **Request History** list. You can select calls in the **Request History** list, and click the **Generate Test Suite** icon .

Related information

[Sending service requests with the generic service client on page 495](#)

[Recording a service test through a client program on page 231](#)

[Recording sensitive session data on page 272](#)

[Sending service requests with WSDL files on page 508](#)

Recording a service test through a client program

You can record tests for SOAP-based, XML, plain text, or binary services with any client program that uses the HTTP protocol. To record the test, the recorder intercepts the service calls and message returns between the client and the service. You can choose between an HTTP or SOCKS proxy recorder or a low-level socket recorder, depending on the capabilities of the client program.

Before you begin

The following recorders are available for recording traffic from an application:

- **SOCKS proxy recorder:** Use this recorder when no proxy connections are required.
- **HTTP proxy recorder:** Use this recorder when a proxy connections is required to connect to the network or when the client program does not support SOCKS.
- **Socket recorder:** Use this recorder for low-level network traffic when the client does not support proxies. This recorder does not support SSL authentication or encryption of any kind and is only available if the HCL OneTest™ Performance Extension *for Socket Protocols* is installed.



Regardless of the recorder that you use, the client program must use the HTTP network protocol. For recording Java™ Message Service (JMS) or IBM® WebSphere® MQ tests, see [Recording a service test with the generic service client on page 229](#).

If you are using Secure Sockets Layer (SSL), the HTTP or SOCKS proxy can cause authentication problems because the proxy recorder relays traffic between the client and the server. Depending on the authentication method in place, the client might require that the proxy recorder authenticate itself as the server and the server might require that the proxy recorder authenticate as the client. If the client program requires an authenticated server, you must either have access to the server certificate keystore and provide it to the proxy recorder or configure the client to accept the default certificate from the proxy recorder instead of the certificate from the actual server.

If you are testing a SOAP-based web service, ensure that you have access to a valid Web Services Description Language (WSDL) file. The wizard can import WSDL files from the workspace, the file system, a remote repository, or from a URL. Ensure that the WSDL files use the correct syntax for the test environment. The generic service client might not work with some WSDL files.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files. See [Configuring the environment for SOAP security on page 228](#) for more information.

To record a service test with a client program:

1. In the Performance Test perspective, click the **New Test from Recording** toolbar button  or click **File > New > Test from Recording**.
2. In the **New Test from Recording** wizard, click **Create a test from a new recording**, select **Service Test**, and click **Next**.
If you are recording sensitive data, you can select a **Recording encryption level**.
3. On the **Select Location** page, select the project and folder to create the test in, type a name for the test, and click **Next**.
If necessary, click **Create Parent Folder**  to create a project or folder.
4. On the **Select Client Application** page, select the type of client program to use.
The program type defines the recorder that can be used. The following client program types are supported for recording a service test:
Choose from:

- **Managed Application:** This option starts a specified program and uses a proxy or socket recorder to record the traffic.

On the **Managed Application Options** page, click **Browse** to specify the **Program path**. If necessary, specify the **Working directory**, and type the command line **Arguments** that the program requires.

If the program requires user input from a command-line interface, select **Open console for user input**.

- Choose a web browser to record traffic that is sent and received with the web browser.
- **Unmanaged Application:** This option enables you to record traffic from one or multiple client programs that use a proxy. You must manually start the client programs and the proxy recorder records all traffic that is sent and received through the specified network port.
- **Generic Service Client:** This option uses the generic service client if you do not have access to a dedicated client for the service calls. See [Recording a service test with the generic service client on page 229](#) for using the generic service client to record service tests.

5. On the **Recorder Settings** page, depending on the type of client program you selected, specify these details:

- If you selected **Managed Application**, specify the recording method.
 - Select **Record traffic with the proxy recorder** to record HTTP or SOCKS traffic through a proxy.
 - Select **Record traffic with the socket recorder** to record low-level network traffic for applications where a proxy cannot be used. This recorder does not support SSL authentication or encryption.



Note: When using proxy recording, you can filter out HTTP or HTTPS requests to a specific endpoints so that any requests to those endpoints are not recorded. See [Proxy recording preferences on page 1190](#)

- If you selected **Record traffic with the proxy recorder**, specify whether the proxy recorder uses HTTP or SOCKS. Select **HTTP** if a connection to proxy is required or if your application does not support SOCKS.

- If you are using SSL authentication, specify the authentication settings for the proxy recorder.

During the recording, the proxy recorder is between the client and the server.

- If the server requires client SSL authentication, provide the client certificate for the proxy recorder to be authenticated by the server as though the proxy recorder were the client. Select **The server requires a specific client certificate**.

To provide single certificate keystore, specify the file name and password of the server certificate keystore. If multiple certificates are required, click **Multiple certificates**, and click **Add** to specify a certificate keystore file name and password for each host name and port.

- To record a secured site using Internet Explorer or Google Chrome on Windows, install the recorder certificate by selecting **Register the recorder root certificate authority**. Before the recording starts, the browser prompts you to install the certificate. After the recording is

stopped, the browser prompts you to uninstall the certificate. To avoid multiple prompts for each recording, select **Keep the recorder root certificate authority after recording**.



Note: If you already had the certificate from a version prior to 9.2.1 and then install the latest version of the product, you might have to install the certificate again.

This option is not available when you record by using the Firefox or Safari browser. To record a secured site on these browsers, manually import the certificate in the browser from the default location `C:\Program Files (x86)__VENDOR_NAME____VENDOR_NAME__IMShared\plugins\com.ibm.rational.test.lt.recorder.proxy_version\SSLCertificate`. For information about how to import the certificates, see the browser's documentation.

- If the client requires server authentication, you must provide the server certificate keystore for the proxy recorder to be authenticated by the client as though the proxy recorder were the server. Select **The client requires a specific server certificate**, and click **Add** to specify a certificate keystore filename and password for each hostname and port. If you do not select this option, the proxy recorder provides its own default certificate.



Note: The keystore must contain the private certificate of the server.

- d. If you selected to use the HTTP proxy recorder, specify how to connect to the network. If necessary, specify an HTTP or SOCKS proxy or point to a proxy auto-configuration (PAC) file. Use this option if you are connecting to the service through a corporate proxy or firewall.

6. Click **Next**. If this is the first time you record a service test and you did not select a web browser for the client application, read the Privacy Warning, select **Accept**, and click **Finish** to proceed.
7. If you selected a proxy recorder with a managed or unmanaged application, change the network settings of the client program to use the proxy recorder.




The method for changing the network settings depends on the client program. However, you must be able to set the following proxy settings in the program:





- SOCKS or HTTP proxy: Specify the protocol that you selected for the proxy recorder in the wizard.
- Host name: Set to `localhost`.
- Port: Specify the port number that you selected for the proxy recorder in the wizard.

To avoid unexpected results, revert to the previous proxy settings before you stop the recording.

8. Use the client program to perform the actions to test.

You can use the **Recorder Test Annotations** toolbar to add comments, record synchronizations, or take screen captures during the recording.

- To add a comment to the recorded test, click the **Insert comment** icon .
- To add a screen capture to the recorded test, click the **Capture screen** icon . Screen and window captures make your tests easier to read and help you visualize the recorded test. You can change the settings for screen captures and add a comment to the image.
- To manually add a synchronization point to the recording, click the **Insert synchronization** icon .

- To manually add a transaction folder to the recording, click the **Start Transaction** icon  and **Stop Transaction**  icon to start and stop the transaction.
 - To insert a split point into the recorded test, click the **Split point** icon . With split points, you can generate multiple tests from a single recording, which you can replay in a different order with a schedule.
9. After you finish the user tasks in the client program, stop the recorder. You can do this by closing the client program or by clicking the button **Stop**  in the **Recorder Control** view.
- If you changed the network settings of the client program as described in step 8, you can revert to the default settings before closing the program.

Result

The **Generate Service Test** wizard opens.

10. If you inserted a split point during the recording, on the **Destination** page, specify the location for the split test or merge the split recordings together.
- See [Splitting an HTTP test during recording on page 194](#) for more information about splitting tests.
11. On the **Service Test Generation Options** page, if you are testing a SOAP-based web service, specify a Web Services Description Language (WSDL) file from the workspace or click **Add** to import a WSDL or to link to a remote WSDL file and click **Next**.
12. Select the domains to include in the test and click **Finish**. The domains that are not selected are not included in the test. You can add them back by generating the test again from the recording.
- To include all the domains for all of the recordings, click the **Select all and remember my decision** check box.
- To enable the filter again for HTTP tests, click **Window > Preferences > Test > Test Generation > HTTP Test Generation**, and, for Service tests, click **Service Test Generation** and then click the **Enable domain review before test generation** check box.
13. Click **Finish**.

Results

A progress window opens while the test is generated. On completion, the **Recorder Control** view displays the `Test generation completed` message, the test navigator lists your test, and the test opens in the test editor.

Related information

[Recording a service test with the generic service client on page 229](#)

[Sending service requests with the generic service client on page 495](#)

[Recording sensitive session data on page 272](#)

[Sending service requests with WSDL files on page 508](#)

Preparing to record a test for the HTTP/2 service

To test a web service that is based on the HTTP/2 protocol, record a test by using the SOA extension of HCL OneTest™ Performance. Before recording the HTTP/2 service, follow the procedure in this topic to configure your computer.

About this task

This configuration is required because this feature is released as Beta and is intended for use in a non-production environment only.

Use Mozilla Firefox or Google Chrome when recording on servers that support the HTTP/2 service.

1. Download the following Application Layer Protocol Negotiation (ALPN) boot jar file <https://mvnrepository.com/artifact/org.mortbay.jetty.alpn/alpn-boot/8.1.8.v20160420>
2. Create or rename the `productInstallDir\jdk` folder to `..\jdk.ibm`. You can rename the folder back to `jdk` later to test with the IBM JDK.
3. Download Oracle Java 1.8.0u92 from <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>. You can then either extract the compressed file or install Java at `productInstallDir\jdk`.
4. Copy the ALPN jar file to `productInstallDir\majordomo\lib`.
5. From `productInstallDir`, open `eclipse.ini` and add the following flags:

```
-Xbootclasspath/p:<productInstallDir>\majordomo\lib\alpn-boot-8.1.8.v20160420.jar
```



Note: If there are any other flags starting with `-X`, delete those flags.

6. Configure HCL OneTest™ Performance Agent to use Oracle Java.
 - a. Stop the Majordomo process.

On Windows systems, run the following command:

```
cd "c:\program files\hcl\hclonetest\majordomo"
ngastop
```

On Linux systems, run the following command:

```
cd /opt/HCL/HCLOneTest/Majordomo ./MDStop.sh
```

- b. Set the environment variable `RPT_JAVA` to the Oracle Java binary or executable file.

On Windows systems, run the following command:

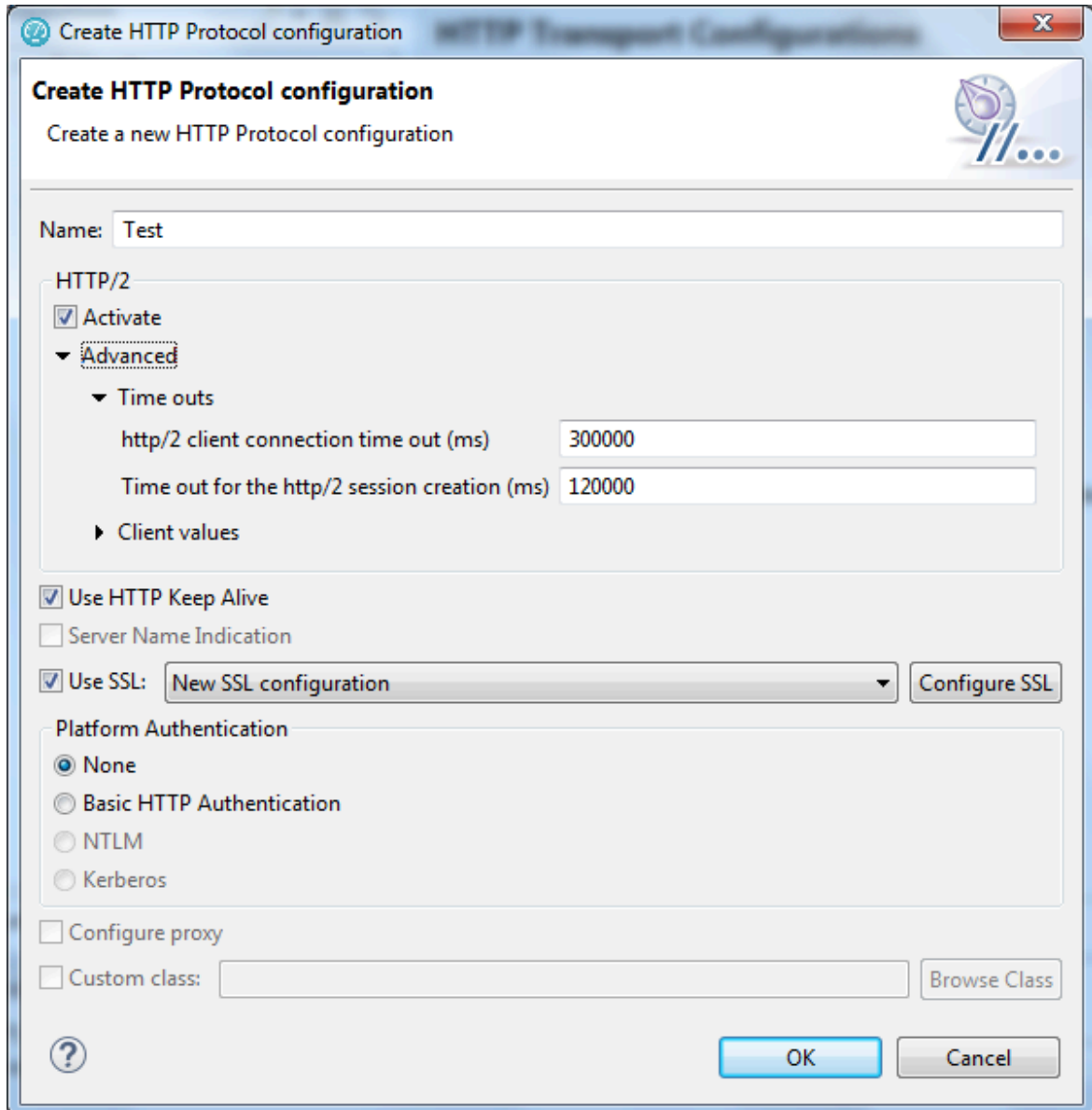
```
set RPT_JAVA=c:\program files\java\jdk1.8.0_92\bin\java.exe
```

On Linux systems, run the following command:

```
export RPT_JAVA=/root/jdk1.8.0_92/jre/bin/java
```


- c. Start the Majordomo process.

7. When you record a service by using a web browser against an HTTP/2 client, the HTTP/2 traffic is automatically captured. But, to record an HTTP/2 service by using the GSC client, in the HTTP transport protocol configuration dialog box, you must select the **Activate** check box.



8. Specify the following configuration options for HTTP/2:

HTTP/2

 **Note:** Testing HTTP/2 service is in the Beta mode. For more information, see [Preparing to record a HTTP/2 service on page 235](#).

To test a service that uses the HTTP/2 protocol, select the **Activate** check box. This check box is automatically selected when you record a service by using a browser. If you use the Generic Service Client component to create a HTTP/2 test, you have to manually select the check box.

HTTP/2 client connection timeout

Specifies the time limit for the HTTP/2 client to connect to the HTTP/2 server.

Time out for the HTTP/2 session creations

Specifies the time limit to create the HTTP/2 session. This time starts after the connection is established.

Enable HTTP/2 Push

The Push functionality of HTTP/2 automatically identifies and passes the related objects or requests to the client when a request is sent to the server. Clear the check box to not use the functionality.

Initial session window

Specifies the buffer size on the sessions.

Initial stream window

Specifies the window size for buffer on each stream after the connection is established.

HTTP/2 Client Input Buffer Size

Specifies the buffer size that is used to read the network traffic.

Maximum Quantity of Messages that can be queued

Specifies the maximum number of messages that can be queued for the HTTP/2 client on a thread.

Maximum Quantity of HTTP/2 thread pool

Specifies the maximum number of thread pools that will be used by the HTTP/2 client to distribute the workload.

Minimum Quantity of HTTP/2 thread pool

Specifies the minimum number of thread pools that will be used by the HTTP/2 client to distribute the workload.

HTTP/2 client bytearray pool size

Specifies the buffer size to receive the unciphred values.

Server Name Indication



Note: Not applicable for HTTP/2.

Clear this check box if you do not want to connect to the host computer by using the Server Name Indication protocol. If the host computer is already configured with Server Name Indication protocol, you should keep this check box selected.

Use HTTP Keep Alive

Select this option to keep the HTTP connection open after the request. This option is not available if you are using IBM® Rational® AppScan®.

Use SSL

Select this option to use an SSL configuration. Click **Configure SSL** to create an SSL configuration or select an existing configuration.

Platform Authentication

In this section, specify the type of authentication that is required to access the service. Select **None** if no authentication is required.

Basic HTTP authentication

Select this option to specify the **User Name** and **Password** that are used for basic authentication.

NTLM authentication

Note: Not applicable for HTTP/2.

Select this option to use the Microsoft™ NT LAN Manager (NTLM) authentication protocol. NTLM uses challenge-response authentication. This view lists what is negotiated (supported by the client and requested of the server) and what is authenticated (the client reply to the challenge from the server).

Kerberos authentication

Note: Not applicable for HTTP/2.

Select this option to use the Kerberos authentication protocol between the client and server.

Connect through proxy server

Note: Not applicable for HTTP/2.

If the HTTP connection needs to go through a proxy server or a corporate firewall, specify the **Address** and **Port** of the proxy server. If the proxy requires authentication, select either **Basic proxy authentication** or **NTLM proxy authentication**.

Proxy authentication

In this section, specify the type of authentication that is required to access the proxy. Select **None** if no authentication is required.

Basic proxy authentication

Select this option to specify the **User Name** and **Password** that are used for basic authentication.

NTLM proxy authentication

Select this option to use the Microsoft™ NT LAN Manager (NTLM) authentication protocol. NTLM uses challenge-response authentication. This view lists what is negotiated (supported by the client and requested of the server) and what is authenticated (the client reply to the challenge from the server).

Custom class



Note: Not applicable for HTTP/2.

Select this option if the communication protocol requires complex, low-level processing with a custom Java™ code to transform incoming or outgoing messages. Click **Browse** to select a Java™ class that uses the corresponding API. This option is not available in IBM® Security AppScan®.

9. Click **OK**. You have configured the workbench to test an HTTP/2 service.

What to do next

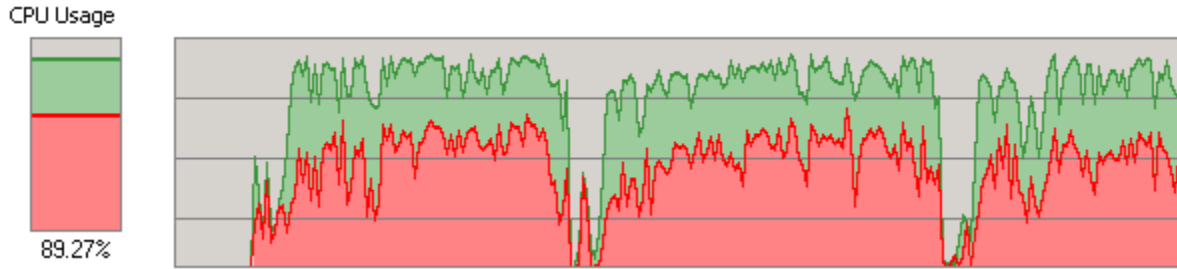
You can now record a regular SOA test for the HTTP/2 service. After the recording, in the Version field of request details, the requests are marked with HTTP/2 indicating that the HTTP/2 traffic is captured. If the test playback fails, check if all the steps are correctly followed.

Optimizing HTTP/2 tests for SOA

HTTP/2 tests require a lot of CPU and memory resources. When you apply load on HTTP/2 tests using computers that do not have enough resources, the tests might fail. You might want to configure or tune the computers that run HTTP/2 tests.

CPU Usage

Ensure that the HTTP/2 tests get adequate CPU resources to run. If there are other processes running on the computer and they are not required, you can stop them. For example, the CPU usage statistic in the image below indicates that the other processes (shown in red) on the computer are consuming a lot of resources whereas the test execution process (shown in green) is getting less resources.



Memory usage and garbage collection

Ensure that enough memory is available for the test execution. You can configure the garbage collector and adjust the memory heap size.

Garbage Collection - Consider using the following values so that the garbage collector does not allocate large amount of temporary memory. By doing so, you are tuning the number of threads allocated for the garbage collector according to the capability of the computer. You apply the values for each location asset of the schedule.

RPT_VMARGS

`-XX:MaxGCPauseMillis=250 -XX:ParallelGCThreads=6 -XX:ConcGCThreads=3 -XX:GCTimeRatio=19`

General Properties

ROOTDIR	=	/tmp/cloud_agent
OPERATING_SYSTEM	=	LINUX
CLOUD_ROLE	=	CLOUD_AGENT
RPT_VMARGS	=	-XX:MaxGCPauseMillis=250...
LOCATION_TEMPLATE	=	/myProj/Cloud-SL.loctemp...
RPT_ENABLE_IP_ALIASING	=	FALSE
RPT_IPA_ENABLE_ALL_INTE...	=	TRUE

Memory heap - Consider using the following values for memory heap:

RPT_VMARGS

`-Xms11024m`

RPT_DEFAULT_MEMORY

`22412m`

▼ General Properties

ROOTDIR	=	/tmp/cloud_agent
OPERATING_SYSTEM	=	LINUX
CLOUD_ROLE	=	CLOUD_AGENT
RPT_VMARGS	=	-Xms11024m
LOCATION_TEMPLATE	=	/myProj/Cloud-SL.loctemp...
RPT_ENABLE_IP_ALIASING	=	FALSE
RPT_IPA_ENABLE_ALL_INTE...	=	TRUE
RPT_DEFAULT_MEMORY	=	22412m

Thread Usage

Ensure that you start load testing with fewer virtual testers and gradually ramp up the workload. This practice helps in observing the changes in the workload, that is, the number of calls per second. In the graph, when the number of calls per second is flat, it indicates that the maximum capacity of the computer is reached and there is no need to add more virtual testers.

Platform tuning

Configure the TCP/Socket capabilities of your system by following these two links:

https://wiki.eclipse.org/Jetty/Howto/High_Load

<https://msdn.microsoft.com/en-us/library/aa560610%28v=bts.20%29.aspx>

Creating a service test from a BPEL model

You can use Business Process Execution Language (BPEL) resources from your workspace to automatically generate a set of service tests that corresponds to the paths that are run in a synchronous BPEL model.

Before you begin

Tests are stored in test projects. If your workspace does not contain a test project, the test creation wizard creates one, enabling you to change its name. To store a test in a specific project, verify that the project exists before you record the test.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

If you are using Java™ Message Service (JMS), ensure that you have configured the environment with the correct libraries and configuration files. Ensure that the WSDL files use the correct syntax for the test environment.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

BPEL models must be synchronous. Asynchronous BPEL models are not supported.

Ensure that the BPEL models refer to the WSDL files in a valid import statement, for example:

```
<bpws:import importType="http://www.w3.org/2001/XMLSchema" location="foo.wsdl" namespace="http://foo"/>
```

Relative file paths, such as: `../../foo.wsdl` are not supported.

Ensure that you have one or more valid Web Services Description Language (WSDL) files and the associated BPEL model in your workspace. Only the calls to services with a valid web service binding are taken into account. For example, if the BPEL model was produced in IBM® Websphere Integration Developer, then services must be exported with the following web service bindings:

```
<bpws:invoke name="myOperation" operation="myOperation" partnerLink="IServicePartner"
portType="ns3:IService" wpc:displayName="myOperation" wpc:id="20">
```

Only BPEL *invoke* activities are considered for generating tests. Any BPEL *receive* and *reply* activities are ignored.

Websphere Integration Developer does not generate the required `soapAction` attributes for the soap operations in the WSDL files. Please edit the generated WSDL files, as follows for every operation: `<soap:operation soapAction="" />`.

To create a service test from a BPEL model:

1. In the Performance Test perspective, click **File > New > Other > Test > Test Assets > BPEL to Web service test**, and then click **Next**.
2. Click **Browse** to select a BPEL file from the workspace, and click **Next**.
3. On the **Web service test generation** page, change the number of paths by specifying how activities and sequences from the flow of the BPEL model are processed. Each path generates one test.
 - a. In the **Flow** section, select how any concurrent sequences that are found in the flow will be converted into paths.
 - b. In the **Switch** section, select whether to test *otherwise* activities from the flow.
 - c. In the **Throw** section, select how *throw* activities from the flow are converted into paths.
 - d. In the **Invoke** section, select whether to test inline catches inside *invoke* activities from the flow.
 - e. Select **Enable data correlation in generated tests** to automatically create references in the generated test elements by propagating variables to the parameters of the web service call and message return elements.
4. Click **Recount paths** to update the number of paths to test, and click **Next**.
One test is generated for each path.
5. For WSDL operations that are bound to multiple ports, you must select one port that is to be used for the test.

Under each test that will be generated, the **Operations** list displays the WSDL operations that are bound to multiple ports.

If no WSDL operations are displayed under the tests, this means that all operations are bound to a single port. In this case, skip step 6.

- a. In the **Operations** list, expand a test and select a WSDL operation that requires binding.
- b. In the **Binding ports** list, select the port that you want to use to test the selected WSDL operation.
- c. Repeat steps a and b for each WSDL operation that requires binding.

6. Click **Next**.
7. Select a location and a name for the new folder where the tests generated from the BPEL model are created, and click **Finish**.

Results

A new folder is created in the Test Navigator containing the generated service tests. These tests are generated with default message content and must be edited with valid input values.

Creating a service test manually

You can create a service test without recording by simply adding the test elements as required and manually editing the test element details in the test editor.


Before you begin

Tests are stored in test projects, which are test projects that include a source folder. You must create a test project before creating a test.

Ensure that you have a valid WSDL file in your workspace. Ensure that the WSDL files use the proper syntax for the test environment.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the proper libraries and configuration files.

1. In the workbench, click **File > New > Other > Test > Test Assets > Web service test** or click the **New Service Test**  toolbar button.
2. Select a project and, in **Name**, type a name for the test, and then, click **Next**.
The name that you type is the base name for the recording, test, and other required files. You see these files in standard Navigator or the Java™ Package Explorer with their distinguishing suffixes, but you see only the simple (test) name in the Test Navigator.
3. Select a web service request to create the test for.
If you select **Web service request** or one of the options in **Specification-based structure**, specify a WSDL port and then configuration properties for the HTTP protocol. If you select, **XML request** and **Text request**, specify the configuration properties for the HTTP, JMS, WebSphere MQ, WebSphere Java MQ, and Microsoft.Net protocols.

For information about the configuration properties of each protocol, see the topics in [Sending service requests with the generic service client on page](#) .
4. Click **Finish**. The service test is created.

Creating a service test for WebSphere® MQ

You can create an IBM® WebSphere® MQ test by adding the test elements as required and editing the test element details in the test editor.


Before you begin

Tests are stored in test projects, which are Java™ projects that include a source folder. You must create a test project before creating a test.



Ensure that you have a valid Web Services Description Language (WSDL) file for a WebSphere® MQ service in your workspace.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

1. In the workbench, click **File > New > Other > Test > Test Assets > Web service test** or click the **New Service Test**  toolbar button.
2. Select a project, and then, in **Test file name**, type a name for the test and click **Next**.
The name that you type is the base name for the recording, test, and other required files. You see these files in the standard Navigator or the Java™ Package Explorer with their distinguishing suffixes, but you see only the simple (test) name in the Test Navigator.
3. In the **Select a service request interface** page, complete one of the following steps:
 - a. To test a service that use a WSDL file, select **Web service request** or **Specification-based structure**, click **Next**, and select a WSDL file.
 - b. To test a service that does not use a WSDL file, select **XML Request**, **Text Request**, **Binary Request** or an **Empty test**.
4. Click **Next** and select the **WebSphere MQ** protocol.
5. In **SOAP Action**, specify the SOAP action to be used to invoke the MQ request.
6. To override the message header and descriptor that was specified in WebSphere MQ transport configuration, click **Override MQ Protocol Configuration values** and specify the customize header and message descriptor.
7. Click **Finish**. The service test is created.
8. On the web service call, click **Update Response**.
This opens the **Response Preview** window, displaying the data that will be used to perform the call.
9. Click **Update Test**.
This action calls the web service and creates a message return element with the return data. If a message return element already exists, then it is updated with latest return data. With the message return test element, you can implement data correlation and content-based verification points.

Creating a service test for WebSphere Java MQ


To test Java-based applications, create a service test and add the WebSphere Java MQ messages. You can create a service test by using Generic Service Client option  or the New Service Test wizard .

Before you begin

Connect to a WebSphere MQ server.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

1. In the workbench, click **File > New > Other > Test > Test Assets > Web Service Test** or click **Create a Service Test** .
2. Select a project, and then, in **Test file name**, type a name for the test.
The name that you type is the base name for the recording, test, and other required files. You see these files in the standard Navigator or the Java™ Package Explorer with their distinguishing suffixes, but you see only the simple test name in the Test Navigator.
3. In the **Select a service request interface** page, complete one of the following steps:
 - a. To test a service that use a WSDL file, select **Web service request** or **Specification-based structure**, click **Next**, and select a WSDL file.
 - b. To test a service that does not use a WSDL file, select **XML Request**, **Text Request**, **Binary Request** or an **Empty test**.
4. Click **Next**, select the **WebSphere Java MQ** protocol, and specify a transport configuration. If necessary, click **New** to create the transport configuration for the call. See [Creating a WebSphere Java MQ transport configuration on page 502](#).
5. Complete the following information in the **General** tab:



Learn more about the UI elements in the General tab:

Queue

Name of the queue as defined on the WebSphere MQ server.

Message type

The types of messages are these:

- *Datagram* means that the message does not require a reply.
- *Request* means that the message requires a reply.
- *Reply* means that the message is a reply to an earlier request message.
- *Report* means that the message is reporting on some expected or unexpected occurrence, usually related to some other message. An example is a request message that contained data that was not valid.

**Message Persistence**

This value indicates whether the message is persistent or not. If the message is persistent, it survives the system failures and restarts of the queue manager. If the message is not persistent, it survives a restart if it is present on a queue having the NPMCLASS(HIGH) attribute. However, even with the NPMCLASS(HIGH) attribute a message does not survive a QMGR class. Nonpersistent messages on queues having the NPMCLASS(NORMAL) attribute are discarded at queue manager restart, even if the message is found on the auxiliary storage during the restart procedure.

Dynamic Reply

Select this check box for the WebSphere MQ server to dynamically create a temporary queue as a reply. If this check box is not selected, the message in Reply Queue is used.

Reply Queue

This is the name of the message queue to which the application that issued the get request for the message should send the reply and report messages.

Reply Manager

This is the name of the queue manager on which the reply-to queue is defined.

Additional properties

Specify the additional properties for the queues.

6. **Optional:** If necessary, complete the following information on the **Config** tab:

**Learn more about the UI elements in the Config tab:****Message Priority**

This is the priority of the message. The lowest priority is 0.

Encoding

This is the numeric encoding of numeric data in the message. This value does not apply to numeric data in the MQMD structure itself.

Expiry Interval

This is the period of time, in tenths of a second, after which the message becomes eligible to be discarded if it has not already been removed from the target queue. The expiry interval is set by the application that put the message.

**Character set**

This is the character set identifier of the character data in the application message data.

7. **Optional:** In the **Report** tab, select the report messages to receive.
8. **Optional:** If necessary, complete the following information in the **Context** tab:

**Learn more about the UI elements in the Context tab:****Application Identity Data**

This information is defined by the application suite. Use it to provide information about the message or its originator.

Application Origin Data

This information is defined by the application suite. Use it to provide additional information about the origin of the message.

Accounting Token

This information is needed by the application to appropriately charge for the work that is done as a result of the message.

User ID

This is the user identifier of the application that originated the message.

9. **Optional:** In the **Identifiers** tab, for the messages that require binary input, specify the ID in the string format in the second column. The first column is filled automatically in the hexadecimal format.
10. **Optional:** In the **Segmentation** tab, select the segment of the message and click **Next**.
11. If you had selected **XML Request**, click **Next**, select a XSD file and click **Finish**.

Result

The new service test is created.

What to do next

You can now enhance the test and run it.

Creating a service test for a plain XML call


You can create a test for a plain XML call over HTTP, JMS, or IBM® WebSphere® MQ, by simply adding the test elements as required and editing the test element details in the test editor.

Before you begin

Tests are stored in test projects, which are Java™ projects that include a source folder. You must create a test project before creating a test.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

1. In the workbench, click **File > New > Other > Test > Test Assets > Service Test** or click the **New Service Test**  toolbar button.
2. Select a project, and then, in **Test file name**, type a name for the test and click **Next**.
The name that you type is the base name for the recording, test, and other required files. You see these files in the standard Navigator or the Java™ Package Explorer with their distinguishing suffixes, but you see only the simple (test) name in the Test Navigator.
3. On the **Select Service Call Interface** page, select whether you want to create a test using a plain **XML call** interface or a **Web service call** interface.
If you select web service call interface, select or add a WSDL file and then, select port to which the call will be binded. Click **Next**.
4. On the **Configure Protocol** page, select either **HTTP**, **JMS** or WebSphere® **MQ** as the protocol and then, specify the options for the selected **Protocol configuration**.
5. On the **Select Root Element** page, you can select an XSD and then, select a root element for the call.
6. Click **Finish**.

Changing service test generation preferences

You can change default test generation values by changing the preference settings. The default settings, however, are appropriate for recording in most cases.

1. Click **Window > Preferences > Test > Web Services Test Generation**
2. Select the setting to change.

Time out delay used for call

This is the default time out for web service calls. If the web service does not respond within this period, an error is produced.

Think time default value

This is the default think time for generated tests.

3. After changing a setting, click **Apply**.

Recording socket and TN3270 tests

When you record a test, the test creation wizard records your interactions with the application under test, generates a test from the recording, and opens the test for editing.

Socket performance testing guidelines

Before you can test the performance of TCP/IP socket-based applications, set up your test environment and incorporate these guidelines to produce reliable performance tests.

Limitations

You can use this extension to test applications that run in a client-server model, where the test simulates multiple clients that connect to one or several servers. Other models, such as peer-to-peer networks, are not supported.

HCL OneTest™ Performance does not support socket recording in the 64 bit versions of Microsoft Windows 2003 and Windows XP. Also, you cannot record 64 bit applications on 64 bit Windows 10 and Windows 2016 systems.

Performance

When you deploy performance tests, use a relevant number of virtual users on a given computer is important. For example, if you deploy too many virtual users on a single computer, the results will reflect more the load of the test computer than the load of the server.

For best results with performance tests on an average test computer with a 1 GHz processor and 1 GB of RAM, do not exceed 1000 concurrent virtual users.

If you exceed the number of virtual users that a single test computer can run, the measured performance of the server will be affected by the performance of the test computer, which will invalidate the final results.

When editing a schedule for long performance tests, use these guidelines:

- In the schedule editor, reduce the **Test log level** to **None**.
- In the schedule editor, set the **Statistics sample interval** to approximately 1/60 of the run time, for example 12 minutes for an estimated 12-hour session.
- When possible, use loops inside test suites rather than loops in the schedule. Using loops inside test suites avoids connection problems that might occur over long duration tests and emphasizes measurement of the send and receive activity rather than connection and close activity.

SSL/TLS Authentication

Socket tests support simple or strong Secure Sockets Layer (SSL) or Transport Layer Security (TLS) authentication mechanisms, also called server authentication and client authentication.

For server authentication, the client must determine whether the server can be trusted. When you are recording or running a socket test with a proxy recorder, the proxy recorder sits between the server and the client. Therefore, you must "trick" the client application into behaving as though the proxy recorder is the certified server by performing either one of the following actions:

- Configure the SSL or TLS settings of the recorder proxy to authenticate itself as the actual server to the client and as the client to the service. This means that you must supply the recording proxy with the adequate certificates.
- Configure a managed client (an external client application) to accept the proxy recorder as though it were the certified server. The recording wizard provides a link to download and import an HCL OneTest™ Performance certificate into the client application.

For client authentication, the server must authenticate the test client according to its root authority. Therefore, you must provide the client certificate that is expected by the server to authenticate the proxy recorder or the test agent as a certified client.

See [Digital certificates overview on page 258](#) for more information about managing digital certificates.

TN3270 performance testing guidelines

Before you can test the performance of TN3270 terminal applications, set up your test environment and incorporate these guidelines to produce reliable performance tests.

Limitations

You can use this extension to test applications that run on a TN3270 terminal emulation client, where the test simulates multiple terminals that connect to one or several servers.

These TN3270 terminal emulation packages are supported:

- IBM® Personal Communications
- Attachmate EXTRA! X-treme

Performance

When deploying your performance tests, use a relevant number of virtual users on a given computer. For example, if you deploy too many virtual users on a single computer, the results reflect more the load of the test computer than the load of the server.

For best results with performance tests on an average test computer with a 1 GHz processor and 1 GB of RAM, do not exceed 1000 concurrent virtual users.

If you exceed the number of virtual users that a single test computer can run, the measured performance of the server is affected by the performance of the test computer, which invalidates the final results.

When editing a schedule for long performance tests, use these recommendations:

- In the schedule editor, reduce the **Test log level** setting to **None**.
- In the schedule editor, set the **Statistics sample interval** value to approximately 1/60 of the run time, for example 12 minutes for an estimated 12-hour session.
- When possible, use loops inside test suites rather than loops in the schedule. Using loops inside test suites avoids connection problems that might occur over long-duration tests and emphasizes measurement of the send and receive activity rather than connection and close activity.

Recording a socket API performance test

You can record a socket API test from any client program on your computer. When you record, the recording wizard automatically starts the client program and records all the data that transits through the socket API.

Before you begin


Tests are stored in performance test projects. If your workspace does not contain a performance test project, the test creation wizard creates one with a name that you can change. To store a test in a specific project, verify that the project exists before you record the test.

Ensure that you have a working client program and that you can connect to the server.

Ensure that the session that you are recording is reproducible. This means that when the recorded actions are replayed by the test, the same responses from the server will be received.

HCL OneTest™ Performance does not support socket recording in the 64 bit versions of Microsoft Windows 2003 and Windows XP. Also, you cannot record 64 bit applications on 64 bit Windows 10 and Windows 2016 systems.

To record a socket test:

1. In the Performance Test perspective, click the **New Test from Recording** toolbar button  or click **File > New > Test from Recording**.
2. In the **New Test from Recording** wizard, click **Create a test from a new recording**, select **Socket Test**, and click **Next**.

If you are recording sensitive data, you can make a selection in **Recording encryption level**.

3. On the **Select Location** page, select the project and folder to create the test in, type a name for the test, and click **Next**.

If necessary, click **Create Parent Folder**  to create a new project or folder.

4. On the **Select Client Application** page, select the type of client program to use to record the test:

Choose from:

- To specify any client program that is located on your computer, select **Managed Application**, and click **Next**.

On the **Managed Application Options** page, click **Browse** to specify the **Program path**. If necessary, specify the **Working directory**, and in **Arguments** type the command-line arguments that the program requires.

If the program requires user input from a command-line interface, select **Open console for user input**.

- To record a TN3270 terminal emulation session, select **IBM Personal Communication** or **Attachmate EXTRA! X-treme** if these programs are installed, and click **Next**.

If required, specify a session file to start the TN3270 session.



Note: Using this method to record a TN3270 session produces a low-level socket API performance test that is based on the TN3270 protocol traffic. To record a TN3270 test, see [Recording a TN3270 performance test on page 254](#).

- To record an HTTP session, select **Microsoft Internet Explorer** or **Mozilla Firefox**, and click **Next**.

If you choose **Mozilla Firefox**, you can specify a Firefox profile.



Note: Using this method to record an HTTP session produces a socket API performance test that is based on the HTTP traffic. To record an HTTP test, see [Recording an HTTP test on page 183](#).

5. If the application uses Secure Sockets Layer (SSL) and Transport Layer Security (TLS) authentication to authenticate the server or the client application, specify the following options, and click **Next**:

Choose from:

- Select **The server requires a specific client certificate** if you are using client authentication. Specify a certificate keystore file name and password. If multiple certificates are required, click **Multiple certificates** and specify a certificate keystore file name and password for each host name and port.
- Select **The client requires a specific server certificate** to provide the certificate keystore file name of the server and a password for each host name and port.

If you do not provide the server certificate, you must configure the client application to authenticate the certificate of the proxy recorder as though the proxy recorder were the actual server. Click **Save this certificate** to save the certificate that is generated by HCL OneTest™ Performance, and import the `.cer` file into the client application.

If necessary, select whether to use SSL 3.0 and TLS 1.0 encryption. See [Socket performance testing guidelines on page 250](#) for more information about SSL and TLS authentication.








6. If this is the first time that you record a socket API performance test, read the **Privacy Warning**, and select **Accept** to proceed.
7. Click **Finish** to start recording.

Result

A progress window opens while the client program starts.

8. Use the client program to perform the actions to test.

You can use the **Recorder Test Annotations** toolbar to add comments, record synchronizations, or take screen captures during the recording.

- To add a comment to the recorded test, click the **Insert comment** icon .
 - To add a screen capture to the recorded test, click the **Capture screen** icon . Screen and window captures make your tests easier to read and help you visualize the recorded test. You can change the settings for screen captures and add comments to images.
 - To manually add a test synchronization to the recording, click the **Insert synchronization** icon .
 - To manually add a transaction folder to the recording, click the **Start Transaction** icon  and **Stop Transaction** icon  to start and stop the transaction.
 - To insert a split point into the recorded test, click the **Split point** icon . With split points, you can generate multiple tests from a single recording, which you can replay in a different order with a schedule. See [Splitting a test during recording on page 194](#) for more information about splitting a test.
9. When you have finished test actions in the program, stop the recorder. You can do this by closing the client program or by clicking the **Stop**  push button in the **Recorder Control** view.

Result

A progress window opens while the test is generated. On completion, the **Recorder Control** view displays the `Test generation completed` message, the Test Navigator lists your test, and the test opens in the test editor.

Related information

[Socket performance testing guidelines on page 250](#)

[Recording a TN3270 performance test on page 254](#)

[Recording sensitive session data on page 272](#)

[Recording an HTTP test on page 183](#)

[Recording service tests on page 225](#)

Recording a TN3270 performance test

You can record a TN3270 test from a terminal emulation client. When you record, the recording wizard automatically starts the terminal emulation client and records all the screen and input activity that transits the socket connection.



Before you begin

Ensure that you have a TN3270 terminal emulation program installed on the local computer.

Tests are stored in performance test projects. If your workspace does not contain a performance test project, the test-creation wizard creates one with a name that you can change. To store a test in a specific project, verify that the project exists before you record the test.







Ensure that the session that you are recording is reproducible. This means that when the recorded actions are replayed by the test, the same responses from the server will be received.


To record a socket test:

1. In the Performance Test perspective, click the **New Test from Recording** toolbar button  or click **File > New > Test from Recording**.
2. In the **New Test from Recording** wizard, click **Create a test from a new recording**, select **TN3270 Test**, and click **Next**.
If you are recording sensitive data, you can select a **Recording encryption level**.
3. On the **Select Location** page, select the project and folder where to save the new test, type a name for the test, and click **Next**.
If necessary, click **Create Parent Folder**  to create a new project or folder.
4. On the **Select Client Application** page, select the type of client program to use to record the test:
Choose from:
 - In most cases, select **IBM Personal Communication** or **Attachmate EXTRA! X-treme**, and click **Next**.
If required, specify a session file to start the TN3270 session.
 - If you are using other TN3270 terminal emulation software, select **Managed Application**, and click **Next**.
On the **Managed Application Options** page, click **Browse** to specify the program path. If necessary, specify the **Working directory**, and in **Arguments**, type the command-line arguments that the program requires.
If the program requires user input from a command-line interface, select **Open console for user input**.
5. If this is the first time you record a socket API performance test, read the **Privacy Warning**, and select **Accept** to proceed.
6. Click **Finish** to start recording.

Result

A progress window opens while the TN3270 terminal program starts.

7. Use the TN3270 terminal program to perform the actions to test.
You can use the **Recorder Test Annotations** toolbar to add comments, record synchronizations, or take screen captures during the recording.
 - To add a comment to the recorded test, click the **Insert comment** icon .
 - To add a screen capture to the recorded test, click the icon  **Capture screen**. Screen and window captures make your tests easier to read and help you visualize the recorded test. You can change the settings for screen captures and add a comment to the image.
 - To manually add a test synchronization to the recording, click the **Insert synchronization** icon .
 - To manually add a transaction folder to the recording, click the **Start Transaction** icon  and **Stop Transaction** icon  to start and stop the transaction.
 - To insert a split point into the recorded test, click the **Split point** icon . With split points, you can generate multiple tests from a single recording, which you can replay in a different order with a schedule. See [Splitting a test during recording on page 194](#) for more information about splitting a test.

- When you have finished test actions in the program, stop the recorder. You can do this by closing the TN3270 terminal program or by clicking the **Stop** push button  in the **Recorder Control** view.

Result

A progress window opens while the test is generated. On completion, the **Recorder Control** view displays the `Test generation completed` message, the Test Navigator lists your test, and the test opens in the test editor.

Related information

[TN3270 performance testing guidelines on page 251](#)

[Recording a socket API performance test on page 252](#)

[Recording sensitive session data on page 272](#)

Changing test generation preferences

You can change the way that the test recorder organizes multiple send and receive elements in a new socket test by changing test generation preferences. To improve the readability of your test, you can merge consecutive send or receive elements that use the same connection.

Before you begin

To change the way that test elements are organized by default in a new test, you can change the test generation preferences before recording the test.

To merge or reorganize elements in an existing test, you can use the **Organize** wizard. See [Merging socket send and receive elements on page 377](#) for more information.

To merge send or receive elements in a new socket test:

- Click **Window > Preferences > Test > Test Generation > Socket Test Generation**.

Result

The **Socket Test Generation** preferences window opens.

- Select **Strategies**.

You can create multiple organization strategies for handling different applications. Only one strategy is active during the recording.

- Select **Default Strategy** or click **New** to create an organization strategy.
- Click **Settings**.

- In **Edit Socket Strategy Settings**, specify how you want the test recorder to generate multiple send and receive elements:

Send elements

Merge consecutive send elements

Select this option to merge together all the consecutive socket send elements that use the same connection.

Manipulate data with custom code

Select this option to force all the selected send elements to enable the **Manipulate data with custom code** setting with the specified **Class name** of a custom Java™ class that uses the API to process data in the socket send element.

Receive Actions**Do not merge**

Select this option to keep receive elements unmodified as they are initially recorded.

Merge consecutive receive elements

Select this option to merge together all the consecutive socket receive elements that use the same connection.

Keep only last receive element

Select this option to discard all multiple consecutive receive elements except the last one recorded.

Response timeout

The maximum delay (in seconds) to receive the first byte of the response. If no data is received before the end of the response timeout delay, the receive action produces an error in the test log. The response timeout counter starts when the receive action starts after the think time; the counter is interrupted when the first byte is received.

End policy

This option specifies when to stop receiving data and to move to the next test element.

- **Receives exact number of bytes:** The receive action stops when the recorded number of bytes is received. Specify a **Timeout** (in seconds) after which the receive action produces an error in the test log, if the correct number of bytes is not received. If **Link data size** is enabled, the receive action expects the number of bytes displayed in the **Data** area. If **Link data size** is disabled, the receive action expects the number of bytes displayed in **Bytes**. This is the default setting
- **Receives until end of stream:** The receive action stops when the connection is closed by the remote computer. If **Accepts empty response** is selected, then the reception of a single byte is *not* required and the **Response Timeout** is ignored. Specify a **Timeout** (in seconds) after which the receive action produces an error in the test log, if the correct number of bytes is not received.
- **Matches a string:** The receive action stops when a specified sequence of bytes is received. Specify a **Timeout** (in seconds) after which the receive

action produces an error in the test log, if the correct number of bytes is not received.

- **Recognizes a regular expression:** The receive action stops when a sequence of bytes that matches a regular expression is received. Specify a **Timeout** (in seconds) after which the receive action produces an error in the test log, if the correct number of bytes is not received.
- **Delegated to custom code:** The receive action stops when a condition is met in a custom Java™ class. This setting allows great flexibility, but requires coding of a custom Java™ class following the HCL OneTest™ Performance extension API. Click **Generate Code** to generate a template based on the API or **View Code** to open the specified class in the Java™ editor.

Except when the **Receives until end of stream** policy is in force, receive actions produce an error in the test log when the connection is closed by the remote computer.

Timeout

For end policies that have a **Timeout** setting, this setting specifies a delay (in seconds) after which the receive action produces an error in the test log if the end policy criteria is not met. The timeout counter starts when the first byte is received.

6. Click **OK** to apply the changes, and close the **Preferences** window.

Digital certificates overview

The digital certificates feature enables you to run tests against servers that use Secure Sockets Layer (SSL) for applications that require client-side digital certificates to authenticate users.

A *digital certificate* is a file that binds a public cryptographic key with an identity (a user or an organization). Trusted certificate authorities issue digital certificates, which are then used to authenticate users and organizations for access to websites, email servers, and other secure systems. A *certificate store* is an archive file that contains almost any number of digital certificates, possibly certificates that are issued from different certificate authorities.

To use digital certificates in tests:

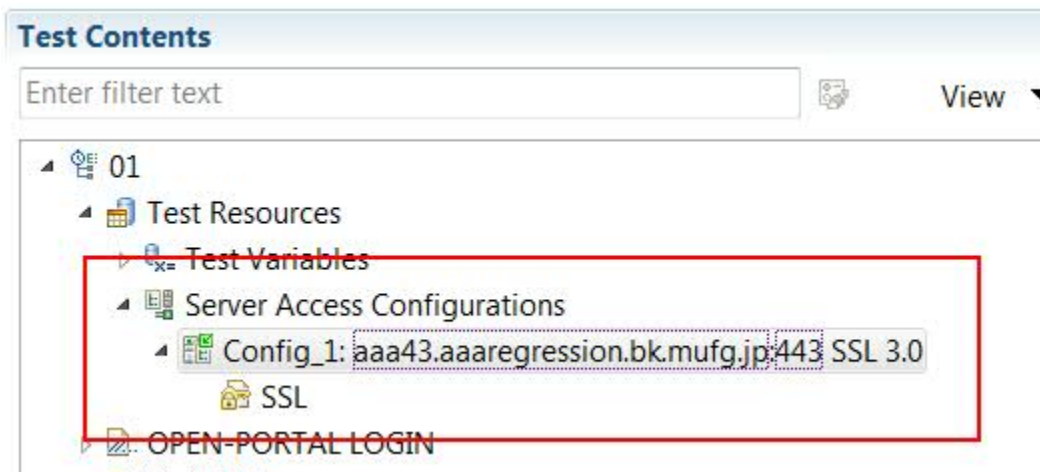
1. Create a digital certificate store. For more information about this subject, see [Digital certificate creation overview on page 259](#) and [Creating a digital certificate store on page 263](#).
2. Record a test that requires that you use a digital certificate. For more information about this subject.
3. Associate a digital certificate with a test for playback. For more information about this subject, see [Playing back a test with a digital certificate on page 266](#).
4. Optionally, you can associate the digital certificates in one or more digital certificate stores with a dataset. For more information about this subject, see [Using a digital certificate store with a dataset on page](#) .

Create a certificate store by running the supplied KeyTool command-line program. The program creates a certificate store that contains digital certificates.

Record a test that requires using a digital certificate. Specify the certificate and password that to use, and then begin recording the test. Browse the website as you typically would to record a test.

After you have finished recording, open the test for editing. On the Common Options page, under Digital Certificates, click **Add**. Type the name of the certificate store that you created previously; then select the certificate that you want to use. Save the test. When you run this test, the digital certificate from the certificate store is submitted to the server.

If you have recorded a test that does not use SSL, you can convert that test to be secure by adding an SSL object to the corresponding Server Access Configuration in the test.



To use a certificate store with a dataset, open the test for editing. On the **Common Options** page, click **Add Dataset**. Create a dataset with two columns that contains a list of the certificates in the certificate store and a list of passphrases for the certificates. Select **Fetch only once per user**. Save the dataset. On the Common Options page, under Digital Certificates, click **Add**. Select the certificate store that you created previously from the **Certificate Store** column. Insert a **Certificate Name** for the digital certificate. Highlight this name, and then select **Substitute from dataset**. Choose the dataset added previously, and then choose the column with the certificate name. Repeat this process to substitute passphrases from the dataset column containing passphrases. Save the test. Add the test to a schedule. When you run this schedule, the certificates from the certificate store are submitted to the server.

Digital certificate creation overview

If you want to use digital certificates to run tests against applications that require client-side digital certificates to authenticate users, work with the appropriate server administrators to determine the types of certificates that you need to create.

In cryptography, a public key certificate is a document that uses a digital signature to bind a public key with a physical identity. These certificates are often referred to generically as digital certificates or client digital certificates. The most common standard for digital certificates is the X.509 standard.

In public key cryptography, each certificate has two associated keys: a public key and a private key. The public key is incorporated into the X.509 certificate and is always available with the certificate itself. The private key is always kept private (meaning, it is never transmitted). For ease of portability, the two keys (and the certificate) can be included in one, encrypted and passphrase-protected, format known as PKCS#12.

In order to verify the authenticity of a certificate, it is digitally signed by another certificate, known as a Certificate Authority (CA). This CA certificate may be one created (and kept secure) by a company hosting a secure application, or it could be created by a company such as Verisign.

When a web application requires digital certificates, an administrator typically creates digital certificates for each authorized user. The administrator digitally signs each certificate using the system CA certificate. These certificates, along with the public and private keys, are distributed to users. Often these keys will be distributed in the PKCS#12 format. Users then import these certificates into their web browsers. When the browser is challenged by the server, it will produce its certificate.

When importing certificates for web applications, select the check box that indicates that the keys be exportable. With this indication, the certificate can be exported to a PKCS#12 formatted file for later use by other programs.

Do not use certificates that are assigned to actual users for performance testing purposes. Use test certificates that do not correspond to actual users.

There are four types of certificates that can be used in testing:

- Self-signed certificates
- Signed certificates
- Certificate authority (CA) certificates
- Unsigned certificates (rarely used)

Self-signed certificates are used when no entity needs to vouch for the authenticity of the certificate. These are the simplest certificates to create and use. Typically, however, a signed certificate is used to represent a particular user.

Signed certificates are used when a certificate needs to be created for and issued to one, and only one, user. Signed certificates are signed by a certificate authority (CA).

Certificate authority (CA) certificates are self-signed certificates used to sign (certify) certificates.

Unsigned certificates are certificates that are neither signed by a CA nor self-signed. Most web applications do not use unsigned certificates.

When you create a self-signed or signed certificate (including CA certificates) you can specify a *subject*. The subject of a certificate is the set of attributes of an X.500 Distinguished Name that is encoded in the certificate. The subject enables the recipient of a certificate to see information about the owner of the certificate. The subject describes the certificate owner, but is not necessarily unique. Think of subjects as entries in a telephone book; there can be multiple entries for Patel Agrawal, but each entry refers to a different person.

The subject can contain many different types of identifying data. Typically, the subject includes the following:

Attribute	Example
COMMON NAME (CN)	CN=Patel Agrawal
ORGANIZATION (O)	O=XYZ Corporation
ORGANIZATIONAL UNIT (OU)	OU=XYZ Software Group
COUNTRY (C)	C=IN
LOCALITY (L)	L=Bangalore
STATE or PROVINCE (ST)	ST=Karnataka
E-MAIL ADDRESS (emailAddress)	emailAddress=agrawal@xyz.com

This information can be typed as one string, using forward slashes to separate the data.

For example, the above subject would be typed as follows:

```
/CN=Patel Agrawal/O=XYZ Corporation/OU=XYZ Software Group/C=IN/L=Bangalore/ST=Karnataka/
emailAddress=agrawal@xyz.com
```

To learn more about using the supplied command-line program to create certificates, see [Creating a digital certificate store on page 263](#).

Creating a digital certificate with OpenSSL

You can use the OpenSSL program to create digital certificates for use with tests.

Before you begin

OpenSSL is available from the OpenSSL Project at <http://www.openssl.org/>.

1. Create a certificate authority (CA).

For the purposes of testing, this CA takes the place of a recognized CA on the Internet, such as VeriSign. You use this CA to digitally sign each certificate that you plan to use for testing.

- a. Create a certificate request (CSR) file. The "subject" (-subj) describes the user of the certificate. Enter dummy values as shown. The following command line sets the password for the certificate to `abcdefg`.

Example

```
openssl req -passout pass:abcdefg -subj "/C=US/ST=IL/L=Chicago/O=HCL Technologies/OU=HCL Software
Group/CN=Rational Performance Tester CA/emailAddress=hop@hcl.com" -new > waipio.ca.cert.csr
```

- b. Create a key file, `waipio.ca.key`, to store the private key.

This removes the password protection from the certificate request file so that you do not have to type the password every time you sign a certificate. Because the password protection has been removed, use the certificate request file for testing purposes only.

Example

```
openssl rsa -passin pass:abcdefg -in privkey.pem -out waipio.ca.key
```

- c. Create an X.509 digital certificate from the certificate request. The following command line creates a certificate signed with the CA private key. The certificate is valid for 365 days.

Example

```
openssl x509 -in waipio.ca.cert.csr -out waipio.ca.cert -req -signkey waipio.ca.key -days 365
```

- d. Create a PKCS#12-encoded file containing the certificate and private key. The following command line sets the password on the P12 file to `default`. HCL OneTest™ Performance uses password of `default` for all PKCS#12 files by default.

Example

```
openssl pkcs12 -passout pass:default -export -nokeys -cacerts -in waipio.ca.cert -out waipio.ca.cert.p12 -inkey waipio.ca.key
```

Result

You now have a CA certificate (`waipio.ca.cert`), which can be installed into the web server under test and a private key file (`waipio.ca.key`) that you can use to sign user certificates.

2. Create a digital certificate for a user.

- a. Create a CSR file for the user. Set the initial password to `abc`. Optionally, provide an appropriate subject.

Example

```
openssl req -passout pass:abc -subj "/C=US/ST=IL/L=Chicago/O=HCL Technologies/OU=HCL Software Group/CN=John Smith/emailAddress=smith@hcl.com" -new > johnsmith.cert.csr
```

- b. Create a private key file without a password.

Example

```
openssl rsa -passin pass:abc -in privkey.pem -out johnsmith.key
```

- c. Create a new X.509 certificate for the new user, digitally sign it using the user's private key, and certify it using the CA private key. The following command line creates a certificate which is valid for 365 days.

Example

```
openssl x509 -req -in johnsmith.cert.csr -out johnsmith.cert -signkey johnsmith.key -CA waipio.ca.cert -CAkey waipio.ca.key -CAcreateserial -days 365
```

- d. **Optional:** Create a DER-encoded version of the public key. This file contains only the public key, not the private key. Because it does not contain the private key, it can be shared, and does not need to be password protected.

Example

```
openssl x509 -in johnsmith.cert -out johnsmith.cert.der -outform DER
```

- e. Create a PKCS#12-encoded file. The following command line sets the password on the P12 file to default.

Example

```
openssl pkcs12 -passout pass:default -export -in johnsmith.cert -out johnsmith.cert.p12 -inkey johnsmith.key
```

Repeat this step to create as many digital certificates as needed for testing. Keep the key files secure, and delete them when they are no longer needed. Do not delete the CA private key file. You need the CA private key file to sign certificates.

Results

Now you can install the CA certificate (`waipio.ca.cert`) into WebSphere®. Optionally, create a user certificate specifically for your web server, and install it into WebSphere®.

You can use user certificates individually to record tests. To use the user certificates (`johnsmith.cert.p12`) during test editing and playback, compress them in ZIP format to a file with the `.rcs` extension. This creates a digital certificate store. To learn more digital certificate stores, see [Creating a digital certificate store on page 263](#). You can also import user certificates into your web browser to interactively test them in your environment.

Creating a digital certificate store

The KeyTool command-line program enables you to create a Rational® Certificate Store (RCS) file that contains digital certificates for use with tests. A Rational® Certificate Store (RCS) file is a compressed archive file that contains one or more PKCS#12 certificates. You can also use the KeyTool program to remove certificates from a certificate store.

About this task

HCL OneTest™ Performance acts as a proxy between the browser and the server application to record the data exchange. When a secured page is recorded using HCL OneTest™ Performance, the proxy certificate of the product is presented to the browser.

1. In the command line tool, navigate to the directory that contains the Keytool utility. By default, the utility is located at `C:\Program Files\HCL\HCLOneTest\jdk\jre\bin`.
2. Type the following command:

Example

```
keytool.exe -genkeypair -alias certificateName -keystore keystoreName -storepass password -validity 365 -keyalg RSA -keysize 2048 -storetype pkcs12
```

For additional information about parameters by certificate generation, review the official [keytool documentation](#).

Option	Description
<code>-genkeypair</code>	Generate public and private keys for key pair.
<code>-alias</code>	Alias for your certificate in the key store. You may never use it, but every new certificate in your key store must have its own alias.
<code>-keystore</code>	Name of the key store file, which will be generated as the result of the command. It holds your certificate and a corresponding private key. You can reuse this key store for next certificates that you might generate. One key store can contain many certificates.
<code>-storepass</code>	Password that protects your key store file. You will have to enter it every time you want to sign a document.
<code>-validity</code>	Number of days the certificate is valid. You can enter more than 365.
<code>-keyalg</code>	Algorithm to generate the cryptographic keys that is corresponding to your certificate. You can use RSA or DSA.
<code>-keysize</code>	Length of the cryptographic keys. The more the length the stronger the signature.
<code>-storetype</code>	Format of the key store file. PKCS#12 (a.k.a PFX) key stores can be understood by a lot of different programs and you can also import a PKCS#12 file in your Windows key store (just double click it and follow the instructions).

3. The certificate generation process prompts you to enter some information about you. Enter the information as prompted.

```

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd C:\Program Files\HCL\SDP\jdk\jre\bin

C:\Program Files\HCL\SDP\jdk\jre\bin>keytool.exe -genkeypair -alias my_certificate -keystore my-keystore.pfx -storepass my_password -validity 365 -keyalg RSA -keysize 2048 -storetype pkcs12
What is your first and last name?
  [Unknown]: Syed Attaullah
What is the name of your organizational unit?
  [Unknown]: HCL
What is the name of your organization?
  [Unknown]: HCL
What is the name of your City or Locality?
  [Unknown]: Chennai
What is the name of your State or Province?
  [Unknown]: TamilNadu
What is the two-letter country code for this unit?
  [Unknown]: IN
Is CN=Syed Attaullah, OU=HCL, O=HCL, L=Chennai, ST=TamilNadu, C=IN correct? (type "yes" or "no")
  [no]: yes_

```

4. If prompted for a password when using the keystore, enter the same password as provided on the command line.

Result

The key store file (.pfx) is stored in your current directory.

Results

You now have a digital certificate store that you can use with tests. Because the KeyTool program has many options, you might want to create an alias or script file to use to invoke KeyTool. Use KeyTool to create and add as many digital certificates as you want. If you want to create a dataset of the names of certificates in the certificate store, run KeyTool again with the `-list` option. This option writes a list of names that can then be imported to a dataset.

What to do next

Before you start recording the application that requires client certification, import the certificate to the HCL OneTest™ Performance project. For information about how to import the certificate and record a test, see the [Recording a test on page](#) topic.

You do not have to use the KeyTool command-line program to create a certificate store. It is possible to use existing PKCS#12 certificates with HCL OneTest™ Performance. PKCS#12 certificates can be exported from a web browser. PKCS#12 certificates encode the private key within the certificate by means of a password.



Note: Do not use certificates associated with real users. Certificates associated with real users contain private keys that should not become known by or available to anyone other than the owner of the certificate. An intruder who gained access to the certificate store would have access to the private keys of all certificates



in the store. For this reason, you must create, or have created for you, certificates that are signed by the correct certificate authority (CA) but that are not associated with real users.

Playing back a test with a digital certificate

After you create a digital certificate store and record a test using a digital certificate, you must associate the digital certificate with the test for playback.

Before you begin

You need to record a test using a digital certificate, and you need a digital certificate store file containing one or more PKCS#12 certificates.

If your certificate extension is not .rcs, then you need to zip the certificate, rename the extension to .rcs, and copy it to the root directory of the project.

To associate a digital certificate with a test for playback:

1. Open the test for editing.
2. On the Security tab, under Digital Certificates, click **Add**.
3. Select or type the name of the certificate store file that you created previously.
You must type or select the file name. You cannot browse to locate the file. The certificate store must be a Rational® Certificate Store (RCS) file. A Rational® Certificate Store file is a compressed archive file that contains one or more PKCS#12 certificates.
4. Select the digital certificate that you want to use, and then click **Select**.
5. When prompted to place the digital certificate in a dataset, click **No**. To learn more about substituting digital certificates, see [Using a digital certificate store with a dataset on page](#) .

Result



Note: If you add multiple certificates to the **Digital Certificates** list on the Common Options page, the first certificate that satisfies the request from the server (in the order by which the certificates were entered) is used during playback.

6. Save the test.

Results

When you run this test, the digital certificate from the certificate store is submitted to the server.

Entrust TruePass authentication overview

Entrust provides digital identity and encryption technologies to governments and private industry. With Entrust TruePass software users can authenticate with secure web applications without installing a digital certificate in their browsers. This makes it convenient for use in kiosks and other public user environments.

You can now run tests against servers that require Entrust TruePass authentication. Roaming mode with TruePass applet version 7.0 and later are supported. Local mode, and versions of the TruePass applet prior to 7.0, are not supported. Recording tests with Entrust TruePass applications works just as regular HTTP recording does.

The Entrust TruePass Authentication object is displayed in the test editor for tests that you record with Entrust TruePass applications. The **Version** field displays the recorded version number of the Entrust TruePass applet. The **Server Name** and **Port** are correlated fields. Click **Substitute** to use the **Data Sources** view to change the server or port number for playback. The **Application Context** displays where the Entrust application is mapped to in the application server. The **User Name** and **Passphrase** fields can be substituted with values from a dataset.

The screenshot displays the Test Editor interface. On the left, the **Test Contents** pane shows a tree view of test elements under the folder *Entrust Technology Portals*. The selected element is **www.entrust.com:443/Authentication**. To the right of the tree are buttons for **Add**, **Insert**, **Remove**, **Up**, **Down**, **Prev**, and **Next**. Below the tree is a **Common properties** section.

The **Test Element Details** pane on the right shows the configuration for **Entrust Authentication (Mode: Roaming)**. It includes the following fields and controls:

- Version:** 8.0
- Connection Information:**
 - Server Name:** www.entrust.com
 - Port:** 443
 - Substitute** button
 - Application Context:** /TruePassApplication
- Authentication Information:**
 - User Name:** testuser
 - Passphrase:** testPassphrase99

Kerberos overview

You can run HTTP tests against servers that use the Kerberos protocol for authentication.

Introduction

Kerberos is a security authentication protocol that requires users and services to provide proof of identity.



Note: Kerberos is supported only for HTTP tests on HCL OneTest™ Performance.

Supported environments

Kerberos is supported on HTTP for web servers running Internet Information Server (IIS) or WebSphere® with the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI). Additionally, the Key Distribution Center (KDC) must be part of the Windows™ Domain Controller Active Directory. Internet Explorer, Mozilla Firefox, Opera, Apple Safari, and Google Chrome browsers are supported for recording tests. Kerberos is not supported on other protocols, environments, or browsers. For example, a KDC running on Linux™ is not supported.

Tips

For best results when you record tests that use Kerberos authentication, specify the host by name, not by numeric IP address. Also, note that user information is case-sensitive. Specify user information using the exact logon name from the user account in Active Directory. The **User logon name** field in the properties for the user in Active Directory displays the correct user name in the correct case. To the right of the user name the realm or domain name is displayed in the correct case. For example:

- User ID: kerberostester
- Password: secret
- Realm: ABC.IBM.COM

User logon names of the form ABC\kerberostester are not supported.

Troubleshooting

Kerberos authentication is a complex process. If you encounter problems when you attempt to record and play back tests that use Kerberos authentication, change the problem determination log level to **All** and run the tests again with only one virtual user. To learn more about the problem determination log, see the help topic on changing the problem determination level. After running a test, the `CommonBaseEvents00.log` file on the agent computer contains information that can help you determine why Kerberos authentication failed.

Terms

Active Directory

Active Directory is an implementation of Lightweight Directory Access Protocol directory services created by Microsoft™ for use primarily in Windows™ environments. The main purpose of Active Directory is to provide central authentication and authorization services for Windows™ computers. With Active Directory, administrators can assign policies, deploy software, and apply critical updates to an organization.

Directory service

A directory service is a software application or set of applications that store and organize information about the users and resources of a computer network.

Generic Security Services Application Program Interface (GSS-API)

The GSS-API enables programs to access security services. The GSS-API alone does not provide any security. Instead, security service providers provide GSS-API implementations, typically in the form of libraries that are installed with their security software. Sensitive application messages can be *wrapped*, or encrypted, by the GSS-API to provide secure communication between client and server. Typical protections that GSS-API wrapping provides include confidentiality (secrecy) and integrity (authenticity). The GSS-API can also provide local authentication about the identity of a remote user or remote host.

Key Distribution Center (KDC)

The authentication server in a Kerberos environment is called the Key Distribution Center.

Lightweight Directory Access Protocol (LDAP)

LDAP is an application protocol for querying and modifying directory services running over TCP/IP. An LDAP directory tree typically reflects political, geographic, or organizational boundaries. LDAP deployments typically use Domain Name System (DNS) names for structuring the highest levels of the hierarchy. LDAP entries can represent many different types of objects including people, organizational units, printers, documents, or groups of people.

Simple and Protected GSS-API Negotiation Mechanism (SPNEGO)

SPNEGO is used when a client application attempts to authenticate to a remote server, but the authentication protocols supported by the remote server are unknown. SPNEGO is a standard GSS-API pseudo-mechanism. The pseudo-mechanism uses a protocol to determine which common GSS-API mechanisms are available, then SPNEGO selects one GSS-API mechanism to use for all future security operations.

Trust Association Interceptor (TAI)

The TAI is a mechanism that establishes a secure connection between WebSphere® and other application software.

Recording Kerberos applications with Internet Explorer

You must configure your browser before you attempt to record Kerberos applications.

Before you begin

The client computer must be a member of the domain for which single sign-on (SSO) has been defined.

To configure Internet Explorer to use Simple and Protected GSS-API Negotiation Mechanism (SPNEGO):

1. Log on to Windows™ with a user ID for the domain for which SSO has been defined.
2. Start Internet Explorer.
3. Click **Tools > Options**.
4. Click the **Security** tab.
5. Define the site to authenticate to using Integrated Windows™ Authentication. Depending on your enterprise policy, you define the site in either the **Local intranet** zone or the **Trusted sites** zone.

Choose from:

- To define the site in the **Local intranet** zone, select **Local intranet**. The instructions that follow assume that you are defining the site in the **Local intranet** zone.
 - To define the site in the **Trusted sites** zone, select **Trusted sites**. In the **Security Settings** for the **Trusted sites**, click **Automatic logon with current username and password**, not **Automatic logon only in Intranet zone**.
6. Click **Sites**.
 7. Click **Advanced**.
 8. Type the URL for the hostname for which you want to enable SSO. For example, type `http://abc.ibm.com`.
 9. Click **Add**.
 10. Click **OK**.

11. Click **OK** again.
12. Click the **Advanced** tab.
13. Scroll down to Security, and select **Enable Integrated Windows Authentication (requires restart)**.
This option is not available on Internet Explorer 5.5. Integrated Windows™ Authentication is always enabled on Internet Explorer 5.5.
14. Click **OK**.
15. Restart Internet Explorer.

Recording Kerberos applications with Mozilla Firefox

You must configure your browser before recording Kerberos applications.

Before you begin

The client computer must be a member of the domain for which single sign-on (SSO) has been defined.

To configure Mozilla Firefox to use Simple and Protected GSS-API Negotiation Mechanism (SPNEGO):

1. Start Mozilla Firefox.
2. In the location bar, type `about:config`.
3. In the **Filter** field, type `network.n`.
4. Double-click the `network.negotiate-auth.trusted-uris` preference.
The `network.negotiate-auth.trusted-uris` preference lists the sites that are permitted to engage in SPNEGO authentication with the browser.
5. In the **Enter string value** window, type a comma-delimited list of URLs of trusted domains.
6. Click **OK**.
7. **Optional:** If the application under test uses the advanced Kerberos feature called Credential Delegation, double-click the `network.negotiate-auth.delegation-uris` preference. Type a comma-delimited list of URLs of trusted domains.
The `network.negotiate-auth.delegation-uris` preference lists the sites to which the browser may delegate user authentication.

Generating tests that use Kerberos

You must supply your Kerberos user name and password when generating tests that use Kerberos.

About this task

The test generation process attempts to detect if Kerberos was used when a test was recorded. If the test generation process detects that Kerberos was used when a test was recorded, you are prompted for the Kerberos credentials.

The test generation process might not differentiate between Kerberos authentication and NT LAN Manager (NTLM) authentication. In that case, you are prompted for the authentication type and credentials.

1. In the **Kerberos** window, type in the **Kerberos Realm name** field the Kerberos realm name that you used during recording.
If the test generation process cannot determine which type of authentication was used, the **Authorization** window is displayed, not the Kerberos window. In that case, click the **Kerberos** radio button before typing the Kerberos realm name.
2. In the **User name** field, type the user name that you used during recording.
3. In the **Password** field, type the password that you used during recording.

Results







The test generation process creates a Kerberos configuration file. The file is `krb5.ini` and it is stored in the root of the project workspace. This file is required to play back Kerberos tests. Typically, you need to ensure that the workbench computer that you use to record the test is in the same Kerberos realm as the agent computers that you use to play back the test. Advanced users and security administrators can edit this file with a text editor to tailor it to a specific test environment.


Annotating a test during recording

You can add comments, add transactions, or change a page name while you record a test. The advantage of adding these elements during (rather than after) recording is that you can place the annotations in the test exactly where you want. In addition, because annotations are part of the recorded test, they are regenerated when you regenerate the test. You can also insert split points into a test during record.

1. Start recording the test. The **Recorder Test Annotations** toolbar opens near the top of the screen.
2. Click the appropriate icon.

You can use the **Recorder Test Annotations** toolbar to add comments, record synchronizations, or take screen captures during the recording.

- To add a comment to the recorded test, click the **Insert comment** icon . You are prompted for a comment.
- To add a screen capture to the recorded test, click the **Capture screen** icon . Screen and window captures make your tests easier to read and help you visualize the recorded test. You can change the settings for screen captures and add a comment to the image.
- To manually add a synchronization point to the recording, click the **Insert synchronization** icon .
- To manually add a transaction folder to the recording, click the **Start Transaction** icon  and **Stop Transaction** icon  to start and stop the transaction. Transactions can be nested.
- To insert a split point into the recorded test, click the **Split point** icon . With split points, you can generate multiple tests from a single recording, which you can replay in a different order with a schedule. See [Splitting a test during recording on page 194](#) for more information about splitting a test.

- When recording an HTTP test, to change the page name, click the **Change page name** icon . In the resulting test, the page element in the test editor uses the new name, however the original name is preserved in the **Page Title Verification Point** area so that page title verification points still work correctly.
3. Close the client program to stop the recording.
 4. If you inserted a split point during the recording, on the **Destination** page, in the **Test Generation** wizard, specify the location for the split test or merge the split recordings together.


Results

The test is generated with the comments, transactions, and page names that you added.

Recording sensitive session data

You can keep recording session (.recsession) files to view the contents of a recording or to regenerate tests. However, if a recorded test contains sensitive information, you can choose to obfuscate, or encrypt, text strings in the recsession file.

To protect test data in a recording session file:

1. In the Performance Test perspective, click the **New Test from Recording** toolbar button  or click **File > New > Test from Recording**.
2. In the **New Test from Recording** window, select **Create a Test from a New Recording**, and select the type of test to create.
3. In **Recording encryption level**, select one of these options:

Choose from:

 - **Obfuscated:** This setting hides text strings to prevent viewing the raw data in recsession files with a text editor outside of the workbench. You can still use recsession file to generate tests and to view recording information.
 - **Passphrase:** This setting uses an AES-128-bit algorithm to encrypt text strings in the recsession files. The encryption strength depends on the length of the passphrase. The recording session file is unrecoverable if the passphrase is lost.
4. On the **Select Location** page, select the project and folder locations to contain the new test, type a name for the test, and click **Next**.
5. If you selected **Passphrase**, on the **Passphrase Protection** page, type the passphrase twice in **Passphrase** and **Confirm passphrase**.

For solid protection, make the passphrase longer than 24 characters if using English words or at least 12 random characters.
6. Click **Next**, and continue the recording session for the type of test that you selected.

Related information

[Creating tests on page 180](#)

Splitting an HTTP test during recording


You can insert split points when you record a test. With split points, you can generate multiple tests from a single recording that you can replay in a different order with a schedule. You can also create a schedule that contains all of the tests that are generated from the split points.

About this task

During the recording process, you can select the option to create a schedule for the tests that are generated from the split points. The schedule will contain these attributes:

- One user
- One user group for the local computer
- All of the tests from the recording, in serial order
- One stage: Run until finished
- Recorded think times, with the maximum think time set to 2 seconds
- Statistics:
 - Statistics log level: All
 - Statistics sample interval: 5 Seconds
 - Only store All Hosts statistics
- Test Log:
 - Show errors and failures: All
 - Also show warnings: All
 - And also show all other types: All
- Problem Determination log level: Warning

To insert split points when you record a test:

1. Start recording the test. The **Recorder Test Annotations** toolbar opens near the top of the screen.
2. To insert a split point into the recorded test, click the **Split point** icon . The **Insert Split Point** window is displayed.

Choose from:

- Click **Test name**, and then type a name for this section of the test.



Tip: You are naming the previous section of the test, not the upcoming section of the test.

Repeat this step between recorded user actions as needed to split tests.

The **Test Generation** window displays the status of generating the tests and schedule and the data correlation. You can view the test generation log from this window.

- When test generation is complete, you can select the test to open and then click **Open Selected Tests**, or you can click **Close** to finish this process.

Results

The schedule and tests are generated using the names that you specified in the wizard.

Generating a new test from a recorded session

You can generate a new test from a recorded session. For example, if you accidentally damage a test during editing, or if you want to change a test preference, you can regenerate the test instead of re-recording it. If split points were inserted in the recording, you can choose to generate a single test without split points.

To regenerate a complete test from a recording that contains split points:

- In the test navigator, select the .recession file of the test recording to regenerate.
- Right-click, and then select **Generate Test**.

Result


The **Generation Test** wizard is displayed.

- If the .recession file is compatible with multiple test types, select the type of test that you want to generate and click **Next**.

Example

For example, select **HTTP Test** to generate an HTTP performance test.

- On the **Select Location** page, select the project and folder where you want to create the test, type a name for the test, and click **Next**.

If necessary, click  **Create the parent folder** to create a new performance test project or a folder

- If the .recession file contains split points, on the **Options** page, select **Generate test without split points** if you want to regenerate the test as a single test.
- Click **Finish**.

Results


The test is regenerated and opened in the test editor.

Organizing test assets by type

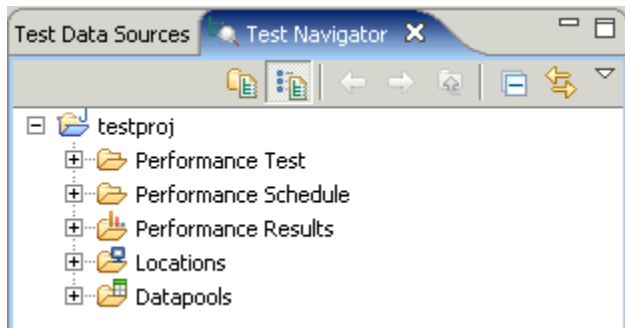
By clicking an icon, you can view your test assets in a logical order, in separate folders for tests, schedules, results, locations, and datasets.

About this task

In the Test Navigator view, you can click **Show Java Content** to see custom code that you created, click **Show Missing Resources** to view unresolved references, and click **Show File Extensions** to view file extensions of test assets.

- On the Test Navigator toolbar, click the **Show the logical test navigator**  icon.
To see the Test Navigator view, click **Windows > Show view** and click **Test Navigator**.

2. Your assets are now grouped logically. To see them, open the appropriate folder.



Note: In the Logical view, only if the asset is available the appropriate folder is displayed. For example, if the Results folder is displayed only after you have executed a test.

Editing tests

After you record a test, you can edit it to include datasets (to provide variable data rather than the data that you recorded), verification points (to confirm that the test runs as expected), and data correlation (to ensure that returned data is appropriate for the corresponding request). You can also add protocol-specific elements to a test. When you edit a test, the modified items appear in italic type. The italic type changes to regular type after you save the test.

Editing HTTP tests

After you record a test, you can edit it to include variable data rather than the data that you recorded and can include verification points to confirm that the test runs as expected. You can also edit the test to include [NB1] transactions, conditional processing, custom code, and standard header or custom header to a request or response.

Redirection support for HTTP tests

When you run HTTP tests, redirect requests are followed automatically, which supports common usage patterns, such as load balancing.

HTTP redirect responses are responses to requests with status codes in the 300 family, which indicate that the requested content is found at a different location. Redirect responses include HTTP status codes such as `301 Moved Permanently` and `302 Found`. Some HTTP applications redirect clients to a specific URL, but the ultimate response to the client request can be handled by one of several servers to balance the load that each server handles. For example, a request that is sent to `http://www.example.com/` might be redirected to `http://www-1.example.com` or `http://www-2.example.com`, depending on traffic and load conditions.

Both expected redirect and unexpected redirect responses are supported when you run tests. Expected redirect responses occur when you record tests. Unexpected redirect responses are received from the server when you run tests, but the responses are not present in the recorded test.

Expected redirect responses are handled by automatic data correlation. To automatically correlate host names and port numbers, click **Window > Preferences > Test > Test Generation > HTTP Test Generation > Data Correlation**, and

then select the **Automatically correlate URL pathname if redirected by response** check box. This option is selected by default.

Unexpected redirect requests are followed until an HTTP status code that is not a redirect response, such as `200 OK`, is returned by the server, or until the maximum number of redirect responses has been reached. By default, the maximum number of redirect responses to follow is 10. When a `200 OK` response is received, references that use the data in the final response are created.

Server access configurations are updated dynamically for unexpected redirect responses. For example, for a `config_1` server access configuration, where the host is `abc.example.com` and the port is 80, if a request that uses that configuration is redirected to port 8080 on the `xyz.example.com` host, all subsequent requests in the test that use the same configuration use port 8080 on the `xyz.example.com` host.

Verification points in a request are applied to the final destination. You can set a `ResponseCode` verification point in a request so that an unexpected redirection can be handled appropriately. If you set an exact `ResponseCode` verification point, it fails when a redirection occurs. A relaxed verification point also fails if the status codes in the 300 family are not part of the relaxed code list.

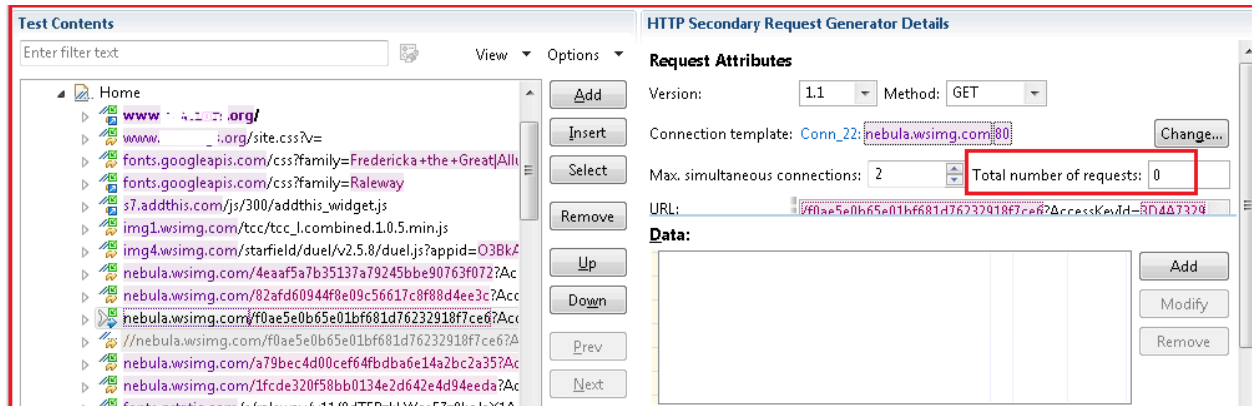
Creating secondary HTTP requests

A recording creates multiple HTTP requests and responses. In some cases, a response from the server can be dynamic, because of which the subsequent requests might need to be modified. While playing back the test, some of these dynamic requests might fail. For example, recording and playback might involve a different set of users with different permission settings or the UI elements might have changed since the time you recorded the test. To ensure that the test is played back without the need to record it again, you can create secondary requests which, based on the response received from the server, fetch the exact values that the test requires .

About this task

You identify the HTTP request for which to create a secondary request, and then create a reference for all the occurrences to dynamically generate the request. If you do not want to create a reference, you can define an array variable that lists the HTTP requests to use from playback. You can use the array variable from custom code.

From version 9.1.0.1 or later, you can also specify the number of requests to be send to the server in the Test Editor itself. After creating the secondary HTTP request, in the HTTP Secondary Request Generator Details area, specify the total number of requests. If the number of requests is greater than 0, the number that you specify in **Total number of requests** takes precedence over the array variable.



You can also use the delays to control the flow of the requests to the server. In the **Advanced** tab, go to the Delay Between Requests section and select the parameters for the flow of the requests. In **Release When**, select when exactly to release the request. For instance, First Character Sent indicates to release the second request after the first character in the first request is sent.

To create a secondary HTTP request:

1. Create a reference. See the [Creating a reference on page](#) topic.
2. Right-click the HTTP request for which to create the secondary request and click **Create Secondary HTTP Request Generator**. You can also select multiple HTTP requests that have common headers or connection attributes. If multiple requests do not have common attributes, you are prompted to select one request as template.

Result

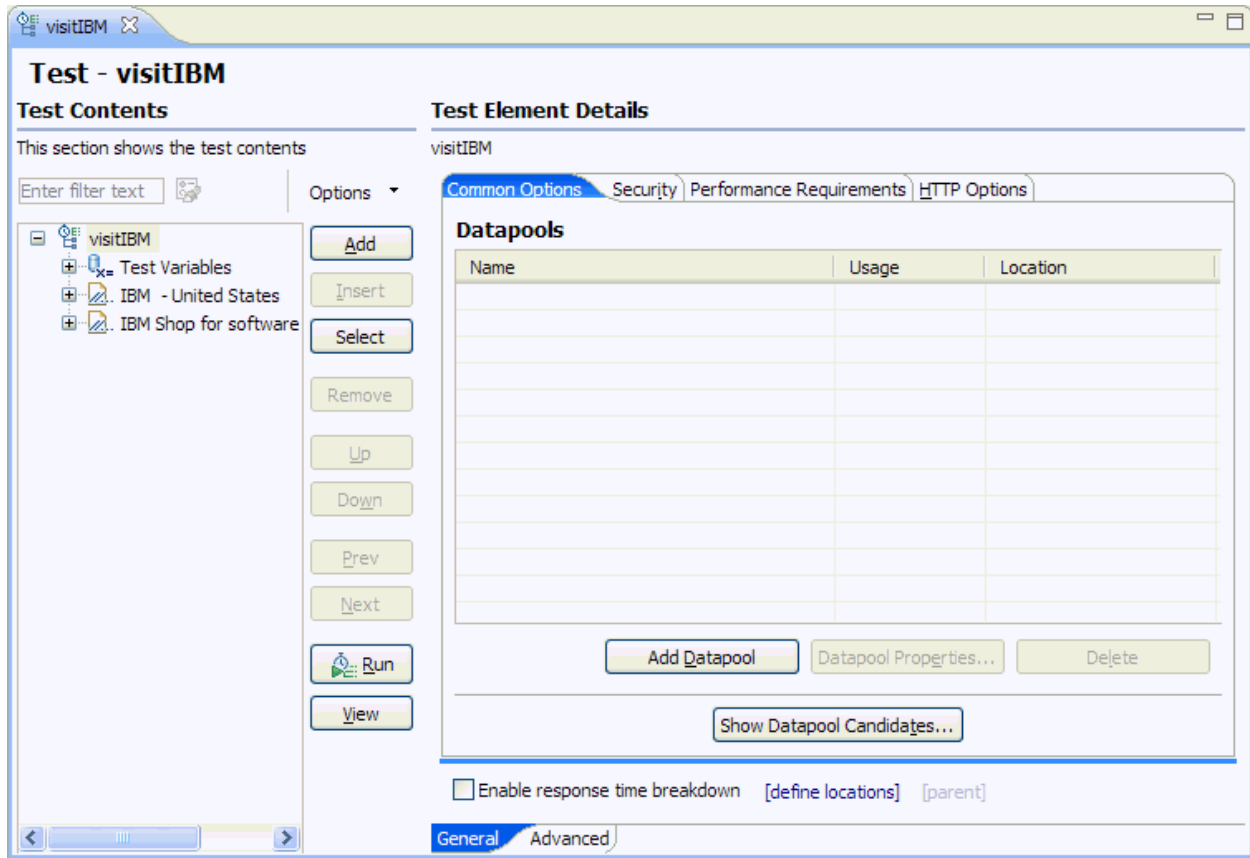
The original HTTP requests are disabled, indicating that the playback will now use only the secondary HTTP request.

3. Run the test.

HTTP test editor overview

With the test editor, you can inspect or customize a test that you recorded.

The test editor lists the HTTP pages for a test, by title. The following example shows the visitIBM test, which was generated from a recording of these tester actions: type `http://www.ibm.com`, under **Shop for** select **Software**, stop recording.



The test editor window contains two main areas. The area on the left, **Test Contents**, displays the hierarchy of the HTTP pages for the test. The area on the right, **Test Element Details**, displays common options and specific information about the HTTP protocol. The HTTP options apply to every page an HTTP test.

The **Test Variables** are listed at the top of the **Test Contents** area. These variables, which are the host names and ports in the test, are automatically created when the test is generated. Click a variable name to see where it is used in the test. By changing these variables, you can share or reuse the test for different configurations and web hosts. User-defined variables are also listed at the top of the **Test Contents** area. For more information on this subject, see related topics.

When you expand a test page, you see a list of the requests for the page, in separate folders, with names that are full web addresses minus the initial `http://`. The following example shows the expanded first page of the visitIBM test with the page selected in the **Test Contents** area. In this example, the settings that are displayed in the **Test Element Details** apply to the selected page.

Test - visitIBM

Test Contents
This section shows the test contents

Enter filter text

Options ▾

- visitIBM
- Test Variables
- IBM - United States**
- IBM Shop for software

Add
Insert
Select
Remove
Up
Down
Prev
Next
Run
View

Test Element Details
IBM - United States

Page title:

Primary request: www.ibm.com/us/en/

Think time: Milliseconds ▾

Test Data Options ▾

Name	Value	Substituted with
js	1	
ts	128568227947...	
lc	http://www.ibm...	
rs	1024x768	
cd	32	
ln	en	
tz	GMT -04:00	
jv	1	

URL Encode

Preview is not available.

Page Title Verification Point
 Enable verification point

The *primary request*, which is listed in bold, is the basis of the page title. The primary request can be the web address that the tester typed into the browser, a hyperlink that was clicked, or a web address to which the tester was redirected from another page request. In the example, the primary request shows that the tester was redirected to www.ibm.com/us/ from the initial page request (www.ibm.com). If the primary request does not return a page title, the test generator creates a unique name for it from the first node of the web address.

Some requests are highlighted in yellow. This highlighting indicates that these requests contain one or both of the following types of information:

- **A dataset candidate:** This is a value, usually one specified by the tester during recording, that the test generator determined is likely to be replaced by values in a dataset. An example of a dataset candidate is a string that you search for in a recorded test. The string is highlighted as a dataset candidate on the assumption that, before playback, you might want to associate the string with a dataset column that contains appropriate substitute values. For more information on this subject, see related topics.
- **Correlated data:** These are values in a test, usually one of them in a response and the other in a subsequent request, that the test generator determined needed to be associated in order to ensure correct test playback. An example is a photograph that is returned to the browser by a test that searches an employee database. The test generator automatically correlates employee names with photographs. Suppose that, before running

the test with many virtual users, you replace the employee name searched for in the recorded test with names in a dataset. Because the test correlates the data, each virtual user searches for a different employee, and the server returns an appropriate photograph. For more information on this subject, see related topics.



Note: To see an illustration of color coding in performance tests, click **Window > Preferences > Test > Test Editor**, and then click the **Colors and Fonts** tab.

When you expand a request, you see the **Response** data for the request. As shown in the following example, requests can also contain **Connection** data. Because the response is selected in the **Test Contents** area, the **Test Element Details** area displays the response data for this request.

The screenshot shows the Test - visitIBM interface. The **Test Contents** pane on the left displays a tree view of test elements, including a **Response** element selected. The **Test Element Details** pane on the right shows the details for the selected response, including the status (302 - Found), response data, response headers, and the content of the response.

Test Element Details

Response: 302 - Found

Response Data

Status: 302 Version: 1.1
Reason: Found

Response Headers

Header Name	Value
Date	Tue, 28 Sep 2010 13:57:54 GMT
Server	IBM_HTTP_Server
Content-Type	text/html
Expires	Fri, 01 Jan 1990 00:00:00 GMT
Pragma	no-cache

Content:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.ibm.com/us/
</body></html>
```

The **Response** data inside each request shows the data that the web server returned to the browser based on that request. Collectively, the requests listed inside a page are responsible for everything that was returned by the web server for that page.

Select multiple responses to display a table under **Test Element Details** that shows the following elements for the selected responses:


- Response code
- Response reason
- Response size
- Binary indicator
- Parent request
- Content verification point
- Response code verification point
- Response size verification point

Click **Add** to add child elements to the selected test element. Alternatively, you can right-click a test element and select an action from a menu. The choices that you see depend on what you have selected. For example, after you select a test, you can add a new page, a block of custom code, or an IF condition. After you select a page, you can add a request or an IF condition.

The **Insert** push button works similarly. Use it to insert a sibling element before the selected element.

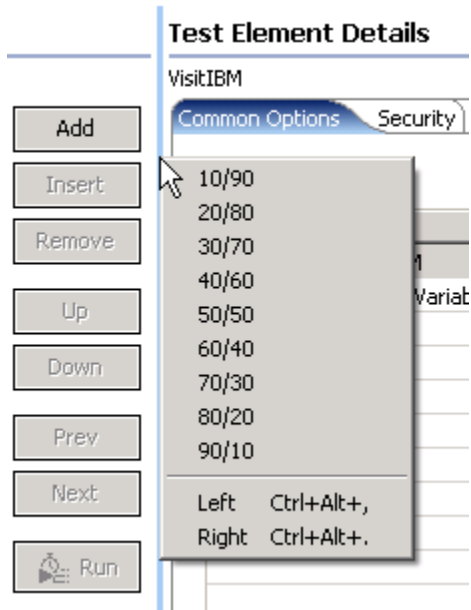
The **Remove**, **Up**, **Down** push buttons, and some **Add** choices (**HTTP Page**, **HTTP Request**, **Basic Authentication**) are primarily for use with tests that you write by hand; these actions or choices are likely to break a generated test. The types of structures that are commonly used in generated tests are explained in [Verifying expected behavior on page 292](#) and [Adding test elements on page 310](#).

If you test Siebel applications, see [Testing Siebel applications on page 310](#) for prerequisites and details about the differences between standard HTTP tests and Siebel tests.

A portion of the test editor can occasionally be obscured. To see the obscured area, move the cursor over one of the blue lines until it changes to , and drag up or down while holding the left mouse button.

To resize the Test Editor window, do one of the following:

- Click Ctrl+Alt+> or Ctrl+Alt+< to enlarge or reduce the window.
- Hover at the left side of the **Test Element Details** area. When you see a vertical blue line, right click the line and select a size ratio from the menu.



The new size remains the next time you open the window. Double click the blue line to return to a 50/50 ratio.

Related information

[Reusing tests on different hosts: Server connection variables on page 307](#)

[Providing tests with variable data \(datasets\) on page](#)

[Correlating response and request data on page](#)

Specifying the number of allowable URL redirects during test runs

When you run a test in a load-sharing environment, an unexpected redirection loop might occur during HTTP processing. An unexpected redirect response occurs when an HTTP request that normally returns a specific document redirects the browser to another location.

About this task

When the system detects an infinite loop of redirects, the infinite loop is broken, an error verdict for the request is issued, and the following message is displayed:

```
Infinite redirection loop detected getting URL n. If this is expected and understood, increase RPT_VMARGS
rptMaxRedirection parameter. Redirected history (from the first URL to the current one).
```

The default number of redirects is set at 10; however, you can edit the maximum number of redirects by updating the `RPT_VMARGS` argument in the `-DrptMaxRedirection` setting.

To modify the number of allowable redirects before an error is reported, complete these steps on the workbench.

1. In the Test Navigator, expand the project until you find the agent computer at the deployment location to change.

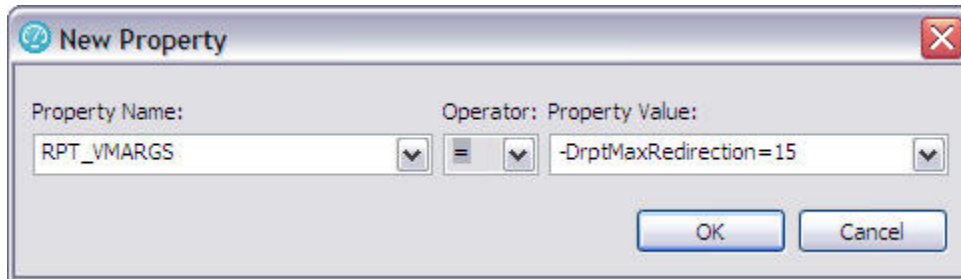
Agent computers are represented by the  icon.

2. Right-click the agent computer, and then click **Open**.
3. Under **Property Groups**, click **General Properties**, and then click **Add**.
4. In the **New Property** window complete these steps:
 - a. In the **Property Name** field, type `RPT_VMARGS`.
 - b. In the **Operator** field, confirm that the operator is `=`.
 - c. In the **Property Value** field, type `-DrptMaxRedirection=n`, where *n* is the maximum number of redirects that can occur before the error message is displayed, and then click **OK**.
 - d. Only one `RPT_VMARGS` argument is allowed for each agent computer location. If multiple `RPT_VMARGS` style properties are required, you must separate `-Dname=Value` with a space. For example,


```
-DrptMaxRedirection=15 -DanotherProperty=Value.
```

Result

The following **New Property** window sets the maximum number of redirects to 15:



Cutting and pasting in tests

You can cut, copy, and paste in HTTP tests.

The test editor supports the standard cutting, pasting, and copying of test elements by using the **Edit** menu or keyboard shortcuts. Test elements include HTTP pages, HTTP page elements, and requests. If you cut a test element, that element is not actually removed from the test until you next cut or paste.

If you copy a test element, that element is not actually copied until you next paste. For this reason, do not close the test from which you copy a test element until you have pasted the test element into another test. If you copy a test element from a test, and then close the test, nothing is pasted when you attempt to paste the test element. When you cut an element, it becomes unavailable (gray) and is displayed in italics. When you paste an element, it is displayed in italics until you save the test.

If you copy a request from one test to another and the connection details are same, the connections are copied. If the server URL in the another test is different, the connection detail is not copied. Click **Change** to add a new connection.

Request Attributes

Version: Method: - Primary request for page

Connection: [Conn_2: api.bing.com 80](#)

URL:



Note: Cutting and pasting can break correlations between test elements. For example, it is possible to cut a test element that contains a reference that a later portion of the test requires. When you cut, copy, or paste HTTP test elements, you are not warned of potential data correlation problems. You must ensure that the editing operation does not cause a data correlation error.

Defining requirements in tests

You can define requirements for elements in a test. These requirements specify acceptable thresholds of performance and validate service level agreements. Starting from version 9.2.0.1, you can define both performance and functional requirements in the tests. The verdict of the test is computed based on the requirements defined in the schedule. You can view the verdict in the Requirements report.

About this task

You can set requirements on protocol-specific test elements, on schedule elements, on data created by custom code, and on collected resource usage data. You define a requirement as *standard* or *supplemental*. A standard requirement determines that the requirement is significant enough to cause the entire run to be declared a failure if it fails. A supplemental requirement, although important, is not significant enough to cause the run to fail. For example, a supplemental requirement might be a request from development to validate a very specific data item provided by WebSphere® PMI monitoring.

To define a requirement for the elements in a test:

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. In the **Test Contents** area, select the page or the request that will have the requirement.
You can select multiple pages or multiple requests.
3. In the **Test Element Details** area, click the **Advanced** tab, and select **Enable Requirements**.

Result

A table of requirements that apply to the page or to the request is displayed.

4. Click the requirement to define, and add a definition, as follows:

Option	Description
Name	You can change the name of a requirement to improve readability. However, changing a requirement name causes a mismatch between the Requirements report, which uses the changed name, and the other reports, which use the default name. Therefore, when you change a requirement name, be sure to keep track of the original name.
Operator	Select an operator.
Value	Type a value.
Standard	Select to make the requirement <i>standard</i> . A standard requirement can cause a test to have a verdict of fail. Clear to make the requirement <i>supplemental</i> . In general, supplemental requirements are used for requirements that are tracked internally. A supplemental requirement cannot cause a run to fail, and supplemental results are restricted to two pages of the Performance Requirements report.

5. Optionally, apply the defined requirement to other test elements:

a. In the **Test Contents** area, select the test elements that will have the requirement.

The elements must be of the same type, for example, all page elements.

b. In the Requirements table, right-click the requirement row, and select **Copy Requirements**.

6. Optionally, select **Hide Undefined Requirements** to hide the shaded rows, which indicate that a requirement is not defined, and improve readability.

7. Select a requirement and click **Clear** to remove its definition. The requirement is still available and can be redefined.

8. After you have defined a number of requirements on test elements, you might want to see all of the requirements defined for the test. To do so:

a. In the **Test Contents** area, click the name (root) of the test.

b. In the **Test Element Details** area, select the **Requirements** category.

Result

The **Requirements** page displays a summary of the performance and functional requirements defined in the test.

c. To navigate to the original requirement definition, double-click the requirement row.

Example

You can define requirements in a test or in a schedule. When you define a requirement in a test, the requirement is defined individually for each test element—even if you select multiple test elements and apply the requirement to all of them at the same time. When you define a requirement in a schedule, the requirement is applied to the aggregate of test elements.

For example, assume that you select every page in a test and define this requirement: `Average response time for page [ms] [For Run]` must be less than 5 seconds. This means that if one page in the test has a response time of 6 seconds, the requirement on that page fails. The other pages, which have a response time of less than 5 seconds, pass.

Assume that you open a schedule and define this requirement: `Average response time for all pages [ms] [For Run]` must be less than 5 seconds. This measures the average response time for all of the pages. One page can have a response time of 30 seconds, but if enough pages have a response time low enough to counter the negative effect of that one page, the requirement passes.

For information on defining requirements in schedules, see [Defining requirements in schedules on page 558](#).

Adding an authentication folder

Web application servers can include an option to force a login. You might have recorded a test with this option disabled but want to run the test with the option enabled. Adding an authentication folder to the appropriate test request lets you do this without recording the test again.

To add an authentication folder to a request:

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. Click the request that will contain the authentication folder.
3. Click **Add** and select **Basic Authentication**.

Result

A folder named **Authentication** is added to the request, and the **Test Element Details** area displays the **Userid**, **Password**, and **Realm** fields.

Adding or removing header in batches

You can add or remove HTTP header from multiple HTTP requests or responses in batches to improve the script efficiency. For example, during the development process, there might be HTTP headers change or addition of new HTTP headers to the requests. These changes in the HTTP headers will result in the test script failure.

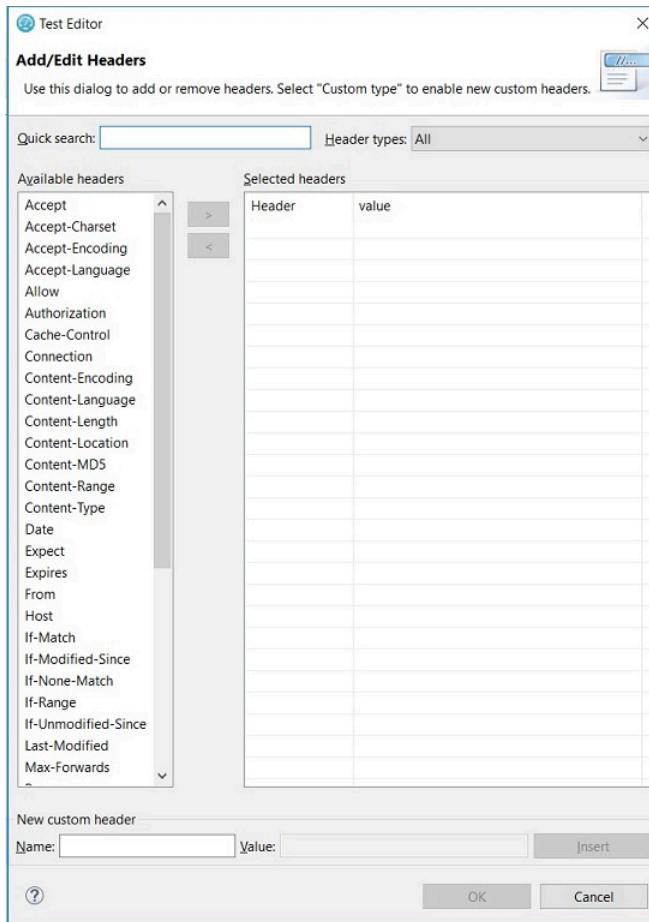
Adding HTTP header to multiple HTTP requests

During the development process, there might be a change in HTTP headers. To modify the test script in a faster and easier way, you can add the HTTP headers in batches to the requests.

About this task

To add HTTP headers to multiple HTTP requests at once:

1. Open the test.
2. In the test hierarchy, select a request (press **Ctrl** key to choose multiple requests).
3. From the **Options** drop-down list, select **Add > HTTP Header**. The **Add/Edit Headers** window opens.



4. To add a standard header:
 - a. In the **Available Headers** list, locate the header to add and click it. Type the name of the header in the **Quick search** field and select the type of header that you are looking for in **Header types** list to quickly locate a header in the **Available Headers** list.
 - b. Click the right-angle bracket (>). The selected header moves into the **Selected headers** list and your cursor is placed in the value column.
 - c. Type the value for the header.
5. To add a custom header:

- a. In the **Header types** list, select **Custom**.
- b. At the bottom of the window, in the **New custom header** area, type the header information in the **Name** field and the **Value** field, and then click **Insert**. The custom header is added to the **Selected headers** list.
- c. When you have finished adding headers, click **OK**.

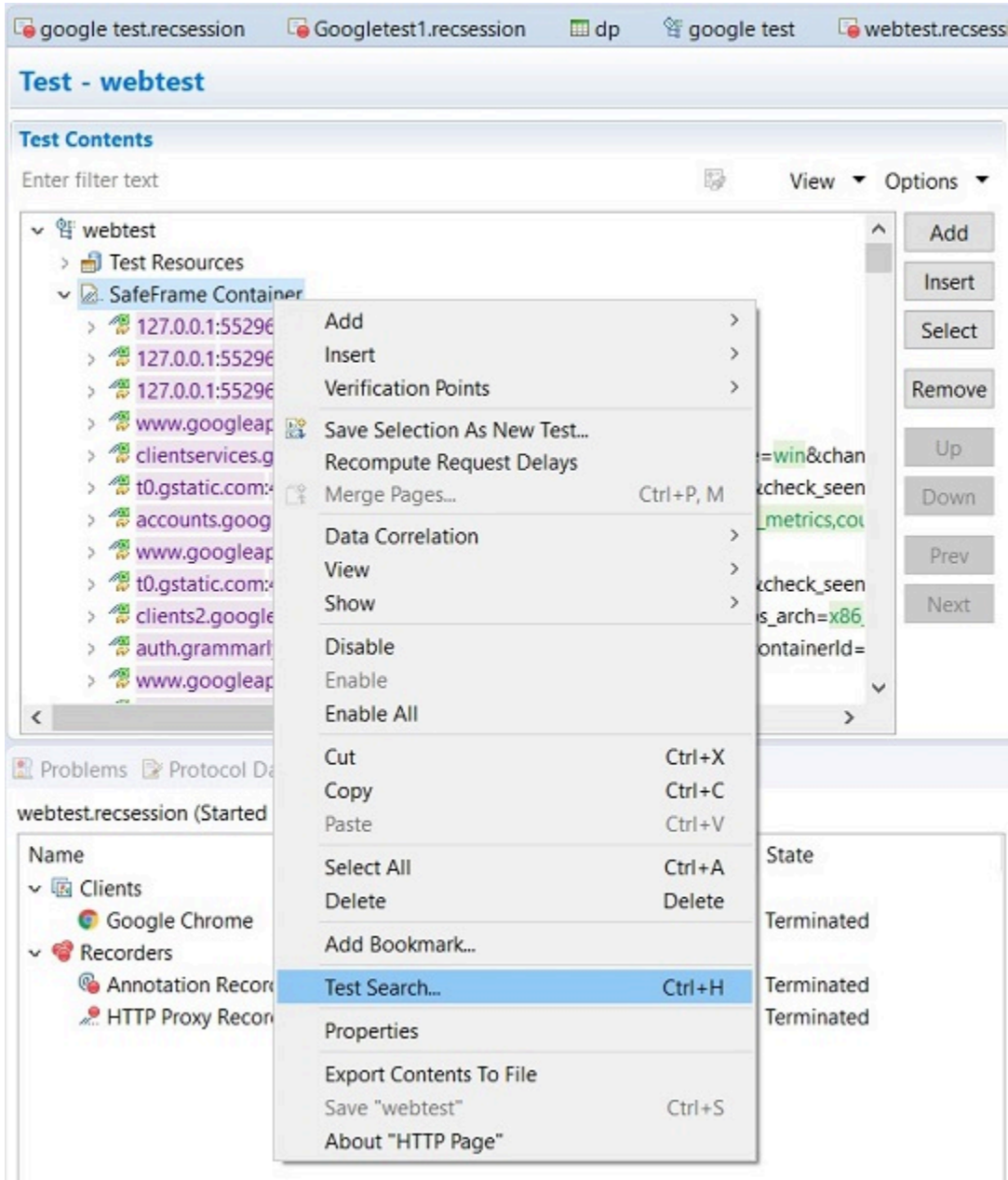
Removing HTTP header from multiple HTTP requests

During the development process, there might be addition of new HTTP headers. To modify the test script in a faster and easier way, you can remove the HTTP headers in batches from the requests.

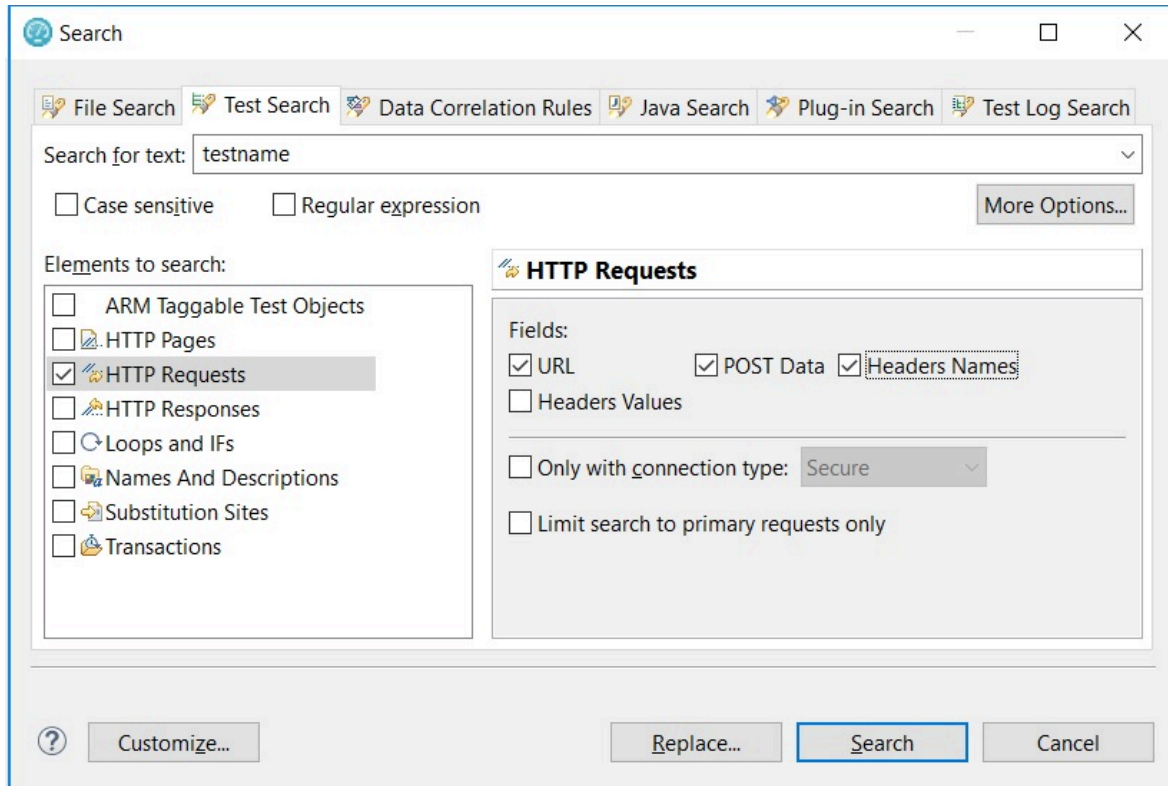
About this task

To remove HTTP headers from multiple HTTP requests at once:

1. In the Test Navigator, browse to the test and double-click it. The test opens.
2. Right-click the test name, and then select **Test Search**.



3. In **Search for text**, type the header name to locate.
4. In the **Elements to search** list, select the **HTTP Requests** check box.
5. On the right-hand side, where you can define how to search a selected element. Select the **Headers Names** or **Header Value** check box depending on the input provided in the **Search for text** field.

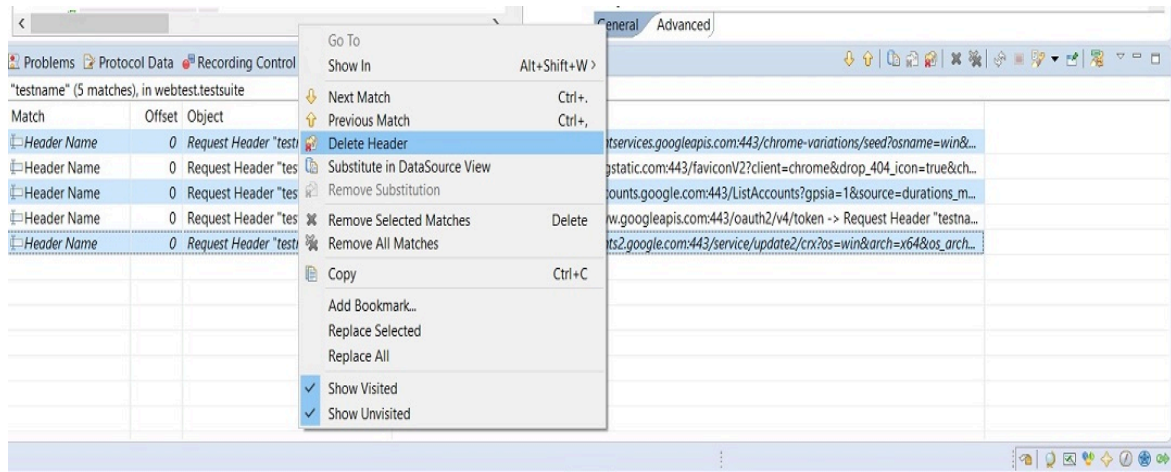


6. Click **Search**. The **Test Editor** window displays with the number of matches found.



7. In the Search view, select a result (press **Ctrl** key to choose multiple results).

8. Right-click the selected results and click **Delete Header**.



Verifying expected behavior

To check whether an expected behavior occurred during a run, you add verification points. When you run a test that contains a verification point, an error is reported if the expected behavior did not occur. When global verification points are disabled (the default), you can enable verification points for a specific test.

Enable verification points globally. Click **Window > Preferences > Test > Test Generation > HTTP Test Generation**.

Specifying the expected page title

Page title verification points verify that the primary request for a page returns the expected page title. If the returned title is unexpected, the test log reports a failed verdict event. Although the comparison is case-sensitive, it ignores multiple white-space characters (such as spaces, tabs, and carriage returns).

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. Right-click the test name or a page, and select **Enable Page Title VPs**.

Your choice determines whether the verification point is added to all pages in the test or to one page.

3. Click the page title to display the editing fields in the **Test Element Details** area.

Page Title Verification Point

- Enable verification point

Expected page title:

Recorded title: IBM United States

4. Ensure that the **Expected page title** field shows the string that you expect to be included in the response. Although you can change the string, the value listed is what was returned between the <title></title> tags during recording.

What to do next

You can also change the preferences so that Page Title verification points are set automatically. To do this:

1. Change the HTML page title. Click **Window > Preferences > Test > Test Generation > HTTP Test Generation > Verification Points**, and select **HTML Page Title**. This changes subsequent tests that you record.
2. Optionally, regenerate existing tests with the changed preference, as shown in [Generating a new test from a recorded session on page 275](#).

Specifying the expected response code

Response code verification points verify that the response code matches an expected value. If the returned code is does not match, the test log reports a failed verdict event. You can specify an exact response code or verify that the code is within the same category.

About this task

You can either change the preferences so that Response Code verification points are set automatically for all the tests or you can configure each test setting. To set response code in the preferences:

1. Change subsequent tests that you record. Click **Window > Preferences > Test > Test Generation > HTTP Test Generation > Automatically include verification point of**, select **HTTP Return Code** and click **Relaxed** or **Exact**. This changes subsequent tests that you record.
2. Optionally, regenerate existing tests with the changed preference, as shown in [Generating a new test from a recorded session on page 275](#).

To set response code setting for a test:

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

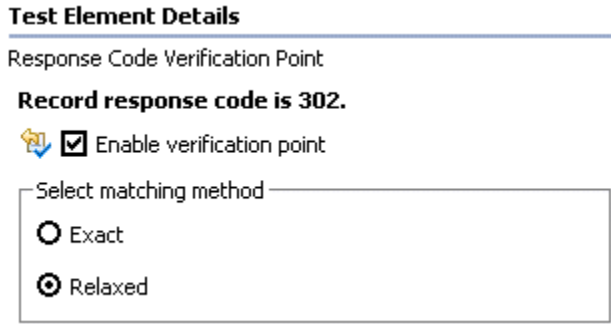
2. Right-click the test name, a test page, or a request, and select **Enable Response Code VPs**.

Your choice determines whether a verification point is added to every request in the test, to every request in a page, or to a particular request. The following figure shows a response code verification point within a test:



Note: When you modify a test, the modified items appear in italic type. The italic type changes to regular type after you save the test.

3. Click the verification point to display the response code editing fields in the **Test Element Details** area.



4. To disable an individual response code verification point, clear the **Enable verification point** field.
5. From the **Select matching method** list, click an option to indicate how closely the returned response code must match the recorded value.

Option	Description
Relaxed	If the recorded response code is 200, 201, 202, 204, 206, 301, 302, or 304, then a returned response code of any of those values causes the verification point to pass. If the recorded response code is any other value, it is the same as if you had specified an exact match.
Exact	An error is reported if the returned response code does not match the recorded value.

Specifying the expected response size

Response size verification points verify that the number of bytes returned in a response is what you expected. If the byte count does not match, the test log reports a failed verdict event. You can specify an exact response size or verify that the byte count is within a range.

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. Right-click the test name, a test page, or a request, and select **Enable Response Size VPs**

Your choice determines whether the verification point is added to all test pages, to a page in the test, or to a particular request. The following figure shows a response size verification point within a test:






Note: When you modify a test, the modified items appear in italic type. The italic type changes to regular type after you save the test.

- Click the verification point to display the response size editing fields in the **Test Element Details** area.

Test Element Details

Response Size Verification Point

Recorded response size is 206 bytes.

 Enable verification point

Select matching method

Exact: bytes

At least:

At most:

Range (bytes):

Range (%):

Verify that the response size is exactly 206 bytes.

Responses to HEAD requests, by definition, have no contents; the size is always 0.

- To disable an individual response size verification point, clear the **Enable verification point** field.
- From the **Select matching method** list, click an option to indicate how closely the response size that is returned in the page request must match the recorded response size.
 - The default matching method for a primary request is **Range**, with a percentage adjustment based on the value in the field on the right. (You can change the percentage basis as well as the matching method.) When the test is run, an error is reported if the actual response size falls outside the adjusted range.
 - For other requests, the default matching method is **Exact**. When the test is run, an error is reported if the response size does not exactly match the expected size.

What to do next

You can also change the preferences so that Response Size verification points are set automatically. To do this:

- Change subsequent tests that you record. Click **Window > Preferences > Test > Test Generation > HTTP Test Generation > Verification Points**, and select **HTTP Response Size**. This changes subsequent tests that you record.
- Optionally, regenerate existing tests with the changed preference, as shown in [Generating a new test from a recorded session on page 275](#).

Specifying the expected content

Content verification points verify that the response contains—or does not contain—an expected string. If the content returned is unexpected, the test log returns a failed verdict event. You can create a content verification point from specific response text.

To create a content verification point from a response or a portion of a response:

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. Click the response that contains the contents that you want to use for the verification point. If you do not see the contents, press Ctrl+Shift+Spacebar in the **Test Element Details** area, under **Content**.
3. Select the content for the verification point, right-click, and select **Add to Content Verification Point**.

Result

The content verification point and the selected string are displayed in the **Test Contents** area.

4. Edit the new content verification point as needed. You can use the **Add**, **Insert**, and **Remove** buttons to manipulate content verification points and content verification point strings. After you select a content verification point or content verification point string in the editor, you can also edit the verification points and strings by using the controls in the **Test Element Details** area. For example, you might want to change part of the hard-coded response contents to a regular expression. To substitute from any data source that exists in the test, select the entire string or a portion of the text string, and then right-click, and select **Substitute**.



Note: Responses to HEAD requests, by definition, have no content. Therefore, a content verification point that attempts to match a string in a response to a HEAD request will fail.

Specifying the expected content for multiple requests

Content verification points verify that the response contains—or does not contain—an expected string. If the content returned is unexpected, the test log returns a failed verdict event. You can advance through a test and create content verification points in multiple requests, or in the entire test.

To create content verification points in multiple requests:

1. Adjust the verification point scope. The default is to create content verification points in primary requests only and to skip responses with binary contents. Click **Window > Preferences > Test > Test Editor > HTTP Test**. Select or clear **Skip responses with binary content** and **Create only in primary responses** as needed.

2. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

3. Right-click the test name, a test page, or a request, and select **Verification Points > Enable Content VPs**. Your choice determines whether the verification point is added to all test pages, to a page in the test, or to a particular request. Confirm that you want to modify the test elements, and click **OK**.

4. In the **Create/Enable Content Verification Point** window, set **Verification fails if** to either **At least one of the checked strings is found** or to **None of the checked strings are found**.
5. In the list of strings in the **Text** column, select the strings that the content verification point should search for.
 - If you are editing an HTTP test, the window lists user-defined strings.
 - If you are editing a Siebel HTTP test, the window also lists strings of interest in Siebel applications.
6. To create a new string from scratch, click **New String**. To create a new string by editing another string, click it and click **Duplicate**. To edit an existing string, click **Edit**. To remove a string, click **Remove**.
7. Optionally, insert a regular expression into the verification point. The most common regular expressions are * for any number of characters, ? for any single character, and \ for an escape to enter literals. For detailed information on Java™ regular expressions, see the [Java™ documentation](#).
8. The **Create/Enable Content Verification Point** window advances through the requests. The preference settings that you selected in the first step determine whether secondary requests and requests containing binary data are affected. Select one of the following:

Option	Description
Skip	Advances to the next request without inserting a verification point in the current request.
Enable	Inserts a verification point into the current request and advances to the next request.
Enable All	Inserts a verification point into every test request (if the scope is the test) or every page request (if the scope is a page).

By default, a string that you set for a content verification point is available to all tests. To make the string available only to a specific test, or to clear the list of strings displayed when you create a verification point, change the preferences in the **Content Verification Points** section of **Window > Preferences > Test > Test Editor > General**.

Specifying error-handling behavior

You can specify how error conditions are handled when running a test or schedule. Error conditions include verification point failures, connection failures, server timeouts, custom code alerts, and problems with data correlation.

About this task

You can specify error-handling behavior for the workbench, schedules, tests, and test elements/steps. The Errors report displays the error conditions and error behavior that occurred in a test or schedule. When you set the error-handling behavior for the workbench, it is applied to all the tests in the workbench. To apply it for the workbench, click **Window > Preferences > Test > Test Execution > Error handling**.

Starting from 9.2, there is a new **Error handling** preference - **Mobile or Web UI Fatal Error** - to allow text execution for a compound test to continue even after a fatal exception in one of the tests in a compound test.

1. In the Test Navigator, browse to a test, and double-click it.

Result

The test opens.

2. In the Test editor, complete one of the following steps:

- a. To specify error-handling behavior for a request/step or other elements in the test hierarchy, in the **Test Contents** section, select the element and in the **Test Details** section, click the **Advanced** tab.
- b. To specify error-handling behavior for an element such as a connection, verification point, substitution, or reference, select the element, and then click **Change**. Skip to step 5.
- c. To specify error-handling behavior for a dataset that is associated with a test, on the **Common Options** page, select the dataset, and then click **Dataset Properties**. Click **Change**. Skip to step 5.

3. Under **Error Handling**, expand **Click to show conditions**.

Result

The error condition table is displayed. The error condition table shows all possible conditions that can be handled in the test.

4. Select the check box next to the condition for which to specify behavior.

Result

A window opens where you can specify the action to take and the message to log when the condition occurs.

5. To apply an action when the specified condition occurs, select the **Override action upon error** check box .

Option	Description
Continue	Click to continue running the test.
Exit transaction	Click to exit a transaction when the specified condition occurs. Select the Innermost or Outermost transaction, or type the name of a transaction.
Exit loop	Click to exit a loop when the specified condition occurs. Select the Innermost or Outermost loop, or type the name of a loop.
Continue to next iteration of loop	Click to continue to the next iteration of a loop when the specified condition occurs. Select the Innermost or Outermost loop, or type the name of a loop.
Exit test	Click to exit the test when the specified condition occurs.
Exit user	Click to stop the virtual user that encounters the specified condition. For a Web UI test, this action will stop the test execution.
Terminate run	Click to stop the run when the specified condition occurs.

6. To contribute to the health of the page, transaction, or loop, select the **Override contribution to health status** check box and select **Yes**. The respective reports display the health of the page, transaction, or loop.
7. **Optional:** To write a message to the test log when the specified condition occurs, select the **Override log message upon error** check box and type a message.
8. Click **OK**.

Exemple

To stop running a test when a substitution failure occurs in data correlation, select the **Substitution Failure** check box. Click **Exit Test**, and then click **OK**. The error-handling behavior that is specified closest to where an error occurs takes precedence. If a specific request is set to continue if a substitution fails, and the schedule is set to stop if a substitution fails, then the schedule will continue running if the substitution fails in that request.

Related reference

[Error conditions on page 1169](#)

How loops affect the state of virtual users

If verification points fail unexpectedly during a run, the cause might be that virtual users in loops do not maintain their original state. To enable each virtual user to enter the loop in the original state, you can modify the test's HTTP options or add custom code.

About this task

By default, the cookie cache for a virtual user is not reset during a test run. This is consistent with a browser's behavior. If a test or schedule contains loops, and a web server sets a cookie during the first iteration of the loop, that cookie is "remembered" on subsequent iterations.

However, in certain instances, you might want to clear all cookies cached for a particular virtual user. For example, if you want each iteration of a loop to appear as a new user, you must reset the cache. If you do not, although the test completes, verification points that you have set within the test may fail.

There are two ways to reset the cookie cache, and each way has different effects.

To reset the cookie cache when looping in the schedule, or when the test follows another test in the schedule, use the following method. This resets the cache whenever the test is entered. Even if your tests do not loop, use this method if you are running back-to-back tests or Siebel tests.

1. In the Test Navigator, browse to the test and double-click it. The test opens.
2. On the HTTP options page, select **Clear cookie cache when the test starts**.

To reset the cookie cache from one loop iteration to the next when you have put a loop around the entire contents of the test, and the loop is inside the test, add custom code to the test and call an API, as follows:

1. Run the test or schedule to add the current Java™ libraries to the class path.
2. Open the test and select the test element located at the point where you want the cookie cache to be reset. Typically, this is at the end of the loop.
3. Click **Add** or **Insert** and select **Custom Code**.
Add appends the custom code to the bottom of the selected element (test or test page). **Insert** adds the custom code above the selected page or page request.
4. Add the following Java™ import statement: `import`
`com.ibm.rational.test.lt.execution.http.util.CookieCacheUtil;`
5. Add the following Java™ code inside the exec method: `CookieCacheUtil.clearCookieCache(tes);`

Exemple

The following example shows a custom code addition that resets the cookie cache. The lines that you add to the generated custom code template are shown in bold:



Note: For another example of custom code that sets and clears cookies, see [Setting and clearing cookies for a virtual user on page](#) .

```
package test;

import com.ibm.rational.test.lt.execution.http.util.CookieCacheUtil;
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

public class Class1131739398417 implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    public Class1131739398417() {
    }
    public String exec(ITestExecutionServices tes, String[] args) {
        CookieCacheUtil.clearCookieCache(tes);
        return null;
    }
}
```

Splitting a test

After you record a test, you can split it into smaller tests. By splitting a test, you can create modular building blocks of smaller tests and combine them to make bigger tests. The original test is unchanged.

About this task

With the test-splitting capability, you can record a relatively long scenario with many functional steps against an application and then, in the editor, dissect the test into many smaller test segments, which you can run in various orders in a schedule. The wizard determines which variables need to persist among the split tests and creates the linkage so that you do not have to write custom code.

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. Select one or more elements in the test for splitting into a new test.

You must select contiguous elements. You can select elements, except for variable containers, that are immediate children of the root node of the test.

3. Right-click the selected elements, and then select **Save Selection As New Test**.

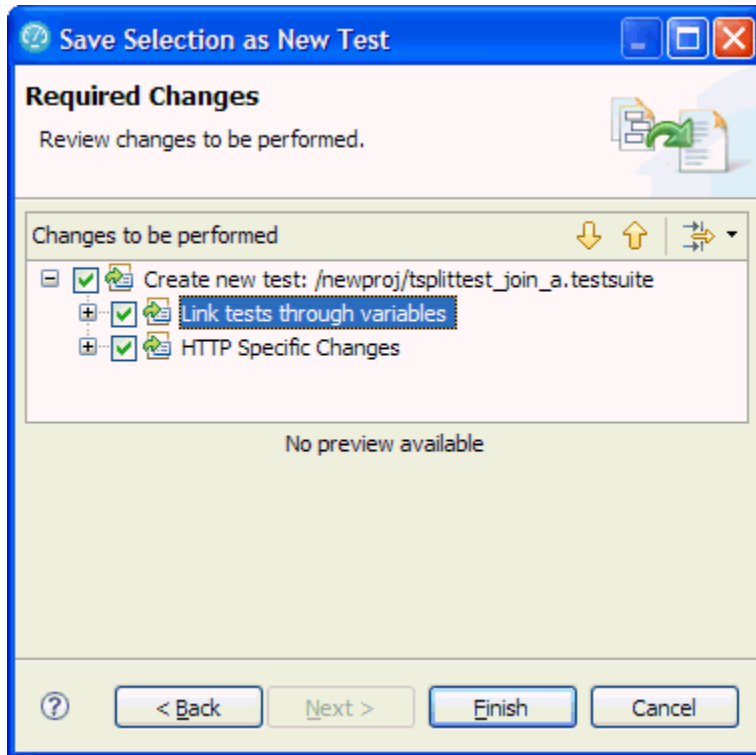
4. In the **Save Selection as New Test** window, type a file name for the new test, and optionally provide a description or comment for the split test.

- a. Select **Also use description text for comment at beginning of each test** to add the **Description/comment** field as a comment at the beginning of the split test.
- b. Select **Mark the selection in the editor** to mark the selection in the original test with marks of the form $[N->$ where N is a positive integer that corresponds to the number of saved selections. For example, the first time you save a selection as a new test, the selection in the original test is marked with this identifier: $[1->$. If you save three selections, the original test is marked with these identifiers: $[1->$, $[2->$, and $[3->$.
- c. Click **Next**.

Selecting **Mark the selection in the editor** makes it easier to split a test into multiple parts. The marks in the editor are removed when you close the test. You cannot save marked test elements. Right-click and select **Clear Range** to remove the marks if you want to save the selected test elements again.

5. **Optional:** On the next page of the **Save Selection as New Test** wizard, examine the changes to be performed as a result of the split.

Typically, you leave **Link tests through variables** and **HTTP Specific Changes** selected; clearing these options might make a split test unusable or produce unpredictable results. However, you can clear specific **Link tests through variables** boxes if you do not want certain data to be correlated between the tests.



6. Click **Finish**.

Results

The new test is created from the selected elements. The test variables that are created by splitting a test are listed in the **Variables Created by Split Test** container in the new test. For best results, open the **Variables Created by Split Test** container in the new test and make sure that the variables are created and assigned with values by a test that is executed before the newly split test. For more information on test variables, see [About test variables on page](#) and [Declaring and assigning test variables on page](#).

The original test is marked if you select the **Mark the selection in the editor** option. To remove the marks, right-click the selected elements in the original test and select **Clear Range**. You are also prompted to delete the new test. Click **Yes** to delete the new test or **No** to preserve it.

Example

Be aware of the choices that you make when you split a test and rearrange the split tests in a schedule. Assume that the visitIBM recorded test contains the following actions:

- Logging on to a server.
- Creating an entry on the server and removing the entry.
- Editing an entry, validating that the change occurred, and restoring the entry.
- Logging off of the server.

You want to split the test into four parts: Logon, Create, Edit, and Logoff. You need to split the test four times.

- Open visitIBM and select the logon actions. Name the new test Logon, which contains the Logon actions.
- Select the create actions. Name the new test Create.
- Select the edit actions. Name the new test Edit.
- Select the logoff actions. Name the new test Logoff.

You then create a schedule that runs virtual users selected from a dataset. Each virtual user runs the Logon test, performs various combinations of the Create and Edit tests, and finally runs the Logoff test.

Be aware, however, that when you split the tests, the Create test might have initialized variables that the Edit test uses. Therefore, if you reverse the order during the run (that is, run the Edit test before the Create test), make sure that the variables that the tests share are initialized and set correctly.

Splitting a test page

You can split an HTTP page into two contiguous pages. The page title, think times, primary request, and delay are automatically recalculated for the affected pages. Customized page titles, think times, primary requests, and delays revert to the default values.

Before you begin

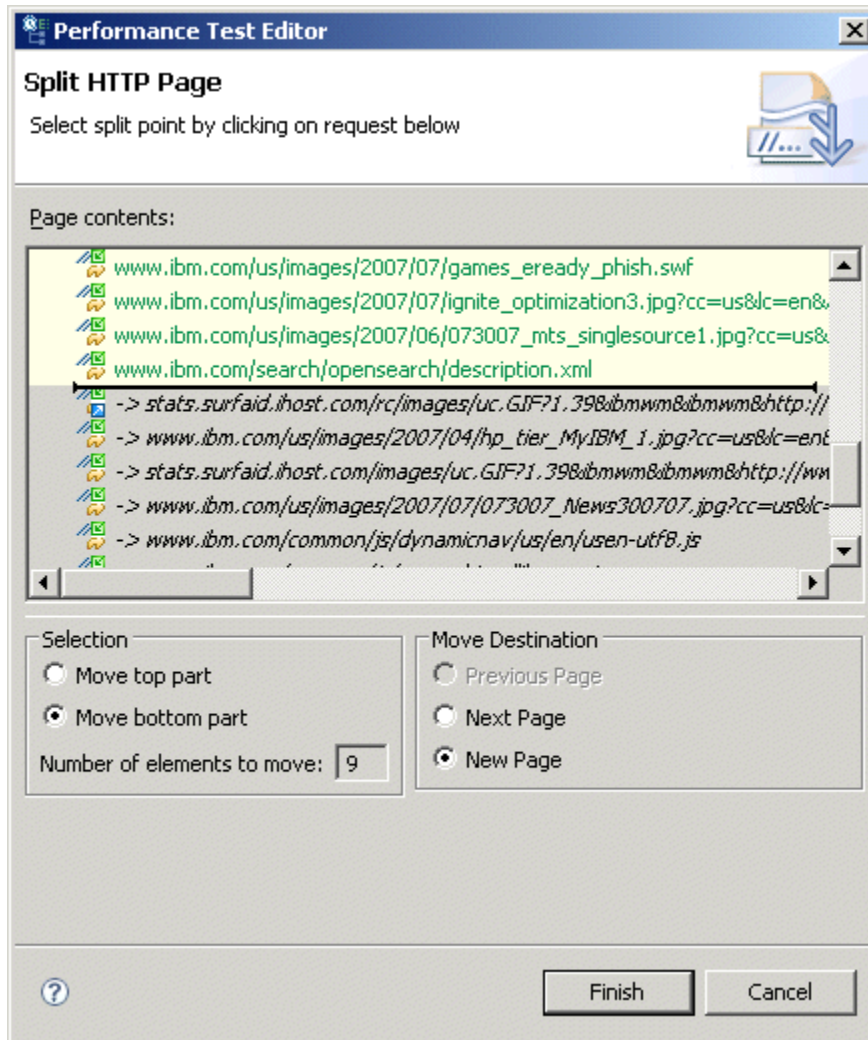
When you inspect a test, you might notice that some page boundaries are not at the correct place. A common cause is that, during recording, you did not wait for the page to fully load. Although you can rerecord a test, if your test is long or complex, it is often easier to split the incorrect page when you edit the test, rather than to rerecord the entire test.

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. In the test, expand the page that you want to split.
3. Right-click on the request where you want the split to occur, and select **Split page here**.
4. In the **Split HTTP Page** window, you can change the location of the split by clicking on another page element. You can also create a new page, combine the elements above the insertion point with the previous page, or combine the elements below the insertion point with the next page.



- To split the page, click **Finish**.

What to do next

You can also set preferences so that new pages are generated when the delay exceeds a specified value. Click **Window > Preferences > Test > Test Generation > HTTP Test Generation**, and click the **Protocol** tab. Enter a value for **Generate new page if delay between requests is >**.

Merging test pages

You can two or more contiguous HTTP pages into one page. The page title, think times, primary request, and delay are automatically recalculated for the affected pages. Customized page titles, think times, primary requests, and delays revert to the default values.

About this task

When you inspect a test, you might notice that some page boundaries are not at the correct place. These extra pages are caused by a variety of reasons. For example, during recording you might hover over a "hotspot" that fetches

images as you move on and off of the spot. In this case, some pages in the test, which properly belong to the previous page, contain only .gif files. Although you can rerecord a test in the hope of fixing this problem, if your test is long or complex, it is often easier to merge the incorrect page when you edit the test, rather than to rerecord the entire test.

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. Select the pages to merge (the pages must be contiguous) and select **Merge Selected Pages**
3. The **Merge Pages** window lists the pages that you are merging. From this list, select the page that will contain the other pages.
4. Optionally, click **Keep empty pages** to keep the same number of pages in the test. For example, if you select this option and merge two pages, one page will contain all of the requests and the other page will be empty.

What to do next

You can also set preferences so that new pages are not generated when the think time is less than specified value. Click **Window > Preferences > Test > Test Generation > HTTP Test Generation**, and click the **Protocol** tab. Enter a value for **Do not generate new page if think time is <**.

Disabling and enabling secondary HTTP requests

You can disable all secondary requests within an HTTP performance test or a subset of requests in the test. Secondary requests are all requests within a page other than the primary request.

About this task

To disable other elements in tests or schedules (for any protocol), see [Disabling portions of a test or schedule on page](#).

1. In the Test Navigator, browse to the test, and double-click it.

Result

The test opens.

2. In the **Test Element Details** area, click the **HTTP Options** tab.
3. At **Secondary request behavior**, click **Modify**.
4. In the Enable or Disable Secondary Requests box, select one or more of the following options:

Option	Description
All secondary	Selects all secondary requests.
Images	Selects all secondary requests that are image-related. This selection includes all secondary requests where the Content-type header of the response contains <i>image</i> or the path of the URI of the request contains a .gif, .png, .jpg, .bmp, or .tif extension.

Option	Description
Host/Port based	Selects all secondary requests that use the specified host:port pair for connections. A list of host:port pairs in the test is displayed.
User-defined	Selects all secondary requests where the user-specified string, or a string matching a user-specified regular expression, is in the request URI.

5. You typically keep the boxes under **Do not disable secondary requests** selected, which leaves them enabled. However, if you have extensive knowledge of the system under test and have already done some troubleshooting, you might clear these boxes in the following cases:
- With responses containing set-cookie headers: If the cookies set in a particular request are not important to the remaining requests, you may disable them. This decision requires you to know how the system under test uses cookies.
 - With data sources used by enabled requests: If a test contains superfluous data correlation, you may disable it. This decision requires you to know how the system under test uses data correlation.
6. Select **Disable** or **Enable** to modify the secondary requests.

Result

The requests are now enabled or disabled.

Adding custom actions to requests

After an HTTP test is generated, you might want to add a few actions before or after a particular request is processed. For instance, for preprocessors, you can modify the headers before sending them to the server. Similarly, for postprocessors, you can extract data from the response and set it to a variable.

About this task

The actions to be processed must be coded in Java. Note that the pre processor is the last thing to be sent to the server after all the processing (data substitution) for the request is complete. Also, a postprocessor is the last thing to be received after all response processing is complete from the server.

You can add preprocessors and postprocessors at the test level and at the request level. When they are added at both the levels, the ones at the request level takes precedence.

The preprocessors and postprocessors for the HTTP extension uses the following interfaces and methods:

- **IHTTPRequestInterface**: Use this interface to process an action before the request is processed.
 - **setURI**: Use this method to set or change the URL that is sent to the server.
 - **getHeader**: Use this method to get the name of the header that requires a value. If there are more than one header with the same name, only the first value is returned.

- **removeHeader**: Use this method to remove the name of the header.
- **setHeader**: Use this method to set the name and value for the header. If the header does not exist, it is created. If the header exists, the value is changed. If there is more than one header with the same name, only the first header is changed.
- **IHTTPResponseInterface**: Use this interface to process an action after the request is processed.
 - **getResponseContent**: Use this method to view the content of the response. If the response content exceeds the specified limit, the content is truncated.
 - **getResponseHeader**: Use this method to view the name of the header. If there are more than one header with the same name in the response, only the first header is returned.
 - **getReturnCode**: Use this method to view the return code of the current response.
- **IHTTPRequestCommonInterface**: Use this interface for both request and response interfaces.
 - **getURI**: Use this method to view the current URI of the request.
 - **getHeaders**: Use this method to view all of the headers associated with the request for pre processing or all of the headers associated with the response for post processing.

To add a preprocessor or postprocessor:

1. In the Test Content area of the Test editor, select a request.
2. In the Test Details area of the Test editor, click the **Advanced** tab.
3. To add a custom action before a request is processed, in Preprocessor, click **Create**.
4. Specify a name for the Java file and click **Finish**.

Result

A new Java file opens. Add the custom actions to be processed before the request and save the file.

5. To add a custom action after a request is processed, in Postprocessor, click **Create** and follow step 4.

What to do next

You can now run the test to verify if the actions specified in preprocessors or postprocessors returned expected results.

Reusing tests on different hosts: Server connection variables

Your tests represent a significant investment in time and effort. You can share or reuse them for different configurations and web hosts by changing the variables for the host name and port.

Before you begin

Before you begin, confirm that test generation preferences are set to support data correlation. Click **Window > Preferences > Test > Performance Test Generation > HTTP Test Generation**, click the **Data Correlation** tab, and verify that **Automatically correlate host and port data** is selected. If not, select that option and regenerate the test.

About this task

Perhaps you develop the tests on a lab computer and then they want to run them on a production server. If the application that you are testing is identical on both computers, you can change the host name and reuse the tests on the production server.

To change the name of the host or proxy server on all requests in a test:

1. In the Test Navigator, browse to the test and double-click it. The test opens.
2. Expand the **Test Variable** section at the top of the test, and click the Server Connection variable that contains the hostname that you want to change.
3. In the **Test Element Details** area, perform the following steps:
 - a. Type the new host name in the **Host** field.
 - b. **Optional:** Type a new port number in the **Port** field.

Result

The new hostname and port combination is correlated to the test variable, which contains the value currently being used.

Results



Note: To change the host names and ports of many tests, put the server connection variable in a dataset and associate the dataset with the tests. When you change the name and port in the dataset, that change is propagated throughout the tests in the dataset. For more information, see [Creating a dataset associated with a test on page](#).

Converting tests to use SSL connections

You can convert a test that was recorded without Secure Sockets Layer (SSL) connections to use SSL connections.

About this task

If you develop performance tests on a lab computer that does not use secure connections, and then you must run the tests against a production server that requires SSL connections, you can add SSL to the server access configurations and reuse the tests.

To convert a server access configuration to use SSL for all associated connections:

1. In the Test Navigator, browse to the test, and double-click it. The test opens.
2. Expand the **Test Resources** section at the top of the test.
3. Right-click the server access configuration that corresponds to the server where you want to add SSL, and then click **Add > SSL**.

Result

An SSL element is added as a child of the server access configuration.

4. In the **Test Element Details** area for the server access configuration, type a new number in **Port** if the server uses a different port for SSL communication. If the port number is correlated with a server connection variable, select the port number, right-click, and select **Go To > Variable: variable_name** to navigate to the variable. Change the port number in the server connection variable.
5. In the **Test Element Details** area for the SSL element, make a selection in **Protocol**, and type or select a name in **Cipher**.
6. Repeat the same steps for all server access configurations to convert to use SSL.

Results

When you run the test, the connections that are associated with the server access configurations use SSL.

Working with Server Name Indication (SNI) recordings

If you have recorded against a server that supports Server Name Indication (SNI), an extension of the TLS protocol, the recording session file displays `true` for the **SNI Extension** field. There might be a need for you to access both SNI and non-SNI applications from the same server. To run the same test without using the SNI extension, you can manually change the value to `false`.

About this task

The screenshot shows the HTTP Proxy Recorder interface. On the left, a 'Packets' table lists various events with their start and end times. The selected packet is 'Proxy opens secured connection 2 based on 1'. On the right, the 'Packet Details' panel shows the following information:

Field	Value
Recorder:	HTTP Proxy Recorder
Connection Id:	2
Parent Connection Id:	1
Connection Type Id:	0
Server Host:	tiles.services.mozilla.com
Server Port:	443
Client Host:	127.0.0.1
Client Port:	54790
Cipher Suites:	SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256
SSL Protocols:	TLSv1.2
HTTP Version Protocol:	http/1.1
SNI Extension:	true

The **Server Access Configurations** resource of the test script also have SSL entries. Each SSL entry displays which TLS version and Cipher value was used. To edit multiple SSL entries, select them and in the Detail area, right-click the entries and click **Edit multiple SSLs**.

Viewing a test in the Protocol Data view

The Protocol Data view enables you to inspect the actual test data. You can see requests, response headers, and response contents, as well as the rendered images that you see through your browser. Use this view to obtain the information you need to add custom code or to manually correlate data. This view also lets you compare the recorded data with the data retrieved during a run.

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. Click the **Protocol Data** tab to open the view.

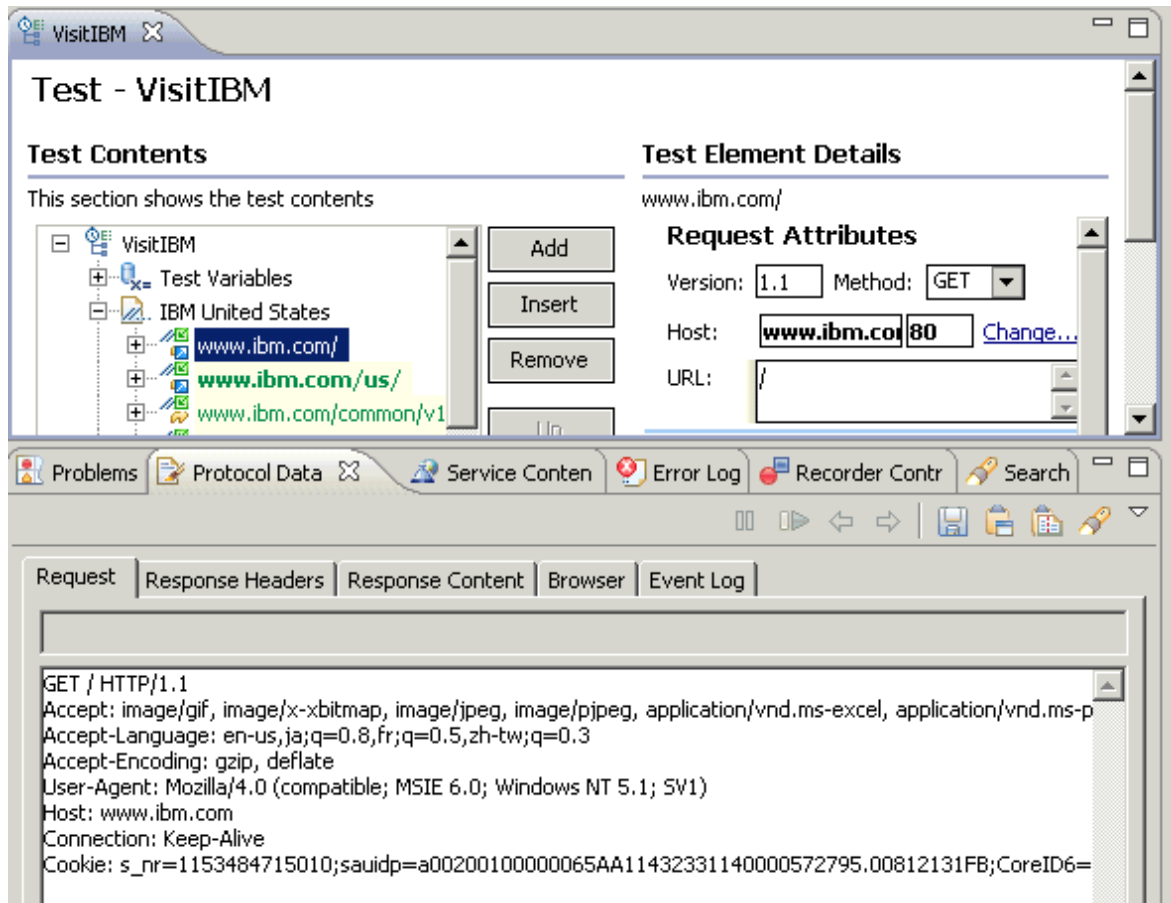
Result

i **Tip:** If you cannot locate the **Protocol Data** tab, click **Window > Show View > Protocol Data**.

3. In the test, click the line that corresponds to the page, request, or response that you want to view.
4. In the Protocol Data view, click the tab that corresponds to the type of data or view of interest.

Result

The selected data is displayed in the Protocol Data view.



What to do next

If you have problems during playback, you can compare the data that was recorded with the data that was retrieved during the run. For information on displaying the data retrieved during the run, see [Inspecting HTTP test logs in the Protocol Data view on page 651](#).

Testing Siebel applications

When you record a Siebel application, a Siebel-specific test is automatically generated. However, before you run this test, install the Siebel Test Automation library and edit the test to use built-in Siebel variables.



Note: HCL OneTest™ Performance 64-bit workbench does not support testing Siebel applications.

Prerequisites for Siebel testing

To test Siebel applications, the Siebel Test Automation and the Microsoft™ C++ runtime libraries must be installed on the workbench computer.

About this task

Siebel applications run only on Windows™ operating system, therefore you must run the Siebel tests only on Windows™. You must not add a Siebel test to a schedule that you deploy to run on operating systems other than Windows™.



Note: HCL OneTest™ Performance 64-bit workbench does not support testing Siebel applications.

To install the prerequisites for testing Siebel applications:

1. Obtain the Siebel Test Automation library, `ssdtcorr.dll`, from Siebel.
2. Copy the `ssdtcorr.dll` file to the workbench computer in this path: `...\ibm\sdp\rpt\external_files\deployable\siebel\...\hcl\hclonetest\hot-perf\external_files\deployable\siebel\`, where `...\ibm\sdp\rpt\...\hcl\hclonetest\hot-perf\` is the product installation directory.

If the Siebel Test Automation library is not installed, Siebel tests will fail, and a warning message will display when you edit a Siebel test. After the library is installed on the workbench computer, it is automatically deployed as needed to any remote location. The library must be on any computer that runs a Siebel test.

You can use the `rptExternal` variable in the `config.ini` file to control where external files must be installed. By default, the `rptExternal` variable is set to the product installation directory. Edit the `rptExternal` variable if you want to install the `.dll` files to a different location.

Example

For example, if you set the `rptExternal` variable to `e:\:\ibm\sdp\rpte:\:\hcl\hclonetest\hot-perf` then you would install the Siebel Test Automation library in `e:\hcl\hclonetest\hot-perf\external_files\deployable\siebel\`. Note that you must use an extra backslash before the colon and backslashes in the path.

3. Copy the appropriate Microsoft™ C++ runtime library to the workbench computer in the same directory as the `ssdtcorr.dll` file. Recent version of `ssdtcorr.dll` (Version 8.1.1.14 for example) require `msvcr110.dll`. Older versions of `ssdtcorr.dll` (like Version 7.7.0.0) need `msvcr70.dll`.

To download the appropriate Visual C++ runtime library from the Microsoft™ web site, download the Windows Server 2003 Resource Kit Tools for `msvcr70.dll` or the Visual C++ Redistributable for Visual Studio 2012

to get `msvcr110.dll`. The file name is case-sensitive, and must be all lower-case. The file must be named `msvcr70.dll`, not `MSVCR70.DLL`.

Once installed in the correct directory on the workbench computer, the Microsoft™ C++ runtime library is automatically deployed as needed to any remote location.

Differences between Siebel tests and HTTP tests

Siebel tests precisely designate dataset candidates, include an additional type of data source, and store variables in a proprietary data structure. In addition, page names are created during test generation to help you find pages of interest.

The primary difference between a Siebel test and a standard HTTP test is in how dynamic data is stored and substituted during a test run:

- In a standard HTTP test, a data source (dataset variable, custom code, or a reference) is linked to a test value that is replaced at run time. Siebel tests support standard HTTP data sources and substitution. Datasets, which are explained in [Providing tests with variable data on page 312](#), work the same way in Siebel tests as in standard HTTP tests, but the dataset candidates in Siebel tests are more precisely designated than in standard HTTP tests. In many cases, dataset substitutions are the only changes that you need to make to a Siebel test.
- In a Siebel test, an additional type of data source, which is called a *built-in data source*, contains variables that you can use to replace a test value.

These variables can be substituted for dates (in defined formats), time stamps, and counters throughout tests. In some cases, **SWE Unique Value** can be used as an alternative to a dataset; for example, to supply variable account names. For detailed instructions, see [Correlating a request value with a built-in Siebel variable on page 313](#).

Siebel variables are stored in a proprietary data structure called a *star array*. A star array stores both strings and their length in hexadecimal format (`length_string`) or integer format (`length*string`). Siebel substituters have a method for substituting data and recomputing the length. You can substitute from a value in a star array (highlight a `length_string` or `length*string` format value, right-click, and then click **Substitute**). You are then asked whether you want a Siebel substitution or a standard HTTP substitution. You typically select Siebel data correlation.

Siebel tests are organized inside the test editor much like standard HTTP tests, but with some differences for pages:

- The first page of a Siebel test is named **Message Bar**, which emulates the ticker-tape message that Siebel application pages display.
- Page names are created during test generation to help you find pages of interest. For example, a typical change to make to a test before running it is to replace the user name and password that you typed during recording with values in a dataset. As shown in the example, the page from which you logged into the Siebel server is named **Login - Send UserName/Password**, to help you find this page quickly.

Correlating a request value with a built-in Siebel variable

If you are editing a Siebel test, you can correlate request values with built-in Siebel variables.

To correlate a request value with a built-in variable:

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. Locate the value that should be replaced by a built-in variable.
3. Highlight the value: with the left mouse button pressed, drag your mouse over it.
4. Right-click the highlighted value and click **Substitute from > Built-in data sources**.

Result

The Built-in Datasource Selection wizard displays the types of variables that can be substituted.

5. Select the type of variable and click either **Next** or **Finish**.

Choose from:

- If you select **Current Date**, click **Next**, select the date format, and then click **Finish**.
- If you select **SWE Counter**, click **Next**, type values for the counter in the **Current Value** and **Maximum Value** fields, and then click **Finish**.

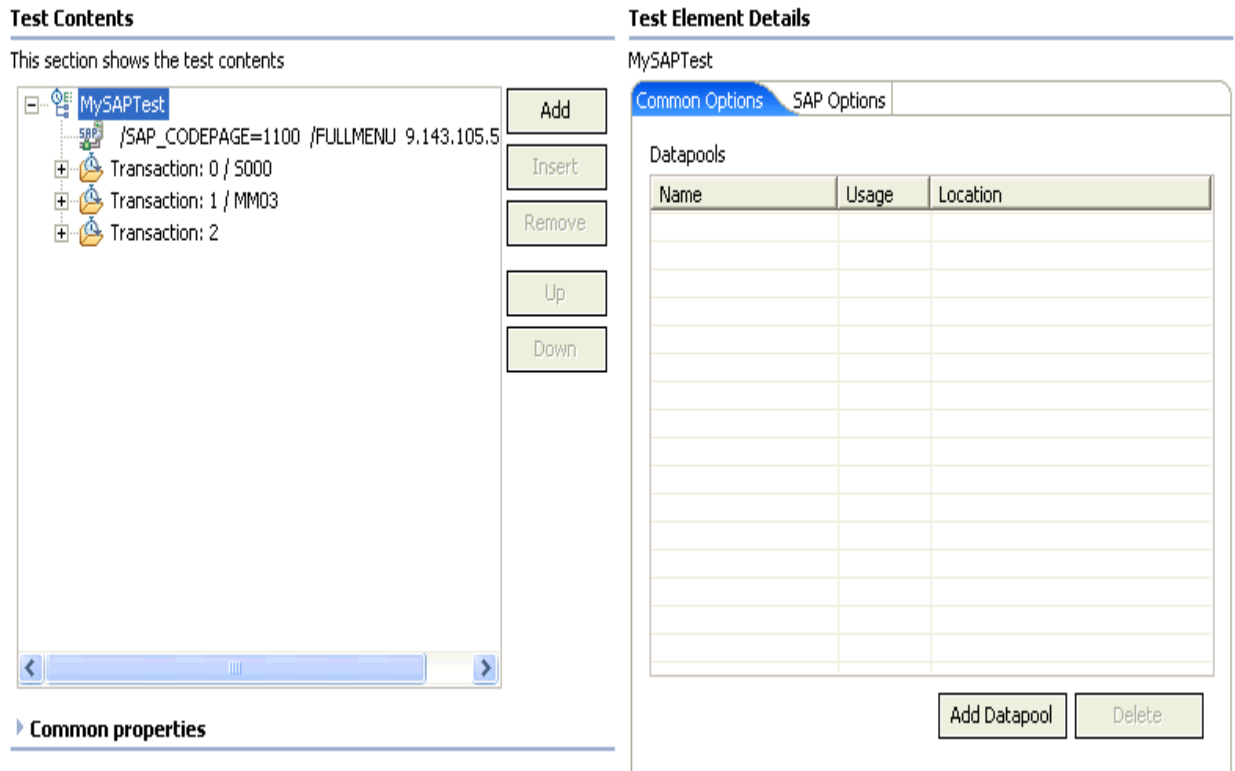
Editing SAP tests

After you record a test, you can edit it to include variable data (rather than the data that you recorded) and verification points (to confirm that the test runs as expected).

SAP test editor overview

You use the test editor to inspect or customize a SAP performance test that you recorded.

The test editor lists the SAP transactions for a test, by title. The following example shows the test MySAPTest, which was generated from a recording of these tester actions: logon to the server, launch a Material Master Display transaction (mm03), view the Basic Data screen for the item SCREW, and stop recording.



There are two main areas in the test editor window. The area on the left, **Test Contents**, displays the hierarchy of the SAP transactions for the test. The area on the right, **Test Element Details**, displays details about the currently selected item (transaction or event) in the test hierarchy. In the preceding example, **Test Element Details** displays information about the test because the name of the test, MySAPTest, is selected in the **Test Contents** area.

When you expand a test transaction, you see a list of SAP screens. Each SAP screen corresponds to a new page or window in SAP GUI and is reflected by the screen capture in the **SAP Protocol Data** view.

When you expand a SAP screen, you see the SAPGUI events for the screen, with names that describe the action. The following example shows the Material Master Display (mm03) transaction expanded with the SAP Easy Access screen. The set event that described the mm03 text input in the SAP GUI is selected in the **Test Contents** area.

Test Contents

This section shows the test contents

Common properties

Test Element Details

SAP Set Event

Information

Name:
 Type:
 Identifier:

Think Time:

Event Attributes

Name:
 Value:

Values can sometimes be highlighted in green. This highlighting indicates that these requests contain one or both of the following types of information:

- **A dataset candidate:** This is a value, usually one specified by the tester during recording, that the test generator determined is likely to be replaced by values in a dataset. An example of a dataset candidate is a string that you search for in a recorded test. The string is highlighted as a dataset candidate on the assumption that, before playback, you might want to associate the string with a dataset column containing appropriate substitute values..
- **Correlated data:** These are values in a test, usually one of them in a response and the other in a subsequent request, that the test generator determined needed to be associated in order to ensure correct test playback. An example is a material price returned to the browser by a test that searches a material database. The test generator automatically correlates material names with prices. Suppose that, before running the test with many virtual users, you replace the material name searched for in the recorded test with names in a dataset. Because the test correlates the data, each virtual user searches for a different material, and the server returns an appropriate price.

To see an illustration of color coding in performance tests or to change the color settings, click **Window > Preferences > Test > Performance Test Editor**, and then click the **Fonts and Colors** tab.

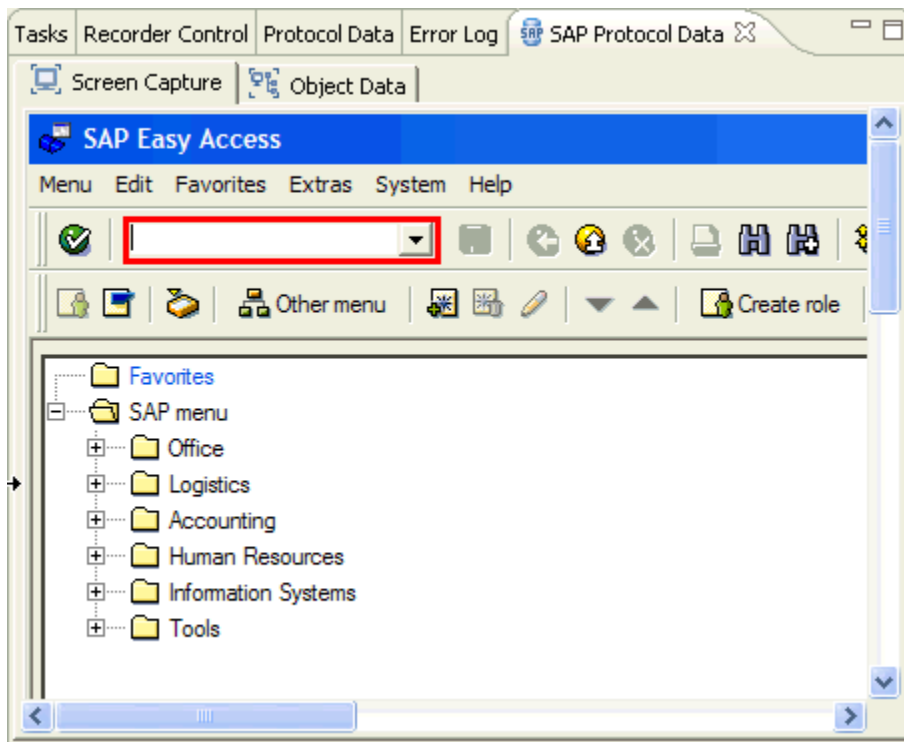
The **Response** data inside each request shows the data that the SAP server returned to the browser based on that request.

Click **Add** to add elements to the selected test element. Alternatively, you can right-click a test element and select an action from a menu. The choices that you see depend on what you have selected. For example, after you select a test, you can add a new event.

The **Insert** button works similarly. Use it to insert an element before the selected element.

The other buttons (**Remove, Up, Down**), are primarily for use when substantially modifying a test; these actions or choices are likely to break a recorded test. The types of structures that are commonly used in recorded tests are explained in [Adding verification points to a SAP test on page 317](#) and [Adding elements to a SAP test on page 319](#).

The test editor synchronizes with the **SAP Protocol Data** view. When you select a test element in the test editor, the corresponding screen is displayed as a screen capture in the **SAP Protocol Data** view. If a specific SAP GUI object is related to the test element, the object is highlighted in the **SAP Protocol Data** view. You can also right-click SAP GUI objects in the **SAP Protocol Data** view to add associated events or verification points to your test.



Sometimes, the area of the editor where you need to work is obscured. To enlarge an area, move your cursor over one of the blue lines until your cursor changes shape (to a vertical line with an up arrow at the top and a down arrow at the bottom) and drag up or down while holding the left mouse button.

Verifying expected behavior

To check whether an expected behavior occurred during a run, you can add verification points. When you run a test that contains a verification point, an error is reported if the expected behavior did not occur.

Adding an SAP verification point

With SAP get and SAP call elements, you can retrieve a value from the SAP GUI to create verification points on an SAP screen element.

Before you begin

When you add SAP verification points, SAP get elements, and some SAP calls, retrieve the data from objects in the SAP GUI, such as windows or text fields. SAP get and SAP call elements are contained in SAP screens in the test suite. SAP screens can be windows, dialog boxes, or transaction screens that are part of a recorded transaction.

You can use either the test editor or the **SAP Protocol Data** view to create or edit SAP get and SAP call elements and place verification points on them. When using the **SAP Protocol Data** view, you can select SAP screen elements from the screen capture to specify the SAP GUI identifier for the get event. Using this method to create or edit an SAP verification point is easier than adding it manually from the test editor.

The **SAP Protocol Data** view contains two pages that are synchronized with each other and with the test editor:

- **Screen Capture** displays a graphical screen capture of the SAP GUI. You can select all GUI objects such as windows, buttons, fields or areas.
- **Object Data** provides information about the selected GUI object: identifier, type, name, text, tooltip, and subtype.

1. Open the test editor and the **SAP Protocol Data** view.

If the **SAP Protocol Data** view is not open, click **Window > Show View > Other > Test > SAP Protocol Data**.

2. In the **Test Contents** area of the test editor, expand a transaction and an SAP screen.

The **SAP Protocol Data** view displays a screen capture of the selected transaction.

3. Inside the transaction, select the item for which you want to enter a new value.

Result

The Screen Capture page of the SAP Protocol Data view displays the screen capture of the SAP GUI with the corresponding GUI object highlighted in red.

4. In the SAP Protocol Data view, right-click the GUI field that you want to verify, and then select **Create Verification Point**.

If you want to create an advanced verification point using an SAP GUI call method, then you can select **Create Element** instead.

5. In the **Create Verification Point** or **Create Element** window, specify the expected value for the verification point.

Choose from:

- If you want to verify a text value in the SAP GUI object, ensure that **Verify text** is selected, and type the **Expected value** that you want to verify; then click **Finish**.
- If you want to verify advanced properties of the SAP GUI object, you can select **Advanced**, and then specify the properties attached to the GUI object as well as the Expected values. Refer to SAP documentation for information about these properties.
- If you selected **Create Element** to create a verification point on an SAP call, then select **Advanced**, choose an **Element type** that returns a value, and then specify the properties attached to the GUI object. Refer to SAP documentation for information about these properties.

What to do next

After creating the event, you can use the test editor to easily change the value. You can also enable and disable SAP verification points on SAP get and SAP call elements in the test editor.

Specifying an expected screen title

Screen title verification points report an error if the title of an SAP screen is different from what you expected.

1. Select the SAP screen in the test editor and ensure that screen title verification is enabled for the SAP screen.

The **Test Element Details** area includes a **Screen Title Verification Point** section.

- If screen title verification was enabled for the entire test, the **Enable verification point** check box is selected for all SAP screens in the test.
- If screen title verification was enabled for a specific SAP screen, the **Enable verification point** check box is selected for the selected SAP screen.

You can enable or disable screen title verification for a specific SAP screen in the test editor by selecting or clearing the **Enable verification point** check box.

2. Ensure that the **Expected screen title** field shows the string that you expect to be included in the page title that is returned when this page is loaded.

When the test was recorded, SAP returned a default title for this screen. This value is displayed in the **Recorded title** field, and is automatically copied to the **Expected page title** field when the **Enable verification point** check box is selected. You can change the string in the **Expected page title** field as needed.

Result

Whenever the test runs with page title verification enabled, an error is reported if the title returned for the page does not contain the expected title. Although the comparison is case-sensitive, it ignores multiple white-space characters (such as spaces, tabs, and carriage returns).

Verifying response times

SAP request response times measure the delay between the moment the user submits a server request and the moment the server responds. Response time data is provided by the server. You can set a verification point on a response time threshold value. If the test encounters a response time above the threshold, the verification point is failed.

Before you begin

When the **Verification points for SAP request response time threshold** option is selected in the SAP Test Generation preferences, all SAP server request elements contain a response time verification point. The default threshold value is calculated by applying a multiplier to the recorded response time. You can change the default multiplier in the SAP Test Generation preferences. The response time measurements are displayed in the SAP server request element, which is the last element in an SAP screen.

To define a response time verification point:

1. In the test editor **Test Content** area, select an SAP server request element inside an SAP screen element.
2. In the **Test Element Details**, select **Enable verification point**, and then enter the **Response time threshold** in milliseconds.

If the test encounters a response time that is higher than the threshold, the verification point is failed.

Adding elements to an SAP test

You can add a variety of elements to an SAP performance test, such as loops, conditions, SAP set, verification point or sequence elements.

Adding an SAP set, SAP get, or SAP call element

You can use SAP set, SAP get, or SAP call elements in performance tests to add items such as a field selection, a keyboard entry, a get element for reference use, or any advanced interaction with the SAP client GUI.

Before you begin

SAP set elements represent user interactions with the SAP GUI, such as entering a value into a field. SAP set elements are contained in SAP screen elements. SAP screen elements can be windows, dialog boxes or transaction screens that are part of a recorded transaction.

SAP get elements enable you to retrieve information from the SAP GUI, such as field values. SAP get elements are contained in SAP screen elements. The primary use of an SAP get element is to create a reference field or a verification point in the test. An SAP verification point is actually an SAP get element with a verification point enabled.

SAP call elements represent all the various user interactions that are neither an SAP set or an SAP get element. For example, selecting an object, scrolling, or pressing the Enter key generates an SAP call.

You can use either the test editor or the **SAP Protocol Data** view to create or edit SAP set, get, or call elements.

When using the **SAP Protocol Data** view, you can select SAP screen objects from the screen capture and copy the information directly to the new SAP set, get, or call element. Using the **SAP Protocol Data** view to create or edit an SAP event is much easier than adding an event manually from the test editor.

The **SAP Protocol Data** view contains two pages that are synchronized with each other and with the test editor:

- **Screen Capture** displays a graphical screen capture of the SAP GUI. You can select all GUI objects, such as windows, buttons, fields or areas.
- **Object Data** provides information about the selected GUI object: identifier, type, name, text, tooltip, and subtype.

To add an SAP set, get, or call element:

1. Open the test in the test editor and the **SAP Protocol Data** view.
If the **SAP Protocol Data** view is not open, click **Window > Show View > Other > Test > SAP Protocol Data**
2. In the **Test Contents** area of the test editor, expand a transaction and an SAP screen.
The **SAP Protocol Data** view displays a screen capture of the selected transaction.
3. Inside the transaction, select the item for which you want to enter a new value.

Result

The **Screen Capture** page of the **SAP Protocol Data** view displays the screen capture of the SAP GUI with the corresponding GUI object highlighted.

4. In the **SAP Protocol Data** view, right-click the GUI object for which you want to create the SAP set, get, or call element, and then click **Create Element**.

Result

This opens the window, which already contains the **Identifier** from the recorded session.

5. In the **Create Element** window, specify the type of SAP element that you want to create. Enter the value that you want to add, and then do one of the following procedures, depending on your testing objectives:

Choose from:

- To create a simple SAP set element, select **Set text**, and then type the text value that you want to input into the SAP GUI.
 - To create an SAP get or an SAP call element, select **Advanced**, and then select **SAP Set**, or **SAP Call** in the **Element type** list. Use a SAP set to input a value into the SAP GUI client. Use a SAP call to call a method for advanced interaction with the SAP GUI. Specify the property to get or the method of the call. Refer to SAP documentation for information about SAP call methods and properties. SAP call elements are created with default values that you can change in the test editor. You can use SAP call methods to define a verification point or for data correlation.
 - To create a non-text SAP set element, select **Advanced**, and then select **SAP Set** in the **Element type** list. Specify the property to set and enter a value if that property requires one.
6. Click **Finish**.

What to do next

After creating elements, you can use the test editor to change values. You can also replace values with a dataset variable or a reference.

Adding an SAP sequence element

You can use SAP sequence elements to specify complex interactions with the SAP GUI that involve multiple actions within a single object.

Before you begin

SAP sequence elements represent complex user interactions with the SAP GUI, and contain multiple SAP set, get, or call child elements where each child element of the sequence is an action that relies on the return result of the preceding child element. In most cases, SAP sequences are recorded when you interact with complex SAP GUI objects. However, with advanced knowledge of the SAP API, you can use SAP sequences to create complex actions or verification points.

SAP sequence elements are contained in SAP screen elements.

You can use either the test editor or the **SAP Protocol Data** view to create or edit SAP sequences. When using the **SAP Protocol Data** view, you can select SAP screen objects from the screen capture and copy the information directly to the new SAP sequence element.

The SAP Protocol Data view contains two pages that are synchronized with each other and with the test editor:

- **Screen Capture** displays a graphical screen capture of the SAP GUI. You can select all GUI objects such as windows, buttons, fields or areas.
- **Object Data** provides information about the selected GUI object: identifier, type, name, text, tooltip, and subtype.

1. Open the test in the test editor and the **SAP Protocol Data** view.

If the **SAP Protocol Data** view is not open, click **Window > Show View > Other > Test > SAP Protocol Data**

2. In the **Test Contents** area of the test editor, expand a transaction and a SAP screen.

The **SAP Protocol Data** view displays a screen capture of the selected transaction.

3. Inside the SAP screen, select where you want to enter the new sequence.

Result

The **Screen Capture** page of the **SAP Protocol Data** view displays the screen capture of the current SAP screen.

4. In the **SAP Protocol Data** view, right-click the SAP GUI object for which you want to create the SAP sequence element, and then click **Create Element**.

Result

This opens the **Create Element** window, which already contains the **Identifier** from the recorded session.

5. Select **Advanced**, select **SAP Sequence** in the **Element type** list, and then click **Finish**.

6. In the **Test Contents** area of the test editor, select the SAP sequence and click **Add > Element** to add a SAP set, get, or call element to the SAP sequence.

7. In the **Test Contents** area of the test editor, select the SAP sequence, and then click **Add** or **Insert** to add sequence child elements as required. In the **Create Element** window, use the **Cast** button to cast the results of the previous child element to match the expected input type. See the SAP API documentation for detailed information on SAP objects and calls.

Adding a batch input transaction

You can import batch input transactions that were recorded from the SAP GUI into an SAP batch input test. A batch input test can contain multiple batch input transactions.

Before you begin

You can add batch input transactions only to a batch input test, not to a regular SAP performance test. First create a new batch input test.

Batch input tests access the SAP server at a low level, bypassing the SAP GUI interface, and therefore cannot contain any verification points or SAP GUI elements. Their main purpose is to simulate a load on the server when added to a test schedule that already contains SAP performance tests. Only the SAP performance tests provide accurate SAP application performance measurement.

To add an SAP batch input transaction to a batch input test:

1. Open the batch input test in the test editor.
2. In the **Test Contents** area of the test editor, right-click the test node, and click **Add > Transaction**.
Placing the batch input transaction inside a transaction is not mandatory, but it is a good practice, because the performance report shows the results for each transaction separately.
3. Right-click the transaction and click **Add > SAP Batch Input Transaction**.

Result

This opens the **SAP Batch Input File Selection** window.

4. Select a batch input transaction file that was recorded with the SAP GUI batch input recorder, and then click **Open**.

What to do next

After creating elements, you can use the test editor to change values. You can also replace values with a dataset variable or a reference. You can also place the batch input transaction inside a loop to make it repeat several times during the test.

Splitting an SAP test

After you record a test, you can split it into smaller tests. Splitting tests enables you to create modular building blocks of smaller tests and combine them to make bigger tests. The original test is unchanged. You can recombine these building blocks in a schedule, including loops and conditions.

Before you begin

When reusing split tests in a schedule, you must ensure that the general test structure is consistent including SAP session logon and logoff details. For example, if split test a contains the session logon details and split test c contains the logoff transaction, you must place them in the correct order in the schedule user group.

SAP tests can be split only at the transaction level. When the test is executed in the schedule, it must start with a SAP logon transaction and end with a SAP logoff transaction.

1. In the Test Navigator, browse to the test and double-click it. The test opens.
2. Right-click a SAP transaction in the test, and select **Split Test**. The page that you click is the first page of the new test.
3. In the **New Test Names** window, confirm the location of the split, optionally provide names and descriptions for the split tests, and click **Next**.
4. In the **Split Test** window, examine the changes to be performed as a result of the split, and click **Finish**.

Exemple

For example, you could record a test that contains the following actions:

- Logging on to a server.
- Creating an entry on the server and removing the entry.
- Editing an entry, validating that the change occurred, and restoring the entry.
- Logging off of the server.

You then split the test into four parts: Logon, Create, Edit, and Logoff. You create a schedule that runs virtual users selected from a dataset. Each virtual user runs the Logon test, performs various combinations of the Create and Edit tests, and finally runs the Logoff test.

Viewing GUI data in the SAP Protocol Data view

The SAP Protocol Data view provides a graphical view of screens as they are displayed in the SAP GUI. In addition, it provides a view of the SAP GUI object data. The data displayed in the SAP Protocol Data view is synchronized with the test elements selected in the test editor.

To view test contents in the SAP Protocol Data view:

1. Open the test.
2. Click the **SAP Protocol Data** tab to activate the view.

Result

As shown in the example, the tab color changes to blue and the tabs for the Protocol Data view are displayed.

If you cannot locate the **SAP Protocol Data** tab, you can open this view by clicking **Window > Show View > Other > Test > SAP Protocol Data**.

3. In the test editor, click the line corresponding to the transaction, screen or SAP event that you want to view.
4. In the **SAP Protocol Data** view, click the tab corresponding to the type of data or view of interest.
 - Clicking **Screen Capture** opens a page that displays the corresponding screen as recorded in the SAP GUI client. If a SAP event is selected, the corresponding field, button or GUI object is highlighted in red.
 - Clicking **Object Data** opens a page that displays the SAP GUI object data for the corresponding object. This data can be used to identify the object in a test element.

Editing Citrix tests

After you record a test, you can edit it to include variable data (rather than the data that you recorded), verification points (to confirm that the test runs as expected), transactions, conditional processing, and custom code.

Citrix test editor overview

With the test editor, you can inspect or customize a test that you recorded.

The test editor lists the window events for a test, in sequential order. New windows are displayed in bold. The Windows™ operating system assigns each window an ID number. This number changes on each execution of the test, but usually remains the same within the test, providing a means of identifying each window object.



Note: In some cases, the operating system recycles destroyed window IDs. The test recorder identifies these properly by appending an extra number at the end of the window ID if necessary.

There are two main areas in the test editor window. The area on the left, **Test Contents**, displays the chronological sequence of events in the test. The area on the right, **Test Element Details**, displays details about the currently selected item (window, mouse event, key event, or screen capture) in the test hierarchy. The **Common Options** and **Citrix Options** apply to the entire test.

Under the test is the Citrix session, which contains information about the connection and Citrix XenApp client options, such as color depth and resolution.

Window events are the primary test elements in a Citrix test and represent graphic objects that are drawn by the Citrix server, such as actual window, dialog boxes, menus, or tooltips. A Window event is recorded each time a window is created, destroyed, moved, or resized. The first occurrence of a window, a create window event, is displayed in bold. Window objects are typically identified by their title. If there is no window title, for example on menus or tooltips, then the test editor uses the window ID number.

Inside windows, you see a list of events for the window, such as create window events, screen captures, mouse or keyboard actions.

Some actions contain data that is highlighted. This highlighting indicates that the data contains one or both of the following types of information:

- **A dataset candidate:** This is a value, usually one specified by the tester during recording, that the test generator determined is likely to be replaced by values in a dataset. An example of a dataset candidate is a string that you search for in a recorded test. The string is highlighted as a dataset candidate on the assumption that, before running the test, you might want to associate the string with a dataset column containing appropriate substitute values.
- **References:** These are values in a test, usually one of them in a response and the other in a subsequent request, that the test generator determined needed to be associated in order to ensure correct test execution. An example is a photograph returned to the browser by a test that searches an employee database. The test generator automatically correlates employee names with photographs. Suppose that, before running the test

with many virtual users, you replace the employee name searched for in the recorded test with names in a dataset. Because the test correlates the data, each virtual user searches for a different employee, and the server returns an appropriate photograph.

To see an illustration of color coding in performance tests, click **Window > Preferences > Test > Test Editor**, and then click the **Fonts and Colors** tab.

Click **Add** to add elements to the selected test element. Alternatively, you can right-click a test element and select an action from a menu.

The choices that you see depend on what you have selected. For example, inside a window, you can add a mouse action or a text input. The **Insert** button works similarly. Use it to insert an element before the selected element. The **Remove** button allows you to delete an item.



Note: Because Citrix performance tests rely on low level interaction with the server, manually changing test elements is likely to break a recorded test.

Citrix synchronization overview

During the run of a Citrix session, the test uses window events or image recognition to ensure that the correct user input actions are maintained on track with the application events coming from the Citrix server.

The synchronization mechanism enables the test to remain synchronized without relying only on the timing, which could vary with the load on the Citrix server.

Synchronization is different from the function of verification points. Verification points check specified values of the application, such as window synchronizations, and produce a test status: *pass*, *fail*, *error*, or *inconclusive*. Synchronization is used for test execution and produces synchronization timeouts in the test log when the test fails to recognize the server output.

Synchronization occurs on window events or through the recognition of a screen area that you specify.

- Window event synchronization is produced automatically by the Citrix server. When the test is run, the virtual users send emulated user actions to the server, such as keyboard or mouse actions and the server responds with window events such as *create*, *activate*, or *destroy*. The test waits for the expected window events to occur before sending the next user actions to the server. The test uses window styles, and optionally, locations, sizes, and window titles, to recognize windows.
- Image synchronization enables you to require additional recognition of screen area contents. The test can synchronize on either a unique bitmap hashcode that is calculated from the image or on a text string retrieved from the image through optical character recognition. In some applications, such as web browsers or word processors, the actual window content changes more frequently than the window objects. In these cases, you must manually add image synchronization to critical parts of the test during the recording so that the test can synchronize with the window contents.

During test execution, the test waits for the window event or the image recognition to synchronize the user actions independently from the load on the server. If an expected window event fails to occur or an expected image is not recognized, then the test produces a synchronization timeout that is reported in the test log. The test will attempt to resume the execution at the next synchronization point.

The base timeout delay is specified in the Citrix test generation preferences; however, the actual delay varies with the level of synchronization.

There are three levels of synchronization for window events and images that can be specified for each element in the test editor:

- **Conditional:** This is the default behavior for main windows and dialog boxes. If the synchronization fails, the test tries to continue and the synchronization timeout is logged in the Citrix performance report and the test log.
- **Mandatory:** If the synchronization fails after a period that is three times the base timeout delay (by default), the test exits with an *error* status and the connection with the Citrix server is closed. The test execution continues to run until test results are finalized.
- **Optional:** Synchronization is not required but is logged in the test log. A timeout occurs after 1/10th of the base timeout delay.

The default timeout values can be overridden for each synchronization element in the test editor. The test recorder automatically sets the recommended synchronization level for window events depending on their nature:

- Main window create events are set to mandatory.
- Other window create events are set to conditional.
- Main window destroy events are set to conditional.
- Other window destroy events are set to optional.
- Image synchronizations are set to conditional.

Verifying application behavior

To check the expected behavior of the application during a Citrix performance test, you can add verification points at strategic points in the test. You can use window verification points to check that a specific window opens during the test, regardless of its contents, or you can use image synchronization to verify the displayed contents. During the run, verification points produce a pass, fail, error, or inconclusive status in the Citrix Verification Point report. You can also measure response times between two test elements.

Enabling Citrix window verification points

You can use verification points on window titles to check whether a window with a specific caption is created during the test. Alternatively you can set a verification point on the synchronization criteria of the window to check whether the window position, size, and style match the expected criteria, regardless of the contents. You can enable window verification points for a specific test or generate them automatically by setting the test editor preferences.

Before you begin

During the run, verification points produce a pass, fail, error, or inconclusive status in the Citrix Verification Point report and in the test log.

Mandatory window synchronization events always have a verification point enabled. This setting causes the test to have a *fail* status if the window does not synchronize.

About this task

To automatically enable verification points each time a window title changes during the recording, click **Window > Preferences > Test > Performance Test Generation > Citrix Test Generation**, and select **Verification point on every window title change**.

When verification points are disabled, you can enable verification points for a specific test:

1. Open the test.
2. In the test editor, select a session or a window event.
Your choice determines whether the verification point that you select is added to all windows in the test or to a particular window in the test.
3. Right-click, and select **Enable Window VPs**.
To inspect or set your verification points, see [Specifying window verification point criteria on page 328](#).



Note: Verification points on mandatory window synchronization elements cannot be disabled.

Enabling Citrix image synchronization verification points

You use verification points on image synchronization elements to check whether the contents of a screen area match either an expected bitmap or a text string. You can add image synchronization verification points to a recorded image synchronization element.

Before you begin

During the run, verification points produce a pass, fail, error, or inconclusive status in the Citrix Verification Point report and in the test log.

To add a verification point to a recorded image synchronization element:

1. In the test editor, select an image synchronization test element that you added during the recording.
2. In the **Test Element Details** area, on the Synchronization page, select the verification method:

Choose from:

- Select **Bitmap hash code** to verify the exact contents of the image.
- Select **Optical character recognition** to verify the recognized text in the captured screen area, and then click **Extract Text** to obtain the text that is expected.

If the text extraction is unsuccessful, try changing the text recognition settings on the **Options** page. However, accuracy of the recognized text is not essential. It is only important that the recognized text is consistent each time the test runs for the verification to pass.

3. In the Test Element Details area, on the Settings page, select **Enable verification point on synchronized image**.

Related reference

[Citrix image synchronization details on page 1222](#)

[Citrix test generation preferences on page 1184](#)

Related information

[Citrix synchronization overview on page 325](#)

[Synchronizing tests with screen areas on page 333](#)

[Adding values to an image synchronization on page 335](#)

Specifying window verification point criteria

Window verification points produce a fail status in the test execution report if they differ from the specified expected criteria.

1. In the test editor, select a window event element.

You can set a verification point on any window event that creates, activates, or destroys a window that contains a title.
2. Ensure that **Enable verification point on synchronized window event** is enabled for the selected window event element.
3. In the test editor, select the window element that contains the window event.
4. Select the event synchronization criteria:
 - **X position** and **Y position**: Select these options to specify that the top left corners of the window must be located at the same coordinates.
 - **Width** and **Height**: Select these options to specify that the window must be the same size.
 - **Title**: Select this option to specify that the window must have the same title.

The window styles are a mandatory criteria and cannot be disabled.

5. If you selected **Title**, the **Window title** field shows the expected title.

If necessary, you can change the expected title in the **Window title** field.

You can use standard regular expressions to specify the expected title by selecting **Use regular expressions**.

Results

Whenever the test runs with a verification point enabled on a window create, activate, or destroy event, a *fail* status is reported in the test log if the criteria returned by the Citrix server for the window does not contain the expected criteria.

Measuring response times

A response time measures the delay between a specified start event and a specified stop event. Typically, the start of the measurement is set to a mouse or keyboard input. Similarly, the stop is set to a window create, window activate, or window destroy event that is a response to the input event, or an image synchronization element that was recorded with the test.

Before you begin

Response time measurements require a start element and a stop element. If either of these are missing, the response time definition is displayed with an error or warning marker.

About this task

When the **Response time for main windows** option is selected in the Citrix Test Generation preferences, recorded tests are generated with a response time measurement each time a main window is created. Generated measurements start on the event that occurred immediately before a main window is created and stop when the main window is created.

You can view all the response times of a test by selecting the Citrix test element in the test editor. Response times are listed in the **Response Time Definitions** table where they can be edited, renamed, or deleted.

1. In the test editor **Test Contents** area, select a test element to start the response time measurement.
2. Press the Ctrl key, and select another test element to stop the response time measurement.
The two elements are selected in the test. The first element is the start of the response time measurement and the second is the stop.
3. Right-click either the start or stop element, and click **Create Response Time**.
4. A **Create Response Time** window displays information about the new response time measurement. If the new response time measurement replaces a previous one, click **Yes**. Otherwise, click **OK**.
5. **Optional:** To view all the response times that are defined for the test, click the test element in the test navigator, and select the **Citrix Response Time** page.



Note: By default, response time measurements are included in the test results, even when the synchronization fails with a timeout. To exclude failed synchronizations from the response time results, clear the **State** option in the **Response Time Definitions** list.

Adding elements to a Citrix test

You can add a variety of elements to a test, such as user input actions, comments, loops or conditions.

Editing a mouse action

You can edit mouse actions to manually specify how the mouse interacts with the Citrix server.

Before you begin

Mouse actions describe low-level user actions using the mouse such as mouse movements, clicks and double-clicks, or drag-and-drop operations. Mouse actions can be added only inside window events or mouse sequences. For example, a drag-and-drop operation is described as a mouse down action at a specific location, a sequence of mouse move actions, and a mouse up action at the destination location.

In most cases, you will rely on the mouse actions that were recorded with the test. However, in some cases, you might want to manually refactor mouse action sequences that are redundant or poorly recorded. For example, a double-click event can sometimes be recorded as a mouse down, a mouse up, and a click. The result will be the same as a double click action but the test will be more difficult to read.



Note: Because Citrix tests contain low level user input and synchronizations, minor changes can prevent the test from working. When editing these tests, you must ensure that they are functionally identical.

1. Open the test in the test editor.
2. Expand a window action.

Choose from:

- To edit a mouse action, select the event in the **Test Contents** area.
 - To create a new action at a specific location, select an action and click **Insert** and **Citrix Mouse**.
3. In the **Test Element Details** area, specify the type of event.
 - **Mouse move:** This indicates that the mouse is moved from the current coordinates to the coordinates specified in **X Position** and **Y Position**.
 - **Mouse click:** This indicates that the mouse is clicked at the coordinates specified in **X Position** and **Y Position**. In the **Buttons** area, select the button that is clicked.
 - **Mouse double click:** This indicates that the mouse is double clicked at the coordinates specified in **X Position** and **Y Position**. In the **Buttons** area, select the button that is double clicked.
 - **Mouse down:** This indicates that a mouse button is pressed at the coordinates specified in **X Position** and **Y Position**. In the **Buttons** area, select the button that is double clicked.
 - **Mouse up:** This indicates that a mouse button is released at the coordinates specified in **X Position** and **Y Position**. In the **Buttons** area, select the button that is double clicked.
 4. You can specify the think time for the mouse action. This emulates the time spent by a user before initiating the current event.

Viewing a mouse sequence

You can use mouse sequences to view complex mouse movements on the screen.

Before you begin

Mouse sequences provide a graphical view of a series of mouse move events. This is particularly useful when a large number of mouse move events are meaningful to the application. For example, drawing the letter e in a paint program is represented as a mouse down event, a mouse sequence, and a mouse up. In the test editor, the sequence graphically displays how the letter e is drawn in the application. If any screen captures were taken during the recorded session, the sequence will be shown over the screen capture.

In most cases, you will rely on the mouse sequences that were recorded with the test. However, in some cases, you might want to manually refactor mouse events into sequence.



Note: Because Citrix tests contain low level user input and synchronizations, minor changes can prevent the test from working. When editing these tests, you must ensure that they are functionally identical.

1. Open the test in the test editor
2. Expand a window event and select the sequence in the **Test Contents** area.
3. In the **Test Element Details** area, you can view the following information:

Display mouse sequences for

This option specifies how you want to display previous, current, or all mouse sequences in the current mouse sequence:

Current® sequence

Only the current mouse sequence is displayed in the test editor. This option is selected by default.

Previous and current sequences

The current mouse sequence is displayed with any previous mouse sequences.

All sequences

All mouse sequences are displayed simultaneously.

Fit screen to visible area

Select this option to adjust the display of the mouse sequence to the available area in the test editor. If disabled, the screen capture will be the actual size, which might require scrolling. This option is enabled by default.

Screen capture area

This area represents the mouse movements on the screen. If a screen capture was recorded, it is displayed in the background. Mouse sequences are displayed as specified.

Editing a keyboard action

You can edit keyboard actions to manually specify how the keyboard interacts with the Citrix server.

Before you begin

Keyboard actions describe low level user actions using the keyboard such as text inputs or keyboard shortcuts. Keyboard actions can be added only inside window events.

In most cases, you will rely on the keyboard actions that were recorded with the test. The recording tries to factor multiple key presses into text input actions. However, in some cases, you might need to manually factor keyboard actions that are redundant or poorly recorded. For example, in some cases, a text input string `He11o` can be recorded as a press on the Shift key, a press on the **H** key, a release of the **Shift** key, a stroke of the **E** key, and finally a text input of **llo**. By manually factoring keyboard actions into text inputs, you can handle text from dataset variables and references.



Note: Because Citrix tests contain low level user input and synchronizations, minor changes can prevent the test from working. When editing these tests, you must ensure that they are functionally identical.

1. Open the test in the test editor
2. Expand a window event.

Choose from:

- To edit a keyboard action, select the action in the **Test Contents** area.
 - To create a new keyboard action at a specific location, select an action and click **Insert** and **Citrix Keyboard**.
3. In the **Test Element Details** area, specify the type of action.
 - **Key Stroke:** This indicates that a key is pressed and released.
 - **Key Down:** This indicates that a key is pressed and held down.
 - **Key Up:** This indicates that a pressed key is released.

The **Key Code** field displays the key code as interpreted by the Windows™ operating system and is translated in the Character field. Use the modifiers to specify the whether the Control key, Shift key, or Alt key is also pressed.

4. You can manually enter any Unicode character that is not normally available through single keystrokes by using the **Character Edition** area. Select the input field and enter the character on your keyboard.

The **Key Code** and **Character** fields display the corresponding character.



Note: The workbench uses some key combinations as keyboard shortcuts. Such combinations can be intercepted and cause undesirable actions instead of displaying a particular character in the Character field..

5. You can specify the think time for the keyboard event. This emulates the time spent by a user before initiating the current event.

Editing a text input

You can edit text inputs to replace a sequence of keyboard events with text strings that are easier to handle as dataset variables or references.

Before you begin

Text inputs describe a series of low-level keyboard events as a single text string. Text inputs can be added only inside window events.

In most cases, you will rely on the text inputs that were recorded with the test. The recording attempts to factor multiple key presses into text input events. However, in some cases, you might need to manually factor keyboard events that are redundant or poorly recorded. For example, in some cases, a text input string `He11o` can be recorded as a press on the **Shift** key, a stroke of the **H** key, a release of the **Shift** key, a stroke of the **E** key, and finally a text input of **llo**. By manually factoring keyboard actions into text inputs, you can handle text from dataset variables and references.



Note: Because Citrix tests contain low level user input and synchronizations, minor changes can prevent the test from working. When editing these tests, you must ensure that they are functionally identical.

1. Open the test in the test editor
2. Expand a window event.
 - Choose from:**
 - To edit a text input, select the action in the **Test Contents** area.
 - To create a new text input at a specific location, select an action and click **Insert** and **Citrix Text**.
3. In the **Test Element Details** area, edit the text string that will be entered during the test.
4. You can specify the think time for the keyboard event. This emulates the time spent by a user before initiating the current event.

Synchronizing tests with screen areas

Image synchronization enables Citrix performance tests to keep track of the contents of a screen area during replay instead of focusing only on window events.


Before you begin

You can use image synchronization elements in a test to facilitate the test replay in applications that do not create or modify many windows, but update the contents of the window regularly.

The contents of an image are processed as a value that is either a calculated bitmap hashcode or a text string obtained by optical character recognition. During test execution, the test waits for the contents of a screen area to match the expected image synchronization value before proceeding with the test. If the value is not matched during the synchronization period, an image synchronization timeout is produced in the test log.

You can also add verification points to image synchronizations in the test editor. The verification point passes if the synchronization succeeds and fails if a timeout occurs.

You insert image synchronizations during test recording.

1. Start recording a Citrix performance test and record a sequence of user actions.
2. To add an image synchronization to the recorded test, in the **Citrix Recorder Control** window, click the **Insert image synchronization**  button, select an area of the screen that will be used for synchronization, and then click the **Insert image synchronization** button again.
3. Continue the recording. When you have completed the sequence of actions to be tested, end the Citrix session cleanly and close the Citrix XenApp client.

Result

A progress window opens while the test is generated. On completion, the **Recorder Control** view displays the message `Test generation completed`, the Test Navigator lists your test, and the test opens in the test editor.

4. In the test editor, select the **Image synchronization** element.
 - a. Set the synchronization state to **Conditional**, **Mandatory** or **Optional**. In most cases, because you are explicitly requesting for a synchronization to occur, you should leave the synchronization state as **Conditional**.
 - b. Specify one of two synchronization methods.
 - Select **Bitmap hash code** if you want the synchronization to occur when the selected area matches exactly the recorded image. A single pixel of difference will cause a synchronization timeout.
 - Select **Optical character recognition** if you want the synchronization to occur when a text string is recognized in the selected area. Click **Extract text** to test the text recognition on the recorded image. Note that for synchronization purposes the text recognition results must be repeatable, not necessarily accurate.



Note: On Windows 2008, it is recommended to use the Bitmap hash code synchronization method for a Citrix test.

You can click **Extract text** several times to make sure that the text recognition results are repeatable. If the recognized text differs, click **Options** to change the **Zoom factor** or other optical character recognition settings. You should not manually correct the recognized text.

You can click **Add** to specify multiple expected text strings, **Use regular expression** to specify a text string as a regular expression, or **Factorize** to automatically generate a regular expression from multiple expected text strings.

- c. Specify whether you want to enable a verification point on the image synchronization. This enables the test to produce a pass or fail status in the Citrix Verification Points report.

Manually adding an image synchronization

You can manually add an image synchronization element to an existing test to ensure that the test synchronizes correctly or to add a verification point. To do this, you must create the image synchronization element, and then perform a single run of the test to retrieve the image hash code or text detected by optical character recognition (OCR).

1. Open the test in the test editor.
2. Select the location where you want to create the image synchronization and click **Insert** and **Citrix Image Synchronization**.

Result

This creates a new image synchronization element that is set to Conditional and has several values disabled. This is because the test does not yet contain the image hash code or recognized text.


3. In the screen capture area, move and resize the black rectangle to specify the zone of the image that will be used for the synchronization.

Alternatively, you can specify the pixel values in **X position**, **Y position**, **Width** and **Height**.

4. Select **File > Save** to save the test, and then, in the test navigator, right-click the test and select **Run As > Performance Test**.

Result

This runs the test once to retrieve the image hash code or recognized text.

5. After running the test, right-click the test report, select **Display Test Log**, and then select the **Events** page.
6. Click the **Events** tab.
7. Expand the top line of the **Events** hierarchy and navigate to the **Image timeout** element that was produced during the run.
8. Select the image timeout to open the **Citrix Image Synchronization** view.
If necessary, you can open the **Citrix Image Synchronization** view manually by selecting **Window > Show View > Other > Test > Citrix Image Synchronization**. This view shows the actual screen area encountered during the test and the expected image, as well as the corresponding hash codes or recognized text values.
9. To add the actual hash code or recognized text to the image synchronization, click the **Add value**  button.

Adding values to an image synchronization

You can replace or add alternative values in an image synchronization element so that synchronization is not restricted to the exact recorded image. Alternative values can also be useful in updating the test if the behavior of the application changes.

Before you begin

If you are using optical character recognition, simply click **Add** in an image synchronization element in the test editor to add an alternative value, and enter a new text string in the **Expected text** field.



If you are using the bitmap hash code synchronization or optical character recognition, you can use the following method to add an alternative hash code or recognized text value.

To add an alternative value to an image synchronization:

1. After inserting an image synchronization element, run the test from the Run menu.
If possible, arrange for the alternative condition to occur before starting the run. The test should produce an image synchronization timeout when the alternative condition is encountered.
2. In the **Performance Test Runs** view, right-click the test log container, and then select **Display Test Log**.
The test log opens in the editor with the **Overview** page selected.

3. Click the **Events** tab.
4. Expand the top line of the **Events** hierarchy and navigate to the **Image timeout** element that was produced during the run.
5. Select the image timeout to open the **Citrix Image Synchronization** view.
If necessary, you can open the **Citrix Image Synchronization** view manually by selecting **Window > Show View > Other > Test > Citrix Image Synchronization**. When an image timeout is selected, this view shows the actual screen area encountered during the test and the expected image, as well as the corresponding hash codes or recognized text values.
6. You can either add the new hash code or recognized text value to the image synchronization as an alternative value or you can replace the existing value with the new value.

Choose from:

- To add the actual hash code or recognized text as an alternative value, click the **Add value**  button.
- To replace the expected value with the new value, click the **Replace value**  button.

Using custom code with a Citrix test

You can write custom Java™ code to expand the functions of HCL OneTest™ Performance.

Before you begin

Custom code requires knowledge of Java™ programming and the use of the HCL OneTest™ Performance API. See [Executing test execution with custom code on page](#) for more information.

About this task

To use custom code for test synchronization:

1. In the test navigator, select the test element location to insert the custom code.
2. Click **Insert > Custom Code**.

Result

A custom code test element is created in the test.

3. On the **Test Element Details** page, click **Generate Code** to create a Java™ class based on the HCL OneTest™ Performance API.

You can click **View Code** to edit an existing class.

4. In the Java™ editor, add the import statement for Citrix tests: `import com.ibm.rational.test.lt.execution.citrix.customcode.*;`
5. Complete the **exec** method to specify the function to create.
6. Save and close the Java™ class.

Example

The following example is custom code class that can be used as a starting point to evaluate the results of a synchronization point. You can use this template to write a class that performs a synchronization when image synchronization and window-event synchronization are not practical for your test.

```
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import org.eclipse.hyades.test.common.event.VerdictEvent;
```

```

import com.ibm.rational.test.lt.execution.citrix.customcode.CitrixCustomCodeImpl2;
import com.ibm.rational.test.lt.execution.citrix.customcode.ICitrixCustomCode2;

public String exec(ITestExecutionServices tes, String[] args) {
    ICitrixCustomCode2 thisCode = new CitrixCustomCodeImpl2(tes);

    // to get the last VP status
    int verdict = thisCode.getLastVerificationPointVerdict();
    if (verdict != VerdictEvent.VERDICT_PASS) {

        // this example reports a message but must be adapted to your specific needs
        tes.getTestLogManager().reportMessage("last VP status: " + thisCode.verdictEventToString(verdict));
    }
    return null;
}

```

The following example demonstrates how you can record a screen capture during playback for debugging purposes.

The screen capture is recorded in the test log and can be viewed in the **Citrix image synchronization** view.

```

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.execution.citrix.customcode.*;

public String exec(ITestExecutionServices tes, String[] args) {

    ICitrixCustomCode2 thisCode = new CitrixCustomCodeImpl2(tes);

    // To capture and log the full screen:
    thisCode.logFullScreenCapture();

    // To capture and log a part of the screen:
    // thisCode.logPartialScreenCapture(x, y, width, height);

    // To capture and log a part of the screen to a file:
    // thisCode.savePartialScreenCapture(filename, x, y, width, height);

    return null;
}

```

What to do next

After creating a custom code test, you can run the test as usual. If you need to debug the test, you can use the monitoring panel to insert breakpoints or to interact with the Citrix environment during execution.

Related information

[Debugging Citrix tests on page 654](#)

Editing service tests

After you record a service test, you can edit the calls and message returns to include variable data (rather than the data that you recorded). You can add verification points (to confirm that the test runs as expected), transactions, conditional processing, and custom code.

Web service test editor overview

With the test editor, you can inspect or customize a test that you recorded.

The test editor lists the web service call elements for a test, in sequential order.

There are two main areas in the test editor window. The area on the left, **Test Contents**, displays the chronological sequence of test elements in the test. The area on the right, **Test Element Details**, displays details about the currently selected item (test, call, message return, or verification point) in the test hierarchy.

Window events are the primary test elements in a Citrix test and represent graphic objects that are drawn by the Citrix server, such as actual window, dialog boxes, menus, or tooltips. A Window event is recorded each time a window is created, destroyed, moved, or resized. The first occurrence of a window, a create window event, is displayed in bold. Window objects are typically identified by their title. If there is no window title, for example on menus or tooltips, then the test editor uses the window ID number.

A service request node name can be updated automatically or you can use custom code or dataset to supply different names. To apply a dataset to a node name, in the test editor, select the node name. In the Request Details area of the test editor, clear the **Update node name automatically**, select the name and substitute it with dataset.

Web service calls can contain web service message return elements, which display the results of the web service call. The XML message content can be displayed either in Form, Tree or Source view. Each of these views displays the same message content in different forms:

- **Form** view provides a simplified view of the call elements focused on editing the values of the XML message content.
- **Tree** view provides a hierarchical view of the XML structure, including elements, namespaces, and the associated values. Tree view also allows you to manipulate XML fragments.
- **Source** view displays the XML contents of a web service or XML call or the plain text contents of a simple text message.

Message return elements can contain verification point elements that check that the actual return results match expected criteria.

Some actions contain data that is highlighted. This highlighting indicates that the data can be used as a dataset candidate or as a reference. See [Data correlation overview on page](#) for more information.

In service calls and message returns, you can use datasets and data correlation on values contained in the XML or on XML fragments. To use data correlation on XML fragments, switch to the Tree view, right-click the XML element and select **Create XML Fragment**.

To view or modify the color coding in web service tests, click **Window > Preferences > Test > Test Editor**, and then click the **Fonts and Colors** tab.

Click **Add** to add elements to the selected test element. Alternatively, you can right-click a test element and select an action from a menu.

The choices that you see depend on what you have selected. For example, inside a web service call, you can add a web service message return. The **Insert** button works similarly. Use it to insert an element before the selected element. The **Remove** button allows you to delete an item.

Verifying application behavior

To check the expected behavior of the application during a service test, you can add verification points after a message return. During the run, verification points produce a pass, fail, error, or inconclusive status in the Web Service Verification Point report.

Adding equal verification points

Equal verification points enable you to check that the contents returned by a service match exactly the contents specified in the verification point.

About this task

When you add verification points, the results from a service response are compared with the expected data specified in the verification point test element. *Equal* or *contain* verification points enable you to directly compare the XML document that the service returns.

- Contain verification points return a Pass status when the response XML document contains the expected XML data.
- Equal verification points return a Pass status when the response XML document matches exactly the expected XML data.

Complex service requests or verification points might have empty XML elements that are not needed in a test script. When playing back the test, you can skip such empty XML elements. In **Window > Preferences > Test > Test editor > Service test** ensure that the **Display the 'Skip if empty' column in XML tree viewer** check box is selected. This option displays a **Skip if empty** column in the tree view of the request. You can then choose the XML elements to skip.

1. Open the test editor, and right click a response element and select **Add > Equal Verification Point**.
2. Select the verification point, and in the **Test Element Details** area of the test editor, type a name for the verification point.
3. Select the verification options:
 - Select **Test using XML namespaces** to perform the verification on the qualified structure of the XML document, including the namespace tagging, instead of the simple name. Disable this option to check only the simple name of the element and the final return value.
 - Select **Text XML text nodes** to include the content of text elements in the verification.
 - Select **Text XML attributes** to include the content of attributes in the verification.
4. On the Message page, select the **Form**, **Tree**, or **Source** view to specify the expected XML data.

For an equal verification point, the expected XML data contains the XML document from the response test element. If necessary, you can edit the expected XML data.

You can specify standard Java™ regular expressions in the **Tree** view. To do this, select the **Regular expression** column on the line of an attribute or text value and type the regular expression in the **Value** column. For example, the following regular expression checks for a correctly formatted email address: `/^[a-zA-Z0-9_\.\\-]+\@(([a-zA-Z0-9\\-]+\\.)+([a-zA-Z0-9]{2,4})+)$/`

When using regular expressions, the number of XML nodes or XML fragments in the verification point must match the quantity of expected nodes.

What to do next

You can enable or disable each verification point by right-clicking the verification point in the test editor and clicking **Enable** or **Disable**.

Adding contain verification points

With contain verification points, you can check that one or several elements of the XML content returned by a service match the XML fragment that is specified in the verification point.

About this task

When you add verification points, the results from a service response are compared with the expected content that is specified in the verification point test element. *Equal* or *contain* verification points enable you to directly compare the XML contents that the service returns.

- Contain verification points return a Pass status when the response XML contents contain the expected XML fragment.
- Equal verification points return a Pass status when the response XML contents match exactly the entire expected XML content.

Complex service requests or verification points might have empty XML elements that are not needed in a test script. When playing back the test, you can skip such empty XML elements. In **Window > Preferences > Test > Test editor > Service test** ensure that the **Display the 'Skip if empty' column in XML tree viewer** check box is selected. This option displays a **Skip if empty** column in the tree view of the request. You can then choose the XML elements to skip.

1. Open the test editor, and select a service response element.
2. In the **Test Element Details** area, click the **Message** tab and select the **Form** or **Tree** view.
3. Expand the envelope line, right click the element that you want to check, and then click **Create Contain Verification Point**. This action creates a contain verification point that includes the XML element from the recorded response.



Note: You can also create a contain verification point with the message response by selecting the message response in the **Test Contents** pane and clicking **Add > Contain Verification Point**. However,



the result is effectively the same as an equal verification point because the verification point contains the entire XML content of the message response.

4. Select the verification point, and in the **Test Element Details** pane, type a name for the verification point.
5. Select the verification options:
 - Select the **Test using XML namespaces** check box to perform the verification on the qualified structure of the XML document, including the namespace tagging, instead of the simple name. Disable this option to check only the simple name of the element and the final return value.
 - Select the **Test XML text nodes** check box to include the content of text elements in the verification.
 - Select the **Test XML attributes** check box to include the content of attributes in the verification.
6. If necessary, select the **Form**, **Tree**, or **Source** views to edit the expected XML fragment.

For an equal verification point, the expected XML data contains the XML document from the response test element. If necessary, you can edit the expected XML data.

You can specify standard Java™ regular expressions in the **Tree** view. Select the **Regular expression** column on the line of an attribute or text value and type the regular expression in the **Value** column. For example, the following regular expression checks for a correctly formatted email address: `/^[a-zA-Z0-9_\. \-]+\@(([a-zA-Z0-9\ -])+\.)+([a-zA-Z0-9]{2,4})+$/`

When using regular expressions, the number of XML nodes or XML fragments in the verification point must match the number of expected nodes. The verification point returns a Pass status when all regular expressions in the XML fragment are matched.

Example

You can use a contain verification point to check that the message response contains only a specific element with a specific value. For example, consider the following message response:

```
<s:Envelope
  xmlns:a="http://www.w3.org/2005/08/addressing"
  xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header>
    <a:Action
      s:mustUnderstand="1">http://www.w3.org/2005/08/addressing/soap/fault</a:Action>
    <a:RelatesTo>uuid:ed9bc447-d739-452f-989d-cd48344d494a</a:RelatesTo>
  </s:Header>
  <s:Body>
    <s:Fault>
      <s:Code>
        <s:Value>s:Sender</s:Value>
        <s:Subcode>
          <s:Value
            xmlns:a="http://schemas.xmlsoap.org/ws/2005/02/sc">a:BadContextToken</s:Value>
          </s:Subcode>
        </s:Code>
        <s:Reason>
          <s:Text
            xml:lang="en-US">The message could not be processed. This is most likely because the action
            &apos;http://Samples.ICalculator/Add&apos; is incorrect or because the message contains an invalid or
```

```

expired security context token or because there is a mismatch between bindings. The security context
token would be invalid if the service aborted the channel due to inactivity. To prevent the service
from aborting idle sessions prematurely increase the Receive timeout on the service endpoint&apos;s
binding.</s:Text>
  </s:Reason>
  <s:Node>http://www.w3.org/1999/xlink</s:Node>
  <s:Role>http://www.w3.org/1999/xlink</s:Role>
  <s:Detail
    xmlns:tns0="http://schemas.com/2003/10/Serialization/"
    xmlns:tns15="http://Samples.Windows"
    tns0:Id="id"
    tns0:Ref="idref">
    <tns15:GetCallerIdentityResponse>
      <tns15:GetCallerIdentityResult>str</tns15:GetCallerIdentityResult>
    </tns15:GetCallerIdentityResponse>
  </s:Detail>
</s:Fault>
</s:Body>
</s:Envelope>

```

To check for the `Subcode` element, the expected content of the contain verification point is the following XML fragment:

```

<s:Subcode
  xmlns:a="http://www.w3.org/2005/08/addressing"
  xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Value
    xmlns:a="http://schemas.xmlsoap.org/ws/2005/02/sc">a:BadContextToken</s:Value>
</s:Subcode>

```

By default, the contain verification point checks whether an element named `Subcode` contains one element named `Value`. You can use the following options:

- **Test using XML namespaces:** With this option, the verification point checks whether an element named `"http://www.w3.org/2003/05/soap-envelope":SubCode` contains one element named `"http://www.w3.org/2003/05/soap-envelope":Value`.
- **Test XML text node:** With this option, the verification point also checks whether the element named `Value` contains the text `a:BadContextToken`.
- **Test XML attributes:** With this option, the verification point also checks that the attributes match the expected XML fragment. In this example, the **Test XML attributes** option is not necessary because the `Subcode` element does not have any attributes.

To check that the `Detail` element properly returns a specific value for `GetCallerIdentityResult`, the expected content of the contain verification point is the following XML fragment:

```

<s:Detail
  xmlns:a="http://www.w3.org/2005/08/addressing"
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tns0="http://schemas.com/2003/10/Serialization/"
  xmlns:tns15="http://Samples.Windows"
  tns0:Id="regular_expression"
  tns0:Ref="idref">
  <tns15:GetCallerIdentityResponse>
    <tns15:GetCallerIdentityResult>IdentityValue</tns15:GetCallerIdentityResult>
  </tns15:GetCallerIdentityResponse>
</s:Detail>

```

```
</tns15:GetCallerIdentityResponse>
</s:Detail>
```

You can use the following options:

- **Test XML text node:** With this option, the verification point also checks whether the element named `GetCallerIdentityResult` contains the text `IdentityValue`.
- **Test XML attributes:** With this option, the verification point also checks that the attribute `Id` referred to by `tns0:Id` has the expected value. You can specify a regular expression for this value by using the **Regular expression** column in the **Tree** view of the verification point. For example, `tns0:Id=" [a-zA-Z] "` checks that the value does not contain numbers.

What to do next

You can enable or disable each verification point by right-clicking the verification point in the test editor and clicking **Enable** or **Disable**.

Adding XPath query verification points

With service query verification points, you can check that a response matches an XPath query.

Before you begin

When you add verification points, the results from a service response are compared with the expected data that is specified in the verification point test element. With *query* verification points, you can check that the number of nodes returned by an XML Path language query matches the expected number of nodes specified in the verification point.

Refer to the XPath specification for details on expressing an XPath query: <http://www.w3.org/TR/xpath>.

You can use the test editor to create or edit verification points.

1. Open the test editor, and select a web service response element.
2. Click **Add**, and select **Query verification point**.
3. In the **Test Element Details** area of the test editor, type a name for the verification point.
4. Type a valid **XPath expression** or click **Build Expression** to open the **XPath Expression Builder**.
The **XPath Expression Builder** helps you build and evaluate XPath expressions based on the recorded contents of the response.
5. Specify a **Comparison operator** (`=`, `>`, or `<`), and the expected number of nodes that the query should return.
Click **Evaluate** to update the Expected Count with the actual result based on the recorded contents of the response.

What to do next

You can enable or disable each verification point by right-clicking the verification point in the test editor and clicking **Enable** or **Disable**.



Note: Because XPath expressions require that the qualified name have a prefix, XPath expressions will return null for the default namespace declared with *xmlns*.

Adding attachment verification points

Service attachment verification points enable you to check that the attachment of a service response matches the specified criteria.

Before you begin

When you add verification points, the results from a service response are compared with the expected data that are specified in the verification point test element. *Attachment* verification points enable you to verify that an expected attachment is delivered with the response.

Attachment verification points return a Pass status when all the criteria of an attachment match the expected criteria specified in the verification point test element. If any of the criteria do not match, the verification point returns a Fail status.

You can use the test editor to create or edit verification points.

To add attachment verification points to a performance test:

1. Open the test editor and select a service response element.
2. Click **Add** and select **Attachment Verification Point**.
3. In the **Test Element Details** area of the test editor, type a name for the verification point, and specify the criteria to be verified. All criteria must match in order for the verification point to pass.
 - a. In the case of multiple attachments, set the **Index of attachments** to the index number of the attachment to be checked. Type `1` if there is only one attachment in the response.
 - b. Specify the expected size in bytes of the attachment.
 - c. Specify the MIME type and encoding of the attachment.

What to do next

You can enable or disable each verification point by clicking **Enable verification point** in the test editor.

Adding Text verification points

To check the text content that is returned by the service response, you can add a text verification point in the service test. When you add the verification point, you can check whether the text matches equally with the response or whether the response contains the text.

1. Open the Test editor, right-click a response element and select **Add > Text Verification Point**.
2. In **Verification Point Name**, specify a name for the verification point.
3. In the **Operator** field, select the basis of comparison between the text to be verified and the response content.
4. To search between the offset values, select **From Offset** and **To Offset** check boxes and specify the offset values.

- To search between two string values, select **From String** and **To String** check boxes and specify the strings. You must also specify the number where the strings occurred.

For example, if there are four occurrences of 'My Text' in the content and you want to verify the text that is between second and third occurrence, you should specify 2 and 3 in **From String** and **To String** respectively.

- To do a case-sensitive match, select the **Case sensitive** check box.
- To ignore carriage return/ line feed in the response, select the **Ignore CL/LF when matching** check box.
- Save the test and run it.

Results

The Service Verification Point Report shows the number of Text Verification Points that passed or failed.

Adding **properties verification points** to a test response

You can add verification points for the properties in a service test so that these properties in the test response are verified and validated when you play back the test.

Before you begin

You must have recorded or created a service test using the test editor.

About this task

When you add verification points, results from a service response are compared with the expected data specified as the verification point test element. You can add the verification point for the properties to an existing test response when the test is manually created or recorded. After you add the verification point for the properties to a test response, you can verify the selected response properties during the test run.

- Identify the service test from **Test Navigator** and double-click the service test to open it in the test editor.
- Select a service response for a service request from the service test.
- Right-click the service response, click **Add > Properties Verification Point**.

Result


The **Properties Verification Point** is added based on the existing properties of the service response.



Note: You can add multiple verification points for the properties, if required.

- Perform any of the following on the verification points for the properties in the **Properties Verification Point Details** pane.

To Do...	Do This...
To add a new property and its value	Click Add .
To edit the value of an existing property	Click Edit .

To Do...	Do This...
To remove the property that you do not want to verify during the test run	Click Remove .  Note: You can remove multiple properties in a group at the same time.

5. Select or clear the **Apply And Operator** check box based on the requirement as follows:
 - To verify all the listed properties, select the **Apply And Operator** check box.
 - To verify one of the listed properties, clear the **Apply And Operator** check box.
6. Optionally, you can substitute the value of one or more properties in the verification point by using a test variable, data set, custom java code, or built-in variables.
7. Verify all the verification points for the properties that you entered, and then click **Save**.

Results

The verification points that you added for the response properties are added to the service test.

What to do next

You can run the test and after the test run, you can view and analyze the `properties verification point` details from the following page and reports:

- **Verdict List** pane in the **Test Log** page. Click any of the verification point from the list and view the details.
- **Response Properties Verification Points** tab in the **Service Verification Point Report**. Click the **Response Properties Verification Points** tab and view the verification point details.
- **Verification points verdicts** pane from the **Functional Test** report page. Click any of the verdict status to verify the expected value and actual value of the verification point for the properties.

Related reference

[Web Service Verification Points report on page 842](#)

Adding XSD verification points

XSD verification points enable you to check that the XML content of a service response comply with the rules defined in an XML Schema Definition (XSD) file.

Before you begin

When you add verification points, the results from a service response are compared with the expected data that are specified in the verification point test element. XSD verification points return a Pass status when the XML contents of the response are compliant with the associated XSD or a Web Service Description Language (WSDL) file that contains XSD information.

If you add multiple XSD files to the verification, then the XML content of the response must comply with all of the XSD files.

You can use the test editor to create or edit verification points.

To add an XSD verification point to a test:

1. Open the test editor and select a service response element.
2. Click **Add** and select **XSD Verification Point**.
3. In the **Test Element Details** area of the test editor, type a name for the verification point.
4. Click **Add XSD** to add a an XSD file to the validation list or **Add WSDL** to add a WSDL that contains XSD information.
Click **Open** to display the XSD or WSDL contents.

What to do next

You can enable or disable each verification point by right-clicking the verification point in the test editor and clicking **Enable** or **Disable**.

Working with Server Name Indication (SNI) recordings

If you have recorded against a server that supports Server Name Indication (SNI), an extension of the TLS protocol, the recording session file displays `true` for the **SNI Extension** field. There might be a need for you to access both SNI and non-SNI applications from the same server. To run the same test without using the SNI extension, you can manually change the value to `false`.

About this task

The screenshot shows the HTTP Proxy Recorder interface. On the left, the 'Packets' pane displays a list of network events with columns for Packet, Start Time, and End Time. The selected packet is 'Proxy opens secured connection 2 based on 1'. On the right, the 'Packet Details' pane shows the configuration for this connection, including Recorder, Connection Id, Parent Connection Id, Connection Type Id, Server Host, Server Port, Client Host, Client Port, Cipher Suites, SSL Protocols, and HTTP Version Protocol. The 'SNI Extension' field is highlighted with a red box and contains the value 'true'.

Packet	Start Time	End Time
Proxy connection type 0	00:00.069	00:00.069
Local addresses	00:00.308	00:00.308
Proxy starts	00:00.310	00:00.310
Proxy initiates connection 1	00:06.271	00:06.271
Proxy opens connection 1	00:06.271	00:06.611
Proxy opens secured connection 2 based on 1	00:06.758	00:07.357
Proxy close connection 2	00:07.551	00:07.551
Proxy close connection 1	00:07.551	00:07.551
Proxy initiates connection 3	00:10.195	00:10.195
Proxy opens connection 3	00:10.195	00:10.259
Proxy opens secured connection 4 based on 3	00:10.268	00:10.408
Proxy close connection 4	00:10.703	00:10.703
Proxy close connection 3	00:10.703	00:10.703
Proxy initiates connection 5	00:10.788	00:10.788
Proxy opens connection 5	00:10.788	00:11.066

Proxy Open Secured	
Recorder:	HTTP Proxy Recorder
Connection Id:	2
Parent Connection Id:	1
Connection Type Id:	0
Server Host:	tiles.services.mozilla.com
Server Port:	443
Client Host:	127.0.0.1
Client Port:	54790
Cipher Suites:	SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256
SSL Protocols:	TLSv1.2
HTTP Version Protocol:	http/1.1
SNI Extension:	true

The **Server Access Configurations** resource of the test script also have SSL entries. Each SSL entry displays which TLS version and Cipher value was used. To edit multiple SSL entries, select them and in the Detail area, right-click the entries and click **Edit multiple SSLs**.

Adding elements to a service test

To improve the reliability of the test script, add the different elements to the service test.

Adding a service request

You can use service request elements in tests to send a request to the service.

About this task

Complex service requests or verification points might have empty XML elements that are not needed in a test script. When playing back the test, you can skip such empty XML elements. In **Window > Preferences > Test > Test editor > Service test** ensure that the **Display the 'Skip if empty' column in XML tree viewer** check box is selected. This option displays a **Skip if empty** column in the tree view of the request. You can then choose the XML elements to skip.

1. Open the test in the test editor, and select the first element in the test.
2. Click **Add** and select a service request.
3. If you selected WSDL service request, select one or several WSDL files in your workspace for the web service that you want to test and click **Next**.
If necessary, you can import a WSDL file into the workspace with the **Add** button.
4. Select either **HTTP**, **JMS**, or **WebSphere MQ** depending on the transport protocol used by the web service, and provide the correct transport protocol configuration to perform the call.
You can create a **New** transport configuration or reuse an existing one.
5. Click **Finish**.

Result

This creates the web service request in the test editor.

6. On the **Message** page of the request, select the **Form**, **Tree**, or **Source** views to edit the service request contents.
7. If any resource files are to be attached to the request, select the **Attachment** tab. Use **Add**, **Remove**, or **Edit** to specify the resources that are to be attached to the request.
8. If the service uses encryption, signature or other security protocols, select the **Security for Request** and **Security for Response** pages to configure the security for this particular service request or to open the WSDL security editor.

What to do next

After creating elements, you can use the test editor to edit service requests. You can create a service response element to test the performance and behavior of the service. You can also replace some content values with dataset variables or a references.

Updating a service response from the service

While you are developing a service test, you can send a request from the test editor to record or update the response element.

Before you begin

Service response elements are children of service request elements. Service tests use response elements to measure the response time between a call and the corresponding response. Response elements can also contain verification points.

You can click **Update Response** in the request element to complete one of the following actions:

- Record a response from the service: This method sends the request and records the actual response from the service. For services that use the IBM® WebSphere® MQ or JMS transport protocols, multiple responses can be recorded.
- Update the current response content: If a response exists, its contents are replaced. If multiple responses are received, the number and order of the responses are updated.



Important: After updating the response content, data correlation or verification points that referred to replaced content might no longer work.

You can use the test editor to create or edit response elements in a service test. There are three methods of adding a service response:

- Generate a response from Web Services Description Language (WSDL): If the service uses WSDL, then the response is created with the content structure that the WSDL specifies.
- Add a text response: In this response type, you specify free formatted content for the response.
- Record a response from the service: This method sends the request and records the actual response from the service.

WebSphere® MQ and JMS requests can contain multiple response elements.

To add a response element to a service test:

1. Open the test in the test editor, and select a service request element.
2. On the **Test Element Details** page, click **Update Response**.
Alternatively, right-click the service request element, and click **Add > Response from Request**.

Result

This action performs the service request. If the request is valid, the **Update Response** window opens and displays the response data.

3. In the **Return Preview** window, review the content of the response to ensure that it is correct.
For the WebSphere® MQ and JMS protocols, if multiple responses are received, then click the arrows to view each response.
 - a. Click the **Message** tab to view the contents of the response in the **Form, Tree** or **Source** view.
 - b. Click the **Attachment** tab to view any resource files that were attached to the response.
 - c. Click the **Response Properties** tab to view the properties of the response.
4. To use the received response in the test, click **Update Test**.
This creates the response elements as a child of the request element or updates the existing response elements with the new data.

What to do next

After creating or updating response elements, you can create verification points on the response contents to test the behavior of the service.

Related information

[Manually adding a response element on page 350](#)

[Verifying application behavior on page 339](#)

Manually adding a response element

You can add service response elements to specify the received content of a service request. You can use the test editor to create or edit response elements in an existing service test.

Before you begin

Service response elements are children of service request elements. Service tests use response elements to measure the response time between a call and the corresponding response. Response elements can also contain verification points. IBM® WebSphere® MQ and JMS requests can contain multiple response elements.

Depending on the type of request, you can manually create several types of response elements:

- Response from Web Services Description Language (WSDL): For web services, this response type uses the WSDL file to create the specified XML structure of the response.
- XML response: This response type creates an empty response element in which you must manually create the expected XML structure. You can use an XML Schema Definition (XSD) document from the XSD catalog to assist you.
- Text response: This response type creates an empty response element, which can contain freely formatted text.

Alternatively, you can automatically create and update response content by recording the actual response content that the service returns. See [Updating a service response from the service on page 348](#) for more information.

To add a response element to a service test:

1. Open the test in the test editor, and select a service request element.
2. Create one of these elements:

Choose from:

- For web service requests, click **Add > Response from WSDL**.
- If the expected response contains XML content, click **Add > XML Response**.
- If the expected response contains plain text, click **Add > Text Response**.

Result

This action creates the corresponding response element in the test. If the request uses the WebSphere® MQ or JMS format, then you can create multiple responses.

3. Edit the message content of the response element to reflect to actual content that the service returns.
 - a. Click the **Message** tab to view the contents of the response in the **Form, Tree** or **Source** view.
 - b. Click the **Attachment** tab to view any resource files that were attached to the response.
 - c. Click the **Response Properties** tab to view the properties of the response.

What to do next

After creating a message return, you can create verification points on the contents to test the behavior of the service.

Related information

[Updating a service response from the service on page 348](#)

[Verifying application behavior on page 339](#)

Managing JMS/MQ connections in a service test

When you run a service test that includes JMS or MQ protocol, the socket connections are created and closed in the background. When you include multiple tests in a compound test or a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule., multiple connections are created and closed. Starting from 9.2, when you run service tests in a schedule, you can select a pooling strategy for these JMS/MQ connections so that when the connections are created, they do not close and are reused subsequently for the other JMS/MQ calls, if required.

About this task

You can set the scope of JMS/MQ connections to a test, compound test, or schedule. When you set the scope to test, existing behavior comes into play wherein duplicate connections could be created and closed. When you set the scope to a compound test or a schedule, the connections are reused for JMS/MQ calls within a compound test or a schedule.

1. In the Test Navigator, browse to the schedule and double-click it. The schedule is displayed.
2. Select a schedule. In the VU Schedule Details area, click the **Advanced** tab and under **Protocol-specific options**, click **Edit Options**.
3. In **JMS/MQ connections scope**, select the scope of the connections.
4. Use the following options to control the underlying MQ Connection Manager to create only the specified number of connections. These options are generally used by the MQ expert:
 - a. In **Maximum quantity of connections**, specify a number to ensure that a certain number of connections are open at a time only for MQ Java.
 - b. In **Maximum quantity of unused connections**, specify the maximum number of connections that should be unused among the open connections.
 - c. In **Connection timeout (ms)**, specify a time after which there is no attempt to establish the connection.

Results

When you run a schedule, the JMS/MQ connections are reused.

Editing WSDL security profiles

To ensure that your service test uses the correct security protocols to access a SOAP-based service, you must specify a security profile for the (Web Service Description Language) WSDL file. After a security profile is set up, it can be reused in multiple web service calls.

WSDL security editor overview

With the WSDL security editor you can create the SOAP algorithm stacks that are associated with a web service operation. Algorithm stacks contain digital certificate information and the security algorithms that are applied to messages to perform secure communication with a web service.

After you create an algorithm stack, you associate it with an operation that is specified in the Web Services Description Language (WSDL) file of the web service. Algorithm stacks remain available in the workspace and you can reuse them with other WSDL files. You can also edit a test to make the same web service call several times with different security configurations.

You use the **WSDL security editor** to create and edit security configurations. The WSDL security editor contains two pages that correspond to the steps of setting up a security configuration:

- Describing a security stack
- Associating a security stack with each WSDL operation

Algorithm stacks

Algorithm stacks contain one or several algorithm blocks that are arranged in a sequence of steps. Each algorithm block modifies or transforms the message content. Algorithm blocks can add timestamps to, add tokens to, encrypt, or sign messages.

Use the **Algorithm Stacks** page of the WSDL security editor to create stacks for service requests and responses. When a message is sent or received, each algorithm block in the stack is executed in the specified order. For example, you can define a request stack for outgoing requests that adds a timestamp, signs, and then encrypts the message content, and you can define a response stack that decrypts incoming responses. You can create as many algorithms as your application requires.

You can edit algorithm blocks and move them up and down in the stack. Encryption and signature blocks can use keystores for digital certificates. Some algorithm blocks display messages that help you enter correct information. If the contents of the algorithm block are invalid, an error icon is displayed.

Raw transaction data view

When a stack is associated with a service request or response, viewing the results of each transformation step that is applied to the XML message content can be useful. You can use the **Raw Transaction Data** view to look at the message content before and after each algorithm in the stack.

Digital certificate keystores

You can add digital certificate keystores to a security stack to use with encryption or signature algorithms. Keystores must be declared with their associated passwords before the algorithms that use them. Digital certificates are contained in Java™ keystore files (KS, JKS, JCEKS, PKCS12, and PEM) that must be located in your workspace.

Associating stacks with WSDL operations

Use the **Algorithms by WSDL operations** page of the WSDL security editor to associate a security algorithm stack with each web service call and message return in the WSDL file.

Creating security profiles for WSDL files

You can create SOAP security profiles for the web service calls or message returns that require message encryption, signature or other advanced security algorithms.

Before you begin

You must have a Web Services Description Language (WSDL) file in your workspace.

If the security profile uses digital certificates for encrypting or signing requests or responses, you must have the corresponding keystore files (KS, JKS, JKECS, PKCS12, or PEM) in your workspace.

About this task

If the WSDL is simple and you want to check its security, in the **Request Stack** tab of the test editor, click **Override Stack > Tools > Analyze Security from Pasted Content**. Paste the SOAP XML message and click **Next**. The next page shows the different security algorithms used in the XML. Click **Finish** to add the security algorithms to the editor.



Note: When you add a secured SOAP XML message in **Message > Source** tab of the test editor, certain security related warnings are displayed in the **Error Message** view. If you are aware of the secured SOAP XML message and do not want to view the warnings, click **Window > Preferences > Generic Service Client > Message Edition** and select the **Analyze pasted SOAP content** check box.

If the WSDL uses WS-Policy, you must configure security as follows:

1. In the test navigator or project explorer, right-click the WSDL file and select **Edit WSDL Security**.

Result

The WSDL security editor is displayed.

2. Click the **Security Algorithms** tab.

Security profiles are described by adding elements to a stack. When a service request is sent or a response is received, each element in the stack is applied to the message in a specified order. If necessary, create one security profile for outgoing requests and one for incoming responses.

3. In the **Security Algorithms** area, click **Add** to create a new algorithm stack, and click **Rename** to change the default name.
4. In the **Algorithm Stack Details** area, click **Add** to add a new algorithm element to the stack.

You can add time stamps, username tokens, encryption, or signatures.

5. Edit each element in the stack according to the requirements of the web service.

You can apply encryption and signature stack elements to portions of the web service call or message return document by specifying an Xpath query in **User Xpath part selection**. For example, you can encrypt one XML element with one encryption stack element, and another element with another stack element. You can use the **Web Service Protocol Data** view to help identify the correct Xpath query for this option.

You can check whether the security stack is valid by clicking **Tools > Validate Selected Algorithm**.

6. When all the stack elements are complete, ensure that the execution order is correct.

If necessary, use the **Up** and **Down** buttons to change the order of elements in the stack.

7. Repeat steps 4 through 7 to create as many algorithms as are required for security profile.

8. Click the **Algorithms by WSDL Operations** tab.

This page enables you to associate a security profile with each request or response operation in the WSDL.

9. In the **WSDL Contents** column, select a service request or response.

10. In the **Algorithm Stack** column, select a security profile from the list.

If necessary, click **<<** to open the stack on the Security Algorithms page.

Results

After saving the security profile, the **Web Service Protocol Data** view displays the effect of the security profile on the XML data of the web service.

Related reference

[WSDL security editor reference on page 1233](#)

Related information

[Using a security policy on page 354](#)

[Adding WS-Addressing to a security configuration on page 366](#)

[Implementing a custom security algorithm on page 363](#)

Using a security policy

The WS-Policy specification enables web services to use XML to publish their security policies either as part of the Web Services Description Language (WSDL) file (compliant with the WS-PolicyAttachment specification) or as a separate XML document. With the WSDL Security Editor, you can create a security profile that uses a policy that complies with the WS-Policy specification.

Before you begin

Before creating a security configuration, you must have a WSDL file in your workspace.

If the security policy uses digital certificates for encrypting or signing requests or responses, you must have the corresponding keystore files (KS, JKS, JKECS, PKCS12, or PEM) in your workspace.

When you import a WSDL that contains a policy (with WS-PolicyAttachment), a security profile is automatically generated for each operation in the WSDL security editor.

1. In the test navigator or project explorer, right-click the WSDL file, and select **Configure WSDL Security**.

Result

This opens the WSDL security editor.

2. Click the **Security Algorithms** tab.

Security profiles are described by adding elements to a stack. When a service request is sent or a response is received, each element in the stack is applied to the message in the specified order.

3. In the **Security Algorithms** area, click **Add** to create a profile, and click **Rename** to change the default name.
4. In the **Algorithm Stack Details** area, click **Add > WS-Policy** to add the WS-Policy element to the stack. You can also add time stamps, user-name tokens, encryption, or signatures.
5. If the policy is included in the WSDL file, click **Use policy included in WSDL (WS-PolicyAttachment)**, and edit the WS-Policy settings as required:

Policy

If you are not using the WS-PolicyAttachment specification, specify the XML policy file. Click **Browse** to add a policy file from the workspace or to import a policy file.

Signature configuration

Select this option to specify a keystore for any signature that is specified in the policy. Click **Edit Security** to add a keystore from the workspace or to import a keystore.

Encryption configuration

Select this option to specify a keystore for any encryption that is specified in the policy. Click **Edit Security** to add a keystore from the workspace or to import a keystore.

Decryption configuration

Select this option to specify a keystore for any decryption that is specified in the policy. Click **Edit Security** to add a keystore from the workspace or to import a keystore.

Retrieve token from security token server (WS-Trust and WS-SecureConversation)

Select this option, and click **Configure** to specify a Security Token Server (STS) to use with the policy.

Additional properties

Use this table to specify settings for the advanced properties or specific implementations of the WS-Security specification. Click **Add** to add a property name and to set a value.

6. Check that the security profile is valid by clicking **Tools > Validate Selected Algorithm**.
7. Click the **Algorithms by WSDL Operations** tab.

On this page, you can associate a security profile with each request or response operation in the WSDL.

8. In the **WSDL Contents** column, select a web service request or response operation.
9. In the **Algorithm Stack** column, select a security profile from the list.

If necessary, click << to open the stack on the **Security Algorithms** page.

What to do next

After saving the security profile, the **Web Service Protocol Data** view displays the result of the security profile on the XML data of the web service.

Related information

[Creating security profiles for WSDL files on page 353](#)

[Adding WS-Addressing to a security configuration on page 366](#)

[Implementing a custom security algorithm on page 363](#)

Adding security stacks

To provide better WSDL security, you can make use of many security algorithms in the service test.

About this task

1. From the Test Navigator view or from the Request Library section of Generic Service Client, right-click the WSDL file and select **Edit WSDL Security**.
2. In the **Security Algorithms** area of **Algorithm Stacks** tab, click **Add** to create a profile.
3. In the **Stack Contents** area, click **Add** and add any of the following security algorithms:

Custom Security Algorithm

If you want to use a Java™ class as a custom security algorithm, then use this stack element to apply the custom algorithm to the service.

Java™ Project

If you have not implemented a custom Java™ class, select **Java Project**, type a name for the new project, and click **Generate** to create a new Java™ class with the default structure for custom security implementations.



Note: If you are using IBM® Security AppScan®, this field is not available.

Implementation class

Specify the name of the class that implements the custom security algorithm.

Click **Browse Class** to select an existing Java™ class from the workspace.

Properties

Use this table to send any specific properties and associated values to the custom security algorithm.

WS-Addressing Algorithm

Use this block if your service uses either WS-Addressing 2004/08 or the WS-Addressing 1.0 Core standard.

Namespace

Specify the namespace for either WS-Addressing 2004/08 or WS-Addressing 1.0 Core.

Action if request uses WS-Addressing

Select the action to complete if WS-Addressing is already in the request.

Replace anonymous address in Reply-to with:

Select this option to generate the specified address in the Reply-to header instead of an anonymous address.

Remove WS-Addressing from response

Select this option to strip any WS-Addressing headers from the response.

Encrypted Key

This block defines an encrypted key that can be used in an XML signature or XML encryption block. The encrypted key block must be before a block that uses the encrypted key.

Actor / Role name

Specify the name of the recipient of the algorithm header element, if required.

Must understand

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

Key name

Specify the name of the encrypted key.

Identifier type

Select the type of key identifier to be used for the key. The following key identifiers are available, as defined in the the Web Service Security (WSS) specification X509 profile and OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- THUMBPRINT_IDENTIFIER
- SKI_KEY_IDENTIFIER

Key size

Specify the size of the key in bits.

Key encoding algorithm name

Specify the algorithm to be used for encoding the key.

Keystore

Select a keystore or click **Edit Security** to define a new keystore or to manage the existing keystores.

Name

Select a key contained in the specified keystore.

Password

Type the password for the selected key name.

XML Signature

The XML signature security algorithm specifies how the XML document is signed. For details on security algorithms, refer to the web service security specification.

Actor / Role name

Specify the name of the recipient of the algorithm header element, if required.

Must understand

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

Security token

Select the type of key identifier to be used for the signature. The following key identifiers are available, as defined in the the Web Service Security (WSS) specification X509 profile and OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- SKI_KEY_IDENTIFIER
- KEY_VALUE
- USER_NAME_TOKEN
- CUSTOM_SYMM_SIGNATURE

In addition, the following identifiers are available when the signature is based on a UsernameToken profile:

- USER_NAME_TOKEN
- CUSTOM_SYMM_SIGNATURE

User XPath part selection

Specify an XPath query that describes parts of the XML document that can be the subjects of the algorithm. By default, the body is the subject. Click the **XPath Helper** button to build the Xpath expression.

Key

Select the key used for the encryption. The details of each key vary.

- **x509 key**: This key specifies the name and password of the x509 key and the keystore where it is located.
- **User name token key**: This specifies a user name and password for the signature.
- **Encrypted key**: This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

Signature algorithm name

Specify the signature method algorithm as described in the XML Signature Syntax and Processing specification.

Canonicalization

Specify the canonicalization method to be used as described in the XML Signature Syntax and Processing specification.

Digest algorithm method

Specify which digest method to be used based on the algorithm method used on the server side.

Inclusive namespaces

Specify whether the canonicalization is exclusive as described in the Exclusive XML Canonicalization specification.

XML Encryption

The XML encryption security algorithm specifies how the XML document is encrypted. For details on security algorithms, refer to the web service security specification.

Actor / Role name

Specify the name of the recipient of the algorithm header element, if required.

Must understand

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

Identifier type

Select the type of key identifier to be used for the encryption. The following key identifiers are available, as defined in the Web Services Security (WSS) specification X509 profile and the OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER

- SKI_KEY_IDENTIFIER
- EMBEDDED_KEYNAME
- THUMBPRINT_IDENTIFIER
- ENCRYPTED_KEY_SHA1_IDENTIFIER

User XPath part selection

This enables you to specify an XPath query that describes parts of the XML document that can be subjects of the algorithm. By default, the body is the subject.

Key

Select the key used for the encryption. The details of each key vary.

- **x509 key**: This specifies the name and password of the x509 key and the keystore where it is located.
- **Raw key**: This specifies the name and the byte value of your SecretKey in hexadecimal.
- **Encrypted key**: This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

Encoding Algorithm Name

Specify the encryption method to be used as defined in the XML Encryption Syntax and Processing specification.

Key Encoding Algorithm

Specify the standard algorithm for encoding the key as defined in the XML Encryption Syntax and Processing specification.

User name token

The user name token security algorithm adds a user name token to the XML document in the message. For details on security algorithms, refer to the web service security specification.

Actor / Role name

Specify the name of the recipient of the algorithm header element, if required.

Must understand

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

Name

Type the name of the user.

Password

Type the password of the user.

Password type

Specify the password type for the security algorithm as defined in the Web Services Security UsernameToken profile.

Use nonce

Select this check box to add the Nonce element to the User Name Token XML code. In most cases, the Nonce ID is required.

Use created

Select this check box to add current timestamp to the Created XML element in the User Name Token XML.

Time Stamp

The time stamp security algorithm adds time stamp information to the XML document in the response. For details on security algorithms, refer to the web service security specification.

Actor / Role name

Specify the name of the recipient of the algorithm header element, if required.

Must understand

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

Expiration delay

Specify the delay after which the time stamp expires.

Millisecond precision

Select this option to produce a time stamp that uses millisecond precision instead of the default (1/100th second).

SAML Assertion Block

To use the self-signed SAML assertion security algorithm, add the SAML Assertion stack to the request or WSDL files.

User XPath part selection

Specify an XPath query that describes parts of the XML document that can be the subjects of the algorithm. By default, the body is the subject. Click the **XPath Helper** button to build the Xpath expression.

Key

Select the key used for the encryption. The details of each key vary.

- **x509 key:** This key specifies the name and password of the x509 key and the keystore where it is located.
- **User name token key:** This specifies a user name and password for the signature.
- **Encrypted key:** This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

Signature algorithm name

Specify the signature method algorithm as described in the XML Signature Syntax and Processing specification.

Canonicalization

Specify the canonicalization method to be used as described in the XML Signature Syntax and Processing specification.

Digest algorithm method

Specify which digest method to be used based on the algorithm method used on the server side.

Inclusive namespaces

Specify whether the canonicalization is exclusive as described in the Exclusive XML Canonicalization specification.

Signed Assertion

Select this check box to self-sign the SAML Assertion.

Issuer

Specify the description of the issuer of the SAML Assertion or protocol message.

Subject

Specify the principal that is the subject of all of the statements in the assertion. It might contain an identifier or a series of one or more subject confirmations.

Subject Qualifier

Specify the Name Qualifier of the Subject

Subject Format

Specify the format used for the Subject.

Subject Locality DNS

Specify the DNS domain name for the system from which the assertion subject was authenticated.

Subject Locality IP

Specify the IP address for the system from which the assertion subject was authenticated.

Statement Type

Specify the authentication method to use for the assertion.

Authentication: The assertion subject was authenticated

Attribute: The specified subject is associated with the supplied attributes.

Authorization decision: Permission to allow a subject to access the specified resource.

Requested Resource

When **Authorization decision** option is used, specify the resource for which you need access.

Action

Specify what action to take to access the resource.

Confirmation number

Confirmation methods define the mechanism by which an entity provides evidence (proof) of the relationship between the subject and the claims of the SAML assertions.

Sender vouches: Select this option when a server needs to share the client identity with SOAP messages on behalf of the client. This method is similar to identity assertion, but it has the added flexibility of using SAML assertions to share not only the client identity, but also client attributes.

Holder of key: Select this option when the proof of the relationship between the subject and claims is established by signing part of the SOAP message with the key specified in the SAML assertion. Because there is key material associated with a holder-of-key token, this token can be used to provide a message-level protection (signing and encryption) of the SOAP message.

Bearers: Select this option when the proof of the relationship between the subject and claims is implicit. No specific steps are taken to establish the relationship. Because there is no key material associated with a bearer token, protection of the SOAP message, if required, must be performed using a transport-level mechanism or another security token, such as an X.509 or Kerberos token, for message level protection.

Version

Specify the SAML version used.

4. **Optional:** To verify simple SAML code, use the **Analyze Security from Pasted Content** option. For more information about that option, see [Creating security for WSDL profiles on page 353](#).

Implementing a custom security algorithm

You can define your own security algorithms for SOAP security profiles by implementing custom security Java™ interfaces that can be used in the WSDL security editor. With custom security algorithms, you can implement proprietary security algorithms that transform the XML before sending and after receiving message content.

Before you begin

The custom security interface and the JAR file that contains it are provided with the product in the `customsecuritydefinition` folder of the `com.ibm.rational.ttt.common.models.core` plugin. You need these interfaces to create your own algorithms. If you are using HCL OneTest™ Performance, see [Extending test execution with custom code on page](#) for more information about extending test capabilities with Java™ code.

1. In the test navigator or project explorer, create a new Java™ class in your web service test project folder.
2. Implement a security algorithm in Java™ using the following interface:

```
/**
 * *****
 * IBM Confidential
 *
 * (c) Copyright IBM Corporation. 2008. All Rights Reserved.
 *
 * The source code for this program is not published or otherwise
 * divested of its trade secrets, irrespective of what has been
 * deposited with the U.S. Copyright Office.
 * *****
 */

package com.ibm.rational.test.lt.models.wscore.datamodel.security.xmlsec;

import java.util.Properties;
import org.w3c.dom.Document;

public interface ICustomSecurityAlgorithm {

    /**
     * The following methods can be used in both case:
     * Execution in the workbench and execution of the test.
     */

    /**
     * Called to process de Document that is sent over a transport.
     * @param subject
     */
    void process(Document subject);

    /**
     * Called to un process a document that is received from a server.
     * @param subject
     */
    void unProcess(Document subject);

    /**
     * Properties defined in the UI of the CustomSecurityAlgorithm.
     * @param map
     */
    void setProperties(Properties map);

    /**
```



```

* The following methods can only be used in terms of cast to test service interface,
* or in terms of access to the previous XML information, when the jar containing
* the custom security algorithm is deployed in the performance test project. In
* this case you cannot use the algorithm directly from the workbench.
*/

/**
* This object corresponds to the ITestExecutionService object.
* This applies only to an algorithm that must link to the execution of the test.
* If you plan to use this object you will need to deploy the jar containing the
* implementation into your performance test project and not directly into the JRE.
*
* In case of a need of the previous xml document received from the execution you can
* obtain the value using:
* IDataArea area =
((ITestExecutionService)executionObject).findDataArea(IDataArea.VIRTUALUSER);
*String previousXML = (String) area.get("PREVIOUS_XML"); //$NON-NLS-1$
*
*/
void setExecutionContext(Object executionObject);

```

The `process` method modifies the XML before it is sent to the server.

The `unprocess` method modifies the XML after it is received from the server.

The `setPropertyies` method retrieves any properties that are defined in the security editor for this custom security interface.

The `setExecutionContext` method is called during test with the object `ITestExecutionServices` that corresponds to the message using this custom security interface.

3. The custom security interface can be used either in the **WSDL security editor** for web services or in XML call elements in the **Local XML security** tab.

Choose from:

- If you are testing a WSDL-based web service, right-click the WSDL file in the test navigator or project explorer to open the WSDL security editor, select the **Security Algorithms** page; then, under **Details of selected security algorithm stack**, click **Add > Custom Security Algorithm**.
 - If you are testing an XML call, open the XML call element in the test editor, select the **Local XML Security** tab, and then, click **Add > Custom Security Algorithm**
4. In custom security, click **Browse Class** to select the class name of the custom security algorithm, for example: `ICustomSecurityAlgorithm`.
 5. Type an **Algorithm name** for the custom security algorithm.
 6. In the properties list, use **Add**, **Remove**, or **Edit** to specify any properties that are used by the `setPropertyies` method in your custom security algorithm.

What to do next

After saving the security configuration or the call element, the **Web Service Protocol Data** view displays the effect of the security algorithm on the XML data of the web service.

Related reference

[WSDL security editor reference on page 1233](#)


Adding WS-Addressing to a security configuration


The WS-Addressing specification provides transport-neutral mechanisms that enable SOAP-based web services to communicate addressing information. You can use WSDL security algorithms to add WS-Addressing to your service tests.

Before you begin

Before adding WS-Addressing to a security configuration, you must have a service test with requests and responses that are related to a valid WSDL.

To add WS-Addressing to a WSDL security algorithm:

1. Open the test, select a service request, and in the **Raw Transaction Data** view, select **Enable the display of the XML document after the security processing**.
2. On the **Request Stack** page, click **Edit WSDL Security** .

 **Tip:** If you need to edit separate security or processing algorithms for incoming responses, click **Show Response Stack** to add a **Response Stack** page to the editor.

Result

The **WSDL security editor** opens.

3. Select the **Algorithm Stacks** page of the WSDL security editor, and in the **Security Algorithm** list, select or create a security algorithm.
4. In the **Stack Contents** list, click **Add > WS-Addressing** and specify the settings that are implemented by the service.

WS-Addressing Algorithm

Use this block if your service uses either WS-Addressing 2004/08 or the WS-Addressing 1.0 Core standard.

Namespace

Specify the namespace for either WS-Addressing 2004/08 or WS-Addressing 1.0 Core.

Action if request uses WS-Addressing

Select the action to complete if WS-Addressing is already in the request.

Replace anonymous address in Reply-to with:

Select this option to generate the specified address in the Reply-to header instead of an anonymous address.

Remove WS-Addressing from response

Select this option to strip any WS-Addressing headers from the response.

5. Save the WSDL security algorithm, and select the test editor.

Result

The WS-Addressing namespace and header XML content is displayed in the **Raw Transaction Data** view.

Related reference

[WSDL security editor reference on page 1233](#)

Related information

[Creating security profiles for WSDL files on page 353](#)

[Implementing a custom security algorithm on page 363](#)

Testing asynchronous services

Use the asynchronous callback services for inter-object communications in a service test.

Asynchronous service testing overview

Asynchronous services use a callback interaction pattern for inter-object communications. Asynchronous services can be used, for example, in publish-subscribe systems that are provided by message-oriented middleware vendors or in system and device management domains.

WS-Notification services

Asynchronous services are standardized in the *WS-Notification* specifications:

- *WS-BaseNotification* defines the web services interfaces for *NotificationProducers* and *NotificationConsumers*. This specification includes standard message exchanges that are implemented by service providers that want to act in these roles, along with the associated operational requirements.
- *WS-BrokeredNotification* defines the web services interface for a *NotificationBroker*. A *NotificationBroker* is an intermediary which, among other things, enables entities that are not service providers themselves to publish messages. It includes standard message exchanges that are implemented by *NotificationBroker* service providers, along with the associated operational requirements of service providers and requestors that participate in brokered notifications.
- *WS-Topics* defines a mechanism to organize and categorize items of interest for subscription known as *topics*. These are used in conjunction with the notification mechanisms defined in *WS-BaseNotification* and *WS-BrokeredNotification*.

You can test web services and XML services that implement the WS-Notification specification by creating an asynchronous request inside a test. The asynchronous request contains the interfaces for the corresponding WS-Notification specification, along with a callback structure.

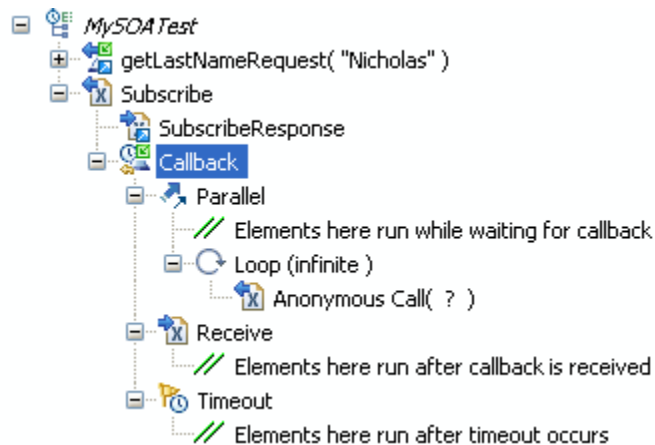
Proprietary asynchronous services

You can test proprietary asynchronous services that do not implement WS-Notification specifications. To test these services, you manually create a service request that contains the interfaces for the service, and then, you can add the asynchronous callback structure to the request.

The XML data of the asynchronous request must contain an endpoint that specifies the URL of the callback receiver. During the test, this endpoint is used to redirect the callback to the tester instead of the real receiver.

Callback structure

To test asynchronous services, you must create an asynchronous request structure in your test as shown in the following diagram:



A web service request or a plain XML request provides the subscription action and contains a callback element, which describes the behavior of the test in three states:

- *Parallel* contains test elements that are run after the subscription request and while waiting for the notification response.
- *Receive* contains test elements that are run when the notification response has been received from the service.
- *Timeout* contains test elements that are run if the notification response is not received after a delay that is specified in the callback element.

When everything contained in the parallel, receive, and timeout elements have finished running, the run continues with the next element in the test after the asynchronous request.

The method for generating the asynchronous callback structure in the test depends on whether the asynchronous service uses the WS-Notification specification:

- *WS-Notification services*: Create the asynchronous request in the test.
- *Proprietary services*: Manually create a web service request or XML request in the test, and then add the asynchronous callback structure to the request.

Creating an asynchronous request structure

You can create an asynchronous request based on the *WS-Notification* specification, which contains an callback structure.

1. In the test editor, select the test, and click **Add**, and then click **Specification-based Structure**.

Result

The **New Web Service Test** wizard opens.

2. On the **Web Services Specification Selection** page, Select **WS-Notification**, and click **Next**.
3. On the **WS-Notification Details** page, if the service has a Web Services Description Language (WSDL) file, click **Add** to associate it with the call.
4. Specify the **Subscription identifier**.
You can select default identifiers for Websphere Application Server or Apache Muse; or if your service does not use a standard identifier, you can select **Custom**, and type the **Name** and **Namespace** of the identifier.
5. In the **Topic** area, replace the default **Name** and **Namespace** values with those of topic of your service.
6. Specify the **Subscription duration**.
Because this is a test environment, the subscription expires after the specified delay to save server resources.
7. If this is a WS-BrokeredNotification service, which implements a notify call when the subscription is received, you can select **Add notify call**, and type the message to be sent.
8. Click **Next**.
9. On the **Configure Protocol** page, select a **Protocol configuration**, and specify the options of the configuration.
Select **Generate SOAP 1.2 envelope** if you are testing a SOAP 1.2 web service.
10. Click **Finish**.

Result

This action generates in the test editor a web service call or an XML request with a callback structure that contains a parallel, a receive, and a timeout element.

What to do next

In the callback structure, add test elements to the parallel, receive, and timeout elements to specify the behavior of the test:

- *Parallel* contains test elements that are run after the asynchronous call has been sent.
- *Receive* specifies the message return of the callback and contains test elements that are run after the callback is received.
- *Timeout* contains test elements that are run if the callback is not received after a specified delay.

Adding an asynchronous callback to a service request

To test a proprietary asynchronous service that does not implement the *WS-Notification* specification, you can add an asynchronous callback to a service request or XML request.

Before you begin

Manually create a web service call or XML call that invokes the asynchronous service. The call must contain an endpoint that specifies the URL of the callback receiver. This endpoint is used to redirect the callback to the tester.

If the service implements the WS-Notification specification, create the asynchronous call structure with the **Create New WS-Notification Request and Callback** wizard instead. See [Creating an asynchronous request structure on page 369](#).

1. In the test editor, select a web service or XML request, click **Add**, and then click **Asynchronous Callback**.

Result

The **Create New Asynchronous Callback** wizard opens.

2. On the **Select Callback Endpoint** page, select the XML element of the request where the endpoint URL of the callback is located.
3. If you have a web Services Description Language (WSDL) file for the web service, click **Next**. Otherwise, skip to step 5.
4. On the **Bind Message to WSDL Port** page, select a port from the WSDL file. If the WSDL file for the service is not listed, click **Add** to add a WSDL file from the workspace or to import a WSDL file.
5. Click **Finish**.

Result

This generates a callback structure that contains a parallel, a receive, and a timeout element, in the test editor.

What to do next

In the callback structure, you can add test elements to the parallel, receive, and timeout elements to specify the behavior of the test:

- *Parallel* contains test elements that are run after the asynchronous request has been sent.
- *Receive* specifies the message return of the callback and contains test elements that are run after the callback is received.
- *Timeout* contains test elements that are run if the callback is not received after a specified delay.

Creating a reliable messaging call structure

You can create a test structure dedicated to testing service calls based on the *WS-ReliableMessaging* specification.

Before you begin

The WS-ReliableMessaging specification provides for a series of SOAP messages to be delivered reliably between distributed applications in the presence of software component, system, or network failures. In the context of a service test, a reliable messaging call structure consists of a series of calls that conform to the specification. The structure can be created either as a sequential list of unique service calls or a loop that contains a call element and uses a dataset to identify the unique calls.

1. In the test editor, select the test, and click **Add**, and then click **Specification-based Structure**.

Result

The **New Web Service Test** wizard opens.

2. On the **Web Service Specification Selection** page, Select **WS-ReliableMessaging**, and click **Next**.
3. Select one or several Web Services Description Language (WSDL) files in your workspace for the web service that you want to test, and click **Next**.
If necessary, you can import a WSDL file into the workspace with the **Import** push button.
4. On the **Configure Protocol** page, select an existing HTTP transport configuration, or click **New** to create a new configuration.
 - a. Specify the **URL** of the service, the HTTP **Method**, and **Version**.
 - b. In the **Header** table, click **Add** to specify any specific headers that need to be added to the call.
 - c. In the **Cookies** table, click **Add** to specify any specific cookies that need to be used by the call.
 - d. Click **Next**.
5. On the **Sequence Options** page, specify how the sequence structure will be created in the test.
 - a. In **Message count**, specify the number of calls in the list or the number loop iterations.
 - b. Select **Create service call list** to generate a list of calls with the number of messages or **Create loop with dataset** to generate a loop with a dataset.
The dataset defines the call number for each call in the loop.
 - c. Select **Shuffle sequence** if you want the call numbers to be generated in a random order.
6. Click **Finish**.

Result

This action generates a reliable messaging service call structure in the test.

Editing Socket tests

Improve the Socket tests by adding test elements.

Socket API test editor overview

You use the test editor to inspect or customize a socket API test that you recorded.

The test editor lists the connections and data exchanges for a test as they occurred during the recording.

The test editor window has two main areas. The area on the left, **Test Contents**, displays the flow of the socket events that constitute the test. The area on the right, **Test Element Details**, displays details about the currently selected test element in the test hierarchy.

Values can sometimes be highlighted in green. This highlighting indicates that these requests contain one or both of the following types of information:

- **A dataset candidate:** This is a value, usually one specified by the tester during recording, that the test generator determined is likely to be replaced by values in a dataset. An example of a dataset candidate is a string that you search for in a recorded test. The string is highlighted as a dataset candidate on the assumption that, before playback, you might want to associate the string with a dataset column that contains appropriate substitute values.

- **Correlated data:** These are values in a test, usually one of them in a response and the other in a subsequent request. An example is a product price returned to the browser by a test that searches a product database. You can use these values as references that can be reused later in the test. Suppose that, before running the test with many virtual users, you replace the product name searched for in the recorded test with names in a dataset. Because the test correlates the data, each virtual user searches for a different product, and the server returns an appropriate price.

To see an illustration of color coding in performance tests or to change the color settings, click **Window > Preferences > Test > Fonts and Colors**.

Click **Add** to add elements to the selected test element. Alternatively, you can right-click a test element, and select an action from a menu. The choices that you see depend on what you have selected. For example, after you select a test, you can add a new event.

The **Insert** button works similarly. Use it to insert an element before the selected element.

Use the other buttons (**Remove, Up, Down**) primarily when you substantially modify a test.



Tip: Performing actions with these buttons or choices are likely to break a recorded test.

Sometimes, the area of the editor where you need to work is obscured. To enlarge an area, move your cursor over one of the blue lines until your cursor changes shape to a vertical line with an up arrow at the top and a down arrow at the bottom, and drag up or down while holding the left mouse button.

Displaying binary data

With the **Socket Details** view, you can display the binary data for send and receive elements in the test. The text area of the **Socket Details** view supports many international character encoding standards. To open the Socket Details view, right-click a socket send or receive element, and click **Show Socket Details**.

In the test log, the **Socket Details** view also displays actual the actual binary data sent and received during a test run.

Manipulating elements in the socket test editor

Socket tests are often made of a long series of send and receive elements to or from various connections. The test editor helps you to locate specific elements and to manipulate certain types of elements in the test editor. The following examples demonstrate how you can select and manipulate large sets of test elements:

- To select all the send or receive elements in the test: Click the **Select** button and click **Socket Send** or **Socket Receive**.
- To locate all the connections in the test: Click the **Select** button and click **Socket Connection**.

- To disable or remove all send and receive actions to or from a specific connection: Right click a test element, click **Manage Socket Connections**, click **Only disable them**, and select the connections that you want to disable or remove.
- To reenable all disabled send and receive actions to or from a specific connection: Right click a connection element, click **Select All Related Actions**, right-click again, and click **Enable**.

Related information

[Merging socket send and receive elements on page 377](#)

[Changing multiple socket send and receive elements on page 378](#)

Adding elements to a service test

To improve the reliability of the test script, add the different elements to the service test.

Adding a socket close

You can manually add a socket close element to a socket API performance test.

Before you begin

In a performance test, socket close elements close the connection to a server. A socket close element relates to a specific socket connection.

Ensure that for each socket connection, there is a corresponding socket close. If not, during the test run, the operating system can run out of socket handles, causing the test to fail.

1. Open the performance test in the test editor.
2. In the **Test Contents** area of the test editor, right-click the test node, and click **Add > Socket Close**.
3. In the **Test Element Details** section, click **Change** and select the socket connection that you want to close.
You can specify a **Think Time** delay before the test establishes the connection.

Results

After you have closed a connection, you can no longer send or receive data from that connection. If any test elements use a connection after the close element, they are flagged with an error in the test editor.

Adding a socket send

You can manually add a socket send element to a socket API performance test.

Before you begin

In a performance test, socket send elements describe how data is sent over a connection. A socket send element relates to a specific socket connection.

1. Open the performance test in the test editor.
2. In the **Test Contents** area of the test editor, right-click the test node, and click **Add > Socket Send**.
3. In the **Test Element Details** section, click **Change**, and select the socket connection that you want to send data to.

You can specify a **Think Time** delay before the test establishes the connection.

4. In the **Data** area, type the data that you want to send.

By default, data is sent as 7-bit alphanumeric characters. To specify hexadecimal bytes, prefix the data with `\x`, for example: `\x00\xff`

Adding a socket receive element

You can manually add a socket receive element to a socket API performance test.

About this task

In a performance test, socket receive elements describe how data is received over a connection. A socket receive element relates to a specific socket connection.

1. Open the performance test in the test editor.
2. In the **Test Contents** area of the test editor, right-click the test node, and click **Add > Socket Receive**.
3. In the **Test Element Details** section, click **Change** and select the socket connection that you want to receive data from.

You can specify a **Think Time** delay before the test establishes the connection.

4. Specify the **End Policy**.

This specifies when the receive element stops receiving data and the test resumes:

- **Detects inactivity:** The receive action stops when no bytes are received from the connection after a delay specified in **Inactivity threshold** (in milliseconds). After this delay, the remote computer has finished sending the response and is considered inactive. This is the default setting.
- **Receives exact number of bytes:** The receive action stops when the recorded number of bytes is received. Specify a **Timeout** (in seconds) after which the receive action produces an error in the test log, if the correct number of bytes is not received. If **Link data size** is enabled, the receive action expects the number of bytes displayed in the **Data** area. If **Link data size** is disabled, the receive action expects the number of bytes displayed in **Bytes**.
- **Receives until end of stream:** The receive action stops when the connection is closed by the remote computer. If **Accepts empty response** is selected, then the reception of a single byte is *not* required and the **Response Timeout** is ignored. Specify a **Timeout** (in seconds) after which the receive action produces an error in the test log, if the correct number of bytes is not received.
- **Matches a string:** The receive action stops when a specified sequence of bytes is received. Specify a **Timeout** (in seconds) after which the receive action produces an error in the test log, if the correct number of bytes is not received.
- **Recognizes a regular expression:** The receive action stops when a sequence of bytes that matches a regular expression is received. Specify a **Timeout** (in seconds) after which the receive action produces an error in the test log, if the correct number of bytes is not received.

5. In the **Data** area, type the data that you expect to receive.

By default, data is sent as 7-bit alphanumeric characters. To specify hexadecimal bytes, prefix the data with `\x`, for example: `\x00\xff`.

Verifying application behavior

Add the different verification points to verify the responses in a socket test.

Verifying received content

With content verification points, you can check that actual received data matches the expected data that is specified in the verification point.

About this task

When you add verification points, the received data is compared with the expected data that is specified in the verification point test element. With content verification points enable, you can directly compare the content data that the server returns and return a Pass status when the criteria is met.

To add a content verification point to a socket test:

1. Open the test editor, right click a socket receive element, and select **Add > Content Verification Point**.
2. Select the verification point, and in the **Test Element Details** area of the test editor, specify the verification criteria:

Comparison operator

Specify the criteria to use to perform the verification, among the following operators:

Equals

The verification point returns a Pass status if the received data exactly matches the text or binary content that is specified in the **Data** area.

Contains

The verification point returns a Pass status if the text or binary content that is specified in the **Data** area occurs at least once in the received data.

Starts with

The verification point returns a Pass status if the text or binary content that is specified in the **Data** area occurs at the beginning of the received data.

Ends with

The verification point returns a Pass status if the text or binary content that is specified in the **Data** area occurs at the end of the received data.

Differs from

The verification point returns a Pass status if the received data does not exactly match the text or binary content that is specified in the **Data** area.

Does not contain

The verification point returns a Pass status if the text or binary content that is specified in the **Data** area does not occur at least once in the received data.

Does not start with

The verification point returns a Pass status if the text or binary content that is specified in the **Data** area does not occur at the beginning of the received data.

Data

Specify the data that is expected to be received through the connection.

Binary

In this view, edit the expected content as binary data.

Raw ASCII

In this view, edit the expected content as raw ASCII data. Bytes are expressed as 7-bit alphanumeric characters or two-digit hexadecimal bytes preceded with \x. Additionally, \r and \n stand for carriage-return and line-feed, while \\ represents the backslash character.

What to do next

You can enable or disable each socket verification point by clicking **Enable verification point** in the test editor.

Verifying received message size

With size verification points, you can check that actual received data matches the expected data size in bytes as specified in the verification point.

About this task

When you add verification points, the received data is compared with the expected data that is specified in the verification point test element. With size verification points, you can check the number of bytes in a socket receive element and return a Pass status when the criteria is met.

To add a size verification point to a socket test:

1. Open the test editor, right click a socket receive element, and select **Add > Size Verification Point**.
2. Select the verification point, and in the **Test Element Details** area of the test editor, specify the verification criteria:

Comparison operator

Specify the criteria that is used to perform the verification with these operators:

- Is
- Is less than
- Is less or equals
- Is more than
- Is more than or equal to
- Is not

Value (bytes)

Specify the size criteria for the verification point.

What to do next

You can enable or disable each socket verification point by clicking **Enable verification point** in the test editor.

Verifying received data with custom Java™ code

With custom verification points, you can use Java™ code to verify the data received through a connection.

Before you begin

Using Java™ custom code requires knowledge of the Java™ programming language and of the HCL OneTest™ Performance API. See [Executing test execution with custom code on page](#) for more information.

About this task

When you add verification points, the received data is compared with the expected data that is specified in the verification point test element. Custom verification points return a Pass status when the custom class returns a Pass status after performing a verification written in Java™ code.

To add a custom verification point to a socket test:

1. Open the test editor, right click a socket receive element, and select **Add > Custom Verification Point**.
2. Select the verification point, and in the **Test Element Details** area of the test editor, click **Generate Code**.

Result

This action generates a Java™ class that is based on the template of the HCL OneTest™ Performance API for socket custom verification points.

3. Edit the Java™ code to define the specific verification action to perform.
4. Save and close the Java™ class.

What to do next

To modify the custom class, click **View Code** to open the code in the Java™ editor. You can enable or disable each socket verification point by clicking **Enable verification point** in the test editor.

Merging socket send and receive elements

With the **Organize** wizard, you can merge consecutive send or receive elements that use the same connection to improve the clarity of your socket test.

Before you begin

The wizard can perform the following actions when merging socket test elements:

- Merge consecutive socket send elements that use the same connection.
- Merge consecutive socket receive elements that use the same connection.
- Delete all socket receive elements except for the last one, which is necessary to synchronize the test.

To perform a merge, the selected elements must be consecutive and must be to or from the same connection.

To merge send or receive elements in a socket test:

1. Select the socket send or receive elements that you want to merge.
Press the **Shift** key to select multiple consecutive elements in the test editor.
2. Right-click the selection and click **Organize Send and Receive Actions**. This opens the **Organize** wizard.
3. Choose one of the following options:
Choose from:
 - Select **Merge all selected send or receive actions** to merge the selected elements.
 - Select **Merge all selected send and keep a single receive** to delete all socket receive elements except the last one.
4. Click **Next**.
5. If your selection contained send elements that could be merged, specify the **Custom Code** settings that should be retained for the merged element, and click **Next**.
6. If your selection contained receive elements that could be merged, specify the **Response Timeout** and **End Policy** settings that should be retained for the merged element, and click **Next**.
7. Click **Finish** to perform the merge.

Related information

[Socket API test editor overview on page 371](#)

[Changing multiple socket send and receive elements on page 378](#)

Changing multiple socket send and receive elements

You can perform global changes to multiple send and receive elements in a socket test.

About this task

With the **Organize** wizard, you can perform the following global changes on multiple send and receive elements:

- Change the Custom Code data manipulation settings on selected send elements.
- Change the Response Timeout and End Policy settings on selected receive elements

You can also apply a global *change strategy* that applies to all the selected send and receive elements. A change strategy can change the settings and merge send and receive elements at the same time.

To perform global changes on a series of send or receive elements:

1. Select the socket send or receive elements that you want to merge.
Press the **Shift** key to select multiple consecutive elements in the test editor. Use the **Select** button to select a specific type of test element. Right-click a connection and click **Select All Related Actions** to select all elements that use a specific connection.
2. Right-click the selection and click **Organize Send and Receive Actions**. This opens the **Organize** wizard.
3. Choose one of the following options:
Choose from:
 - If you want to modify the selected elements, select **Change settings on all selected send and receive elements**.
 - If you want to modify all elements that use a specific connection, select **Change settings on all elements related to a specific connection**, click **Next**, and then select the connections for which you want to change the settings.
4. Click **Next**.
5. If you want to change send elements, select **Globally change**, specify the new custom code settings for data manipulation that you want to apply to all send elements, and then click **Next**.
6. If you want to change receive elements, select **Globally change**, specify the new response timeout or end policy settings that you want to apply to all receive elements, and then click **Next**.
7. Click **Finish** to apply the changes.

Related information

[Socket API test editor overview on page 371](#)

[Merging socket send and receive elements on page 377](#)

Splitting a socket test

After you record a test, you can split it into smaller tests. Splitting tests enables you to create modular building blocks of smaller tests and combine them to make bigger tests. The original test is unchanged. You can recombine these building blocks in a schedule, including loops and conditions.

Before you begin

When reusing split tests in a schedule, you must ensure that the general test structure is consistent including socket connection and close elements. For example, you must ensure that all socket send and receive elements are preceded with a corresponding socket connection and followed by a socket close element. When the split tests are recombined in the schedule, you must place them in the correct order.

1. In the Test Navigator, browse to the test and double-click it. The test opens.
2. Right-click a socket send or receive element in the test, and select **Split Test**. The page that you click is the first page of the new test.

3. In the **New Test Names** window, confirm the location of the split, optionally provide names and descriptions for the split tests, and click **Next**.
4. In the **Split Test** window, examine the changes to be performed as a result of the split, and click **Finish**.

Example

For example, you could record a test that contains the following actions:

- Logging on to a server.
- Creating an entry on the server and removing the entry.
- Editing an entry, validating that the change occurred, and restoring the entry.
- Logging off of the server.

You then split the test into four parts: Logon, Create, Edit, and Logoff. You create a schedule that runs virtual users selected from a dataset. Each virtual user runs the Logon test, performs various combinations of the Create and Edit tests, and finally runs the Logoff test.

Using custom code to specify an end policy

You can write a custom Java™ class to specify when a socket receive element stops receiving. This offers the most flexibility, but requires that you write your own Java™ class using the HCL OneTest™ Performance extension API.

Before you begin

The end policy specifies how the receive element stops receiving and allows the test to resume. There are several predefined end policies that you can choose from, for example after a certain number of bytes has been received, or when a specific string is detected. However, in some cases, a complex condition must be defined. This can be done by delegating the decision to custom code.

To create a new custom code class:

1. In the test editor, select a socket receive element.
2. In the **End policy** section, select **Delegated to custom code** and click **Generate Code**.

Result

This creates a Java™ class template that follows the HCL OneTest™ Performance extension API. The Java™ class is created in the `src` folder of the current project.

3. Write the custom code by extending the generating class. See [Extending test execution with custom code on page](#) for more information about extending HCL OneTest™ Performance with Java™ code.
4. Save the custom code and the test.

You can click **View Code** to edit the Java™ class later.

Example

The following example is a sample custom class that demonstrates how to configure a custom end policy for the internet time protocol:

```
package test;

import java.text.DateFormat;
```



```

import java.util.Date;
import java.util.TimeZone;

import com.ibm.rational.test.lt.execution.socket.custom.ISckCustomReceivePolicy;
import com.ibm.rational.test.lt.execution.socket.custom.ISckReceiveAction;
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

/**
 * Custom receive policy CustomReceive_TimeReceiver.
 * For javadoc of ITestExecutionServices, select 'Help Contents' in the Help menu and select
 * 'Extending HCL OneTest™ Performance functionality' -> 'Extending test execution with custom code'
 */
public class CustomReceive_TimeReceiver implements ISckCustomReceivePolicy {

    // static {
    // static blocks are called once on each run and allow for example to bind
    // to an external dynamic library
    // }

    ISckReceiveAction receiveAction;
    ITestExecutionServices testExecutionServices;

    public CustomReceive_TimeReceiver() {
        // The constructor is called during the test creation, not at the time of the execution of
        // the customized receive action
    }

    public void setup(ITestExecutionServices tesRef,
        ISckReceiveAction receiveActionRef) {
        testExecutionServices = tesRef;
        receiveAction = receiveActionRef;
    }

    public boolean onRead(int readByte) {
        // TIME protocol (RFC 868): a connected server returns 4 bytes and closes the connection
        // Those 4 bytes are the number of seconds since 1900/1/1
        // The test is simply made of a connection to one TIME server on port 37
        // (public servers are listed here: Got time server host name from
        http://tf.nist.gov/service/time-servers.html),
        // Then a receive delegated to this custom code class,
        // Then a close
        try {
            if (readByte == EndOfStream) {
                /* In case of success: */
                receiveAction.receiveSuccess();
                String message = extractAndCheckTime(receiveAction.getConnectionHolder().getFinallyReceivedBytes());
                /* A message is appended in the Test Log just after this receive action: */
                testExecutionServices.getTestLogManager().reportMessage(message);
                return true;
            }
        } catch (Throwable t) {
            /* In case of exception: */
            receiveAction.handleException(t);
            return true;
        }
        if (receiveAction.getConnectionHolder().getCurrentlyReceivedBytesCount() > 4) {
            /* Unexpected condition: */
            receiveAction.handleException(new Exception("Time protocol server returned more than 4 bytes"));
        }
    }
}

```

```

    return true;
}
/* We need further bytes to complete this receive */
return false;
}

private String extractAndCheckTime(byte[] bytes) {
    // This is network order, i.e. big endian
    long remoteTime = (((long)bytes[0]) & 0x00000000000000ff) << 24) +
        (((long)bytes[1]) & 0x00000000000000ff) << 16) +
        (((long)bytes[2]) & 0x00000000000000ff) << 8) +
        (((long)bytes[3]) & 0x00000000000000ff);
    // 1900 to 1970: a difference of reference, see RFC 868 and java.util.Date javadoc
    remoteTime -= 2208988800L;
    Date remoteDate = new Date(remoteTime*1000);
    Date localDate = new Date();
    DateFormat dateFormat = DateFormat.getDateInstance();
    dateFormat.setTimeZone(TimeZone.getTimeZone("GMT"));
    String message = "Remote time: " + dateFormat.format(remoteDate) + " GMT (TIME server is " +
        receiveAction.getConnectionHolder().getHostName() + ", port 37)\n" +
        "Local time: " + dateFormat.format(localDate) + " GMT\n";
    long diff = localDate.getTime()/1000 - remoteTime;
    if (diff == 0) {
        message += "-> No difference";
    } else {
        message += "-> Difference (seconds): " + diff;
    }
    return message;
}
}
}

```

Using custom code to manipulate data

You can write a custom Java™ class to manipulate data in a send element. This offers flexibility for injecting data in test, but requires that you write your own Java™ class using the HCL OneTest™ Performance extension API.

Before you begin

In some cases, complex methods of generating data are required to send specific content to the server. This can be done by manipulating data with custom code.

To create a new custom code class:

1. In the test editor, select a socket send element.
2. Select **Manipulate data with custom code** and click **Generate Code**.

Result

This creates a Java™ class template that follows the HCL OneTest™ Performance extension API. The Java™ class is created in the `src` folder of the current project.

3. Write the custom code by extending the generating class. See [Extending test execution with custom code on page](#) for more information about extending HCL OneTest™ Performance with Java™ code.
4. Save the custom code and the test.

You can click **View Code** to edit the Java™ class later.

Editing Kerberos tests

You can change the Kerberos realm, user name, and password when editing tests.

1. In the Test Navigator, browse to the test, and double-click the test name. The test opens.
2. Click the **Security** tab.
3. Expand **Kerberos**.
To edit the Kerberos information in a test, **Enable Kerberos authentication** must be selected.
4. Edit the Kerberos client realm name, client key distribution center (KDC), user name, and password as necessary.

Result

The realm name, user name, and password are used for all Kerberos connections and requests in the test.

5. Select **Server is in a different realm than client** to edit the server realm name and server KDC if the server is in a different realm than the client.
6. **Optional:** Click **Edit krb5.ini**, and then type a realm name, to edit the Kerberos configuration file.

What to do next

Kerberos user names and passwords can be associated with a dataset. See the related topic on providing tests with variable data to learn more about datasets.

Related information

[Providing tests with variable data \(datasets\) on page](#)

Adding test elements

You can add a variety of elements to a test, such as transaction blocks, IF-THEN conditions, loops, and comments.

Adding a comment

You can add a comment to document a test.

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. In the test, select the item that you want to comment.
3. Click **Add** and then click **Comment**.

Result

The comment icon is placed before the selected item and the **Comment text** field opens.

4. Add the comment to the **Comment text** field. The comment is added to the test.

Adding a transaction to a test

A *transaction* is a specific group of test elements whose performance you are interested in. When viewing the test results, you can view performance data about any transactions that you have added.

To put a group of test elements into a transaction:

1. In the Test Navigator, browse to the test and double-click it.
Result
The test opens.
2. In the test, select the test elements to group together. Use Shift+click to select multiple contiguous elements; use Control+click to select multiple noncontiguous elements.
3. Click **Add** (to place the transaction after the selected element) or **Insert** (to place the transaction immediately before the selected element or block), and click **Transaction**.
4. You are prompted whether to move the selected objects into the transaction. Click **Yes** or **No**.
5. **Optional:** In the transaction details, you can give the transaction a meaningful name. This is useful in the Transactions report, which lists transactions by name.

Adding conditional logic

You can add IF-THEN conditional logic around portions of a test/compound test or a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule. to make those portions run if a specific condition is met.

Before you begin

A conditional block can run portions of a test depending on the value of a reference or field reference. The reference or field reference must exist in the test and precede the conditional block. If the reference or field reference that the conditional block uses for input does not exist, you must create the reference as explained in [Creating a reference or field reference on page 463](#).

About this task

The test might already contain the test elements that are to be run. If the test does contain the elements to be run, you must select the requests in step 2 of the procedure, and click **Insert**. The following instructions explain how to add a conditional block that contains such requests.

Otherwise, you can create an empty conditional block at the end of the selected item (test or request). Click the object, and then click **Add**.

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens. You can also open a compound test or a schedule.

2. Click a page or page request.

The conditional block is inserted before the selected item. By completing step 5, you can move the selected items into the block.

3. Press **Shift** or **Ctrl** when clicking to select multiple pages or requests to be moved into the block.
In step 6, if you add an **Else** block, you can select one or more of these items to be moved into the **Else** branch.
4. Right-click the item and select **Insert > Condition (IF)**.

Result

A prompt **Would you like to move selected objects into the new IF?** is displayed.

5. Click **Yes** or **No**.

Result

The If block is inserted into the test. If you click **Yes**, as shown in the example, the items that you selected are moved under If in the **Test Contents** area. The following example shows an If block with an HTTP test.

Test Contents
This section shows the test contents

Enter filter text

Options ▾

- [-] Browse
 - [+] Test Variables
 - [+] HomePage
 - [+] Login
 - [+] HomePageAfterLogin
 - [+] Flowers
 - [+] If
 - [+] Coleus
 - [+] Flowers2
 - [+] Lily
 - [+] Flowers3
 - [+] Fruits&Veg
 - [+] Grapes
 - [+] Fruits&Veg2
 - [+] Strawberries
 - [+] Trees
 - [+] Crabapple
 - [+] Accessories

Buttons: Add, Insert, Select, Remove, Up, Down, Prev, Next, Run, View

Test Element Details
Conditional (IF) Block

Condition
Type the text to compare or select a data source object from the list

First operand: true Data Source...

Selected data type: **text value**

Operator: Equals

Second operand: true Data Source...

Selected data type: **text value**

Options

- Negate the operator (NOT(op))
- Case-sensitive comparison

If the condition above evaluates to "true" execute:

- [+] Coleus
- [+] Flowers2
- [+] Lily

6. To add an Else block:

- a. In the **Test Contents** area, under If, select the items to be moved to the Else block. Press Shift or Ctrl when clicking to select multiple items.
- b. Right-click and select **Insert > Condition (IF) - ELSE Block**.

Result

A prompt **Would you like to move selected objects into the new ELSE?** is displayed.

- c. Click **Yes** or **No**.

Result

The **Else** block is inserted into the test. If you click **Yes**, as shown in the example, the items that you selected are moved under **Else** in the **Test Contents** area and into the **Else** field in the **Test Element Details** area. The following example shows an If-Then-Else block with an HTTP test.

The screenshot displays the HCL OneTest Performance interface. On the left, the **Test Contents** panel shows a tree view of test elements. An **If** block is expanded, showing a **Then** section with **Coleus** and **Flowers2**, and an **Else** section with **Lily**. On the right, the **Test Element Details** panel shows the configuration for the **Conditional (IF) Block**. The **Condition** section is configured with the following values:

- First operand:** true
- Operator:** Equals
- Second operand:** true

The **Options** section is configured with the following values:

- Negate the operator (NOT(op))
- Case-sensitive comparison

The **Options** section also includes a list of actions to be executed if the condition evaluates to "true":

- Coleus**
- Flowers2**

The **Else** section is configured with the following value:

- Lily**

7. In the **Test Element Details** area, under **Condition**, add conditions:

- Next to the **First operand** field, click **Data Source**, and then select a data source to be compared with the string in the **Second operand** field, or type a value in the **First operand** field.
- In the **Operator** field, indicate the basis of comparison of the two operands.
Note that the two operands are strings.

Starting from 9.1.1.1, you can use the **Matches regex** operator to match the operands. Click **Verify** to verify whether the regular expression finds a match in the data source. If there is a match, the **Verify** button is disabled. If there is no match, a message is displayed. If the regular expression is invalid, a message is displayed for that also.

- Next to the **Second operand** field, click **Data Source**, and select a data source to be compared with the **First operand**, or type a value in the **Second operand** field.
When the defaults are used (both operand fields set to `true` and the **Operator** field set to `Equals`), the block is always processed.

8. In the **Test Element Details** area, under **Options**, choose the required comparison type by selecting or clearing the check boxes.

Synchronizing users in tests

By inserting a synchronization point, you can coordinate the activities of a number of virtual users by pausing and resuming activities. You can synchronize all virtual users at the beginning of a test and stagger the release times so that the users do not overload the system. Synchronization points are also useful in stress testing.

About this task

You can insert a synchronization point into a schedule or a test. You typically insert synchronization points into schedules, because they are more visible at the schedule level and you can set the **Release** and **Timeout** options within a schedule only, not within a test. However, in the following cases, insert a synchronization point into a test:

- You must control where the synchronization point is encountered. For example, you can insert a synchronization point just before a test sends a request to a server.
- You have edited a test, and the execution of a synchronization point depends on the logic that you have added.

Synchronization points within loops are not reset. In other words, after a synchronization point has been released (in the first iteration of a loop), the synchronization point stays released for all further iterations.

To insert a synchronization point into a test:

1. In the Test Navigator, browse to the test, and double-click it.

Result

The test opens.

2. Click the test element just below the place to add the synchronization point, and then click **Insert > Synchronization point**. Depending on the nature of the test element, you can insert a synchronization point at some points in the test hierarchy but not at others.
3. Type a name for the synchronization point, or select the name of an existing synchronization point.

Result

The synchronization point opens in the test. Note that **Release Type** and **Timeout** are not available for synchronization points in tests. **Release Type** and **Timeout** are available only for synchronization points in schedules.

4. To change the **Release Type** or **Timeout**, open the synchronization point within a schedule, and make the changes. The changes affect all instances of the synchronization point.

Adding a loop to a test

You can define part of a test as a loop that runs a specified number of times. If the loop contains a synchronization point, the synchronization point is released after the first iteration of the loop and stays released for all further iterations.

About this task

You can set a loop within a schedule or a test. The following table shows the advantages of both methods:

Loop location

Results

Schedule - Loops in schedules are easy to locate and modify. Loops in schedules close the server connection at the end of each iteration and reopen it at the beginning of the next iteration. This action models the behavior of a user closing and reopening a browser. Use this method to run a test at a set rate.

Test - Loops in tests can be more granular, and thus provide a higher level of control.

Loops in tests reuse the server connection during each loop iteration.

Use this method, with loops that have high iteration counts, to stress test a server.

1. In the Test Navigator, browse to the test, and double-click it.

Result

The test opens.


2. Click the page or the request that will be inside the loop.
Press Ctrl when clicking to select multiple pages or requests.
3. Click **Insert**, and select **Loop**.
4. You are asked whether you want to move the selected elements into a the loop. Click **Yes**.
If you click **No**, an empty loop is inserted into the test.
5. In the **Loop Details** area, type the number of iterations for the loop to repeat.

Option	Description
Count-based	Runs for the number of iterations that you select.
Time-based	Runs for at least the time that you specify. The loop always finishes the iteration. For example, if you select a time of 1 second and a loop takes 10 seconds to run, the loop finishes one iteration, and then checks the time.
Infinite	Runs until the test stops.




Note:

If a test or a schedule includes multiple loops with dataset values and a new dataset value is required for the first iteration of the second loop, then a dataset increment is required before the second loop runs. To do this, you must insert a data source controller by clicking **Insert > Data Source Controller** before the second loop starts and then select the required dataset. You can then select the **Increment**

 option for the data source from the **Data Source Controller Details** pane that triggers the retrieval of the dataset value to automatically choose the new dataset value.

6. Optional: Select **Control the rate of iterations**, and type your preferences for the pacing rate. In specifying a number of iterations for a unit of time, you set a fixed period for the iterations to complete. If you select **Randomly vary the delay between iterations**, the total delay is randomly distributed. If you clear this check box, the same delay occurs between each iteration.

 **Note:** Statistically, the **Randomly vary the delay between iterations** option sets delay amounts at random from a negative exponential distribution with the same mean as the fixed delay value. The negative exponential distribution has a long "tail," which means that a very small number of delays will have very large values. Therefore, make sure that the application you are testing is not negatively affected by long periods of inactivity (such as a timeout that disconnects the user).

Adding Dataset Mapper

You can include a Dataset Mapper in a compound test or a schedule to assign the dataset values to the variables that are defined in multiple tests. In previous releases, to apply the dataset values to multiple tests, you had to associate the dataset to each test. The Dataset Mapper is able to map the dataset columns with the variables.


Before you begin

You must have created at least one dataset. See [Creating a dataset in a workspace on page 410](#).

About this task

For the Dataset Mapper to fetch the test variables, in the Variable Details section of the Test editor, you must set the **Visible In** field for the variable to **All tests for this user**. You can also fetch the variables from the custom code calls.

If the compound test or the schedule includes a Dataset Mapper that retrieves values from one dataset and a test in the compound test or schedule is also associated with another dataset, the run uses both the datasets.

 **Note:** When you run the schedule or compound test with a Dataset Mapper, by default the test picks up the dataset values from the first row. For the test to pick up all of the dataset values, you must put the test in a loop.

1. In the Schedule or Compound test editor, click **Add > Dataset Mapper**.
2. In the **Select Dataset** dialog box, select a dataset to use for the tests and click **OK**.
To change the dataset after it is associated, in **Dataset Mapper Details**, click **Browse** and select another dataset.
3. Select the **Open mode** for the dataset. This mode determines the view that virtual users have of the dataset. This option is useful when you do a parallel test run.

Option	Description
<p>Shared (per test execution) (default)</p>	<p>When you choose the Shared (per test execution) option, the virtual users running in the test share the dataset values in sequential order.</p> <p>For example, if your dataset has 10 rows, the dataset values are taken from row 1, row 2, row 3, and so on when you select this option.</p>
<p>Private</p>	<p>Virtual users draw from a private view of the dataset, with dataset rows apportioned to each user in sequential order.</p> <p>This option ensures that each virtual user gets the same data from the dataset in the same order. However, because each user starts with the first row of the dataset and accesses the rows in order, different virtual users will use the same row. The next row of the dataset is used only if you add the test that is using the dataset in a loop with more than one iteration.</p>
<p>Shared (for all test executions)</p>	<p>When you choose the Shared (for all test executions) option, the virtual users running in multiple tests share the dataset values from the current row.</p> <p>For example, if your dataset has 10 rows and when you set the current row as row 5, the dataset values are taken from row 5 instead of row 1 when you select this option. If you had set the current row as row 1 and used the dataset values until row 5, the dataset values are retrieved from row 6 when you run the test next time.</p>

4. Select the **Access mode** for the dataset:

- **Sequential:** The rows in the dataset are accessed in the order in which they are physically stored in the dataset file, beginning with the first row and ending with the last.
 - **Random:** The rows in the dataset are accessed in any order, and any given row can be accessed multiple times or not at all. Each row has an equal chance of being selected each time.
 - **Shuffled:** Before each dataset access, the order of the rows is changed, and a different sequence results. Rows are accessed randomly but all rows must be selected once before a row is selected again.
5. Select whether the test will reuse data when it reaches the end of the dataset.


By default, when a test reaches the end of a dataset or dataset segment, it reuses the data from the beginning. To force a test to stop at the end of a dataset or segment, clear the check box **Wrap when the last row is reached**. Forcing a stop might be useful if, for example, a dataset contains 15 records, you run a test with 20 virtual users, and you do not want the last five users to reuse information. Although the test is marked as *“Fail”* because of the forced stop, the performance data in the test is still valid. However, if it does not matter to your application if data is reused, the default of wrapping is more convenient. With wrapping, you need not ensure that your dataset is large enough when you change the workload by adding more users or increasing the iteration count in a loop.

6. Select whether the test will make the data in the dataset record permanent for each virtual user.

By default, one row is retrieved from the dataset for each execution of a test, and the data in the dataset row is available to the test only for the duration of the test. Select **Fetch only once per user** to specify that every access of the dataset from any test being run by a particular virtual user will always return the same row.

To illustrate how these options affect the rows that are returned, assume that a test contains a loop which accesses a dataset. The loop has two iterations. The following table shows the row that is accessed in each iteration:

Dataset option	Iteration 1	Iteration 2
Sequential and Private	row 1	row 2
Shared and Shuffled	row x	row y
Fetch only once per user	row x	row x

7. In the **Columns mapping** table, the **Column** is automatically filled with the column names from the dataset.
8. To use the variable names from the test, click the cell and click the Ellipsis button  and select the variable. By default, the variable names are also created with the same names as the dataset columns.
9. If the dataset that you selected in step 2 was generated by HCL® OneTest™ Data, you can choose to update the data by clicking **Update dataset** or update the data automatically for every run by selecting the **Update dataset during deployment** check box.
10. To fetch all the dataset values, put the Dataset Mapper in a loop. Select the **Datapool Mapper** in the schedule and click **Insert > Loop**.
11. Save the changes.

Adding data source controller

Use this test element to control how the data is fetched from the data sources to be used by the test. You can use data from dataset, array variables, built-in data sources, and correlation.

About this task

In the Test editor, the pages/requests succeeding the data source controller makes use of the data source associated to the controller. So, if there are 3 pages in your test and the first controller is placed before the first page and the second controller is before the last page, the first two pages use data associated with the first controller. The last page will use data associated with the second controller.

1. In the Test editor, select the pages for which you want to substitute data and click **Insert > Data Source Controller**.
2. Select the data source from where you want the succeeding pages to pick data and click **Select**.

Result

The data source controller is added to the test.

3. In Data Source Controller Details section, select **Increment** for the test to automatically pick the next value from the data source or select **Reset** to pick the first available value from the data source.
4. **Optional:** To change the data source, click **Data Source** and save the test.

Controlling the flow of test

A test is usually run in the order it was recorded. However, you might want to add some conditions to the test that the users would actually face when interacting with the application under test. For example, if you select a product, you want to know its manufacturing date. If the date is not available, you want to exit. You can now add such control to the test.

1. In the Test Navigator view, browse to the test and double-click it.
2. Select the request or page that uses the **IF** condition and click **Insert > Test Flow Control**.
3. In the **Test Flow Control Details** section, specify one of the following actions:

Action	Description
Continue	Continue running the test.
Exit Transaction	Exit the transaction and continue running the test. If there are multiple levels of transactions, specify whether you want to exit the innermost or outermost transaction.
Exit Loop	Exit the loop and continue running the test. If there are multiple levels of loops, specify whether you want to exit the innermost or outermost loop.

Action	Description
Continue to next iteration of loop	Continue to run the next iteration of the loop after the existing iteration completes.
Exit test	Exit the test.
Exit user	Exit the currently running virtual user and start running with the next user.
Terminate Run	Stop the A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule..

Searching within tests

Search request data or response content by right-clicking in the data or content and selecting **Find**. To search for specific element types and to display the results in a table, click **Select**. For a still more powerful search and replace, use the **Test Search** function.

About this task

You can use a number of different methods to search within a test.

- Use the **Find** option to search within the **Test Details** area and, optionally, to replace text.
- Use the **Select** button to search within the **Test Contents** area and display a table of like test elements.
- Use the powerful **Test Search** function to search within the **Test Contents** and the **Test Details** areas. For example, you can search for a type of verification point and also declare whether the result should include enabled verification points, disabled verification points, or verification points in both states. The specific data that you can search for and the search options are protocol-dependent.

Locating specific types of test elements

A test script can include multiple requests and responses with many test elements and attributes. To locate elements of a specific type in the **Test Contents** area, click **Select**. The results are displayed in a table, and you can sort the table columns. This option is also useful for viewing attributes of test elements that are the same type.

1. In the Test Navigator, browse to the test and double-click it.

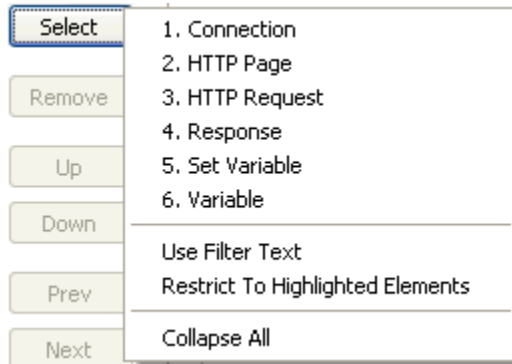
Result

The test opens.

2. To restrict the search to specific instances of elements, select them in the test. For example, you might want to search for text in specific responses, rather than in all responses.
3. Click the test editor tree to give it focus, and then click **Select**.

Result

A list of element types opens. This list is generated dynamically based on the contents of the test. For example, if a test does not contain verification points, they are not listed.



4. To include only the elements that you selected, select **Restrict To Highlighted Elements**.
5. To include only the elements that contain specific text from the **Test Contents** area, select **Use Filter Text**.
6. Select the type of test element to view from the list.
7. If you selected **Use Filter Text**, enter the filter text in the prompt, and then click **OK**.

To use regular expressions in the filter text, click the **Search Options** icon to the right of the prompt, and then select **Regular expression**. By default, if **Regular expression** is not selected, the asterisk (*) and question mark (?) are interpreted as wildcard characters. To search for a URL that contains an asterisk or question mark, type a backslash (\) before the asterisk or question mark.

Results

The **Test Element Details** area displays the results under the heading **Multiple Items**.

Example

The following example shows results for an HTTP request. Other protocols might display less detailed information. Double-click a table row to locate the element within the test.

Test Element Details

Multiple Items

	URL	Delay	Response	Size
GET	www.ibm.com/	0 ms.	302 - Found	206
GET	www.ibm.com/us/	0 ms.	200 - OK	31,458
GET	www.ibm.com/common/v16/hp/js/h...	0 ms.	200 - OK	5,742
GET	www.ibm.com/common/v16/css/all.css	0 ms.	200 - OK	138
GET	www.ibm.com/common/v16/css/scr...	0 ms.	200 - OK	18,188
GET	www.ibm.com/common/v16/css/scr...	156 ms.	200 - OK	99,371
GET	www.ibm.com/common/v16/css/us/...	0 ms.	200 - OK	15,847

Searching and replacing text in tests

With the **Test Search** function, you can search for text in a test or search within specific test elements and optionally replace the found text.

1. In the **Test Navigator**, browse to the test and double-click it.

Result

The test opens.

2. Right-click the test name, and then select **Test Search**.
3. In **Search for text**, type the text to locate.

You can leave this field blank, depending on your search strategy. For example, if you know that a string occurs in elements or element instances that you are not interested in, by using the options described in steps 4, 6, and 8, you can locate the elements or element instances of interest before entering the search text into this field.

4. If you have selected pages or requests within the test, click **More Options**, and then select **Restrict search to elements highlighted in Test Contents**.

This restricts the search to the selected pages and requests.

5. To perform a case-sensitive search, select **Case sensitive**. To search with regular expressions, select **Regular expression**.

In regular expression mode, press Ctrl and type a space in **Search for text** for content assistance. Content assistance lists the regular expression patterns and the content that they match.

6. To highlight found elements in the **Test Contents** area, click **More Options**, and then select **Highlight found elements in Test Contents**.

You can use this option with the option that is described in step 4 to designate the element instances of interest before specifying the text of interest.

7. To have the search include children of the selected element, click **More Options**, and then select **Recursive**.

This option is selected by default. If **Recursive** is cleared, then only the selected element is searched.

8. To have the search locate both encoded and decoded versions of the specified text, click **More Options**, and then select **Match encoded and decoded values**.

This option is selected by default. The type of encoding that the search supports varies depending on the protocol.

Example

For example, when searching in HTTP data, `abc%123` and `abc%25123` match.

9. In the **Elements to search** list, select all test elements to search.

Selecting the check box in step 4 restricts the elements that you can select in this step to the instances that are selected in the **Test Contents** area. For example, if you select **HTTP Pages** here and only one page is selected in the **Test Contents** area, only that page is found. If the check box in step 4 is cleared, every test page is found.

10. **Optional:** Click selected elements to define how to search them.

A new area opens, where you can define how to search a selected element.

To locate items, continue to the next step. To replace found strings, click **Replace**, and go to step 12.

11. Click **Search**.

Result

The results of your search are displayed in two views

- The Search view, which lists the objects that contain matches
 - The Test Search Match Preview view, which displays the matches that were found
12. In the Search view, complete any of these search actions:
 - To preview a found string in the Test Search Match Preview, click the object.
 - To open your test at the location where an instance is found, double-click the object.
 - To perform a different search action (such as proceed to the next match or previous match, replace), right-click the object, and select your choice.
 13. If you clicked **Replace** in step 9, the **Replace** window opens. In the **With** field, type the replacement text.
 14. Select the replacement action by clicking the appropriate push button.

Result

If you are making selective replacements, found instances are displayed in the same order as in the **Test Search Match Preview** view. Click **Replace** or **Skip** until all found instances have been displayed.

Exporting a test

To share the test scripts with manual testers or reviewers who do not have the workbench, export the test scripts to text files. You can export one file at a time.

1. In the Test Navigator view, double-click a test.
2. In the test editor, right-click the root node of the test and click **Export Contents to File**.
3. Select a project and specify the name of the file to export to.
4. To add a separator between two steps or lines, select the **Add line separators after each step** check box and click **Finish**.

Results

The text file opens on another tab in the workbench and is saved in the directory it is exported to.

Copying test assets with dependencies

You can export test assets, and then import them into another project or workspace without losing any dependencies. Test assets include projects, schedules, and tests. You can export and import test assets to collaborate with other testers.

Before you begin

If you plan to export assets with dependencies, make sure that you have migrated the test assets to the current version of the product before you start to export.

About this task

When you copy a test with dependencies, any datasets or custom code referred to by the test are also copied. When you copy a schedule with dependencies, any locations or tests referred to by the schedule are also copied. When you copy results, any schedules or tests referred to by the results are copied.

1. In the **Test Navigator** view, right-click the test assets to export, and then click **Export**.
You can export projects, schedules, tests, and test results with dependencies.

2. In the **Export** window, expand the **Test** folder, and then click **Test Assets with Dependencies**.

You can export test assets with dependencies if the test assets were created in the current version of the product. You cannot export test assets with dependencies if the test assets were created in a previous version of the product and the assets have not been migrated to the current version of the product.

3. Click **Next**.
4. Specify the path and name of the archive file into which you want to export the selected test assets.
5. Click **Finish**.

The assets are exported to the archive file. You are prompted if the total size of the test assets is larger than 4 GB, or if any individual test asset file is larger than 4 GB. To copy test asset files that are larger than 4 GB, copy the files manually.

6. If the target workspace is on a different computer, transfer the archive file to a location that is accessible to that computer.
7. In the **Test Navigator** view, select the test project into which you want to import the test assets.

The target project must have the same name as the source project. Optionally, you can import test assets with dependencies into a workspace where no projects exist. If you import test assets with dependencies into a workspace where no projects exist, the Import wizard creates projects based on information from the archive file. To import test assets into a project with a different name, you must first import the test assets into a project with the same name, and then manually move the assets into the project with the different name.

8. Click **File > Import**.
9. In the **Import** window, expand the **Test** folder, and then click **Test Assets with Dependencies**.
10. Click **Next**.
11. In the **Import with dependencies** window, click **Browse**, and then select the archive file.

The test assets are displayed in the **File contents** list.

12. Click **Finish** to import the test assets with dependencies from the archive file into the target project.

If a file that you are attempting to import already exists in the target workspace, you will be prompted to choose whether to overwrite the file. You can also choose to overwrite all files that already exist in the target workspace, or not to overwrite any files that already exist in the workspace. If you choose to overwrite all files that already exist in the target workspace, you will be prompted again if the import process encounters a `.classpath` or `.project` file in the source archive file.

Copying projects

You can export a test project from a workspace and import it into another workspace.

About this task

If you export test assets to an archive file and then import them to another project, ensure that both project names are the same. Otherwise, you might not be able to locate your imported test assets.



Note: You can also export the test project with all the dependent assets in to an archive file. See [Copying test assets with dependencies on page 396](#) for the instructions.

1. Start HCL OneTest™ Performance, and select the source workspace.
2. Export the test project to an archive file.
For instructions, see [Exporting resources to an Archive file](#). Datasets can be located either in the same project as the tests that use them or in different projects. Be sure to export all the datasets that the exported tests require.
3. If the target workspace is on a different computer, transfer the archive file to a location that is accessible to that computer.
4. Start HCL OneTest™ Performance, and select the target workspace.
5. Click **File > Import**. Expand the **General** folder, and click the **Existing Projects into Workspace** icon; then click **Next**.
6. Click **Select archive file**, and then click **Browse** to select the archive file. Click **Finish** to import the source project from the archive file into the target workspace.
7. **Optional:** If the imported project contains custom code or tests that have been run, you might need to change the Java™ build path.

The following examples are cases that might require a change to the Java™ build path:

- The Java™ build path was manually changed in the project from which it was exported. In this case, the same changes need to be made in the imported project. While importing, you are asked whether to overwrite the class path file, which stores the Java™ build path for project. Answering **Yes** reduces the likelihood that the build path will need to be changed.
- The project was imported onto a different computer with a different Java™ installation configuration. In this case, missing libraries must be removed from the build path.
- The project was imported into a workspace on a different disk drive. When you are asked whether to replace the class path file, answering **No** reduces the likelihood that the build path will need to be changed.

For instructions on changing the build path, see the [Java™ Build Path page](#).

What to do next

If you encounter errors after importing a test project or when using an existing workspace with a new version of the product, you might need to delete .java files from the `src` folder in the workspace:

1. Click **Window > Open Perspective > Resource** to open the Resource perspective.
2. In the **Navigator** window, expand the test project folder, and locate the `src` folder.
3. Delete all .java files in the `src` folder, except for those that contain custom code.
4. Return to your test perspective: Click **Window > Open Perspective**, and select **Performance Test** (or **Service Test**, if you are using Rational® Service Tester).

Disabling portions of a test

When you disable portions of a test, you can still see the disabled portion, but it is not executed during a run. You can also disable portions of a schedule by using the following procedure.

About this task

To disable secondary HTTP requests, see [Disabling and enabling secondary HTTP requests on page 305](#).

To disable an element:

1. In the Test Navigator, browse to the test, and double-click it.

Result

The test opens.

2. Right-click the element that you want to disable, and select **Disable**.

The element and the dependent child elements, which are disabled automatically, are shaded and preceded by two forward slashes (//) to remind you that they are disabled.



Note: To change the color or symbol that represents disabled elements, click **Windows > Preferences > Test > Test Editor**, and then click the **Colors and Fonts** tab.

Result

Although a disabled test element does not run, you can still work with it. For example, you can insert a test into a disabled user group for later use.

3. To enable a disabled element, right-click it, and select **Enable**. Select **Enable All** to enable all disabled elements.

Example

Disabling an element affects other elements in the following ways:

Disabled element	Result
User group (percentage)	The percentages in the remaining user groups are recalculated. When you enable the user group again, remember to return all of the affected user groups to their original percentage.
User group (absolute)	The number of users in the remaining groups might not match the total number of users specified in the schedule. If so, new virtual users are redistributed among the remaining user groups so that the numbers will match. When you enable the user group again, remember to return all of the affected user groups to their original totals.

Disabled element	Result
Request or step that contains a data correlation reference	Substitution in the remaining actions that depend on this request does not work.
Request or step that contains a data correlation substituter	Substitution does not occur because the entire action is omitted. The substituter that uses the disabled data source is also disabled. To re-enable the substituter, select an enabled data source for substitution.
HTTP request that contains a server connection	No effect. The connection is automatically created in the next request.
Portion of custom code	Custom code with disabled arguments is flagged. If the disabling causes an unexpected number of arguments passed to custom code elements, you receive an error at runtime. To fix this, modify the custom code to check the number of arguments.
IF <i>data_source</i> construct	An IF construct is marked as invalid if it contains a disabled data source.
Test element and child are disabled	<p>If you disable a child element and then disable its parent (for example, a request and then a page), the disabled child element will have two prefixes: one created manually and one inherited. In the following example, the first request has inherited the disabled state. The second request has been manually disabled and has also inherited the disabled state:</p> <pre data-bbox="829 1247 1422 1339">//disabled page //request ///disabled request</pre> <p>Do one of the following to re-enable the second request:</p> <ul data-bbox="878 1430 1377 1541" style="list-style-type: none"> • Re-enable the request, and then re-enable the page. • Right-click the page and select Enable All.
A data source or a range of text that will be replaced	The Data table displays this text in gray.

Running test elements in random order

You can record multiple user scenarios in a test and then run each scenario in a random order. To do this, you put each scenario under a random selector and then select the proportion of time that the scenario should be run.

About this task

For example, you can record a test that includes logging on to a system, browsing through items in the system, buying various items, and then totaling the order. In this case, you could run the logging in and the totaling scenarios once, but put the browsing and buying scenarios under a random selector.

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. Click the test element that will be controlled by the random selector, and then click **Insert > Random Selector**. Use Shift+Click to select multiple elements.
3. You are asked whether you want to move the selected elements into a new random selector. Click **Yes**. Click **No** to insert an empty random selector into the test.



Note: To set whether or not elements are moved automatically, or whether you are prompted, click **Window > Preferences > Test > Test Editor**, and click the **General** tab.

4. Set the weight of the random selector. The weight determines the statistical probability that a specific element will be selected.
 - a. If you have added a number of test elements, the **Create weighted blocks** window is displayed. You can select adjacent elements and group them. Each element—whether in a group or by itself—must be weighted.
 - b. If you have added only one test element, the weighted block is displayed in the **Test Element Details** area with a default of 1.

Exemple

When a selector contains many different weights, you can mathematically determine the likelihood that a block will be executed. To do this, add the weights together and divide the weight for each block by that total.

For example, assume a selector contains six blocks set to the following weight:

- two blocks set to a weight of 1
- one block set to a weight of 2
- two blocks set to a weight of 5
- one block set to a weight of 9

The total of the weights is: $1 + 1 + 2 + 5 + 5 + 9 = 23$. Therefore, the statistical likelihood of selection is:

Weight of block	Likelihood of block being selected
1 (two blocks)	$1/23 = 0.0435$, or about 4.35% (for each block)
2	$2/23 = 0.0870$, or about 8.70%
5 (two blocks)	$5/23 = 0.2174$, or about 21.74% (for each block)
9	$9/23 = 0.3913$, or about 39.13%

Note that a higher weight increases the likelihood, but does not guarantee, that a block will be executed. Some variation might occur. For example, if you run a test 23 times, you cannot predict that the first and second blocks will execute exactly once, the third block exactly twice, the fourth and fifth blocks exactly five times, and the sixth block exactly nine times. However, the more times that the blocks are executed, the more accurate this prediction is.

Renaming test assets

As your test assets increase and become more complex, you might want to rename them. Use the Eclipse **Rename** function or save the assets under a different name.

Use either of the following steps to rename a test asset:

1. When you use the Eclipse **Rename** function, the new name is visible in the Test Navigator, but the underlying file system name is not changed. To use the Eclipse **Rename** function:
 - a. In the Test Navigator, right-click the test asset, and then select **Rename**
 - b. Type the new name, and then click **Enter**
Be sure to click **Enter**, or the file will not be renamed.
2. When you rename a test asset by saving it under another name, the underlying file system name is changed, but you must perform manual cleanup. To save a test asset under another name:
 - a. In the Test Navigator, browse to the test and double-click it. The test asset opens.
 - b. Click **File > Save As**, and save the asset under a different name.
 - c. Delete the original asset.

Example

The following table summarizes how renaming an asset affects the other assets in your workspace.

Renamed asset	Effect on other assets
Project	Do not rename a project. Renaming a project might result in lost or corrupted project assets.
Schedule	Renaming a schedule has no affect on other assets, but note that results cannot be renamed.
Test	When you use Rename , schedules that contain the old test name will still run correctly. To avoid confusion, manually update the schedule to use the new test name. After you use Save As , manually update each schedule that uses the renamed test.
Custom code	If you rename the custom code class (.java file), then the reference to the class in the custom code action of the test will not work. Typically rename the custom code

Renamed asset	Effect on other assets
	<p>class in the Resource perspective or the Java™ perspective.</p> <p>If you change the name of the custom code class in the test editor that implements the custom code action, the modification does not change the corresponding .java file; instead the modification causes the custom code action to refer to a different (and possibly new) custom code class.</p>
Dataset	<p>When you use Rename and open a test that contains the dataset, you are prompted to save the changes (in this case, the renamed dataset that the test now uses).</p> <p>After you use Save As, manually update each test that uses the dataset.</p>
Location	<p>When you use Rename, locations (agent computers) are automatically updated in the schedules that use them.</p> <p>When you use Save As, manually update each schedule that uses the test.</p>
Results	You cannot rename results.
Weighted block	Renaming a weighted block has no affect on other assets. To rename a weighted block, click the block in the test, and type the new name in the Name field.

Deleting test assets

As your test assets grow and become more complicated, you might want to delete the assets that you no longer use.

In the Test Navigator, right-click the test asset, and then select **Delete**.

Result

The following table summarizes how deleting an asset affects the other assets in your workspace.



Note: If you are deleting a test asset, you can choose to delete it from other test assets that refer it and you can choose to delete other test assets that are referenced only by the test asset that you are deleting.

For example, if you delete a dataset, the **Remove references to test asset name in other test assets** option lets you delete the dataset from all the test assets that uses it.

If you delete a test, the **Delete files that are referenced only by test asset name** option lets you delete all the test assets such as recession and dataset that are referenced only by the test that you are deleting. If the



dataset is used by another test too, it will not be deleted. The **Preview** button lets you see the assets that are referenced by the test.

Deleted asset	Effect on other assets
Project	<p>You are prompted whether to delete the project contents. If you click Yes, the contents are physically deleted. If you click No, you will not see the contents in the Test Navigator, but the project remains in your workspace, which is, by default, <code>C:\Documents and Settings\user-name__VENDOR_NAME____SDP_FULLL_SHORT-NAME__n.n/workspace</code>).</p>
Schedule	<p>Deleting a schedule has no effect on other assets.</p>
Test	<p>If you delete the test in the Test Navigator, the test is physically deleted.</p> <p>If you open a schedule and delete a test, the test is deleted from the schedule, but the test remains available as a test asset.</p>
Custom code	<p>If you delete the custom code class (.java file), then the reference to the class in the custom code action of the test will not work. Typically you delete the custom code class from the Resource perspective or the Java™ perspective.</p> <p>If you delete the name of the custom code class that implements the custom code action, the deletion does not change the corresponding .java file.</p>
Dataset	<p>If you delete a dataset in the Test Navigator, the dataset is physically deleted. When you open a test that uses the dataset, you are prompted to take one of these actions:</p> <ul style="list-style-type: none"> • Locate the dataset • Remove the dataset reference from the test • Leave the invalid reference in <p>You must correct or delete the reference to run the test successfully.</p>

Deleted asset	Effect on other assets
	If you open a test and delete the dataset from the Common Options tab, only the reference to the dataset is deleted.
Location	You are not asked to confirm the deletion, nor are you warned if a user group uses the location. The user group is marked with a red x when you open the schedule that contains it.
Results	You are asked to confirm the deletion, and the results are physically deleted.

Debugging custom code for tests and compound tests

If you have custom code added to a test or a compound test, you can debug the custom code for any errors by clicking the **Debug** button.

Before you begin

This procedure can only be done when custom code is part of a test or compound test. If there are multiple custom code classes added to a test or compound test, the debug action debugs all the custom code classes.

You can use the debug option only in full Eclipse mode of the product. You cannot debug in the streamline mode.

1. Open a test or a compound test from the Test Navigator view.
2. Click the **Debug** button.

Result

The Debug view opens. If there are any breakpoints in the custom code, the test run pauses at the breakpoint. Press F8 to resume the test run.

Providing tests with variable data (datasets)

You can produce more realistic tests by changing them to use datasets. During execution, a test that uses a dataset replaces a value in the recorded test with variable test data that is stored in the dataset. This substitution allows each virtual user to generate a different request to the server.

Dataset overview

Datasets provide tests with variable data during a run. When you record a test, you perform a sequence of steps that you expect a typical user to perform. After the recording, a test is generated that captures these interactions. When you run this test, it uses the same data that you used during recording. To vary the data in the test, you use a *dataset*, that contains variable data. At run time, this variable data is substituted for the data in the recorded test.

If you need to create a dataset with many records, you can initialize the dataset quickly by importing data from a comma-separated-value (CSV) file. Also, you can export test data from your dataset into a CSV file to enable you

to maintain large volumes of test data as a spreadsheet for reuse. Earlier to 9.5, the dataset (formerly known as datapool) was in .datapool format and starting from the 9.5 release, the dataset is in the csv format.

You can copy the CSV file and paste into your project to import the data from a CSV file and create a dataset. Similarly, to export the dataset values as a CSV file, you must copy the dataset from your project and paste it into your local machine.



Note: Alternatively, you can use the **Import** option available in the **CSV editor** to import the data from a CSV file. For more information, see [Editing datasets on page 420](#).

Perform the following steps should you plan to create a test that searches the HCL® website for three items: HCL OneTest™ Performance, HCL OneTest™ UI, and IBM® Rational® Manual Tester:

1. Record a test that searches for one item. See [Recording an HTTP test on page 183](#).
2. Create a dataset and associate it with the test. See [Creating a dataset associated with a test on page 406](#).
3. Associate a request in the test with a column in the dataset. See [Associating a test value with a dataset column on page 417](#).
4. Add a loop in the test to fetch the values from different rows of a dataset. A test without a loop fetches the value only from the first row of the dataset. See [Adding a loop to a test on page 387](#).

Creating a dataset associated with a test

You can create a dataset that contains variable data for tests to use when they run. This is the preferred way to create a dataset because the dataset is automatically associated with a test. You can create anything from an empty dataset that contains one column, which you can edit later, to a fully functioning dataset.

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. In the **Test Contents** area, click the name of the test.
3. In the **Common Options** tab, click **Add Dataset**.

The options listed in the following table, enable you to create anything from a simple dataset that you can edit later to a complete dataset.

To create	Do this in the Test Editor - Add Dataset window
A one-column dataset with a default access mode.	In Existing datasets in workspace , select <i>New Dataset<testname>.csv</i> , and click Finish . You can optionally name the dataset column in this session, and you can add other columns and data later.
A one-column dataset and choose the access mode.	In Existing datasets in workspace , select <i>New Dataset<testname>.csv</i> , and click Next . You can optionally name the dataset column in this session and


To create	Do this in the Test Editor - Add Dataset window
	you are prompted for the access mode. You can add other columns and data later.
An association between the test and an existing dataset.	Select the dataset. The dataset is associated with the test, and you can optionally set the access mode in this session.
A new, fully functioning dataset.	Select a project and click Use wizard to create new dataset .


4. Select **Open mode** for the dataset. This mode determines the view that virtual users have of the dataset. Different tests can open the same dataset differently, and you can change the open mode later by opening the test and double-clicking the dataset title.

Option	Description
Shared (per test execution) (default)	<p>When you choose the Shared (per test execution) option, the virtual users running in the test share the dataset values in sequential order.</p> <p>For example, if your dataset has 10 rows, the dataset values are taken from row 1, row 2, row 3, and so on when you select this option.</p>
Private	<p>Virtual users draw from a private view of the dataset, with dataset rows apportioned to each user in sequential order.</p> <p>This option ensures that each virtual user gets the same data from the dataset in the same order. However, because each user starts with the first row of the dataset and accesses the rows in order, different virtual users will use the same row. The next row of the dataset is used only if you add the test that is using the dataset in a loop with more than one iteration.</p>
Shared (for all test executions)	<p>When you choose the Shared (for all test executions) option, the virtual users running in multiple tests share the dataset values from the current row.</p> <p>For example, if your dataset has 10 rows and when you set the current row as row 5, the dataset values</p>

Option	Description
	are taken from row 5 instead of row 1 when you select this option. If you had set the current row as row 1 and used the dataset values until row 5, the dataset values are retrieved from row 6 when you run the test next time.

5. If you are setting how the test accesses the dataset during this session, select one of the following options:
- **Sequential:** The rows in the dataset are accessed in the order in which they are physically stored in the dataset file, beginning with the first row and ending with the last.
 - **Random:** The rows in the dataset are accessed in any order, and any given row can be accessed multiple times or not at all. Each row has an equal chance of being selected each time.
 - **Shuffled:** Before each dataset access, the order of the rows is changed, and a different sequence results. Rows are accessed randomly but all rows must be selected once before a row is selected again.
6. Select one of the following options.

Option	Description
Wrap when the last row is reached	<p>By default, when a test reaches the end of a dataset or dataset segment, it reuses the data from the beginning. To force a test to stop at the end of a dataset or segment, clear the check box Wrap when the last row is reached. Forcing a stop might be useful if, for example, a dataset contains 15 records, you run a test with 20 virtual users, and you do not want the last five users to reuse information. Although the test is marked as <i>"Fail"</i> because of the forced stop, the performance data in the test is still valid. However, if it does not matter to your application if data is reused, the default of wrapping is more convenient. With wrapping, you need not ensure that your dataset is large enough when you change the workload by adding more users or increasing the iteration count in a loop.</p> <p> Note:</p> <ul style="list-style-type: none"> ◦ With Random access order, Wrap when the last row is reached option is unavailable because you never reach the end of the row.

Option	Description
	 <ul style="list-style-type: none"> With Shuffled access order, if you select Wrap when the last row is reached option, you resume selecting from the beginning of the row with the same access order after each row has been selected once. No more selections are required if you clear the Wrap when the last row is reached option.
Fetch only once per user	<p>By default, one row is retrieved from the dataset for each execution of a test, and the data in the dataset row is available to the test only for the duration of the test. Select Fetch only once per user to specify that every access of the dataset from any test being run by a particular virtual user will always return the same row.</p>

Example

To illustrate how these options affect the rows that are returned, assume that a test contains a loop which accesses a dataset. The loop has two iterations. The following table shows the row that is accessed in each iteration:

Dataset option	Iteration 1	Iteration 2
Sequential and Private	row 1	row 2
Shared and Shuffled	row x	row y
Fetch only once per user	row x	row x

- If you are creating a fully functioning dataset, you can optionally import the data from a CSV file during this session by copying the CSV file and pasting into your project. For more information on importing dataset, see [Editing datasets on page 420](#).

What to do next

After you have created a dataset and added data to it, the next step is to associate a value in the test with a column in the dataset, as discussed in [Associating a test value with a dataset column on page 417](#).

Creating a dataset in a workspace

You can create datasets in a workspace containing variable data that tests use when they run. You can use this method to create a dataset if you have not yet created the test that will use it.

1. Click **File > New > Dataset**.
2. In the **New Dataset** window, click the project that contains the dataset. The project is displayed in the **Enter, create, or select the parent folder** field.
3. In the **Name** field, type the name of the dataset, and then click **Next**.
4. In the window for describing the dataset, add a description.
5. In the **Dimensions** field, specify the number of rows and columns for the dataset that you want to create.
6. Click **Finish**.

Results

The new dataset opens in a browser. For instructions on how to add data to or edit the dataset, see [teditdp_perf.dita on page 420](#).

What to do next

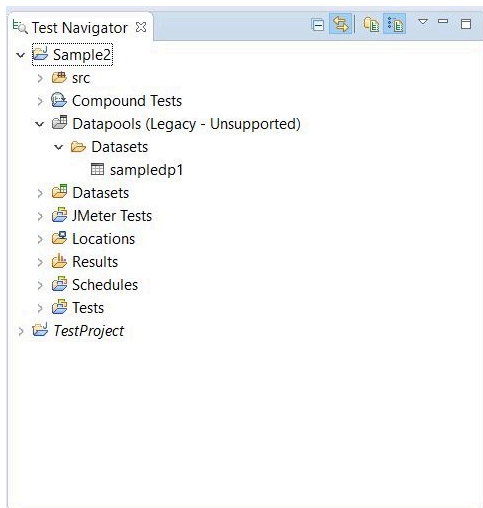
After you have created a dataset and added data to it, you must associate a value in the test with a column in the dataset.

Converting an existing datapool to a dataset

Starting from 9.5 the dataset formerly known as datapool is in the CSV format. You can convert any existing datapool to a dataset.

About this task

When you open the workspace earlier to 9.5 in HCL OneTest™ Performance 9.5, the existing datapools in the workspace are stored in the Datapools (Legacy-Unsupported) folder as shown in the following figure.



To convert the existing datapool to a dataset:

1. In the Test Navigator, browse and select the existing datapool.
2. Right-click and select **Convert to Dataset...**. Verify that the name of the dataset is the name of the existing datapool and format is .csv.
3. Click **Finish**. The converted datapool opens in a CSV editor.

What to do next

After you have created a dataset and added data to it, you must associate a value in the test with a column in the dataset.

Creating datasets with multiple substitutions

Earlier to 9.2, you could substitute one dataset value at a time. Starting from 9.2, after the test is generated, you can view all the dataset candidates, add multiple candidates as dataset values, substitute values, and create a new dataset out of it. You can also substitute multiple dataset candidates for an existing dataset.

About this task

When you substitute multiple dataset candidates to create a new dataset, the same number of columns are created in the dataset. The names of the candidates become the names of columns and values in the dataset. When you substitute multiple dataset candidates in an existing dataset, the column names in the dataset are retained. If the number of substitutions chosen was greater than the number of columns in the dataset, the extra number of substitutions are added as columns in the dataset. For instance, if a dataset has three columns and you substitute five dataset candidates, two new columns are created by using the names of the dataset candidates.

To create a dataset from multiple dataset candidates:


1. In the Test Editor, select the name of the test and from the Test Details section, select **Common Options** and click **Show Dataset Candidates**.
Alternative: After the test generation when you open the test, you are prompted that *"Some test data may need to be correlated or substituted"*. If you click **Yes**, you can see the list of dataset candidates.
2. Select the dataset candidates that you want to add as values to the dataset and click **Substitute multiple candidates**.
The **Add Dataset** dialog box shows the list of datasets that are in the project but not associated with the test.
3. To associate an existing dataset with the test and assign the selected dataset candidates as values and substitutions, select a dataset and click **Next**. To associate a new dataset with the test, click the **Use wizard to create new Dataset** and click **Next**.
4. Select **Open mode** for the dataset. This mode determines the view that virtual users have of the dataset. Different tests can open the same dataset differently, and you can change the open mode later by opening the test and double-clicking the dataset title.


Option	Description
<p>Shared (per test execution) (default)</p>	<p>When you choose the Shared (per test execution) option, the virtual users running in the test share the dataset values in sequential order.</p> <p>For example, if your dataset has 10 rows, the dataset values are taken from row 1, row 2, row 3, and so on when you select this option.</p>
<p>Private</p>	<p>Virtual users draw from a private view of the dataset, with dataset rows apportioned to each user in sequential order.</p> <p>This option ensures that each virtual user gets the same data from the dataset in the same order. However, because each user starts with the first row of the dataset and accesses the rows in order, different virtual users will use the same row. The next row of the dataset is used only if you add the test that is using the dataset in a loop with more than one iteration.</p>
<p>Shared (for all test executions)</p>	<p>When you choose the Shared (for all test executions) option, the virtual users running in multiple tests share the dataset values from the current row.</p> <p>For example, if your dataset has 10 rows and when you set the current row as row 5, the dataset values are taken from row 5 instead of row 1 when you select this option. If you had set the current row as row 1 and used the dataset values until row 5, the dataset values are retrieved from row 6 when you run the test next time.</p>

5. If you are setting how the test accesses the dataset during this session, select one of the following options:

- **Sequential:** The rows in the dataset are accessed in the order in which they are physically stored in the dataset file, beginning with the first row and ending with the last.
- **Random:** The rows in the dataset are accessed in any order, and any given row can be accessed multiple times or not at all. Each row has an equal chance of being selected each time.
- **Shuffled:** Before each dataset access, the order of the rows is changed, and a different sequence results. Rows are accessed randomly but all rows must be selected once before a row is selected again.

6. Select one of the following options.

Option	Description
<p>Wrap when the last row is reached</p>	<p>By default, when a test reaches the end of a dataset or dataset segment, it reuses the data from the beginning. To force a test to stop at the end of a dataset or segment, clear the check box Wrap when the last row is reached. Forcing a stop might be useful if, for example, a dataset contains 15 records, you run a test with 20 virtual users, and you do not want the last five users to reuse information. Although the test is marked as <i>Fail</i> because of the forced stop, the performance data in the test is still valid. However, if it does not matter to your application if data is reused, the default of wrapping is more convenient. With wrapping, you need not ensure that your dataset is large enough when you change the workload by adding more users or increasing the iteration count in a loop.</p> <p> Note:</p> <ul style="list-style-type: none"> ◦ With Random access order, Wrap when the last row is reached option is unavailable because you never reach the end of the row. ◦ With Shuffled access order, if you select Wrap when the last row is reached option, you resume selecting from the beginning of the row with the same access order after each row has been selected once. No more se-

Option	Description
	 ections are required if you clear the Wrap when the last row is reached option.
Fetch only once per user	By default, one row is retrieved from the dataset for each execution of a test, and the data in the dataset row is available to the test only for the duration of the test. Select Fetch only once per user to specify that every access of the dataset from any test being run by a particular virtual user will always return the same row.

Example

To illustrate how these options affect the rows that are returned, assume that a test contains a loop which accesses a dataset. The loop has two iterations. The following table shows the row that is accessed in each iteration:

Dataset option	Iteration 1	Iteration 2
Sequential and Private	row 1	row 2
Shared and Shuffled	row x	row y
Fetch only once per user	row x	row x

7. Click **Finish**.

How dataset options affect values that a virtual user retrieves

The Open, Access, and Wrap modes that you select for a dataset affect the values that a virtual user retrieves.

The following table lists the most common types of datasets and the options that you select to create them.

Dataset purpose	Access mode		
	Open mode selection	Wrap mode selection	Access mode selection
The virtual user retrieves the value from the current row of the dataset in a random order for every attempted transaction. Note that before accessing each row of the dataset the order of the rows is rearranged.	Shared (for all test executions)	Shuffled	Fetch only once per user

Dataset purpose	Access		
	Open mode selection	Access mode selection	Wrap mode selection
The virtual user retrieves the value from the current row of the dataset in sequential order for every attempted transaction.	Shared (for all test executions)	Sequential	Fetch only once per user
The virtual user retrieves the value from the beginning of the row of a dataset in a random order for every attempted transaction.	Shared (per test execution)	Random	Wrap when the last row is reached
The virtual user retrieves the value from the current row of a dataset in sequential order for every attempted transaction. When a test reaches the end of a dataset, it reuses the data from the current row selection of the dataset.	Shared (for all test executions)	Sequential	Wrap when the last row is reached

Enabling a test to use a dataset

Before a test can use variable data from a dataset, you must update the test to include a reference to that dataset.

1. In the Test Navigator, browse to the test and double-click it. The test opens.
2. Right-click the test name, and click **Add > Dataset**.

Result

The **Select Dataset File** window is displayed listing the datasets available to the test. If a test is already using a dataset, it does not appear in the list.

3. In the **Existing Dataset in workspace** list, click the name of the dataset that your test will use and click **Next**.
4. Select the **Open mode** for the dataset. This mode determines the view that virtual users have of the dataset. This option is useful when you do a parallel test run.

Option	Description
Shared (per test execution) (default)	<p>When you choose the Shared (per test execution) option, the virtual users running in the test share the dataset values in sequential order.</p> <p>For example, if your dataset has 10 rows, the dataset values are taken from row 1, row 2, row 3, and so on when you select this option.</p>

Option	Description
Private	<p>Virtual users draw from a private view of the dataset, with dataset rows apportioned to each user in sequential order.</p> <p>This option ensures that each virtual user gets the same data from the dataset in the same order. However, because each user starts with the first row of the dataset and accesses the rows in order, different virtual users will use the same row. The next row of the dataset is used only if you add the test that is using the dataset in a loop with more than one iteration.</p>
Shared (for all test executions)	<p>When you choose the Shared (for all test executions) option, the virtual users running in multiple tests share the dataset values from the current row.</p> <p>For example, if your dataset has 10 rows and when you set the current row as row 5, the dataset values are taken from row 5 instead of row 1 when you select this option. If you had set the current row as row 1 and used the dataset values until row 5, the dataset values are retrieved from row 6 when you run the test next time.</p>

5. Select the **Access mode** for the dataset:

- **Sequential:** The rows in the dataset are accessed in the order in which they are physically stored in the dataset file, beginning with the first row and ending with the last.
- **Random:** The rows in the dataset are accessed in any order, and any given row can be accessed multiple times or not at all. Each row has an equal chance of being selected each time.
- **Shuffled:** Before each dataset access, the order of the rows is changed, and a different sequence results. Rows are accessed randomly but all rows must be selected once before a row is selected again.

6. Select whether the test will reuse data when it reaches the end of the dataset.

By default, when a test reaches the end of a dataset or dataset segment, it reuses the data from the beginning. To force a test to stop at the end of a dataset or segment, clear the check box **Wrap when the last row is reached**. Forcing a stop might be useful if, for example, a dataset contains 15 records, you run a test with 20 virtual users, and you do not want the last five users to reuse information. Although the test is marked as *Fail* because of the forced stop, the performance data in the test is still valid. However, if it does not matter to your application if data is reused, the default of wrapping is more convenient. With wrapping,

you need not ensure that your dataset is large enough when you change the workload by adding more users or increasing the iteration count in a loop.

7. Select whether the test will make the data in the dataset record permanent for each virtual user.

By default, one row is retrieved from the dataset for each execution of a test, and the data in the dataset row is available to the test only for the duration of the test. Select **Fetch only once per user** to specify that every access of the dataset from any test being run by a particular virtual user will always return the same row.

To illustrate how these options affect the rows that are returned, assume that a test contains a loop which accesses a dataset. The loop has two iterations. The following table shows the row that is accessed in each iteration:

Dataset option	Iteration 1	Iteration 2
Sequential and Private	row 1	row 2
Shared and Shuffled	row x	row y
Fetch only once per user	row x	row x

8. Click **Finish**.

Result

A reference to the dataset is added to the test, and the **Test Details** area is updated with the dataset information.

9. Save the test.

What to do next

Now that you have created a reference between the test and the dataset, the next step is to associate a value in the test with a column in the dataset.

Associating a test value with a dataset column

After you have created a dataset and have enabled your test to use the dataset, you can associate a specific value in the test with a specific dataset column.

1. In the Test Navigator, browse to the test and double-click it. The test opens.
2. Locate and click a request that contains a value to replace with variable data.

Clicking a test page displays a table that lists dataset candidates and correlated data on that page. (If correlated data is not displayed, right-click the table and verify that **Show References** is selected.) References are color coded in blue and dataset candidates are color coded in black.

Test Data

Name	Value	Substituted with
searchFor	doe%2C+john	

If the contents of the Value column corresponds exactly with column data in your dataset, click the row, and then click **Substitute**. The **Select Data Source** window is displayed. Skip to step 6. You can ignore step 8 because the URL encoding is preselected.

Otherwise, double-click the row to navigate to the page request that contains the value to replace from a dataset, and continue to the next step.

The value to replace from a dataset might not be listed in any page table. In this case, manually locate the request string that includes the value.

- If the value to replace from a dataset is part of a string that has been designated a dataset candidate, you must remove the light green highlight: right-click and select **Remove Substitution**.
For example, if you searched for **doe, john** in your test, the dataset candidate in your test is displayed as **doe%2C+john**. Suppose that you do not want to associate this candidate with a single dataset column that contains data in the format **doe, john**. Instead, you want to associate **doe** and **john** with separate dataset columns. In this case, you must first remove the substitution.
- Highlight the value: With the left button pressed, drag your mouse over the value.
- Right-click the highlighted value, and select **Substitute > Select Data Source**.

Result

The **Select Data Source** window is displayed.





Note: To use a dataset that is not listed, click **Dataset**: the **Select dataset column** window is displayed.

- Click the name of the dataset variable, or column, to associate with the test value.
- Click **Select**.

Result

To indicate that the association has been set, the highlighting for the selected test value turns dark green, and the dataset table for this page is updated as shown in the example.

Test Data

Name	Value	Substituted with
doe%2C+	john	 "firstname" variabl...
searchFor	doe	 "lastname" variabl...

- Optional:** Encode variable data when it is substituted from a dataset.

If a test value contains special characters such as spaces or commas, click the row and select **URL Encode**. With this option, special characters are encoded when variable data is substituted from a dataset. For example, data that is space-separated in a dataset column might need to be encoded. When the URL encoding is enabled, **John Doe** is substituted as **John%20Doe**. If the URL encoding is not selected, the variable data that is substituted is literal. Do not enable URL encoding for datasets that contain data that is already encoded.

- Optional:** If you substitute an element of a page with a dataset column, to view the substitutions in the Page Elements report, in the Test Elements Details area of the request click the **Use the substituted URL in performance reports** check box.
- Save the test.

Related information

[Adding data source controller on page 392](#)

Viewing dataset candidates when you open a test

Dataset candidates are displayed automatically when you open a test for the first time. From the dataset candidates window you can view the dataset candidates in the test, bookmark locations of interest, and add or remove dataset references.

- Record a test.

Result

When the test opens for the first time in the Test Navigator, the **Show Dataset Candidates** window is displayed. The **Show Dataset Candidates** window is displayed only if there are dataset candidates and if **Always display this dialog when a test is first opened** is selected. To prevent the **Show Dataset Candidates** from being displayed when a test opens, clear the **Always display this dialog when a test is first opened** check box in the **Show Dataset Candidates** window.

- Do one of the following:

Option	Description
<p>To view details about the dataset candidates in a test</p>	<p>Navigate through the Dataset Candidates field to see them previewed in the Preview pane. Click the Next and Previous icons to move the selection down or up in the list of dataset candidates. Click the Show as Tree icon to toggle between tree format and list format. Click the Sort icon to sort the list of dataset candidates. Click the Bookmark icon to bookmark a location for later review.</p>
<p>To select a data source for a dataset candidate</p>	<p>Select the dataset candidate in the Dataset Candidates field, and then click Substitute. The Select Data Source window opens.</p>
<p>To find more values in the test that have the same value as the selected dataset candidate</p>	<p>Click Find More and Substitute. These values can be reviewed and substituted interactively as needed.</p>
<p>To remove a substitution</p>	<p>Select a substitution site, and then click Remove Substitution.</p>

- Click **Close** to close the **Show Dataset Candidates** window and proceed to the test in the test editor.
To display the **Show Dataset Candidates** window again while in the test editor, click the root node of the test. Then click the **Common Options** tab under **Test Element Details**, and then click **Show Dataset Candidates**.

Editing datasets

You can add, modify, or remove data from a dataset by using the CSV Editor. The working principle of the CSV Editor is similar to that of a spreadsheet.

Before you begin

You must have created a dataset. See [Creating a dataset in a workspace on page 410](#).

About this task

In HCL OneTest™ Performance V9.5.0 or later, you can use the CSV Editor to view and edit data in the dataset.

To have seamless access to a dataset CSV editor use any one of the following web browsers on Windows or Linux or Mac operating systems:

- Mozilla Firefox
- Google Chrome
- Microsoft Edge based on Chromium

You can also view the datasets in other editors by right-clicking the dataset and selecting the **Open With** option.

When you want to run a test asset with different dataset values, you can either edit the existing dataset or create a new dataset to use it during the test asset run. When the number of edits is minimal, it is easier to edit the dataset within the CSV Editor.

You can perform basic tasks in the CSV Editor by right-clicking any row, column, or cell of the dataset to organize your data in a better way. For example, updating the data in a cell, inserting or deleting rows and columns, or renaming column names.

When you edit the dataset in a CSV Editor, you can use the following keyboard shortcuts to control the cursor selection in the CSV Editor:

- **Tab** - To move the cursor control to the next available option.
- **Shift-Tab** – To move the cursor control to the previous option.
- **Shift+F10** – To open the context menu from the dataset cell.



Note:

You cannot resize the width of rows in the CSV Editor. When you have a large amount of data in a cell, you can right-click the cell and select **Copy** (or Ctrl+C), and then paste it into a text-editing program to view the content. Alternatively, you can hover the mouse over the cell to view the content.



1. Double-click the dataset that you want to edit in the **Test Navigator**.







Result








The dataset opens in the CSV Editor in a browser.



2. Perform the following actions to use the options available in the CSV Editor:


Option	Description
Set as current row	<p>Right-click any cell in a row and select Set as current row.</p> <p>When rows are deleted:</p> <p>If you delete any row between row 1 to current row, the current row data is taken from the next row.</p> <p>For example, when you set the current row as 6, and then you delete any row between row 1 to row 6, the current row remains at row 6, but the content of row 7 is moved to row 6.</p> <p>When rows are inserted:</p>

Option	Description
	<p>If you insert any new row between row 1 to the current row, the current row data is taken from the previous row.</p> <p>For example, when you set the current row as 6, and then you insert any row between row 1 to row 6, the current row remains at row 6, but the content of row 5 is moved to row 6.</p>
<p>Find and Replace </p>	<p>To find:</p> <ol style="list-style-type: none"> Click the Find and Replace icon . Enter the content that you want to search in the Find field. Select any or all the following options to find the search content more effectively: <ul style="list-style-type: none"> Select the Case sensitive check box to search the content that is the exact letter case of the content entered in the Find field. Select the Match entire cell contents check box to search for cells that contain only the characters that you have entered in the Find field. Select the Search using regular expression check box to search the pattern that matches strings. <p>For example, to search a cell that contains any number between 0 to 9, do the following:</p> <ol style="list-style-type: none"> Enter <code>\d</code> in the Find field. Select the Search using regular expression check box. Click Find. Click Find. If the text is found, the cell containing that text is selected. Click Find again to find further instances of the search text.

Option	Description
<p data-bbox="435 310 618 338">Find and Replace </p>	<p data-bbox="857 268 1068 296">To find and replace:</p> <ol data-bbox="911 352 1425 1310" style="list-style-type: none"> <li data-bbox="911 352 1344 380">Click the Find and Replace icon . <li data-bbox="911 394 1398 464">Enter the content that you want to search in the Find field. <li data-bbox="911 478 1406 548">Enter the content that you want to replace in the Replace field. <li data-bbox="911 562 1425 1150">Select any or all the following options to find and replace the content more effectively: <ul style="list-style-type: none"> <li data-bbox="1003 638 1425 785">▪ Select the Case sensitive check box to find the content that is the exact letter case of the content entered in the Find field. <li data-bbox="1003 800 1425 989">▪ Select the Match entire cell contents check box to find and replace for cells that contain only the characters that you have entered in the Find and Replace fields. <li data-bbox="1003 1003 1365 1150">▪ Select the Search using regular expression check box to find and replace the pattern that matches strings. <li data-bbox="911 1165 1373 1234">Click Replace to replace the individual instances. <li data-bbox="911 1249 1417 1310">Click Replace All to replace every instance of the content throughout the dataset.
<p data-bbox="508 1360 607 1388">Undo </p>	<ol data-bbox="911 1360 1398 1465" style="list-style-type: none"> <li data-bbox="911 1360 1182 1388">Click the Undo icon . <li data-bbox="911 1402 1398 1465">Select the recent changes from the list that you want to undo, and then click the list. <p data-bbox="857 1507 1425 1654">The Undo option undoes anything you do in the dataset. The CSV Editor saves the unlimited undo-able action. You can perform the undo action even after you save your changes made to the dataset.</p>
<p data-bbox="505 1711 607 1738">Redo </p>	<ol data-bbox="911 1711 1398 1816" style="list-style-type: none"> <li data-bbox="911 1711 1190 1738">Click the Redo icon . <li data-bbox="911 1753 1398 1816">Select the recent changes from the list that you want to redo, and then click the list.

Option	Description
<p data-bbox="496 359 613 390">Import </p>	<p data-bbox="857 264 1360 289">The CSV Editor saves the unlimited redo action.</p> <ol data-bbox="911 359 1414 510" style="list-style-type: none"> Click the Import icon . Click Choose File and select the CSV file that you want to import in the Import File dialog box. <p data-bbox="951 558 1068 600"> Note:</p> <p data-bbox="1008 632 1406 856">If the CSV file contains test data with Unicode characters in it, you must save the CSV file in <code>UTF-8</code> format. You can then choose the CSV file and import the test data from the CSV file into the dataset.</p> <ol data-bbox="911 930 1422 1276" style="list-style-type: none"> Optional. Click Overwrite to add the rows and columns from the selected CSV file from the beginning of the dataset. Optional. Click Append to add rows and columns from the selected CSV file to the end of the dataset. Optional. Select the First row contains headers check box if your CSV file contains the header.
<p data-bbox="496 1339 613 1371">Export </p>	<p data-bbox="857 1339 1398 1409">Click the Export icon  to download the dataset as a CSV file.</p>
<p data-bbox="370 1461 743 1493">Dataset configuration settings </p>	<p data-bbox="857 1440 1414 1587">In the Configure Dataset window, you can change the row and column settings and configure the string values in the dataset that contains variable data for tests to use when they run.</p> <ol data-bbox="911 1629 1422 1776" style="list-style-type: none"> Click the Menu icon  and select Configure option. Configure the following options in the Configure Dataset window:

Option	Description
	<ul style="list-style-type: none"> ▪ Column header - To change the column header, use an up-down control button to increment or decrement the value. ▪ Data start point - To change the data starting pointer, use an up-down control button to increment or decrement the value. ▪ Current row - To set the current row, use an up-down control button to increment or decrement the value. You can also see the Current row number on the upper-right side. ▪ Treat empty text as null -If a dataset contains any blank cells, the value of those blank cells is interpreted as null when you select this field. ▪ Treat as null - Enter a string value that is to be treated as null when running test. ▪ Treat as empty - Enter a string value that is to be treated as empty when running test.
<p style="text-align: center;">Discard </p>	<p>Click the Menu icon  and select Discard to discard the changes made to the dataset.</p>

3. Click the **Save** icon  to save the changes made to the dataset and close the CSV editor.

Encrypted datasets overview

You can encrypt one or more columns in a dataset. If you want to encrypt confidential information such as a set of passwords or account numbers that are used during a test, you can use an encrypted dataset.

Dataset columns are encrypted using the RC4 private-key algorithm. You can use only one password to encrypt columns in any given dataset. Encrypted datasets are not supported on agent computers that are running the z/OS® or AIX® operating systems.



Important: If you forget the password to a dataset, there is no way to recover the password.

When you run a test that uses a dataset that contains encrypted variables, you are prompted for the dataset password. If the test uses multiple encrypted datasets, you must enter the password for every encrypted dataset that the test uses.

When you run a test that uses a dataset with an encrypted column, the value of the column is decrypted at a run time. The column value is sent as a cleartext string in the requests to the server. The actual values of the encrypted dataset variables are not displayed in the test log. The test log displays asterisks for the encrypted dataset variables.

To see the actual values of variables that are sent to the server at run time, you must use custom code. You can send the dataset column value to custom code that writes the value to a file other than the test log. If the custom code writes to the test log using `tes.getTestLogManager().reportMessage()`, then asterisks are displayed instead of the decrypted variables.

Encrypting a dataset column

To secure test data, you must encrypt datasets. You can encrypt data in the columns of a dataset by using an encryption key. When you run a test that utilizes a dataset with encrypted variables, you must enter the encryption key for the encrypted column that the test uses.

Before you begin

You must have created a dataset. See [Creating a dataset in a workspace on page 410](#).

1. Double-click the dataset in the Test Navigator.

Result

The dataset is displayed in a browser.

2. Right-click any cell in a column that you want to encrypt and select **Encrypt column data**.

Result

The **Encrypt Column** window is displayed.

3. Enter an encryption key in the **Encryption Key** field to encrypt the data in the column.



Remember: When you have already encrypted other columns in the dataset, you must enter the same encryption key that you used previously. You can use only one encryption key to encrypt columns in a dataset.



Important: The encryption keys you use to encrypt data in a dataset are not stored on the server nor can be retrieved from the server. Therefore, you must remember to store the encryption keys in a secure location. You must use the same encryption keys to view the encrypted values, to decrypt data, or enable the use of the encrypted dataset during test runs.

4. Click **Encrypt Column**.

Result

Asterisks are displayed instead of actual data for the encrypted column.

Results

The dataset column is encrypted.

Decrypting a dataset column

To view the content of an encrypted dataset, you can decrypt the dataset. Removing encryption from a dataset revokes the protection offered to the test data.

Before you begin

You must have created a dataset with at least one encrypted column. See [Creating a dataset in a workspace on page 410](#) and [Encrypting a dataset column on page 426](#).

1. Double-click the dataset in the Test Navigator.

Result

The dataset is displayed in a browser.

2. Right-click encrypted cells that display the contents with asterisks, and then select **Decrypt column data**.

Result

The **Decrypt Column** window is displayed.

3. Enter the encryption key that you used to encrypt the data in the column in the **Encryption Key** field.
4. Click **Decrypt Column**.

Result

Asterisks are replaced with the actual data in the decrypted column.

Results

The encryption is now removed from the selected column in the dataset. When you run a test that uses a dataset that contains decrypted data, the variable data is substituted for the data in the recorded test without prompting for the encryption key.

Using a digital certificate store with a dataset

You can associate the certificates in one or more certificate stores with a dataset to use multiple digital certificates during testing.

1. Open a test for editing. On the **Common Options** page, click **Add Dataset**.
2. Create a dataset with two columns that contains a list of the certificates in the certificate store and a list of passphrases for the certificates. To learn more about adding datasets, see [Creating a dataset in a workspace on page 410](#). You can use the supplied KeyTool program to generate a list of names of certificates in a certificate store. To learn more about KeyTool, see [Creating a digital certificate store on page](#) .
3. Select **Fetch only once per user**.
4. Save the dataset.
5. On the **Security** page, under Digital Certificates, click **Add**.

6. Select a certificate from the certificate store that you created previously.
7. Type the passphrase for the selected certificate.
8. When prompted to dataset the digital certificate, click **Yes**.
9. In the **Select dataset column** wizard, choose the dataset that you added previously, and substitute the appropriate columns for the certificate name and passphrase.
10. Save the test, and then add the test to a schedule.


Results

When you run this schedule, the certificates from the certificate store are submitted to the server.

Navigating between a dataset and a test

After you have created a dataset or imported a comma-separated values (CSV) file into a dataset, you can navigate between the dataset and associated tests in the test editor. You can enlarge the test and the dataset, list the datasets that a test uses, navigate from a row in a dataset to the corresponding element in the test, see the data for a page or request, and add or remove dataset references.

1. In the Test Navigator, browse to the test and double-click it. The test opens.
2. Do one of the following actions:

Option	Description
<p>Maximize the test window</p>	<p>Double-click the test tab (for example, ). Do not click the x, or you will close the test. To return to the default perspective, click Window > Reset Perspective.</p>
<p>View the datasets that a test uses</p>	<p>In the Test Contents area, click the first line of the test, which is the test name.</p>
<p>Navigate from a row in a dataset to its corresponding element</p>	<ol style="list-style-type: none"> a. In the Test Contents area, click the test name, which displays the dataset. b. Expand the dataset to display the rows. c. Double-click the row.
<p>View the data for a page or request</p>	<p>In the Test Contents area, click the page or request.</p>
<p>To add a reference to a dataset</p>	<p>In the Test Element Details area, drag your cursor over the candidate, right-click, and select Substitute > Select Data Source. The Select Data Source window opens. If you have not already added the dataset to the test, click Dataset, and then add the new dataset.</p>

Option	Description
Remove a reference to a dataset	In the Test Element Details area, drag your cursor over the reference, right-click, and select Remove Substitution .

3. Save the test, if you have made any changes.

Test variables

A test variable is a user-defined, name-value pair that stores and refers to information throughout a test and between tests.

A variable is declared in the test variables section of the test. You can use it throughout the test as a reference for any field that can be substituted. Substituting data from a test variable is achieved using the **Test Variables** page of the Test Data Source view. You can do the following actions to a test variable:

- Provide a default value to the variable during declaration.
- Change the value of the variable using Set Variable statement. You can use the Add and Insert menus of the Test Editor to create Set Variable statements.
- Set hard-coded value or value retrieved from a data source, such as dataset or reference that appears before the Set statement to the variable.

If a variable is initialized at various places such as test, compound test, schedule, or user group, the product uses the following order to initialize the value of the variable when running the test. The variable set in the variable table of the compound test editor takes the highest precedence followed by others:

1. Compound test setting in the variable table UI
2. Compound test specified in a var file
3. User group setting in the variable table UI
4. User group specified in a var file
5. Schedule specified setting in the variable table UI
6. Schedule specified in a var file
7. Command line



Note: You must select **All tests for this user** from the **Visible in** drop-down list to take the precedence of variable initialization.

Sharing variables among tests

In order to share variables between tests, all the tests must contain the variables with the same name. The variables must also have **Visible in** set to **All tests for this user**. When these conditions are met and multiple tests have been placed in a schedule, then variable in the dataset of one test can be used in the other test.

A common reason to share data between tests is to perform data correlation. With data correlation, a variable is set to a response that comes from a request in one test and is used in requests performed in a different test. Assume that you are testing an employee database. The Create Employee test creates an employee record and the Modify Employee test modifies an employee record. When a new record is created, it is assigned a record ID. Variables can be used to pass the record ID from a response in the Create Employee test to the Modify Employee test. A user-defined variable is not shared among different virtual users. The variable is shared only among the different tests of the same virtual user. If you set **Visible in to This test only**, then dataset from a test is not available to another even if both tests contain variables with the same name.

If you want to share variables between the different types of test scripts in your product, consider the following points:

- Declare the test variables with the same name across all the test scripts for the variables to communicate with each other. Set **Visible in to All tests for this user**.
- Include the required test scripts into a compound test.

Using test steps, you can share the default values of the variables to another test script. You can also assign new values to the variables and use the latest values in another test script.

If you want to share variables between the test scripts of different testing products such as HCL OneTest™ UI, HCL OneTest™ Performance, or HCL OneTest™ API, you must consider the following points:

- If you are using Installation Manager, you must shell-share or install the products in the same package group.
- If you are using InstallAnywhere, you must use the installer that installs both HCL OneTest™ UI and HCL OneTest™ Performance together.



Note: By using InstallAnywhere, you cannot shell-share your product with HCL OneTest™ API.

- Declare the test variables with the same name across the HCL OneTest™ UI and HCL OneTest™ Performance test scripts. Set **Visible in to All tests for this user**.
- Include the required test scripts into a compound test.
- If you are using HCL OneTest™ API test scripts, you must map your tags with the test variables of HCL OneTest™ UI or HCL OneTest™ Performance.

Using variables to access datasets

You can define variables so that they share data from a dataset throughout tests. This is achieved by having the value field of a Set Variable statement substituted from a dataset. By doing so, the first test which appears in the schedule can set the variable from a dataset and share it with the other test in a schedule.

Assume that you have two tests that log in to an application using a user ID from a dataset. The first test can set the value of a variable from the dataset, and both tests can use the variable, instead of directly using the dataset. In this case both use the same record from the dataset. This is similar to the fetch-only-once-per-user behavior of a dataset. However, fetching once means that during playback, a virtual user gets only one record from the dataset. The one-record limit holds even if the tests are in a loop, and are run several times by the virtual user. By using the user-defined variables, the virtual user retrieves a new record each time through the loop, and both tests can use the same record.



Note: Assignment (set) operators can not only have a variable value substituted from a dataset, but also in the declaration of a variable. You can substitute the assignment operator and variable value from any data source, and thus that value can be shared between tests as well.

Array variables

You create an array variable to add multiple values to a variable. If you create a secondary HTTP request, add complete paths of the requests in the array variable that can be used a custom code during playback.

Declaring and assigning test variables

When you declare a variable, you can create a container for it, initialize it to a string or a dataset value, and set its scope. Then, within the test, you can reassign another value to the variable.

About this task

If the data that you want to assign to a variable is only available after a specific test step, instead of initializing the variable, you need to add a variable assignment further down in the test, so that when the assignment occurs, the data that you need to use is available. Otherwise, when you try to initialize the variable (or do the assignment), the value that you want to use will not be available and will not show up as an option to select.

If a variable is initialized at various places such as test, compound test, schedule, or user group, the product uses the following order to initialize the value of the variable when running the test. The variable set in the variable table of the compound test editor takes the highest precedence followed by others:

1. Compound test setting in the variable table UI
2. Compound test specified in a var file
3. User group setting in the variable table UI
4. User group specified in a var file
5. Schedule specified setting in the variable table UI
6. Schedule specified in a var file
7. Command line



Note: You must select **All tests for this user** from the **Visible in** drop-down list to take the precedence of variable initialization.

To create, initialize, and assign a value to a test variable:

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. To create a container for the test variables that you create in a test:

- a. Open the test, and in the **Test Contents** area, click **Test Variables**.

- b. Select **Add > Test Variable Container**.

Result

A container named **Test Variables** is created for the user-defined variables.

- c. Select the container to rename it.

Result

The **Test Element Details** area opens for you to type a new name in the **Name** field.

3. To declare or define a test variable:

- a. Open the test, and in the **Test Contents** section, click the user-defined container to contain the variable.

- b. To create a variable, select **Add > Variable Declaration**. To create an array variable, select **Add > Array Variable Declaration**.

- c. Type the name of the variable, and click **OK**.

Result

The variable is added as the last element in the container and the **Test Element Details** area opens.

- d. In the **Test Element Details** area, set the scope and initial value for the variable.

Visible in: Select **This test only** to restrict data to the current test only. Even if another test has a variable with the same name, that variable will not change. Select **All tests for this user** to share the value of this variable when the test runs in a schedule. For the variable to be shared, both tests must have a variable with the same name and must have this option enabled.

Check Value: Select **When first used** to check whether or not a variable is initialized only after the test execution reaches the first request that uses a variable. Select **At test start** to check whether or not a variable is initialized when starting the execution of the test. If the variable is not initialized, then an error message is displayed, depending on the behavior set.

If not initialized, set to: Select **Text** to initialize the variable to a specific value whenever the test runs in the schedule. Select **Dataset value** and, in the **Select Data Source** window, select the dataset that will initialize the variable.

Run-time error if variable not initialized: Select the action for the run when it encounters an uninitialized test variable. If you select **Issue test log warning** or **Issue test log error**, verify that the

Test log page in the schedule sets errors, failures, and warnings to **All**, which is the default setting. If you select **Exit the test**, the schedule continues to run although the virtual users that have the uninitialized variable stop. If you select **Do nothing**, the test continues to run.

4. To assign or initial a value to a test variable:

a. Open the test, and in the **Test Contents** area, select a test element.

b. Select **Insert > Variable Assignment**, which inserts the assignment before the selected element.

Result

The **Test Editor** window opens and lists the variables available to the test.

c. Select the variable that you are assigning a value to and, in the **Set to** box in the **Test Element Details** area, set the value for the variable.

You can set the value to a text string, to any data source that exists in the test before the assignment statement, or to **Not initialized**.

Result

A `set` statement is added to the test, with the value you chose.

Initializing variables from the command line

To initialize test variables from an XML file, you can run the test from the command-line interface using the `varfile` option.

Before you begin

- Declare the variables using HCL OneTest™ Performance.
- You must have set the **Web Reports** preferences in the desktop client to view the status of the test run from the command line. See [Accessing reports remotely on page 800](#).
- Create an XML file that contains the variables with values. The XML file would have a structure similar to the following image

```
<?xml version="1.0" encoding="UTF-8"?>
<inits>
  <variable_init value="9.999.99.999" name="hostname"/>
</inits>
```

About this task

If a variable is initialized at various places such as test, compound test, schedule, or user group, the product uses the following order to initialize the value of the variable when running the test. The variable set in the variable table of the compound test editor takes the highest precedence followed by others:

1. Compound test setting in the variable table UI
2. Compound test specified in a var file

3. User group setting in the variable table UI
4. User group specified in a var file
5. Schedule specified setting in the variable table UI
6. Schedule specified in a var file
7. Command line



Note: You must select **All tests for this user** from the **Visible in** drop-down list to take the precedence of variable initialization.

1. Navigate to the directory that contains the `cmdline.bat` and `cmdline.sh` files. On Windows™ operating systems, this directory is typically found as `productInstallationDirectory/cmdline`. For example, `C:\Program Files\HCL\HCLOneTest\cmdline`.
2. Issue the following command:

Example

```
cmdline -workspace workspace_full_path -project proj_rel_path -eclipsehome eclipse_full_path
-plugins plugin_full_path -schedule sched_rel_path -suite suite_rel_path -importzip file_full_path.zip
-varfile variable_file_full_path -servicename service -serviceargs service_args -configfile
file_full_path -results result_file -overwrite {"true" | "false"} -quiet -users nn -vmargs JVM_args
-rate RateRunnerGroupName=iterationNumber/duration, iterationNumber/duration-duration
Stage1=durationOfStage; Stage2=durationOfStage -publish serverURL#project.name=projectName
-publish_for {ALL,PASS,FAIL,ERROR, INCONCLUSIVE} -exportlog log_full_path -exportstats local_dir_path
-timerange "all, 5 Users, 10 Users" -exportstatshtml local_dir_path -compare "result_path1, result_path2"
-exportstatreportlist stats_list -execsummary local_dir_path -execsummaryreport reportID -usercomments
"any user comment" -publishreports "FUNCTIONAL, MOBILE_WEBUI, STATS, TESTLOG" -stdout -swapdatsets
existing_dataset_file_path:new_dataset_file_path -history jaeger,testlog,null -overridermlabels "label name
1,label name 2"
```




Note:




- The workspace is locked after you issue the command. To check the progress of the test or schedule during the run, invoke another workspace and open the project through that workspace.
- On Linux operating system, the command must start with `cmdline.sh`.
- The command line does not provide a way to specify the secure storage password for resource monitoring. You must provide the password in the workbench and ensure that it is stored and persisted in the schedule before you execute the schedule from the command line.



If a value contains spaces, enclose the value in quotation marks. To see the online help for this command while you are in the directory that contains the `.bat` file, type `cmdline -help`.


The following table explains each option:


-work-space	Required. The complete path to the Eclipse workspace.
-project	Required. The path, including the file name of the project relative to the workspace.
-eclipse-home	Optional. The complete path to the directory that contains <code>eclipse.exe</code> .
-plugins	Optional. The complete path to the folder that contains the plugins. Typically, on Windows operating systems, this folder is located at <code>C:\Program Files\HCL\HCLIMShared\plugins</code> . Required. This option is required only if the folder is at a different location.
-schedule	Optional. However, in a command, it is mandatory to use one of the following options: <ul style="list-style-type: none"> ◦ -suite ◦ -schedule ◦ -servicename <p>You must not use the -schedule option along with the other options. The path includes the file name of the schedule to run relative to the project.</p> <p>Starting from V9.2.1.1, you can execute multiple schedules simultaneously.</p> <p>For example, -schedule <code>sch1:sch2:sch3</code></p>
-suite	Optional. However, in a command, it is mandatory to use one of the following options: <ul style="list-style-type: none"> ◦ -suite ◦ -schedule ◦ -servicename <p>You must not use the -suite option along with the other options. The path includes the file name of the suite to run relative to the project.</p> <p>Starting from V9.2.1.1, you can execute multiple tests simultaneously.</p> <p>For example, -suite <code>test1:test2:test3</code>.</p>
-importzip	Optional. To import the project as test assets with dependencies into your workspace, use the -importzip option. This command is available from V9.2.1.1 and later. You can execute test assets from the imported zip file, but you must specify the -importzip <i>{complete path where the zip file is stored on your computer}</i> option along with the -schedule or -suite options. For example, <code>C:\User\Desktop\test1.zip</code>


-varfile	Optional. You can use this option to specify the complete path to the XML file that contains the variable name and value pairs.
-service-name	<p>Optional. However, in a command, it is mandatory to use one of the following options:</p> <ul style="list-style-type: none"> ◦ -suite ◦ -schedule ◦ -servicename <p>You must not use the -servicename option along with the other options. The path includes the file name of the service to run relative to the project. Instead of running a performance test, the specified service is run when it is available.</p>
-serviceargs	<p>Optional. You can use this option to specify a series of arguments to pass to the service specified.</p> <p>For example, -serviceargs "<i>myserviceparm1 myserviceparm1value</i>"</p> <p>The values are in quotation marks as they contain spaces.</p>
-config-file	<p>Optional. You can use this option to specify the complete path to a file that contains the parameters for a test or schedule run. Each parameter must be on a single line. To create a configuration file, you must use an editor that does not wrap lines. Any parameters, whether required or optional, can be set in the configuration file. The command line parameters override the values in this file.</p> <p> Notes:</p> <ul style="list-style-type: none"> ◦ If you are creating a config file manually, the file must be in the UTF-8 format. You must not use quotation marks in this file even for values that contain spaces. ◦ You can create command line config file from the product, which you can use while running tests or schedules from the command-line interface or Maven. For more information about how to create a command line config file from the product, see related links.
-results	<p>Optional. You can use this option to specify the name of the results file. The default result file name is the test or schedule name with a time stamp appended. You must specify a folder name that is relative to the project to store the test results.</p> <p>For example, -results <i>folder/resultname</i></p>

-over-write	Optional. Determines whether a results file with the same name is overwritten. The default value, <code>false</code> , indicates that the new results file is created. If the value is <code>true</code> , the file is overwritten and retains the same file name. You must use double quotes "" for values <code>true</code> or <code>false</code> .
-quiet	Optional. Turns off any message output from the launcher and returns to the command shell when the run or the attempt is complete.
-users	Optional. Overrides the default number of virtual users in the run. For a schedule, the default is the number of users specified in the schedule editor. For a test, the default is one user. This option creates a new copy of the schedule that contains the specified number of users.
-vmargs	Optional. To specify the Java™ maximum heap size for the Java™ process that controls the command line playback, use the -vmargs option with the <code>-Xmx</code> argument. For example, when you use -vmargs -Xmx4096m , specify a maximum heap size of 4096m. This method is similar to specifying <code>-Xmx4096m</code> in the <code>eclipse.ini</code> file for the workbench when playing back the test from the user interface.
-rate	Optional. You can use this argument to specify a rate that you want to achieve for a workload in the Rate Runner group. For example, -rate "Rate Runner Group 1=1/s, 3/m; Rate Runner Group 2=5/s, 10/s" . Here, <code>Rate Runner Group 1</code> is the name of the Rate Runner group that has two stages. The desired rate for the first state is one iteration per second and the rate for the second stage is three iterations per minute.  Note: The Rate Runner group name must match with the name in the Rate Schedule.
-duration	Optional. You can use this argument to specify the duration of the stages in the Rate Schedule. For example, -duration Stage1=10s; Stage2=3m  Note: The stage number specified must exist in the Rate Schedule.
-publish	Optional. You can use this parameter to publish test results to the HCL OneTest™ Server. You can use the following options along with the -publish parameter: <ul style="list-style-type: none">◦ <code>serverURL#project.name=projectName</code>: You can use this option to specify the project name. If the project name is not specified, the name of the current project is used.  Note: If you provide the server and the project details under Window > Preferences > Test > HCL OneTest Server in the product, and if you use <code>serverURL#project.name=projectName</code> along with the -publish parameter, the

	<p> server details in the command-line interface takes precedence over the product preferences.</p> <ul style="list-style-type: none"> ◦ <code>no</code>: You can use this option if you do not want to publish test results after the run. This option is useful if the workbench preferences are set to publish the results, but you do not want to publish them. <p> Important: You must provide the offline user token for the server by using the HCL_ONETEST_OFFLINE_TOKEN environment variable before you use the -publish parameter in the command-line interface.</p>
<p>-publish-for</p>	<p>Optional. You can use this option to publish the test results based on the completion status of the tests:</p> <ul style="list-style-type: none"> ◦ ALL - This is the default option. You can use this option to publish test results for any text execution verdict. ◦ PASS - You can use this option to publish test results for the tests that have passed. ◦ FAIL - You can use this option to publish test results for the tests that have failed. ◦ ERROR - You can use this option to publish test results for the tests that included errors. ◦ INCONCLUSIVE - You can use this option to publish test results for the tests that were inconclusive. <p>You can add multiple parameters separated by a comma.</p>
<p>-exportlog</p>	<p>Optional. You can use this parameter to specify the file directory path to store the exported HTTP test log.</p> <p>Starting from V10.0.1, by using the -exportlog parameter, you can provide multiple parameter entries when running multiple tests. You must use a colon to separate the parameter entries.</p> <p>For example: -exportlog c:/logexport.txt:c:/secondlogexport.txt</p> <p>If there are multiple -suite parameter entries with a single -exportlog parameter entry, then the -exportlog parameter generates the appropriate number of test logs by appending 0, 1, 2, and so on to the -exportlog parameter entry name.</p> <p>For example: -suite "sampletest1:sampletest2:sampletest3" -exportlog c:/logexport.txt The command generates the following test logs:</p> <ul style="list-style-type: none"> ◦ logexport_0.txt ◦ logexport_1.txt ◦ logexport.txt <p>The last test log generated has the same name as that of the initial -exportlog entry.</p>

	 Note: If there are multiple -suite and -exportlog parameter entries, the number of -suite entries must match with the number of -exportlog entries. Otherwise, the following error message is displayed: <pre>Error, number of -suite and -exportlog entries do not match.</pre>
-export-stats	Optional. You can use this option to export reports in comma-separated values (CSV) format, with the file name derived from the report name. This directory can be relative to the project or a directory on your file system. If the -exportstatreportlist option is not specified, the reports specified on the Export Reports page of the Performance Test Report preferences are exported.
-timerange	Optional. You can use this option along with -exportstats , -exportstatshtml , and -execsummary to export test results within one or more time ranges. The value is the time range that you specify in the schedule. For example, <i>"all, 5 Users, 10 Users"</i> . You must separate time ranges with a comma and use double quotation marks (") when there is space in a time range.
-export-statshtml	Optional. When you want to export web analytic results, you can use this option. The results are exported in the specified directory. You can then analyze the results on a web browser without using the test workbench.
-compare	You can use this argument along with -exportstatshtml and -execsummary to export the result in compare mode. The value can be paths to the runs and are relative to the workspace. You must separate the paths by a comma.
-export-statereportlist	Optional. You can use this option to specify a comma-separated list of report IDs along with -exportstats or -exportstatshtml to list the reports that you want to export in place of the default reports, or the reports selected under Preferences . To view this setting, navigate to Window > Preferences > Test > Performance Test Reports > Export Reports . To copy the report IDs list into your command line, navigate to Window > Preferences > Test > Performance Test Reports > Export Reports . Under Select reports to export , select the required reports, and click Copy ID to clipboard . You can then paste the clipboard content on to your command line editor
-exec-summary	Optional. You can use this option to export all of the reports for the test run in a printable format, also known as an executive summary, to the local computer. You must specify the path to store the executive summary.
-exec-summaryreport	Optional. You can use this option to export a specific report as an executive summary for the test run to the local computer. You must specify the ID of the report to export.

	<p>For example, to export an HTTP performance report, specify <code>http</code>. You must use this option along with <code>-execsummary</code>.</p> <p>To copy the report IDs list into your command line, navigate to Window > Preferences > Test > Performance Test Reports > Export Reports. Under Select reports to export, select the required reports, and click Copy ID to clipboard. You can then paste the clipboard content on to your command line editor</p>
-user-comments	<p>Optional. You can add text within double quotation mark ("") to display it in the User Comments row of the report.</p> <p> Note: You can use the file <code>CommandLine.exe</code> to run the command to add comments in a language that might not support Unicode characters on Windows operating system.</p>
-publishre-ports	<p>Optional. You can use this option to publish test results in HCL OneTest™ Server. The parameters that you can use with it are the following:</p> <ul style="list-style-type: none"> ◦ FUNCTIONAL ◦ MOBILE_WEBUI ◦ STATS ◦ TESTLOG <p>For example, <code>-publishreports "STATS, TESTLOG"</code></p> <p>You must prefix with "!" to publish all the reports except the specified one.</p> <p>For example, <code>-publishreports "! TESTLOG"</code></p> <p>All the reports except the TESTLOG report is published to HCL OneTest™ Server after executing the command.</p>
-stdout	<p>Optional. You can use this option to display the information about the test or schedule on the command line.</p> <p>After you run a test or schedule from the command line, the following outputs are displayed to give you the overall information of the test or schedule:</p> <ul style="list-style-type: none"> ◦ --VERDICT: The verdict of the test or schedule. ◦ --REMOTE_RESULT: The URL of the result published to HCL OneTest™ Server. ◦ --REMOTE_RESULT_UI: The URL of the result published to HCL OneTest™ Server and can be opened in a browser to analyze the result. ◦ --LOCAL_RESULT: The path of the result saved locally. <p>For example, <code>-workspace workspace_full_path -project proj_rel_path -schedule sched_rel_path -publishpublish_url -stdout</code></p>

<p>-swap-datasets</p>	<p>Optional. Use this option to replace dataset values during a test or schedule. If a test or schedule is associated with a dataset, you can replace the dataset at run time while initiating the run from the command line.</p> <p>You must ensure that both original and new datasets are in the same workspace and have the same column names. You must also include the path to the dataset when you run the <code>-swap-datasets</code> command.</p> <p>For example, <code>-swaptatasets /project_name/ds_path/ds_filename.csv:/project_name/ds_path/new_ds_filename.csv</code></p> <p>You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon.</p> <p>For example, <code>-swaptatasets /project_name1/ds_path/ds_filename.csv:/project_name1/ds_path/new_ds_filename.csv;/project_name2/ds_path/ds_filename.csv:/project_name2/ds_path/new_ds_filename.csv</code></p>
<p>-history</p>	<p>Use this command when you want to view a record of all events that occurred during a <i>test</i> or <i>schedule</i> run. However, you must use the command suffixed with any of the following options:</p> <ul style="list-style-type: none"> ◦ <code>jaeger</code>: To send test logs to the Jaeger UI during the <i>test</i> or <i>schedule</i> run. ◦ <code>testlog</code>: To send test logs as traditional test logs in HCL OneTest™ Performance during the <i>test</i> or <i>schedule</i> run. ◦ <code>null</code>: To send no test logs either to the Jaeger UI or HCL OneTest™ Performance during the <i>test</i> or <i>schedule</i> run. <p>For example:</p> <pre style="background-color: #f0f0f0; padding: 5px;">-workspace workspace_full_path -project proj_rel_path -suite suite_rel_path -stdout -history comma delimited list of modes</pre> <pre style="background-color: #f0f0f0; padding: 5px;">-workspace C:/Users/HCL/hclonetest/test_ws -project Project1 -suite test1.testsuite -stdout -history jaeger</pre> <p> Note: You can add multiple options separated by a comma to send test logs during the <i>test</i> or <i>schedule</i> run to HCL OneTest™ Performance and the Jaeger UI.</p> <p>For example:</p> <pre style="background-color: #f0f0f0; padding: 5px;">-workspace C:/Users/HCL/hclonetest/test_ws -project Project1 -suite test1.testsuite -stdout -history jaeger, testlog</pre> <p>For more information about how to view test logs in the Jaeger UI and HCL OneTest™ Performance, see related links.</p>
<p>-override-ermlabels</p>	<p>Optional. By using the <code>-overridermmlabels</code> command, you can control the Resource Monitoring sources that are required to collect in a performance schedule during the schedule run.</p>

You can use this command if you want to perform any of the following actions:

- To enable the **Resource Monitoring from Service** option for a performance schedule if the **Resource Monitoring from Service** option is not enabled from the schedule editor in HCL OneTest™ Performance.
- To ignore Resource Monitoring sources that were set in the performance schedule and to change for a label matching mode.
- To replace an existing set of Resource Monitoring labels that were set in the performance schedule and run the schedule with a new set of Resource Monitoring labels.



Note: You must add the Resource Monitoring labels to the Resource Monitoring sources on the **Resource Monitoring** page in your HCL OneTest™ Server project. You can use these labels for adding the Resource Monitoring sources to run the performance schedule through the command line interface.

The command accepts a comma-separated list of labels.

For example, if you have added a label in HCL OneTest™ Server for a Resource Monitoring source as `rm1`, then run the following command to collect data from the source as follows:

```
-workspace workspace_full_path -project proj_rel_path
-suite suite_rel_path -overridermLabels "rm1"
```



Note: You can add multiple labels to a performance schedule separated by a `comma` to collect data from the multiple sources during the schedule run. For example:

```
-workspace workspace_full_path -project proj_rel_path
-suite suite_rel_path -overridermLabels "rm1,rm2,rm3"
```

If your label contains a `comma (,)`, then when running the `-overridermLabels` command, you must replace the **single comma** with the **double comma** in the label.

For example, if you have added a label to a Resource Monitoring source as (`rm1,test`), then you must run the following command to collect data from source as follows:

```
-workspace workspace_full_path -project proj_rel_path -suite suite_rel_path -overridermLabels "rm1,,test"
```

To stop the test run, you can open another command prompt window and use one of the following options with the cmdline option:

Command	Description
-stoprun	Optional. Stops the test run after the specified number of seconds. The block is executed, and the test log is transferred before stopping the run. You must use the -workspace command and specify the location of the workspace.
-abandonrun	Optional. Stops the test run immediately. You must use the -workspace command and specify the location of the workspace.



Note: Messages are displayed to indicate when the test or schedule is launched and when it is completed unless you include the **-quiet** option.

Exemple

```
cmdline -workspace C:/RPTWorkspace -project testProj -eclipsehome C:\Program Files\__BRAND_NAME__\__SDP_PATH__
\eclipse.exe -schedule MySchedule -varfile C:/Assets/testProjVar.xml
```

Initializing variables from Engineering Test Management

If you want to run an HCL OneTest™ Performance test from IBM® Rational® Quality Manager, you can pass the execution variables defined in Engineering Test Management to the HCL OneTest™ Performance test.

Before you begin

- Configure the Engineering Test Management adapter in HCL OneTest™ Performance. For more information, see the [Configuration on page 132](#) topic.
- Variable names must be the same in Engineering Test Management and HCL OneTest™ Performance.
- The **Visible in** value for the variable in the HCL OneTest™ Performance test must be set to **All tests for this user**.

About this task

When you pass an execution variable to a HCL OneTest™ Performance test, the value initialized in the test is replaced by the value in the execution variable. If you modify the value that is initialized in the test, after the test is executed, the modified value is passed back to the execution variable in Engineering Test Management.

To initialize an execution variable value to a test, run the test from Engineering Test Management. For information about execution variables, see [Using execution variables in manual test](#).

Correlating response and request data

For a test to run correctly, a request that is sent to a server might need to use a value that was returned by a previous request. By ensuring that this data is correlated accurately, you can produce better performance tests.

Data correlation overview

A request can include data that was returned in the response to a previous request. Associating data in this manner is called *data correlation*.

Interactions with an application are typically related to each other. For example, consider the following interactions with a web-based application, in which each request depends on information returned from a previous response:

1. A payroll clerk types the web address for an application, which sends a login prompt. When the clerk logs in, the web server returns a page that indicates that login has succeeded and a unique session ID to the web browser that the clerk is using.
2. The clerk clicks a link on the returned page, which requests that the web server open the page for searching the employee database. The web browser includes the session ID when sending the request. Based on the session ID, the web server knows that the request comes from someone who is already logged on, and so opens the search form for the employee database. The clerk then searches for a specific employee. The web server returns a photograph of that employee and the employee's unique ID.
3. The clerk clicks a link that requests the web server to return the payroll record for the employee. With this request, the web browser sends two IDs:
 - The session ID, so that the web server knows that the request comes from some who is logged on
 - The employee ID, so that the web server can locate and return the correct information

In this example, request 2 depends on request 1, and request 3 depends on requests 1 and 2.

If you record these interactions in a test, before running the test with multiple users, you would vary the test data. For example, you would replace the user name and password values, the employee name search values, or both, with values that datasets contain. When you run the test, each virtual user returns a different employee payroll record, based on the contents of the datasets.

In a generated test, where data in a request depends on data that is contained in the response to a previous request, the request data is substituted from the response data on which it depends. The term for this internal linking of response and request data is *data correlation*. When you run a test with multiple users and varied data, data correlation is required to ensure that the test runs correctly.

A *reference* is a value in a test (typically in a response) that can be used by a subsequent value in the test (typically in a request). When the test generator detects that a request value must be substituted from a previous value, it designates the earlier value as a reference and correlates the subsequent request value with the reference. This process is called *automated data correlation*. You can also manually correlate any two values in a test or unlink existing correlations.



Note: You can change or disable automated data correlation. To do so, click **Window > Preferences**, expand **Test**, and then click **Test Generation**.

By default, the empty strings are not correlated because it might increase the time taken to generate a test. However, sometimes empty strings such as spouse name or middle initial become important to correlate. To correlate the

empty strings, click **Window > Preferences > Test > Test Generation > HTTP Test Generation > Data Correlation** and select the **Create substitutions for empty strings** check box.

Generally, the HTML response content after the recording appears as `<input type="username" name="User" id="aaa" value="John"/>`. Some applications dynamically update the `name` attribute. Therefore, when you play back the test, the HTML response content appears as `<input type="username" name="idt020" id="aaa" value="John"/>`. Because the `name` attribute changes dynamically, data correlation does not occur and the playback fails. For data correlation to correlate the response content based on the `ID` attribute, ensure that you have selected **ON** in the **Prioritize correlation based on ID** option at **Window > Preferences > Test > Test Generation > HTTP Test Generation > Data correlation**.

To help you work with correlated data, the test editor uses color coding and provides navigational aids:

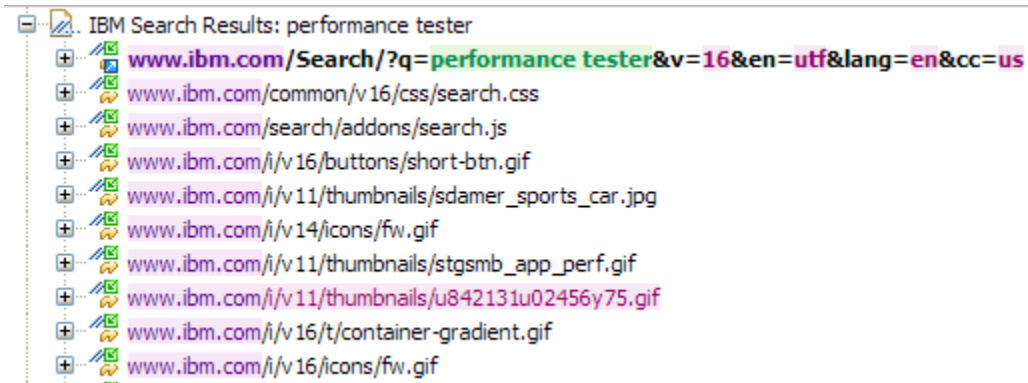
- When you click a page, you see a Test Data table for that page. By default, related dataset candidates are shown in green text on a light green background, values that are already associated with a dataset are shown in white text on a green background, and references are shown in blue text.

Test Data Options ▾

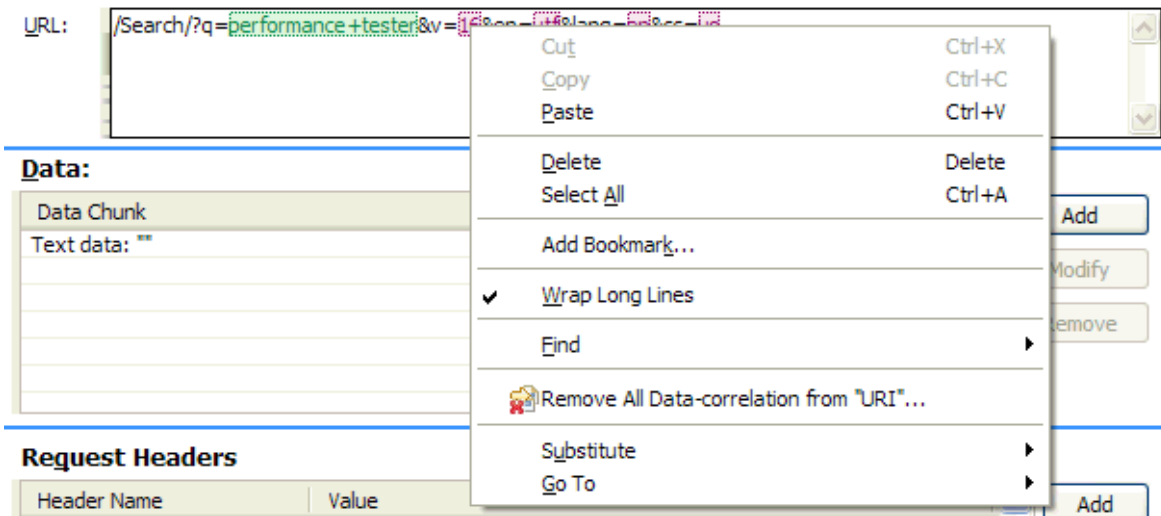
Name	Value	Substituted with
action	shopping	
category	1	
action	getimage	= "action2" ("getimage" - Content)
inventoryID	V0001	= "inventoryID7" ("V0001" - Content)
action	getimage	= "action2" ("getimage" - Content)
inventoryID	V0005	= "inventoryID6" ("V0005" - Content)
action	getimage	= "action2" ("getimage" - Content)
inventoryID	V0003	= "inventoryID5" ("V0003" - Content)

URL Encode

- If correlated data is not displayed, right-click the table and verify that **Show References** is selected. To navigate directly to a page request containing correlated data, double-click a table row. To associate correlated data from this table with a dataset, click the row, click **Substitute**, and then click **Select Data Source** to open the **Select Data Source** window. You can also use the **Test Data Sources** view to make substitutions. In the test editor, right-click the **Test Data** table, and then select **Link with Test Data Sources View**. When you click a row in the **Test Data** table, the **Test Data Sources** view displays information about the selected substitution site.
- When you expand a page, green text indicates page requests that contain dataset data or candidates. Blue text indicates page requests that contain references.



- When you click a highlighted request, dataset candidates are highlighted in light green, data that is associated with a dataset is highlighted in dark green, and correlated data is highlighted in red. If you right-click a value for correlated data, as shown in the example, you can then click **Go To** to see its reference:



- References are highlighted in dark blue.

Viewing data correlation

You can switch between viewing all test elements in the test editor and viewing only elements related to data correlation in the test editor. Viewing only data correlation elements makes it easier to add and remove substitutions.

1. In the Test Navigator, browse to the test, and double-click it. The test opens.
2. In the **Test Contents** area, click **Options**.
3. Click **Show > Data Correlation**.

Result

The test editor window displays only elements that are related to data correlation. Alternately, click **View** under **Test Contents** to switch between **Display all Test Contents** and **Show Substitutions**.

4. Select a single test element in the **Test Contents** area to see the current data source and to remove or change the substitution in the **Test Element Details** area. Select multiple elements in the **Test Contents** area to see

the data in tabular form in the **Test Element Details** area. Different controls are available depending on the type and number of elements that you select in the **Test Contents** area.

5. **Optional:** In the Test Elements Details area, click **Substitute > Select Data Source** to open the **Select Data Source** window, where you can specify the data source for the selected substitution site.

What to do next

To view all test elements, click **Options > Show > Data Correlation** again.



Note: If you select a test element while viewing all test contents, and then switch to viewing only data correlation elements, then the corresponding substituters and dataset candidates are selected. For example, if you select an HTTP page in the test editor, and then switch to viewing only data correlation elements, then all substituters and dataset candidates for all requests from the HTTP page are selected.

Data correlation rules overview

You can use the data correlation rules editor to customize how data is correlated. You can control how references and substitutions are generated in tests, and store these rules so that you do not have to manually correlate data in every test that you record against a particular application.

You create data correlation rule sets in the rules editor. Data correlation rule sets are also known as rules files. Each rule set can contain multiple rule passes, and each rule pass can contain multiple rules. When you re-correlate test data with data correlation rules, each rule set is applied in the order that you specify. Within each rule set, each rule pass is applied in order. Within each rule pass, each rule is applied in order.

You can use data correlation rules to do these tasks:

- Create a reference, substitution, variable, or dataset column
- Link a substitution to a reference
- Rename a reference or substitution
- Encode a substitution
- Unlink a substitution from a reference
- Remove a specific reference, substitution, or variable
- Remove all references or substitutions

Typically, you create a substitution and then link a reference to the substitution. References are located in the data that the server under test returns, while substitutions are in the data that is sent to the server. To create a substitution and then link a reference to the substitution in the rules editor, see [Example: Linking references to substitutions with rules on page 453](#).

Rule sets are hierarchical trees. You can insert child rules, which accept values generated by parent rules as input. To find a particular reference by name, first add a `Find a reference` rule, and then add a child `Reference name` rule. In the rules editor, you can also combine rules by using `And` and `Or` and `Not` rules.

Creating data correlation rule sets

To use rules-based data correlation, you must create a data correlation rule set. Data correlation rule sets are also known as rules files.

1. Click **File > New > Data Correlation Rule Set**.

Result

The **New Data Correlation Rule Set** wizard opens.

2. Select a parent folder, and then in **File name** provide a name.
3. Click **Finish**.

Result

The data correlation rules editor opens.

Results

An empty rule set is created. Data correlation rule sets are XML files with the `.dcrules` file extension. Typically, you use the rules editor to edit data correlation rule sets. You can also use any XML editor to edit a data correlation rule set file.

Creating a reference using data correlation rules

You can create a data correlation rule that creates a reference from a regular expression when the rule is applied to test data.

1. Open a data correlation rule set in the rules editor.
A new rule set contains one empty rule pass.

2. Click **Insert**, and then select **Create a reference**.

Result

An empty `Create a reference` rule is inserted in the rule pass.

3. Under **Details**, supply information for all fields that are marked with asterisks and shaded in red. For **Reference field**, click the down arrow to select the field in which to create the reference. Use the push buttons at the top of the window to select a protocol, and then select a field.

Example

For example, to create a reference in the content field of an HTTP response, click the down arrow, and then click **HTTP > Content**. To create references in multiple fields, click the **Add field** push button to add another **Reference field**.

4. In **Regular expression**, type an expression to use to locate the reference. If only a part of the regular expression is required for a reference, enclose that part in parentheses.
 - a. Open the test, and locate the response to create the reference in.
 - b. Copy the text from the response to the clipboard.
 - c. In the rules editor, click the **Toggle regular expression assistant** push button to open the regular expression assistant.

- d. Paste the text from the clipboard to the **Test regular expression** page of the regular expression assistant window.
If the **Test regular expression** page is empty, the contents of the clipboard is automatically pasted in.
- e. Click the **Captured group** tab in the regular expression assistant.

Result

The overall group is displayed, and the captured groups are displayed. If no groups are displayed, edit the regular expression accordingly.

Example

To create more than one reference using the same regular expression, enclose each part in parentheses. For example, two references can be created from this regular expression: `name=(\S+)\svalue=(.+?)`.

5. In **Reference names**, provide names for the references.

The names that you specify are available to child rules. To use the references as arguments in child rules, enclose the reference name in percent signs.

Example

For example, if you specify a reference name of `name`, you can use `%name%` as an argument in a child rule.

6. If an attribute contains multiple matches for the regular expression, in **Occurrence**, type or select the occurrence to use to create the reference.
7. Under **Create reference only if used**, select **true** to create a reference only if a substitution site uses the reference. Select **false** to create the reference regardless of whether a corresponding substitution site is found.
8. Under **Overlapping site action**, select the action to take when a new reference overlaps with existing references.

Option	Description
Always remove existing	If the new reference overlaps with other references, the other references are removed.
Keep existing	If the new reference overlaps with other references, the other references are not removed.
Keep existing only if used	If the new reference overlaps with other references, the other references are removed only if the references are unused.

9. Under **Create reference even if overlapping**, select **true** to create a reference even if the new reference overlaps with existing references.
10. In **Log level**, select the level of error data to be written to the error log. With logs, you can see which rules worked and which did not. When you are debugging data correlation rules, use the **Action** log level. If the **Action** log level does not provide enough data for troubleshooting, use the **Detail** log level. The **Detail** log level produces a significantly higher number of log entries. Typically, when you are sure that the data correlation

rules that you have written work correctly, use the **None** or **Summary** log levels to reduce memory and disk-space consumption and unrequired entries in the error log.

Option	Description
None	Nothing is logged.
Warning	A message is logged when there are potential problems that are detected when the rule is applied.
Summary	One message is logged for the rule, no matter how many times the rule is applied.
Important	A message is logged every time the rule is applied in a manner that is not typical. This is the default log level.
Action	A message is logged every time the rule is applied.
Detail	A detailed message is logged every time the rule is applied.

11. In **Label**, type a label for the rule. If you do not type a label name, the rule is given a default name. The default name is the base name with the regular expression appended.
12. In **Description**, describe the rule. Descriptions can be useful if you share rule set files with other testers.

Creating a substitution with data correlation rules

You can create a data correlation rule that creates a substitution from a regular expression that is applied to test data.

1. Open a data correlation rule set in the rules editor.
A new rule set contains one empty rule pass.
2. Click **Insert**, and then select **Create a substitution**.
Result
An empty `Create a substitution` rule is inserted in the rule pass.
3. Under **Details**, supply information for all fields that are marked with asterisks and shaded in red. For **Field**, click the down arrow to select the field for which to create a substitution. Use the push buttons at the top of the window to select a protocol, and then select a field. To create a substitution in the data field of an HTTP request, click the down arrow, and then select **HTTP > Data**.
4. In **Regular expression**, type a regular expression to use to locate the substitution. If only a part of the regular expression is required for a substitution, enclose that part in parentheses.
 - a. Open the test, and locate the request to create the substitution in.
 - b. Copy the text from the request to the clipboard.
 - c. In the rules editor, click the **Toggle regular expression assistant** push button to open the regular expression assistant.

- d. Paste the text from the clipboard to the **Test regular expression** page of the regular expression assistant window.
If the **Test regular expression** page is empty, the contents of the clipboard is automatically pasted in.
- e. Click the **Captured group** tab in the regular expression assistant.

Result

The overall group is displayed, and the captured groups are displayed. If no groups are displayed, edit the regular expression accordingly.

To create more than one substitution site using the same regular expression, enclose each part in parentheses. For example, two substitution sites can be created from this regular expression: `(.+?)(.*?)`.

Example

To use values that were created in a parent `Create a reference` rule as arguments in the regular expression, enclose the reference names in percent signs. For example, if a parent rule created a `name` reference and a `value` reference, you could use them in this regular expression: `\:{%name%, %value%\}`. When the rule is run, `%name%` and `%value%` are substituted with the values extracted by the parent rule, and then the resulting regular expression is evaluated. Two substitutions are created, each linked to the corresponding reference.

5. In **Substitution names**, type names for the substitutions.
6. Under **Decode the field**, select **true** to decode the specified attribute before searching for a matching reference. Select **false** to search for a matching reference without decoding the attribute. The type of encoding depends on the selected protocol and attribute. For example, HTTP data can be URL-encoded.
7. Under **Create substitution only if used**, select **true** to create a substitution only when a matching data source is found. Select **false** to create the substitution regardless of whether a matching data source is found.
8. Under **Conflict action**, select the action to take when a new substitution overlaps with existing substitutions.

Option	Description
Always replace existing	If the new substitution overlaps with other substitutions, the other substitutions are removed.
Replace existing if dataset candidate	If the new substitution overlaps with other substitutions, and all the other substitutions are dataset candidates, other substitutions are removed. If at least one current substitution site that overlaps with the new substitution site is associated with a reference, then no current substitution is changed, and the new substitution is not created.
Replace existing if enclosed in the new site	If the new substitution overlaps with other substitutions that are all completely enclosed in the new substitution, other substitutions are removed. If at least one current substitution overlaps with the new substitution without being completely enclosed by the new

Option	Description
	substitution, then no current substitution is changed, and the new substitution is not created.
Replace existing if dataset candidate or enclosed	If the new substitution overlaps with other substitutions, and each of the other substitutions is either a dataset candidate or is completely enclosed in the new substitutions, other substitutions are removed. If at least one current substitution overlaps with the new substitution without being completely enclosed by the new substitution, or if at least one current substitution site that overlaps with the new substitution site is actually associated with a reference, then no current substitution is changed, and the new substitution is not created.
Keep existing	If the new substitution overlaps with other substitutions, the other substitutions are not removed. The new substitution is not created.

9. In **Log level**, select the level of error data to be written to the error log. With logs, you can see which rules worked and which did not. When you are debugging data correlation rules, use the **Action** log level. If the **Action** log level does not provide enough data for troubleshooting, use the **Detail** log level. The **Detail** log level produces a significantly higher number of log entries. Typically, when you are sure that the data correlation rules that you have written work correctly, use the **None** or **Summary** log levels to reduce memory and disk-space consumption and unrequired entries in the error log.

Option	Description
None	Nothing is logged.
Warning	A message is logged when potential problems are detected when the rule is applied.
Summary	One message is logged for the rule, no matter how many times the rule is applied.
Important	A message is logged every time that the rule is applied in a manner that is not typical. This is the default log level.
Action	A message is logged every time the rule is applied.
Detail	A detailed message is logged every time the rule is applied.

10. In **Label**, type a label for the rule. If you do not type a label name, the rule is given a default name. The default name is the base name with the regular expression appended.
11. In **Description**, describe the rule. Descriptions can be useful if you share rule set files with other testers.

Example: Linking references to substitutions with rules

You can create data correlation rules to link references to substitutions.

About this task

If you know which field in a request or in POST data must be correlated, write a `Create a substitution` rule for that field, and then insert a `Create a reference` rule as a child of the `Create a substitution` rule.

1. Open a data correlation rule set in the rules editor.
A new rule set contains one empty rule pass.
2. Create a `Create a substitution` rule. See [Creating a substitution with data correlation rules on page 450](#) to learn more about creating a `Create a substitution` rule.
3. Right-click the **Create a substitution** rule, and then click **Insert Item > Find data source for substitution**.
4. Right-click the **Find data source for substitution** rule, and then click **Insert Item > Create a reference**. If the reference exists, select **Find a reference** instead of **Create a reference**.

A `Create a reference` or `Find a reference` rule is inserted as a child of the `Find data source for substitution` rule. To use the value of the substitution extracted by the parent rule, type `%subname%` for the `Regular expression`, where `subname` is the name of the substitution that is created by the parent rule. See [Creating a reference using data correlation rules on page 448](#) to learn more about creating a `Create a reference` rule.

Results

When you recorrelate test data using this rule set, the references and substitutions that you defined are created and linked.

Example

Assume that a URI in your test is `http://host:port/RPThelp/index.jsp?topic=datacorrelation.html`. Assume that some of the response data from a previous request includes `...<id=2 docHelpName=recordtest> <id=23 docHelpName=datacorrelation> <id=24 docHelpName=rules>...` Write a rule that creates a substitution site in the URI for `datacorrelation` and sets the substitution name to `helpname`. Then, add rules that create a reference as a child of the substitution rule. The regular expression for the reference rule is `docHelpName=%helpname%`. Thus, the regular expression in the reference rule evaluates to `docHelpName=datacorrelation`. This regular expression ensures that the correct reference is linked to the substitution site.

Example: Linking substitutions to references with rules

You can create data correlation rules to link substitutions to references.

About this task

If you are familiar with the application under test and know the exact location of the reference in the response data, write a `Create a reference` rule that uses a regular expression to locate the reference data, and then insert a `Create a substitution` rule as a child of the `Create a reference` rule.

1. Open a data correlation rule set in the rules editor.
A new rule set contains one empty rule pass.
2. Create a `Create a reference` rule. See [Creating a reference using data correlation rules on page 448](#) to learn more about creating a `Create a reference` rule.
3. Right-click the **Create a reference** rule, and then click **Insert Item > Link with substitutions**.
4. Right-click the **Link with substitutions** rule, and then click **Insert Item > Create a substitution**. To use the value of the reference that is extracted by the parent rule, type `%refname%` for the **Regular expression**, where *refname* is the name of the reference that is created by the parent rule. See [Creating a substitution with data correlation rules on page 450](#) to learn more about creating a `Create substitution` rule.
A `Create a substitution` rule is inserted as a child of the `Link with substitutions` rule.

Results

When you recorrelate test data with this rule set, the references and substitutions that you defined are created and linked.

Example

Assume that a response in your test data contains an ID that is present in URIs and in POST data throughout the test. For example, the response includes `...<NeededID=ID123 docHelpName=rules>...`. Assume that a URI in the test is `http://host:port/RPThelpID123/index.jsp?topic=datacorrelation.html` and the POST data includes `...nameID123=ID123...`. Because you know the exact location of the ID in the response, you can write a rule that creates a reference for the ID. Specify the **Reference name** as `ID`. Then, add rules for two substitutions as children of the reference rule. Add one substitution with a URI attribute and one with a data attribute. Specify `%id%` in **Regular expression** for both substitution rules. Thus, the regular expression in the substitution rule becomes the `ID123` value when the rules run. This regular expression creates substitution sites in every location where the `ID123` value is found in URIs and in POST data. In this example, three substitution sites would be created: `RPThelpID123`, `nameID123`, and `nameID123=ID123`.

Example: Linking substitutions to built-in data sources with rules

You can create data correlation rules to link substitutions to built-in data sources. Built-in data sources include the dynamically calculated variables such as **Current Date**, **Random Number**, and **Timestamp**.

1. Open a data correlation rule set in the rules editor.
A new rule set contains one empty rule pass.
2. Create a `Create a built in data source` rule.
3. Right-click the `Create a built in data source` rule, and then click **Insert Item > Link with substitutions**.
4. Right-click the **Link with substitutions** rule, and then click **Insert Item > Create a substitution**. See [Creating a substitution with data correlation rules on page 450](#) to learn more about creating a `Create substitution` rule.
A `Create a substitution` rule is inserted as a child of the `Link with substitutions` rule.

Results

When you recorrelate test data with this rule set, the built-in data sources and substitutions that you defined are created and linked.

Recorrelating data with rules

After you have created a data correlation rule set, you can recorrelate data in tests.

Before you begin

Record a test, and create a data correlation rule set.

1. Open a test for editing. To recorrelate data in tests that are not open in the editor, select multiple tests in the **Test Navigator** window.
2. To recorrelate a test that is open in the test editor, click **Edit > Re-correlate test and transform data**. To recorrelate multiple tests, right-click the selected tests in the **Test Navigator** window, and then select **Apply data correlation rules**.

Result

The **Data Correlation and Transformation** window opens.

3. Select **Do rule-based data correlation**.
4. **Optional:** To remove data correlation from the test data, select **Clear existing data correlation**. Select **Do automatic data correlation** to recorrelate the test data by using the automatic data correlation. You can control automatic data correlation in the preferences. Click **Window > Preferences > Test > Test Generation**. Click **HTTP Test Generation**, and then click the **Data Correlation** tab for additional controls over automatic HTTP data correlation.
5. Click **Add**.

Result

The **Rules File Selection** window opens.

6. Expand the list of resources to locate the data correlation rules file to add.
7. Select the data correlation rules file to add, and then click **OK**.

Data correlation rules files have the `.dcrules` extension by default.

Result

The rules file is added to the list under **Rules Files**.

8. **Optional:** Click **Add** to add more rules files. Use the **Up** and **Down** push buttons to move rules files in the list. The data correlation rules are applied in the order in which the files are listed.
9. Click **Finish**.

Result

The test data is recorrelated.

Results

If you recorrelate test data in multiple tests that are not open in the test editor, the original tests are backed up. The backup copies are displayed in the list in the **Test Navigator** window. To restore the original version of the test, right-click the backup copy in the **Test Navigator** window, and then select **Restore test**.

Recording tests with data correlation rules

After you have created a data correlation rule set, you can record a test that uses those rules.

Before you begin

Create a data correlation rule set.

1. In the **Performance Test** perspective, click the **New Test From Recording** toolbar button or click **File > New > Test From Recording**.
2. In the **New Test From Recording** wizard, click **Create a test from a new recording**, select the type of test to create, and click **Next**.
3. On the **Select Location** page, select the project and folder to create the test in, type a name for the test, select **Customize automatic data correlation**, and click **Next**.
4. Continue recording a test.

Result

After you have recorded the test, the **Data Correlation** window opens.

5. Select **Do rule-based data correlation**.
6. **Optional:** Select **Do automatic data correlation** to correlate the test data using the automatic data correlation. You can control automatic data correlation in the preferences. Click **Window > Preferences > Test > Test Generation**. Click **HTTP Test Generation**, and then click the **Data Correlation** tab for additional controls over automatic HTTP data correlation. For example, when recording an HTTP test with data correlation rules, you might select **Automatically correlate host and port data**, **Automatically correlate URL pathname if redirected by response**, and **Automatically correlate Referers**, and then clear **Enable all other data correlation**.
7. Click **Add**.

Result

The **Rules File Selection** window opens.

8. Expand the list of resources to locate the data correlation rules file to add.
9. Select the data correlation rules file to add, and then click **OK**.

Data correlation rules files have the `.dcrules` extension by default.

Result

The rules file is added to the list under **Rules Files**.

10. **Optional:** Click **Add** to add more rules files. Use the **Up** and **Down** push buttons to move rules files in the list. The data correlation rules are applied in the order in which the files are listed.
11. Click **Finish**.

Result

The recorded test data is correlated.

Generate data correlation rules in the test editor

Instead of writing rules in the rules editor, you can generate rules automatically based on data correlation adjustments that you make while editing tests.

To see what data correlation looks like in rule form, generate rules while you edit tests, and then examine the rules in the rules editor. To generate a rule set file that you can use to correlate data on multiple tests that run against the same application, save all manual correlations in a rule set file.

Generating rules while you edit

You can generate rules automatically as you make changes to data correlation in the test editor. With this feature, you can see what a typical data correlation action looks like in rule form.

Before you begin

Record a test.

1. Rules accumulation is enabled by default. To enable rules accumulation, right-click in the **Test Contents** window, and then select **Data Correlation > Allow rules accumulation**.
2. Adjust data correlation as you typically would in the test editor. For example, create or delete references, substitution sites, datasets, or variables.
To learn more about adjusting data correlation, see [Guidelines for adjusting data correlation on page 469](#).
3. Right-click in the **Test Contents** window, and then select **Data Correlation > Show accumulated rules**.

Result

The **Save Data Correlation Rule Set** wizard opens. The data correlation rules that are generated while you manually edited the test are displayed.

4. Clear the check boxes of any rules that you do not want to save, and then click **Next**.
5. Type a file name, and then click **Save**.

What to do next

Examine the data correlation rules in the rules editor. To learn more about the types of data correlation rules available, see [Data correlation rules on page 1168](#).

Saving manual data correlation in a rule set file

You can generate data correlation rules based on the data correlation changes made to a performance test. This feature analyzes the data correlation adjustments that you have made to a test and saves those adjustments as a data correlation rule set file. You can use the rule set file to automatically correlate the data for other tests that run against the same application.

Before you begin

Create a performance test. Adjust the data correlation so that the test runs correctly. To learn more about adjusting data correlation, see [Guidelines for adjusting data correlation on page 469](#).

1. Open a performance test for editing.
2. Adjust data correlation as you typically would in the test editor. For example, create or delete references, substitution sites, datasets, or variables.
3. Right-click in the **Test Contents** window, and then select **Data Correlation > Save All Manual Correlation in Ruleset file**.

Result

The test is analyzed to determine the set of data correlation rules that represents the adjustments that you have made to data correlation. The **Save Data Correlation Rule Set** wizard opens. The data correlation rules are displayed.

4. Clear the check boxes of any rules that you do not want to save, and then click **Next**.

Typically, when saving manual data correlation, you save all the generated rules, because the generated rules represent the data correlation required for the test to run correctly.

5. Type a file name, and then click **Save**.

What to do next

You can use the generated rules file to perform data correlation on other tests that run against the same application.

To learn more about applying data correlation rules to existing tests, see [Recorrelating data with rules on page 455](#).

Viewing data correlation rule usage


To see a record of all the elements that changed when data correlation rules are applied, open the data correlation rules log. You can use the data correlation rules log to determine which rules worked and which did not.

About this task

The **Data Correlation Rules Log** view opens automatically when you apply data correlation rules to a test.

1. In the **Data Correlation Rules Log** view, consider the following options:

Choose from:

- You can apply data correlation rules when you record a test, or you can recorrelate existing test data. To learn more about recorrelating test data, see [Recorrelating data with rules on page 455](#).
 - You can specify the logging level in the data correlation rules file, or on the **Data Correlation and Transformation** wizard page.
 - You can use the **Action** log level to debug data correlation rules. If the **Action** log level does not provide enough data for troubleshooting, use the **Detail** log level.
 - You can view the data correlation actions in the chronological order. Click the **Chronological log view** icon  to view test elements that are created or removed by the data correlation rules.
 - When you are sure that the data correlation rules that you wrote work correctly, use the **None** or **Summary** log levels to reduce memory, disk-space consumption, and unnecessary entries in the error log.
2. Expand the first element of the log, and navigate to the detail that you want to see. Use the icons in the upper-right corner of the view to navigate through the log. For example, to navigate to the corresponding element in the test editor, select a log entry, and then click **Go to test element**.

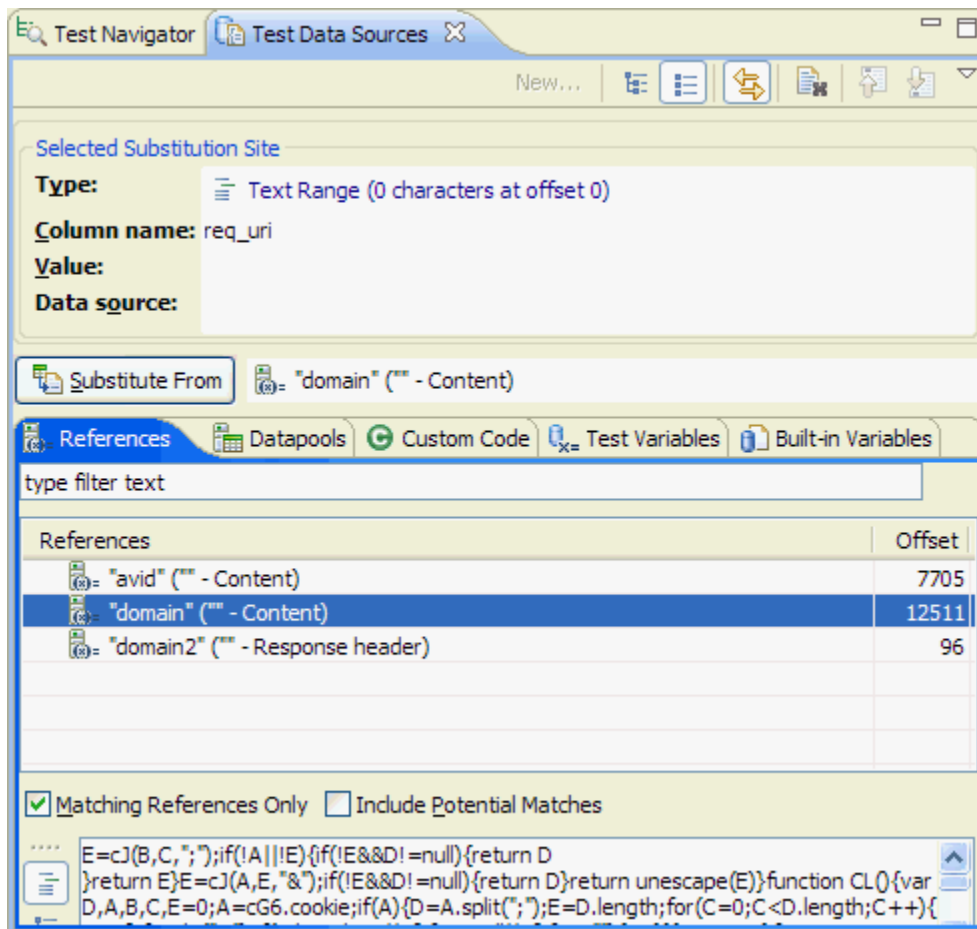
Test data sources overview

Use the **Test Data Sources** window to add or change data correlation for any supported test element.

The test generator attempts to perform automated data correlation. You can also manually correlate request values with other test data. The **Test Data Sources** window displays the following types of test data that you can substitute from:

- References
- Datasets
- Custom code
- Test variables
- Built-in datasources

You can right-click references, custom code, and built-in data sources to display a menu that contains commonly used commands. In addition, you can use the toolbar and menu at the top of the **Test Data Sources** window to complete common operations, such as creating a dataset or switching from tree view to list view. The **Substitute From** push button is enabled when you select a substitution site in the test editor and a data source from one of the five categories in the **Test Data Sources** window. Click **Substitute From** to correlate the data source and the substitution site.



References

The data sources that have been created in the test and the potential matches that are available for a selected substitution site. For example, text that is found in a response and used in a subsequent request is automatically created as a reference. Clear the **Matching References Only** check box to see all the references that occur before the substitution site in the test. Select **Include Potential Matches**

to see a list of locations that might match the substitution site. Replace the `type filter text` string with keywords to filter the list of references. Select a reference in the **References** table to display the reference in the pane at the bottom of the window. Use the controls to the left of the preview pane to switch between inline view and tree view.

Datasets

The datasets that have been added to the test. To add a new dataset, click the **Add new Dataset** icon



Custom Code

The custom code that is available as data sources for this substitution site, if you have written Test Execution Services (TES) custom code.

Test Variables

The test variables and their types that are available as data sources for this substitution site. Replace the `type filter text` string with keywords to filter the list of test variables.

Built-in Datasources

The dynamically calculated data sources, such as **Current Date**, **Random Number**, **Sequential Number** and **Timestamp**, that are available as data sources for this substitution site. To create a built-in data source, right-click inside the **Built-in Datasources** page, and select **New**, or click the **New** push button at the top of the window.

To assign a unique value for every substitution, when creating the built-in data sources, select the **Get New Value Each Time Used** check box.

Detaching the Test Data Sources window

The **Test Data Sources** window is detachable. To detach the **Test Data Sources** window, right-click the **Test Data Sources** tab, and then select **Detached**. When detached, the **Test Data Sources** window is always displayed in front of the other windows that make up the workbench.

Correlating a request value with test data

If a test runs without error but does not generate the results that you expect, you might need to correlate a value in a request with other test data.

About this task

You can correlate a request value with the following types of test data:

- References
- Datasets
- Custom code
- Test variables
- Built-in datasources

For example, if you recorded a test and searched on a date, you might want to substitute the built-in data source *Current Date* so that the test will search on the playback date, not on the recorded date. For information on the different types of test data, see [Test data sources overview on page 458](#).

1. In the **Test Navigator**, browse to the test, and double-click it. The test opens in the test editor.
2. Locate the value that the other test data will replace.
3. Highlight the value: Press and hold the left mouse button and drag your mouse over the value.
4. In the **Test Data Sources** window, click the appropriate tab: **References**, **Datasets**, **Custom Code**, **Test Variables**, or **Built-in Datasources**. To see all references, clear the **Matching Only** check box.
5. In the **Test Data Sources** window, select the test data to use. For references and custom code, you can double-click the entry in the **Test Data Sources** window to find the data source in the test editor.
6. Click **Substitute From**.

Result

The value is shown in purple text to indicate that it has been correlated and the correlation is added to the Test Data table, which contains the substitution sites for the page.

Substituting request content with file contents

You can substitute the content portion of a protocol request with the contents of a file. This feature is only supported in certain sections of a test, depending on the protocol. For example, HTTP tests support file content substitution only in the POST data section of a request. SOA tests support file content substitution for MIME and DIME attachments, XML node values and fragments, and text content. File content substitution works in the same manner as other substitutions. All standard test data sources such as test variables, datasets, and references can be used. The data from the source is treated as a full path to a file. The file is opened, its contents are read, and then those contents are used in the substitution.

Before you begin

You must copy the files that contain the substitution content to the agent computers. You must record a test with locations for substitution from a file. For example, record an HTTP test that contains multipart MIME data in a POST request.

1. Create a data source that contains the full path to the file from which you want to substitute content. Specify an absolute path to the file. Use path separator characters appropriate to the operating system of the agent computer running the test. Optionally, specify a character set to use in reading the file. The existence of the file is not validated. If the file cannot be opened when the test runs, a message is written to the test log. If you use path separator characters that are not appropriate for the operating system of the agent computer, the substitution cannot be completed. For example, if you use a path of `D:\DataFiles\file1` on an agent computer running Linux™, the substitution cannot be completed, because Linux™ uses forward slashes as path separator characters.
2. In the test editor, navigate to the request where you want to substitute content, and then select the request data that you want to substitute.
3. Right-click, and then select **Substitute > Select Data Source**.

4. Select the data source that contains the path to the file from which you want to substitute.
5. Right-click the substitution site, and then select **File Contents Substituter**.

Results

When the test runs, the content in the protocol request is substituted with the specified file contents.

HTTP POST data is displayed in the test editor in chunks. You can create a file contents substitution in the POST data of an HTTP POST request by selecting the data chunk that you want to correlate, and then clicking **Substitute**. The test data source that you select is automatically treated as a file contents substituter. The entire data chunk is replaced with the contents of the file when the test runs, even if only a portion of the text in a text data chunk is selected by the substituter.

Built-in Datasources

You can use built-in data sources instead of creating custom data sources to substitute the recorded values. The Built-in Datasources section in the Test editor displays the data sources that have been used and unused. You can modify their properties from a location.

The built-in data sources are Current Date, Random Number, Sequential Number, and Timestamp. The values of these data sources are dynamically calculated and submitted to the test.

Assigning random numbers to users or clients

To assign unique random numbers to all the virtual users in a test, you can create a random number data source in the workbench. This data source will generate unique integers or floating point numbers for the users or clients. You can choose to distribute the numbers of the virtual users or clients in a uniform, normal, or negative exponential way.

1. In the Test editor, select the root node (name of the test) and from the Test Details area, select **Built-in Datasources**.
2. Select the **Random Number** data source and click **Add Built-in Datasource**.
3. Type a name for the data source.
4. Select how do you want to generate the numbers:
 - **Uniform**: Click this option to generate random numbers with a uniform distribution. Specify the minimum and maximum values for the generated numbers.
 - **Normal**: Click this option to generate random numbers with a normal or Gaussian distribution. Specify the average and the standard deviation for the generated numbers.
 - **Negative Exponential**: Click this option to generate random numbers with an exponential distribution. Specify the average for the generated numbers.
5. Select how do you want to format the numbers. You can select **Common** to format the numbers in decimal or scientific notations.
You can choose **Custom** to specify a **Format mask** by using the standard Java formatting syntax. The changes that you specify can be previewed in **Formatted output**.
6. To substitute the built-in data source every time with a new value for the requests, select the **Get new value each time used** check box.

Assigning sequential numbers to users

To assign unique sequential numbers to all the virtual users in a test, you can create a sequential number data source in the workbench. This data source will generate unique integers or floating point numbers for the users.

Before you begin

You can use the sequential built-in data source option wherever data correlation substitutions are permitted, such as Transactions or Delays.

About this task

You define an initial value that should be assigned to the first virtual user and a step value that is a number by which the current value increments after each retrieval by a virtual user. If initial value is 1 and step value is 5, the workbench generates numbers in the sequence of 1, 5, 10, 15, and so on and each number is mapped to a virtual user. If a test is run on multiple agent machines, the workbench assigns a sequential value to all the users in all the agent machines.

You can also assign a full sequence of numbers of one virtual user. The sequence number increments in the request for each time the request in the multi-request generator is executed.

1. In the Test Contents area of the test, click an element in the test where data correlation substitution is permitted, such as a transaction name or delays.
2. In the Test Element Details area, right-click the name of the element and click **Substitutue > Built-in Datasources**.
3. In **Built-in Datasource Selection Wizard**, click **Sequential Number** and click **Next**.
4. Assign a name for the data source.
5. In **Initial Value**, type a number to be assigned to the first virtual user.
6. In **Step Value**, type a number.
7. In **Formatting Options**, you can format the number in the manner you want to use.
8. **Optional:** To assign a full sequence of numbers to one virtual user, select the **Execute for individual user** check box.
9. **Optional:** To substitute the built-in data source every time with a new value for the requests, select the **Get new value each time used** check box.
10. Click **Finish**.
11. Save and run the test.

Results

After you add the test to a schedule and run the schedule, the test log displays each element name where the data source is applied with the unique sequential number assigned to the virtual user.

Creating a reference or field reference

When you designate a test value as a reference or designate a set of test data as a field reference, you can use the data elsewhere in the test.

About this task

A *reference*, which is typically located in response data, points to a specific value that you want to use in a subsequent test location, typically a request. You can substitute a request value with a reference. This substitution is called *data correlation*. You can also use a reference as input to an IF-THEN condition in a test or as input to custom Java™ code that your test calls.

A *field reference* points to an entire block of test data. For example, an entire HTTP response can be designated as a field reference. You can use a field reference as input to custom Java™ code that your test calls.

1. In the Test Navigator, browse to the test, and double-click it. The test opens.
2. Locate the value or set of data to designate as a reference or field reference.

Different protocols support different references. For HTTP tests, you can create references and field references in these fields:

- A response header value, the Value column of a Response Headers table
- Response content, the Content field

For HTTP responses, you can create field references in these fields:

- The Status field
- The Reason field

3. Create the reference:
 - a. For response contents, highlight the value. For response header contents, click the row in the Response Headers table, and then click **Modify**.
 - b. Right-click, and then click **Create Reference**.

Result

The value is highlighted in light blue to indicate that it is an unused reference. When you use it, the highlight changes to dark blue. The reference is given a name automatically. To see the name of the reference, right-click the value, and then select **Properties**. To edit the regular expression that is used to locate the reference, click the **Toggle regular expression assistant** push button on the **Properties** window. The regular expression assistant displays the response content matched by the regular expression and the groups captured by the regular expression. To ensure that the details about the reference is always logged, select a reference and click **Properties**, and then click the **Always log details** checkbox. To create a reference that will be used by the HTTP secondary request, you must select **All occurrences**. You can also match the reference within a given range of all the occurrences.



Note:

A reference that is created to be used by the HTTP secondary request cannot be used by custom code or other data sources.

If you select the **Always log details** checkbox, the details will be logged irrespective of the logging level set for a schedule. You must use this option only for debugging purpose.



You can always log the details of Substituters, Data Sources, and Requests.

4. To create a field reference, do not highlight the value. Instead, right-click the value, and then click **Create Field Reference**.
 - a. Field references are not automatically given names. To name a field reference, right-click the field reference, and then select **Properties**. Type a name in the **Name** field, and then click **OK**.

Result

The entire field is highlighted in yellow to indicate that it is a field reference.

Selecting a reference in a response

When a response contains multiple matches for the regular expression that defines a reference, you can select which match is used subsequently as the data source. You can specify a particular occurrence, or you can specify a random occurrence.

About this task

An application under test might return responses that contain multiple matches for a regular expression that defines a reference. For example, a response might contain multiple links to rows of data, where each row represents a different user. You can control which occurrence of the regular expression is used as the data source in subsequent data correlation.

If you edit the **Regular Expression** that is associated with a reference, and then click **Verify** or **OK**, and the new regular expression still connects to the highlighted string in the preview window, then the **Specific occurrence number** is updated automatically, overwriting any changes.

1. In the Test Navigator, browse to the test, and double-click it. The test opens.
2. Locate the response that contains the reference that you want to specify.
3. In the **Content** field under **Test Element Details**, right-click the reference, and then select **Properties**.
4. **Optional:** To edit the regular expression that is used to locate the reference, click the **Toggle regular expression assistant** push button on the **Properties** window. The regular expression assistant displays the response content that is matched by the regular expression and the groups that are captured by the regular expression.
5. On the **Properties** page for the reference, select which **Occurrence** to use as the data source. By default, the first occurrence of a match for the **Regular Expression** is used as the data source.

Choose from:

- To specify a particular occurrence, select **Specific occurrence number**, and then type the number of the match. For example, type 4 to specify the fourth match of the regular expression in the response.
 - To specify a random occurrence, select **Random occurrence**.
 - To specify the last occurrence, select **Last occurrence**.
6. Click **OK**.

Result

The occurrence that you specified is used as the data source for data correlation when you run the test.

Viewing references

You can use the **Test References** window to view, modify, or verify references in a test.

1. In the Test Navigator, browse to the test, and double-click the test name. The test opens.
2. **Optional:** To view references in only part of a test, select test elements in **Test Contents** before continuing.
3. In the **Test Contents** area, click **Options**.
4. Select **Display References**.

Result

The **Test References** window opens. All references in the test or the selected test elements are displayed in tabular format.

5. Do one of these tasks:

Option	Description
<p>To view details about the references in a test</p>	<p>Navigate through the References table to preview the references in the Preview area. Click the Next and Previous icons to move the selection down or up in the list of references. Click the Show as Tree icon to toggle between tree format and list format. Click the Show Usage icon to view the substitution sites that are associated with each reference. Click the Bookmark icon to bookmark a location for later review.</p>
<p>To verify regular expressions that are associated with references</p>	<p>Select the check box next to each reference to verify, and then click Verify Checked. The verification procedure completes this procedure:</p> <ol style="list-style-type: none"> a. Checks that the regular expression finds the correct content. Regular expressions can stop working if you modify a test. b. Checks that there are no references with duplicate names. Troubleshooting data correlation problems is easier if references have unique names. c. Checks that there are no overlapping correlations.
<p>To find more locations in the test that have the same value as the selected reference</p>	<p>Select a reference, and then click Find and Substitute. These locations can be reviewed and substituted interactively as needed.</p>
<p>To modify a reference</p>	<p>Select a reference, and then click Properties.</p>
<p>To use the regular expression assistant to edit the regular expression used to locate a reference</p>	<ol style="list-style-type: none"> a. Select a reference, and then click Properties. b. Click the Toggle regular expression assistant push button on the Properties window. The

Option	Description
	<p>regular expression assistant opens. The response data that is matched by the current reference is automatically copied into the Test regular expression window.</p> <p>c. Click the Captured group tab in the regular expression assistant. The regular expression assistant displays the response content that is matched by the regular expression and the groups that are captured by the regular expression. If no groups are displayed, edit the regular expression accordingly.</p>
To remove a reference	<p>Select a reference, and then click the Clear Reference icon. The reference is removed from the test when you close the Test References window.</p>

- Click **Close** to close the **Test References** window and return to the test in the test editor.

When you close or save a test, you are prompted if any changes that you made to the test might affect the integrity of references in the test.

Correlating multiple fields in a test

Some tests are structured in such a way that you must correlate data for multiple fields. For example, assume that you plan to dataset an item that a virtual user is buying. For the test flow to be correct, you must also dataset all occurrences of that item in the test. You can find and correlate all instances of that item in one procedure. Typically, you use **Find More and Substitute** in the **Show Dataset Candidates** window to correlate data for multiple fields. See [Viewing dataset candidates when you open a test on page 419](#). Alternatively, you can use the **Test Search** page to correlate data for multiple fields.

To find all instances of a field in a test and correlate some or all of the instances with a data source, such as a dataset:

- In the Test Navigator, browse to the test, and double-click the test. The test opens.
- Locate the item or the substitution site to change or create a reference for. If the item is plain text, select the item. If the item is an existing reference, click the highlighted area.
- Right-click, and then click **Find > More Substitution Sites**.
- Click **OK**.
- On the **Test Search** page, select **Case sensitive** to perform a case-sensitive search or **Regular expression** to perform a search using regular expressions. In regular expression mode, press Ctrl+spacebar key in **Search for text** for content assistance. Content assistance lists the regular expression patterns and the content that they match.
- Click **More Options**, and then select the appropriate options:

Restrict to elements highlighted in Test Contents

Search only in elements that are selected in the **Test Contents** area.

Highlight found elements in Test Contents

Highlight found elements in the **Test Contents** area.

Recursive

Searches the child test elements in addition to the element. For example, if you search an HTTP page, select this option to search the requests and responses within the page.

Match encoded and decoded values (protocol-specific)

When selected, searches for matches of the unencoded and URL-encoded versions of the specified text. For example, when searching in HTTP data, `abc%123` and `abc%25123` match.

Include matches with overlapping data correlation

Include sites that are contained in, or overlap with, an existing substitution site. If you decide to substitute, the conflicting substitutions are automatically removed.

Include matching substituters

Click to return elements that originally matched the search string but have since been substituted. Clear to skip existing substitution sites when results are returned.


7. Click **Close**.
8. Click **Search**. The search results are displayed in the **Search** view.
9. In the **Search** view, select the matches to substitute, and then right-click the selection.
10. Optional: To select all matches, right-click the test name.
11. Click **Substitute in DataSource View**.

Result

This action sends the selected matches to the **Test Data Sources** window.

12. In the **Test Data Sources** window, click the tab that corresponds to the type of data source to use:

Option	Description
<p>References</p>	<p>The data sources that have already been created in the test and the possible matches that are available as data sources for the selected substitution site. For example, text that is found in a response and used in a subsequent request is automatically created as a reference. Clear the Matching References Only check box to see all the references that occur before the substitution site in the test.</p>

Option	Description
Datasets	The datasets that have been added to the test. To add a new dataset, click the Add new Dataset icon ().
Custom Code	If you have written test execution services (TES) custom code, the custom code that is available as data sources for this substitution site.
Test Variables	The test variables and their types that are available as data sources for this substitution site.
Built-in Datasources	The dynamically calculated data sources (Current Date , Random Number , Sequential Number , and Timestamp) that are available for this substitution site. To create a new built-in data source, right-click inside the Built-in Datasources page, and select New .

13. Select the data source, and click **Substitute From**.

Result

The **Substitute Multiple Items** window is displayed, showing information about the data source and substitutions sites that you selected.

14. For each site with a selected check box, click **Substitute Checked** to substitute the data source or clear the check box to skip the site.
Click **Always Prompt** to examine every substitution site one at a time. Click **Prompt on overlapping data correlations** to examine a site only if the site you are substituting into is contained in, or overlaps with, an another substitution site. If you decide to substitute, the conflicting substitutions are automatically removed.

Results

The selected instances of the field are correlated with the data from the data source.

Guidelines for adjusting data correlation

When you run a test, you might notice that the server is not under the expected load or that your database is not being updated as expected. Incomplete or incorrect data correlation can cause these problems.

To identify data correlation problems:

1. Use the **Potential Correlation Errors** view to find missing or incorrect data correlations. See [Finding data correlation errors on page 472](#) for more information.
2. Run a test individually or in a schedule with the **Log Level** for errors, failures, and warnings set to **All**.
3. After the run, open the test log as explained in [Viewing the test logs on page 799](#).
4. As explained in [Inspecting test log details in the Protocol Data view on page 651](#), verify that each call to the server returned the expected data.

The data correlation algorithms that are used during test generation are based on well known best practices. However, because these practices continually evolve, various types of errors can occur during automated data correlation:

- **Insufficient correlation:** Test values that must be correlated are not. Some possible causes follow:
 - Two parameters that must be correlated have different names.
 - A value must be correlated with a previous value that does not occur in the expected location.
 - A parameter or value must be correlated with a previous parameter or value that does not occur in the test because it is a computed value.
- **Superfluous correlation:** Unrelated test values are correlated.
- **Incorrect correlation:** Test values that must be correlated are correlated incorrectly.

Insufficient correlation: Parameters have different names or occur in unexpected locations

When two parameters that must be correlated have different names, automated data correlation does not recognize that the two parameters are related. For example, consider this request: `http://www.example.com?id=12345`. Suppose that this request must be correlated with the server response that contains `customer_ID=12345`, not `ID=12345`. In this case, the `ID` parameter must be correlated with `customer_ID`.

Data correlation typically links a response value that was returned from the server with a subsequent request value. The automated correlation algorithms search in the URL and the POST data for potential matches; however, other schemes for returning parameters are possible. For example, consider this request: `http://www.example.com?id=12345`. Suppose that this request must be correlated with the server response that contains the name and entity pair `href name="customer_ID" entity="12345"`, not `ID=12345`. In this case, the `ID` parameter must be correlated with `name="customer_ID"` and value `12345` must be correlated with `entity="12345"`.

Here are some additional causes of insufficient correlation:

- Siebel uses the star array format. Standard correlation algorithms can neither retrieve from this format nor substitute into this format.
- SOAP designates correlation parameters in external XML files. The correlation algorithms cannot correlate parameters in the external file with parameters in the test.

To manually correlate data in these cases:

1. In the test editor, use search or browse to locate the two parameters for correlation.
2. Navigate to the parameter that occurs later in the test, and select the parameter. This is the substitution site.
3. In the **Test Data Sources** window, click the **References** tab.
4. Select the data source to use as a reference, and then click **Substitute From**.

Insufficient correlation: One parameter is unnamed

Sometimes a parameter or value must be correlated with a previous parameter or value that is not named in the test, because it is computed, for example, by a JavaScript™ program. In this case, in order to correctly correlate the data,

you must understand how and where the parameter or value is computed, and then use a custom code block. See [Extending test execution with custom code on page 657](#) for more information about custom code.

For example, consider the web address `http://www.example.com?login_stamp=12345_Apr_11_07`, where the value for `login_timestamp` is the concatenation of the login ID and the current date. In this case, you must generate a custom code that concatenates the login ID and the date.

For another example, suppose that the server returned the login ID and date as separate entities: `href "customer_id=12345" Date="Apr_11_07"`. In this case, you can put these parameters in separate references and, in subsequent requests that use customer ID and date, substitute them separately.

Superfluous correlation

Automated data correlation is based on pattern matching: A parameter or parameter value is correlated with a subsequent parameter or parameter value with an exact or similar name. But sometimes parameters with exact or similar names are in fact unrelated. In the best case, unneeded correlation is either harmless or adds a slight load that is inappropriate. In the worst case, the application does not expect a correlation and fails during playback.

To remove a superfluous data correlation:

1. In the test editor, search or browse to locate the substitution site that must not be correlated. By default, purple letters indicate correlated data.
2. Right-click the substitution site.
3. Click **Remove Substitution**.

Incorrect correlation

A parameter that requires data correlation might occur many times throughout a test. For example, a session ID parameter that is used initially when a user logs in might also be used in every subsequent request. If multiple instances of a parameter in a test are not same, the correlation algorithms might use the wrong instance.

With the HTTP Test Generation preferences, you can optimize automatic data correlation for accuracy or for efficiency.

- **Accuracy:** Each occurrence of a parameter is correlated with the nearest previous occurrence. This is the default setting.
- **Efficiency:** Each occurrence of a parameter is correlated with a single previous occurrence.



Note: If you do not manually apply a correlation in the Referer field in an HTTP request header, then the Referer field is automatically correlated as needed. If you manually apply a correlation in the Referer field in an HTTP request header, then no automatic correlation is performed.

Incorrect correlations are more likely to happen when **Optimize automatic data correlation for execution** is set to **Efficiency**. To fix an incorrect correlation:

1. In the test editor, search or browse to locate the value that is incorrectly correlated.
2. Right-click the substitution site.
3. Click **Remove Substitution**.
4. Right-click the substitution site again.
5. Click **Substitute**, and select the correct parameter.

Generally, the HTML response content after the recording would look like `<input type="username" name="User" id="aaa" value="John"/>`. Some applications dynamically update the name attribute. So, when you play back the test the HTML response content would look like `<input type="username" name="idt020" id="aaa" value="John"/>`. Because the name attribute is changing dynamically, data correlation would not occur and the playback would fail.

Such correlations are the result of the tool using the *name* attribute as the basis for correlating other attributes in the response code instead of the *ID*. To correlate the responses based on ID, select **ON** in **Window > Preferences > Test > Test Generation > HTTP Test Generation > Data correlation types > Prioritize correlation based on ID**.

Finding data correlation errors

You can use the **Potential Correlation Errors** view to find missing or incorrect data correlations.

Before you begin

Run a test or a single-user schedule. The **Potential Correlation Errors** view does not support multiple-user schedules. If verification points fail while you are running a test, you are prompted to open the **Potential Correlation Errors** view when the test run is complete.

To find data correlation errors:

1. In the **Test Navigator**, select the result of the test run where you want to find correlation errors.
2. Right-click the result, and then select **Find Data Correlation Errors**. You can choose **Missing Correlation**, **Incorrect Correlation**, or **All**.
3. The **Potential Correlation Errors** view opens.

After the test log is processed, the view is populated. Depending on the size of the test log, it can take significant time to populate the view. The potential missing or incorrect data correlations are displayed, in descending order of the likelihood that the correlation is incorrect. Selecting an item in the **Potential Correlation Errors** view automatically selects the corresponding element in the test editor, so that you can fix the potential error.
4. Use the **Compare with Test Log** toolbar button in the upper-right corner of the view to compare the request or response in the test with the same object in the test log.
5. For missing correlations, use the **Suggest Fix** toolbar button in the upper-right corner of the view to search for other instances of the value in all responses in the test. If a matching value occurs in an earlier response in the test, create a reference in that response.

Disabling data correlation

You can disable a data correlation source or a substitution site. When you disable a data source, none of the substitution sites that use the source will be correlated when you run tests. When you disable a substitution site, only

that specific substitution site is disabled. Other substitution sites that use the same reference will be correlated when you run tests. You can also disable data correlation entirely for subsequent tests that you record.

To disable a data correlation source or substitution site:

1. In the **Test Navigator**, browse to the test and double-click it. The test displays in the test editor.
2. In the **Test Contents** area, click a request.
3. In the **Test Element Details** area, locate the data correlation source or substitution site.
4. Right-click the data value and select **Disable** from the menu.

To re-enable a disabled data source or substitution site, right-click the data value and select **Enable** from the menu.

Results

The data correlation source or substitution site is disabled.



Note: To disable data correlation for the entire workspace, click **Window > Preferences > Test Generation**, and clear **Enable automatic data correlation**. Subsequent tests that you record or regenerate will not include data correlation.

Re-correlating test data

If you disabled automatic data correlation before recording a test, you can regenerate the test with automatic data correlation enabled.

1. Click **Window > Preferences > Test > Test Generation**.

Result

The **Test Generation** preferences window opens.

2. Click the **Data Correlation** tab.
3. Select the types of data correlation to enable, and then click **OK**.
4. In the **Test Navigator**, browse to the test and double-click it. The test displays in the test editor.
5. Click **Edit > Re-correlate test data**.

Results

The test is regenerated with the types of automatic data correlation that you selected.

Data transformation

You can transform HTTP application specific data such as binary data and encoded data to a more readable format to use data correlation.

Viewing binary data

You can view binary data in tests. Use the binary editor to inspect test data, to determine if the binary data is of interest or should be transformed so that it can be correlated. You can also edit binary data in tests. To edit binary

data you need to have in-depth knowledge of the data format in question. Typically, you do not edit binary data in tests.

Before you begin

Record a test that contains binary data.

1. In the Test Navigator, browse to the test, and double-click it.

Result

The test opens.

2. Select a test element that contains binary data.
3. In the **Test Element Details** area, press the Ctrl key and click in the **Content** field. Alternately, type Ctrl+Shift+Space.

Result

The **Test Editor - Content** window opens.

4. Select the bytes to inspect, and then right-click to manipulate the selected data.

Option	Description
<p>Select</p>	<p>Use this page to programmatically select binary data by string or by specifying the number of characters to select.</p> <p>Click Null terminated string from caret to make a selection that starts at the current cursor position and ends at the next null character. If you select Select NULL character also, the null character is included in the selection.</p> <p>Click characters from caret, and type a number to select that number of characters starting from the current cursor position.</p> <p>Click Selection contains number of characters to select to select the number of characters specified by the current selection in the binary editor. For example, if you have 08 selected in the binary editor, this will select the next eight characters after the 08 byte. This control is not available if no data is selected in the binary editor.</p> <p>Click Sign to choose signed or unsigned data.</p> <p>Click Endianness to choose between big endian or little endian representation.</p>

Option	Description
Edit Integer value	<p>Use this page to edit data that you have selected in the binary editor. This page is available only when the binary editor is not in read-only mode.</p> <p>Select Update contents from editor selection to update the Value field when you change the selection in the Test Editor window. Clear this check box to prevent the Value field from being updated when you change the selection in the Test Editor window.</p> <p>Type the new data in the Value field. Select the appropriate base, such as decimal or hexadecimal, from the list.</p> <p>Click negate to negate the value of the selected data.</p> <p>Click Size to choose the size, in bytes, of the selected data.</p> <p>Click Sign to choose signed or unsigned data.</p> <p>Click Endianness to choose between big endian or little endian representation.</p> <p>The Preview area shows how the bytes will change in the binary editor after you click Apply or OK.</p>
Show Integer value	<p>Use this page to show the integer value of selected binary data in different formats. This page is available only when the binary editor is in read-only mode.</p> <p>Select the appropriate base, such as decimal or hexadecimal, from the list. The input field is not available.</p> <p>Click Size to choose the size, in bytes, of the selected data.</p> <p>Click Sign to choose signed or unsigned data.</p> <p>Click Endianness to choose between big endian or little endian representation.</p>
Binary Padding	Use this page to insert and overwrite binary data.

Option	Description
	<p>Type the data to insert in the Pad with field. Select the appropriate format, such as bytes or ASCII, from the list.</p> <p>Click Pad selection only to replace the bytes that you have selected in the binary editor. This control is not available if you have not selected any bytes in the binary editor.</p> <p>Click Number of occurrences, and then type the number of times to repeat the binary padding.</p>
Go to Offset	<p>Use this page to move the cursor to a different position in the binary data.</p> <p>Type the offset in the Enter offset field.</p> <p>Select Make selection with previous and new offset so that the bytes between the current cursor position and new cursor position are selected when you click OK.</p> <p>Click Absolute or Relative to choose an absolute offset or relative offset. An absolute offset starts with the first byte of data. A relative offset is measured from the current cursor position.</p> <p>Click Forward or Backward to choose the direction for relative offsets. This control is not available for absolute offsets.</p>
Find/Replace	<p>Use this page to search for binary data and to replace binary data, if necessary. This page is available only when the binary editor is not in read-only mode.</p> <p>Type the data to search for in the Find field. Select the appropriate format, such as bytes or ASCII, from the list.</p> <p>Type the replacement data in the Replace field. Select the appropriate format, such as bytes or ASCII, from the list.</p>

Option	Description
	<p>Click Forward or Backward to choose the direction to search from the cursor position.</p> <p>Click All or Selection to choose between searching all of the binary data or searching only the data selected in the binary editor.</p>
Find	<p>Use this page to search for binary data. This page is available only when the binary editor is in read-only mode.</p> <p>Type the data to search for in the Find field. Select the appropriate format, such as bytes or ASCII, from the list.</p> <p>Click Forward or Backward to choose the direction to search from the cursor position.</p> <p>Click All or Selection to choose between searching all of the binary data or searching only the data selected in the binary editor.</p>
Encodings	<p>Select from the list the encoding to use for displaying binary data.</p>
Read-only	<p>Click Read-only to toggle between read-only and writable states. If the binary data contains a data-correlation reference, it is read-only and cannot be changed to writable.</p>
Paste	<p>This page is displayed only if you attempt to paste data from the clipboard into the editor and the data on the clipboard includes characters that are not valid in the current encoding scheme. For example, this page is displayed if you attempt to paste accented characters when the encoding is set to ASCII.</p> <p>Click Paste only the valid character to paste only the characters that are valid in the current encoding scheme. All characters that are invalid in the current encoding scheme will be discarded.</p>

Option	Description
	Click Replace invalid character by 00 byte to paste the string from the clipboard, replacing all characters that are invalid in the current encoding scheme with a null character.

Transforming binary data in tests

You can transform binary data in tests to view the data in a more readable format and to assist with data correlation. Binary data in a test might contain values that must be correlated for the test to play back properly.

Before you begin

Record a test that contains binary data.

To transform all the binary data in a test:

1. Open a test for editing.
2. Click **Edit > Re-correlate test and transform data**.
3. In the **Data Correlation and Transformation** window, select the **Apply data transformation** check box.
4. From the list, select a transformation to perform:
 - To convert the binary data to the equivalent ASCII representation, select the **Convert To Ascii Text** check box. Characters that are not part of the readable ASCII character set are shown as escaped hexadecimal values.
 - To convert the binary serialized representation of a Java™ object into XML format, select the **Convert Java Serialized Object to XML** check box. You can also use this transformation to examine and perform data correlation on tests against applications that use serialized Java™ objects.
 - To transform Adobe Flex objects into XML format, select the **Convert AMF to XML** check box.
 - To transform the Windows Communication Foundation binary code into XML, select the **Convert Microsoft WCF-Binary to XML** check box.

Result

All binary data in the test is transformed by using the transformation that you specified.

5. To apply built-in data correlation rules, select the **Apply corresponding data correlation rules if transformation succeeds** check box, and then click **Finish**.



Note: To remove data transformation from the test, clear the check boxes in the list of available transformers and generate the test.

Transforming binary data in specific requests

You can transform binary data in requests to view the data in a more readable format.

Before you begin

Record a test that contains binary data.

1. Open a test for editing.
2. In the **Test Contents** area, select the request that contains binary data to transform.
3. Scroll to the bottom of the **Test Element Details** area, and then click the **Advanced** tab.
4. In the **Data Transformation** area, click **Change**.

The current data transformation is displayed in the **Applied Transform** field in the **Data Transformation** area.

Result

The **Select Transformation Adapter** window opens.

5. Select the data transformation to perform. To remove data transformation from the request, select **[none]**.
If the application under test uses serialized Java™ objects, do not transform individual requests. Instead, configure the product to automatically apply the **Convert Java Serialized Object to XML** transformation to all binary data.

Result

The binary data in the request is transformed using the specified transformation adapter.

Viewing or transforming GWT encoded data

When you record an application that is developed on Google Web Toolkit(GWT), the recorded data is encoded. To view the encoded data, open the test editor. You can also transform the encoded data into XML format, which you can use for data correlation.

Before you begin

- Record an application that is based on GWT. For supported GWT versions, see the download document at https://hclpnpsupport.hcltech.com/kb_view.do?sysparm_article=KB0069531.
- You must have access to the classes that contain the exchanged objects.
- You must add the JAR file that contains the classes that are serialized and exchanged between client and server to your project. To add the JAR file, in the **Test Navigator** view, right-click the project, click **Properties**, and, on the **Library** tab add the JAR file.

1. To view the encoded data, in the Test Navigator, browse to the test, and double-click it.
2. Select a test element or request that contains the encoded data.

Result

The **Content** field in the Test Element Details area contains the encoded data.

3. To transform the encoded data, click **Edit > Re-correlate test and transform data**.
4. Confirm that the **Apply data transformation** check box is selected and select **Convert GWT RPC to XML**.
5. **Optional:** To use data correlation after the transformation is successful, select the **Apply corresponding data correlation rules** check box.
6. Click **Finish**.

Result

The encoded data transforms into XML format.

Transforming GraniteDS or BlazeDS data

To test an application that is developed on Granite Data Services (GraniteDS) or Adobe BlazeDS framework, you must deserialize or transform the objects that are encoded in the AMF format to the XML format.

Before you begin

- HCL OneTest™ Performance supports GraniteDS 2.3.2.
- For GraniteDS transformation, you must add the `lt-granite.jar` and `granite.jar` to the class path, along with any other JAR files that are required for the deserialization process.
- For BlazeDS, you must add the JAR files that are required for the deserialization process to the class path. To add `lt-granite.jar` or any other JAR files to the class path perform the steps as follows:
 1. Right-click the project in the Test Navigator view, and then click **Properties**.
 2. Click the **Libraries** tab, and then click **Add External JARs**.
 3. Browse to `PathToTheProduct\HCLIMShared\plugins`
`\com.ibm.rational.test.lt.datatransform.adapters_VersionNumber` to select `lt-granite.jar` or to any other folder to select other JAR files.
- You must set the JAR files and ensure that you have the following order by clicking the **Order and Export** tab:
 1. JRE.
 2. Plug-in Dependencies.
 3. `lt-granite.jar`.
 4. Any Application JAR files.
 5. BlazeDS JAR files if you use BlazeDS. For instance, `flex-messagin-common.jar` and `flex-messaging-core.jar`.
 6. `granite.jar`.

To transform the GraniteDS or BlazeDS data:

1. Record the test.
2. On the toolbar, click the **Re-correlate test and transform data** icon.
3. Click the **Convert Granite Data Service (AMF) to XML** check box and click **Finish**.

Result

The AMF data is transformed to XML.

What to do next

You can now correlate the data.

Transforming SAP Web DynPro XML to ASCII

To test applications that are built using the SAP Web DynPro framework and correlate data, you must transform data to the ASCII format.

About this task

By default, when you generate a test that contains Web DynPro XML, HCL OneTest™ Performance transforms data to the ASCII format. If the data is not transformed automatically, you must manually select the transform option and rerun the test.

To manually select the transform option:

1. In HCL OneTest™ Performance, click **Window > Preferences > Test > Test Generation**.
2. Select the **Convert SAP Web DynPro XML Encoding to Ascii** check box.
3. Click **OK**.

Creating custom data transformations

You can create your own data transformations to transform binary data in tests. Data transformations can convert binary data to a more readable format and can assist with data correlation.

Before you begin

To create a data transformation, you must be familiar with developing Java™ methods in Eclipse.

To create a data transformation:

1. Click **File > Switch Workspace > Other**, and then type a workspace name to create a new workspace.

Result

The product restarts in the new workspace.

2. Extract the `MyTransformProject.zip` archive file.

The `MyTransformProject.zip` archive file contains the `MyNewDataTransformFeature` and `MyTransformProject` projects. By default, this file is installed in the `C:\Program Files\HCL\HCLIMShared\plugins\com.ibm.rational.test.lt.sampleversion_date\installdirectory`.

3. Import the `MyNewDataTransformFeature` and `MyTransformProject` projects into the new workspace.

See [Importing existing projects](#) for more information about importing projects.

4. Click **Window > Open Perspective > Resource** to open the **Resource** perspective.
5. Open the `MyTransformer.java` file for editing.
6. Implement the `transformData` and `unTransformData` methods to create your data transformations.
7. **Optional:** If you need other `.jar` files to implement your transformations, add the `.jar` files to a user library.

- a. Right-click the `MyTransformProject` project in the **Resources** view, and then click **Build Path > Configure Build Path**.

- b. Click the **Libraries** tab.

- c. Click **Add Library**.

Result

The **Add Library** window opens.

- d. Select **User Library**, and then click **Next**.

- e. Click **User Libraries**.

- f. Click **New**, and then type a library name.
- g. Click **Add JARs**, and then select the `.jar` files to add.

Result

The code in the `.jar` files is now available to the MyTransformProject project.

8. Start another copy of the workbench to debug your transformations.

To learn more about debugging, see [Local Debugging](#) in the Eclipse documentation.

 - a. Click **Run > Debug Configurations**.
 - b. Select **Eclipse Application**.
 - c. Click the **New launch configuration** icon.
 - d. Edit the new configuration, and then click **Debug**.
9. In the copy of the workbench started for debugging, record a test, and then open the test for editing.
10. Click **Edit > Data Transformation**.

Result

The **Select Transformation Adapter** window opens. The MyTransformer transformation is listed along with the default transformations supplied with the product.

11. Select the MyTransformer transformation, and then click **OK**.

Data correlation is removed when you apply a transformation. To correlate the test data again, click **Edit > Re-correlate test data**.
12. Right-click MyNewDataTransformFeature, and then click **Export**.
13. Select **Plug-in Development > Deployable features**, and then click **Next**.

Result

The **Deployable features** window opens.

14. Select MyNewDataTransformFeature.
15. On the **Destination** page, type or click **Browse** to specify a directory, and then click **Finish**.

Result

The installable feature is exported to the specified directory. The exported files can be compressed into archives to make it easier to copy them to other computers.

Using custom data transformations

After you have created a custom data transformation, you can apply it to binary data in tests. To use a custom data transformation, you must create and install the corresponding feature.

Before you begin

Create a custom data transformation.

To use a custom data transformation:

1. To install the feature on computer different from the computer where you created the transformation, click **Help > Install New Software**.
2. On the **Available Software** window, click **Add**.

Result

The **Add Repository** window opens.

3. Click **Local**.
4. Navigate to the installable feature, and then click **OK**.
5. On the **Available Software** window, clear the **Group items by category** check box.

Result

MyNewDataTransformFeature is displayed in the list of available software.

6. Select MyNewDataTransformFeature, and then click **Finish**.

Results

Your custom data transformation is available when you click **Edit > Data Transformation**.

Compound tests

You can create compound tests to help you organize smaller tests into scenarios that can then be run end-to-end. You can combine tests from different extensions to achieve end-to-end flow.

If you need to combine various tests into a single workflow or end-to-end scenario, you can organize the tests into a compound test. Each test may perform a part of the scenario. Each test may also run in a different domain, for example, different web browsers. A typical example of a compound test is an online buying workflow. You may have built smaller tests for each part of an online purchase transaction, such as "log on", "log out", "view item", "add to cart", and "check out". You can combine these tests into a single flow in a compound test. When the compound test is run, its individual tests are run in sequence.

The types of tests you can combine into a compound test depend on the testing capabilities you have purchased. You can also shell-share HCL OneTest™ Studio family products to add multiple tests into a compound test.

To build the scenario you require in a compound test, you can also add the following annotations:

- Comments
- Synchronization points
- Loops
- Delays
- Transaction folders
- IF-THEN-ELSE
- Tests that are mandatory, using the **Finally** blocks
- Tests to be run in random order, using the **Random Selector**

Creating a compound test

You can create compound tests to help you organize smaller tests into scenarios that can then be run end-to-end. You can combine tests from different extensions to achieve end-to-end flow.

1. Create a test workbench project.
2. In the Web UI Test perspective, in the Test Navigator, right-click the test workbench project and click **New**, and then click **Compound Test**.

3. In the **New Compound Test** dialog box, specify the name of the compound test and the location where it must be stored. By default, the test is stored in the workspace of the test workbench project you selected. You can select a different project location if desired.

Result

The file extension `testsuite` is added to the file name, and the new compound test is added to the Compound Tests folder of the test workbench project, visible in the Logical View. The new test is also visible in the Resource View, under the test workbench project. The contents and test element details are displayed in the compound test editor in the right panel.

4. In the compound test editor, add the components of the compound test.

The types of tests you can combine into a compound test depend on the testing requirements and on the components that you have licensed. For example, if you have the appropriate licenses, you can add Web UI tests, performance tests, mobile web tests, and functional tests into a compound test.

5. To build the scenario you require in a compound test, you can also add the following annotations by clicking **Add** and selecting the appropriate option:

- Comments
- Synchronization points
- Loops
- Delays
- Transaction folders
- Tests that are mandatory, using the **Finally** blocks
- Tests to be run in random order, using the **Random Selector**

6. Save your changes.

Viewing compound tests

You can view a compound test in the Compound Test Editor.

About this task

When you open a workspace, the tests and projects that reside in the workspace are listed in the Test Navigator.

You can view compound tests in the Logical and Resource Views in the Test Navigator. From any of these views, you can open the test in the Compound Test Editor.

- In the Logical View of the Test Navigator, compound tests are listed in the Compound Tests folder under the project into which they were imported. Double-click the compound test under the Compound Tests folder to open it in the Compound Test Editor.

Result

In the Resource View, all tests under a project are shown in the project folder. Double click the compound test under the project folder to open it in the Compound Test Editor.

- In the Java perspective, compound tests under a project are shown under the root project folder. Double click the compound test under the project folder to open it in the Compound Test Editor.

- The Compound Test Editor contains two panels - the **Compound Test Elements** panel, where the elements of the workflow are listed. Click one of the elements, and its details are displayed in the far right portion of the right panel, which is the **Compound Test Element Details** panel. Double-click any of the test or the test elements to view its details. The name of the test, test path, source type and execution mode are displayed.

Adding tests into a compound test

After creating a compound test, you can add the smaller test pieces that contribute to the larger workflow you are constructing with the compound test. When you run a compound test, each of the tests added to it are invoked in the sequence defined.

You can add many tests of the same type, or different types, to a compound test, depending on the testing requirements.

To add tests to a compound test, complete these steps:

1. In the Test Navigator, double-click the compound test to which you want to add a test. The contents of the compound test are shown in the **Compound Test Contents** panel in the Compound Test editor.
2. Do one of the following:
 - Click **Add** to add a test as the first element in the compound test.
 - To insert a test before a specific element in the compound test, select the element and click **Insert**. The **Select Tests** dialog box is opened, and the tests found in the Eclipse Client workspace are displayed.
3. Select the test you want to add to the Compound test, and click **OK**. The test is added to the compound test, and is displayed as part of the elements of the compound test in the **Compound Test Contents** panel. When you click the test you added, its details are displayed in the **Compound Test Element Details** panel in the Compound Test editor.
4. Save your changes.

In addition to the tests that you can add to a compound test, you can also add the following elements to construct the workflow you need:

- Comments to document the test
- Delays in the test
- Synchronization points
- Loops
- Transaction folders
- Parts of the test that are mandatory
- Tests to be run in random order

Modifying a compound test

You can modify a compound test in the Compound Test Editor.

About this task

A compound test is a testing workflow comprising smaller tests and other test elements in a certain sequence. You might want to order the tests and test elements to suit your workflow requirement, or add further tests and elements.

1. In the Test Navigator, double-click the compound test that you want to modify. Its elements are shown in the **Compound Test Contents** right panel in the Eclipse Client.
2. To add a test or test element at the beginning of the compound test elements list, select the compound test in the **Compound Test Contents** panel, click **Add**, and then click **Test**. To insert a test or test element into the test, select the test element before which the insertion must be made, and click **Insert**.
3. Add or insert the test or test element you need, and click **OK**. The modified compound test displays its updated elements in the **Compound Test Contents** right panel.
4. Save your changes.

Running compound tests

When you run a compound test, its test elements are run in the order defined in the compound test.

About this task

When you run a compound test, you are prompted to open the Test Execution perspective, in which details of the test run are displayed. When the test run is complete, the Test Log displays the run results.

Prior to 9.2, text execution would terminate on a fatal exception in any of the tests in a compound test. Starting from 9.2, there is a new preference to allow text execution for a compound test to continue after a fatal exception in one of the tests. To set the preference, see **Window > Preferences > Test > Test Execution > Error handling > Mobile or Web UI Fatal Error**.

1. In the Test Navigator, select the compound test to run.
2. Click the Run As icon on the toolbar. The test runs. To run a launch configuration option, click the arrow beside the Run As icon and select Run Configuration. Select a configuration option and run the test.

Result

The **Confirm Perspective Switch** dialog box is opened, prompting you to switch to the Test Execution perspective. Click **Yes**.

3. Select an option to run the test.

Result

The Test Execution perspective is opened and the test runs. On completion, the test log is displayed.

Results

You can work with the test log by exporting it into a flat file.

Generating compound test result reports

When a compound test run is completed, a Test Log is shown in the Test Execution perspective. You can work with the information in the test log and also generate test result reports.

Exporting the Test Log

When a compound test run is completed, a Test Log is displayed in the Test Execution perspective.

About this task

The Test Log displays the following details:

- The General Information tab displays the name of the compound test and its description. The location of the test log file is also shown.
- The Common Properties tab shows the verdict of the test results.
- The Verdict Summary and Verdict List tabs provide a pie chart of verdicts for different components of the test, and a list of the first 20 verdicts. You can view details about the verdicts by clicking the links in the Verdict List tab.

You can export the contents of the test log to a full-text file.

1. To export the contents of the test log to a full-text file, right-click the test run result under the Results folder of the compound test, and click **Export Test Log**.
2. In the **Export Test Log** dialog box, specify where the test log should be exported to, in the **Location** field.
3. Select the format in which the log must be exported, from the list in the **Export Format** field. You can select either Flat Text - Default Encoding or Flat Text - Unicode Encoding.
4. Click **Finish**.

Result

The test log is exported as a full-text file, with the test results run name, to the location you specified.

Generating a functional test report

You can generate a functional test report from the test run results as a HTML file.

About this task

When you generate a functional test report as a HTML file, the following details are displayed in the report:

- A global summary, which lists the number of tests run, verification points, defects
- A test summary which displays the name of each test, the start and end times and the verdicts.

1. Test run results are displayed under the Results folder of a project. Right-click the test run result you want to view and click **Generate Functional Test Report**.

Result

The **Generate Functional Test Report** dialog box is opened.

2. Select the parent folder in which the report must be stored.
3. By default, the name of the compound test and the date and time stamp is displayed as the name of the report in the **Name** field. You can change the name.
4. Click **Next**.

5. Select the report template to be used. If you select the Common Functional Test Report (XSL) format, the report is generated as a HTML file. If you select the Common Test Functional Report format, you can select either the HTML or PDF output format.
6. Click **Finish**.

Result

The report is generated and displayed. The report is listed under the Functional Reports folder under the compound test in the Test Navigator.

Creating an executive summary

You can create an executive summary or test statistics report from the test run results. Executive summaries are generated according to the type of test.

About this task

An executive summary displays the tests and methods that were run, and their success or failure information. This information is shown in summary charts as well as in bar graphs.

1. Under the Results folder of the project, right-click the test run result you want to view and click **Create Executive Summary**.

Result

The **Generate Functional Test Report** dialog box is opened.

2. Select the type of test report you want to generate.
3. Click **Finish**.

Result

The report is generated and displayed. The report is listed under the Functional Reports folder under the compound test in the Test Navigator.

Adding a compound test to a Test Workbench project

You can create a compound test in a test workbench project. If you have an existing compound test, you can import the test to a test workbench project.

Creating a compound test in a test workbench project

You can create a compound test in a test workbench project.

1. Create a test workbench project.
2. In the Web UI Test perspective, in the Test Navigator, right-click the test workbench project and click **New**, and then click **Compound Test**.
3. In the **New Compound Test** dialog box, specify the name of the compound test and the location where it must be stored. By default, the test is stored in the workspace of the test workbench project you selected. You can select a different project location if desired.

Result

The file extension `testsuite` is added to the file name, and the new compound test is added to the Compound Tests folder of the test workbench project, visible in the Logical View. The new test is also visible

in the Resource View, under the test workbench project. The contents and test element details are displayed in the compound test editor in the right panel.

4. In the compound test editor, add the components of the compound test.

The types of tests you can combine into a compound test depend on the testing requirements and on the components that you have licensed. For example, if you have the appropriate licenses, you can add Web UI tests, performance tests, mobile web tests, and functional tests into a compound test.

5. To build the scenario you require in a compound test, you can also add the following annotations by clicking **Add** and selecting the appropriate option:
 - Comments
 - Synchronization points
 - Loops
 - Delays
 - Transaction folders
 - Tests that are mandatory, using the **Finally** blocks
 - Tests to be run in random order, using the **Random Selector**
6. Save your changes.

Importing a compound test into a Test Workbench project

You can import a compound test into a test workbench project.

1. In the Web UI Test perspective, in the Test Navigator, right-click the test workbench project into which you want to import the compound test and click **Import**.
2. In the **Import** dialog box, expand **General** in the source list, select **Import test assets with dependencies** and then click **Next**.
3. Specify the directory in which the compound test resides. Click **Browse**.

Result

By default, the compound test is imported into the test workbench project folder.

4. The compound test assets in the folder you selected are displayed. Select the components you want to import.
5. Click **Finish**.

Result

The imported compound test is displayed in the **Compound Test Elements** panel in the Compound Test editor.

Adding compound tests to schedule

To test the performance of multiple tests, you can add all the tests to a compound test and add the compound test to a user group or a rate runner group. When you run a schedule, all the tests in the compound test are run in a sequential order.

To add a compound test:

1. In the schedule editor, add a user group or a rate runner group.
2. Select the group and in **Behavior**, click **Use compound test**.
3. Select a compound test and click **OK**. If there are no compound tests in the project, click **Create**, specify a name for the compound test, and click **Finish**. If there are test variables associated with a compound test and also defined in the schedule, the variables with the compound test take precedence while running the schedule.
4. Save the schedule.

Related information

[Schedule overview on page](#)

Simulating services with stubs

Service stubs enable you to simulate the behavior of an actual service for a wide variety testing or integration purposes.

Service stub overview

Service stubs are simulations of an actual service, which can be used to functionally replace the service in a test environment. A stub server replaces the actual application server.

From the point of view of the client application, the service stub looks identical to the actual service that it simulates. To use a service stub in replacement of the actual service, you must be able to replace the URL of the original service in the client application with the URL of the stub server.



Important: For version 8.7 and later, you cannot use the schedule option of HCL OneTest™ Performance to deploy stub servers remotely. If you have already deployed stub servers remotely, you must install IBM® Rational® Service Tester for SOA Quality or HCL OneTest™ Performance on those computers and then deploy the stub servers locally.

Use case examples

There are several cases where it can be useful to deploy a stub services instead of using the actual services for your tests:

- If you are testing a local service that uses data from another remote service, you might need to inject specific content to the service under test from the remote service. You can simulate the remote service with a service stub to ensure that the local service responds properly to some specific input.
- Some commercial services charge users for each call. If you are testing such a service, you can develop and debug your test against a stub service, which is based on the WSDL of the actual service, without being charged by the commercial service.
- During integration of a large application involving multiple clients and services, some services might not yet be operational, although their WSDL specifications are available. You can simulate the missing services with service stubs, which will allow you to proceed with the integration work.

Service stub architecture

You create a service stub by providing an existing WSDL specification. The service stub is generated with the exact same ports and bindings as the original service so that it can be addressed with exactly the same interface. Each operation in the service returns a default response of the type defined by the WSDL.

You can edit the service stub in the stub editor to change the default response or to create conditional responses that simulate the actual responses of the original service.

When you have finished editing the service stub, you can deploy it on a local stub server, which runs in the workbench. The stub server simulates an actual application server and can host multiple service stubs. You control the stub server from the stub monitor view.

Finally, to use the service stub instead of the original service, you change the URL used by the client application to point to the local stub server instead of the original application server. This URL, as well as the WSDL of the service stub, is provided in the stub monitor view.

Creating a service stub


You can use a WSDL (Web Service Description Language) specification file to generate a service stub that can simulate the behavior of the original service and uses the exact same interface.

Before you begin

Service stubs are stored in test projects. If your workspace does not contain a test project, the test creation wizard creates one, enabling you to change its name. To store a service stub in a specific project, verify that the project exists before you create the stub.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

The wizard can import WSDL files from the workspace, the file system, a remote repository, or from a URL. Ensure that the WSDL files use the correct syntax for the test environment. Service stub generation might not work with some Web Services Description Language (WSDL) files.

1. In the workbench, click **File > New > Other > Test > Test Assets > Service Test** or click the **New Service Stub**  toolbar button.
2. Select the WSDL of the service that you want to simulate. If necessary, you can import the WSDL from the file system, a URL, or a WSRR or UDDI repository.
3. Click **Next**.
4. Select a project location and a name for the new service stub. Click **Finish**.

Results

The wizard generates a working service stub that reproduces the interface of the original service as defined in the WSDL specification. Each operation is reproduced with a default response. You can edit the service stub with the stub editor to change the default response or to create conditional responses.

Editing a service stub

Service stubs are generated with a single default response for each operation in the WSDL specification. You can edit the service stub to change the default responses or to add conditional responses that can simulate the actual service.

To edit the behavior of a service stub:

1. In the test navigator, double-click the stub to open the stub editor.

Each operation simulated by the stub is represented by an operation element, which contains **Case** elements that describe a condition. Each case contains a response element. Case elements are similar to test verification points and use the same presentation.
2. To change the default response of an operation:
 - a. Expand the operation and the **Case : Default** element, and then select the response element.

The Case : Default element describes the response of the service stub when no other case condition is met.
 - b. Edit the **Message** content to specify the XML content returned by the service stub.
3. To add a conditional response case:
 - a. Right-click the operation and select **Add > Equals Case, Contains Case, or Query Case**.

These conditional case types are similar to the *Equals*, *Contain* and *Query* verification points in service tests.

 - Use **Equal Case** to specify a response that is returned by the stub when the entire incoming message content fully matches the specified message content.
 - Use **Contains Case** to specify a response that is returned by the service stub when a portion of the incoming message content matches the specified message content.
 - Use **Query Case** to specify a response that is returned by the service stub when an XPath query meets the specified criteria.

You can add as many case elements as necessary to simulate the behavior of the original service. Use the **Up** and **Down** buttons to change the order in which the case conditions are evaluated. Only the first matching condition is executed.

The default case cannot be removed and is always the last case element in the operation.

- b. Select the response element and edit the **Message** content to specify the XML content returned by the service stub. Use the **Form**, **Tree**, and **Source** views to change the XML content display mode.
4. Select **File > Save** or click the **Save** toolbar button.

What to do next

When you have finished editing the service stub, you can deploy the stub to a stub server.

Deploying service stubs

You deploy and run service stubs on a stub server, which is a small application server dedicated to running service stubs. The client application, or test, addresses the stub server instead of the actual application of the original service.

Before you begin


The local stub server runs in the workbench on the local computer. Service stubs can be accessed locally. The local stub server is automatically stopped when you close the workbench.

To use a service stub instead of the original service, you must be able to change the endpoint of the client application or service test to replace the URL of the original application with the URL of the stub server.

1. In the stub editor, click the **Deploy** button.
Alternatively, you can right-click the stub in the test navigator and select **Deploy On > Local stub server**

Result

This opens the **Stub Monitor** view.

2. In the **Stub Monitor** view, click  **Run**.
If you make any changes to the service stub, the stub is redeployed to the stub server after saving.
3. To add more service stubs to the stub server, click **Add** and select a service stub from the workspace.
4. Copy the URL of the service stub from the **Stub Monitor** view and paste it into the configuration of the client application.
You can also directly access the WSDL specification of the service stub, which is a copy of the original WSDL with replaced URL endpoints.

What to do next

You can validate that the service stub is responding correctly by using the generic service client to invoke a call.

Recording service stub activity in a log file

With service stub logging, you can monitor the interactions between an application and the stub server. When the option is enabled, one log file is created for each deployed stub. The log files are presented as a formatted HTML report.

Before you begin

You must have created one or several service stubs.

To log service stub activity:

1. Add the following virtual machine (VM) argument to the `eclipse.ini` file: `-DSTUB_LOG_LEVEL=log_level`.

Use one of the following values for the `log_level` variable:

- 0: Disable the log.
- 1: Log stub activity without details.
- 2: Log stub activity including content of sent and received messages.
- 3: Same as level 2 with HTTP headers of received messages.
- 4: Same as level 3 with attached files.

You can also add the following optional arguments:


- `-DSTUB_LOG_KEEP_PREVIOUS=true`: This option creates a separate log file each time the service stub is redeployed. If the value is not `true` or if the option is not present, the log file is erased if the service stub is redeployed or when the stub server is stopped.
- `-DSTUB_LOG_SERIALIZE_XML=true`: This option displays the XML content (with log levels 2, 3, and 4) without formatting or indentation. If the value is not `true` or if the option is not present, the XML content is formatted and indented in the log.

The `eclipse.ini` file is located in the same directory as the `eclipse.exe` launcher binary file that is used to run the product.

Example

For example, to enable logging with basic content, add the following line to the end of the `eclipse.ini` file:

```
-DSTUB_LOG_LEVEL=2.
```

2. Restart the workbench, and in the **Stub Monitor** window, click the **Run** icon  to restart the stub servers.
3. If the server was launched by a schedule in the performance testing application, then corresponding logs are automatically created in the workspace. If not, complete the following steps to retrieve the log files from the stub server:



Important: The stub server must be running.

- a. After running your tests, to view the service stub log files, open the **Stub Monitor**, and click the tab for the stub server.
- b. Click the **Synchronize** toolbar button for the selected server.

Result

An HTML log file is created and displayed for each deployed service stub.

Result

The stub log reports are located in a folder named `stubLogs`, which is in the same folder as the corresponding service stub.

Setting log level for service stubs


While recording a service test, you can set the level of the log details that you want to collect for debugging purposes.

Before you begin

You must stop the stub server.

About this task

The log level that you set in this way takes precedence over the log level setting that you specify in the [eclipse.ini on page 494](#) file.

1. In the **Stub Monitor** view, in the Service Stubs section, click the **Edit log options** icon .
2. Select one of the log level options and click **OK**.

What to do next

Start the server again for the changes to take effect.

Sending service requests with the generic service client

The generic service client enables you to send requests to services for which you do not have a convenient client and to view the responses returned by the service.

Creating transport protocol configurations

Read these topics to configure various transport protocols.


Creating an HTTP transport configuration

You can create an HTTP transport configuration that describes the transport settings for a service request. Transport and security settings can be associated with any service request.

Before you begin


If you are using Secure Sockets Layer (SSL) authentication, ensure that you have valid key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

1. Click the **Generic service client**  toolbar button to open the generic service client and click the **Transport** tab.

Result

This opens the **Transport Configurations** page.

2. On the **Transport Configurations** page, click **Create an HTTP configuration**  to create a new HTTP transport configuration.
3. Type a **Name** for the new transport configuration.
4. Specify the following options for the HTTP transport:

HTTP/2



Note: Testing HTTP/2 service is in the Beta mode. For more information, see [Preparing to record a HTTP/2 service on page 235](#).

To test a service that uses the HTTP/2 protocol, select the **Activate** check box. This check box is automatically selected when you record a service by using a browser. If you use the Generic Service Client component to create a HTTP/2 test, you have to manually select the check box.

HTTP/2 client connection timeout

Specifies the time limit for the HTTP/2 client to connect to the HTTP/2 server.

Time out for the HTTP/2 session creations

Specifies the time limit to create the HTTP/2 session. This time starts after the connection is established.

Enable HTTP/2 Push

The Push functionality of HTTP/2 automatically identifies and passes the related objects or requests to the client when a request is sent to the server. Clear the check box to not use the functionality.

Initial session window

Specifies the buffer size on the sessions.

Initial stream window

Specifies the window size for buffer on each stream after the connection is established.

HTTP/2 Client Input Buffer Size

Specifies the buffer size that is used to read the network traffic.

Maximum Quantity of Messages that can be queued

Specifies the maximum number of messages that can be queued for the HTTP/2 client on a thread.

Maximum Quantity of HTTP/2 thread pool

Specifies the maximum number of thread pools that will be used by the HTTP/2 client to distribute the workload.

Minimum Quantity of HTTP/2 thread pool

Specifies the minimum number of thread pools that will be used by the HTTP/2 client to distribute the workload.

HTTP/2 client bytearray pool size

Specifies the buffer size to receive the unciphred values.

Server Name Indication

Note: Not applicable for HTTP/2.

Clear this check box if you do not want to connect to the host computer by using the Server Name Indication protocol. If the host computer is already configured with Server Name Indication protocol, you should keep this check box selected.

Use HTTP Keep Alive

Select this option to keep the HTTP connection open after the request. This option is not available if you are using IBM® Rational® AppScan®.

Use SSL

Select this option to use an SSL configuration. Click **Configure SSL** to create an SSL configuration or select an existing configuration.

Platform Authentication

In this section, specify the type of authentication that is required to access the service. Select **None** if no authentication is required.

Basic HTTP authentication


Select this option to specify the **User Name** and **Password** that are used for basic authentication.

NTLM authentication

Note: Not applicable for HTTP/2.


Select this option to use the Microsoft™ NT LAN Manager (NTLM) authentication protocol. NTLM uses challenge-response authentication. This view lists what is negotiated (supported by the client and requested of the server) and what is authenticated (the client reply to the challenge from the server).

Kerberos authentication

 **Note:** Not applicable for HTTP/2.

Select this option to use the Kerberos authentication protocol between the client and server.

Connect through proxy server

 **Note:** Not applicable for HTTP/2.

If the HTTP connection needs to go through a proxy server or a corporate firewall, specify the **Address** and **Port** of the proxy server. If the proxy requires authentication, select either **Basic proxy authentication** or **NTLM proxy authentication**.

Proxy authentication

In this section, specify the type of authentication that is required to access the proxy. Select **None** if no authentication is required.


Basic proxy authentication

Select this option to specify the **User Name** and **Password** that are used for basic authentication.

NTLM proxy authentication

Select this option to use the Microsoft™ NT LAN Manager (NTLM) authentication protocol. NTLM uses challenge-response authentication. This view lists what is negotiated (supported by the client and requested of the server) and what is authenticated (the client reply to the challenge from the server).

Custom class

 **Note:** Not applicable for HTTP/2.

Select this option if the communication protocol requires complex, low-level processing with a custom Java™ code to transform incoming or outgoing messages. Click **Browse** to select a Java™ class that uses the corresponding API. This option is not available in IBM® Security AppScan®.

See [Creating SSL configurations on page 506](#) for more information about SSL authentication.

5. Click **OK** to create the new configuration.

What to do next

Once created, you can use your new configuration with any service request that uses the HTTP transport protocol. You can use the **Configurations** list in the generic service client to edit existing configurations or to create duplicate configurations.

Configuring the workbench for NTLMv2 authentication

NTLMv2 authentication requires access to a third-party library. To record and execute a test that contains NTLMv2 authentication, you must download the library and place it at the right location.

Before you begin

Before you can test SOAP-based services that use security algorithms, you must obtain and install a third-party library file.

About this task

By default, the HTTP test generation does not enable NTLMv2 authentication, even if it was part of the recording. To automatically enable the correct NTLM version from the recording, set the **Generated NTLM Version** setting to **Guess from recorded data** in the HTTP Test Generation preferences.

To configure the workbench to enable NTLMv2 authentication

1. Download the `jcifs-1.3.19.zip` file from <https://www.jcifs.org/src/>.
2. Unarchive the zip file and copy the JAR file to the installation directory: `InstallationDirectory\plugins\com.ibm.rational.test.lt.provider_<version>`
3. To automatically enable the correct NTLM version from the recording, in the workbench, click **Window > Preferences > Test > HTTP Test Generation** and set the **Generated NTLM Version** setting to **Guess from recorded data**.

Results


When a test was recorded with NTLMv2, the **Generated NTLM Version** setting is selected in the test editor, under **NTLM Authentication**.

Creating a JMS transport configuration

You can create an JMS transport configuration that describes the transport settings for a service request that uses the Java™ Message Service (JMS) protocol, including JBoss and IBM® WebSphere® JMS. Transport and security settings can be associated with any service request.

Before you begin

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

1. Click the **Generic service client**  toolbar button to open the generic service client and click the **Transport** tab.

Result

This opens the **Transport Configurations** page.

2. On the **Transport Configurations** page, click one of the following buttons:

Choose from:

- **Create a basic JMS configuration** (JMS) to create a new generic JMS transport configuration.
 - **Create a JBoss JMS configuration** (JMS) to create a JMS configuration preconfigured for JBoss.
 - **Create a WebSphere JMS configuration** (JMS) to create a JMS configuration preconfigured for WebSphere® JMS.
3. Type a **Name** for the new transport configuration and select whether the service is a **queue** or a **topic** destination.
 4. Type the address of the JMS end point.
 5. Select **Use temporary object** to provide the address of the reception point to the service as a temporary object. If you disable this setting, you must manually specify the reception point address.
 6. If the service requires authentication, select **Basic Authentication** and type the user name and password to access the service.
 7. If the service requires a custom Java™ Naming and Directory Interface (JNDI) adapter, you can provide your own Java™ class that extends the Apache Axis class. In this case, select Custom Adapter and specify the name of the custom Java™ class. See [Extending test execution with custom code on page](#) for more information about custom code.
 8. Specify whether the message type is **Text** or **Binary**.
 9. If necessary, click **Add** or **Edit** to specify the **Context factory properties** or **Connector properties** required to access the service.
 10. Click **OK** to create the new configuration.

What to do next

Once created, you can use your new configuration with any service request that uses the JMS transport protocol.

You can use the **Configurations** list in the generic service client to edit existing configurations or to create duplicate configurations.

Creating a WebSphere® MQ protocol configuration

When you want to send requests to a service that uses WebSphere MQ transport protocol, you can create a protocol configuration to describe the transport settings for a service request.


Before you begin

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

About this task

By default, messages are sent in bytes. Starting from V10.1.0, you can select message type as Text Message. After you create the protocol configuration, you can change the message format by selecting the **Text Message** check box in the **Message Structure**.

Transport and security settings can be associated with any service request. You can edit the existing configuration or duplicate the default configuration. You must have configured the environment with the correct libraries and configuration files when you use SOAP security.

1. Click the **Generic service client**  toolbar button, and then click the **Transport** tab.
2. From the **Protocol** list, right-click **MQ**, and then click **New MQ protocol configuration**.
3. Enter a name for the new transport configuration in the **Name** field.
4. Enter a name for the queue manager that receives the call in the **Queue Manager Name** field.
5. Enter a name for the queue managed by the queue manager in the **Send Queue Name** field.
6. Select the **Authentication** check box and specify the user name and password to authenticate with the MQ server.

Alternatively, add or update the login credentials in the **Protocol Configuration** tab of a service test.

7. Select the **Use Local Queue Manager** check box when the WebSphere MQ server is running on the local computer.
8. Perform the following steps if the MQ server is installed on a remote computer:
 - a. Clear the **Use Local Queue Manager** check box.
 - b. Enter the remote WebSphere MQ server details in the following fields:
 - The IP address or host name in the **Address** field.
 - Listener port number in the **Port** field.
 - Server connection mode channel name in the **Client Channel** field.
9. Select the **Use Temporary Queue for Response** check box if you want the server to create a temporary queue for receiving messages.
10. Perform the following steps to specify the queue that receives the response messages from the queue manager:
 - a. Clear the **Use Temporary Queue for Response** check box.
 - b. Enter a name for a queue in the **Receive Queue Name** field.
11. **Optional:** Specify the name of the target service in the **Target service** field when you are using the Microsoft .NET framework with SOAP over MQ.
12. **Optional:** Select **Use RFH2 header** when you are using SOAP over MQ. Otherwise, specify the **Message Descriptor** and **Encoding** options for the message header.
13. **Optional:** Click **Configure SSL** to select an existing SSL configuration or to create a new one when the service requires SSL authentication.
14. Click **OK** to create the protocol configuration.

Results

You have created a configuration for the WebSphere MQ transport protocol.

What to do next

- You can use the protocol configuration for the WebSphere MQ with any service request.
- You can change the message format by selecting the **Text Message** check box in the Message Structure.

Related information


[Creating SSL configurations on page 506](#)

Creating a WebSphere® Java MQ transport configuration

You can create a transport configuration that describes the transport settings for a service request that uses the IBM® WebSphere® Java MQ protocol. Transport and security settings can be associated with any service request.

About this task

This topic has instructions to specify the MQ server settings. If you have a single MQ server, you can choose to use the **Default Java MQ protocol configuration** option. If, for a new request, you must point to another MQ server, you can use the instructions in this topic to create a new transport configuration.

1. Click **Generic service client**  and click the **Transport** tab.
2. To create a new Java MQ transport configuration, in **Configurations**, select **Java MQ**.
3. In **Create Java MQ protocol configuration**, specify a name for the transport configuration.
4. Complete the following steps in the **Settings** tab:
 - a. **Host:** Specify the host name or IP address of the MQ server.
 - b. **Port:** Specify the port number that is used on the MQ server.
 - c. **Channel:** Name of the MQ communication channel that is used for sending and receiving messages and specified on the server. This field is case-sensitive.
 - d. **Queue Manager:** Name of the MQ queue manager as specified on the server.
 - e. **Optional: Use credential:** To access the secure server, specify the login credentials that is needed by the connection.
5. **Optional:** If necessary, complete the following steps in the **SSL** tab:



Learn more about the UI elements in the SSL tab:


- Select the **Use MQ SSL** check box when the connection to the Queue manager uses SSL.
- **Peer Name:** Distinguished Name (DN) of the queue manager to be used by SSL. The Distinguished Name is available in the SSL certificate. In MQ, a DN pattern is specified by using the **sslPeerName** variable of **MQEnvironment**. Connections succeed only if Peer Name matches the pattern that is specified.
- **Cipher Suites:** Select one of the available cipher suites to use for encrypting the transport communications.
- **Fips Required:** This option specifies whether the requested cipher suite must use FIPS-certified cryptography in WebSphere MQ.
- **KeyResetCount:** The total number of non-encrypted bytes that can be sent and received within an SSL conversation before the secret key is renegotiated. If left blank or set to zero (default),



the secret key is never renegotiated. This value is ignored if no cipher suite is specified. Valid values are integers 0 - 999,999,999.

- **SSL Configuration:** Select a SSL setting for the connection or click **Configure SSL** to create a new SSL configuration. See [Creating SSL configurations on page 506](#).

6. **Optional:** Use the **Options** tab to configure actions such as read, write, and browse on the selected MQ

Queues. Click  to select the configuration options.

7. **Optional:** Use the **Advanced** tab to specify the number of queue manager connections for reading messages, temporary destination settings, and to associate a reply with a request.

8. To test the connection, click **Test Transport** and then click **OK**.

Result

You have created a new transport configuration to point to a MQ server.

What to do next

You can now send the Java MQ requests to the configured server. See [Sending WebSphere Java MQ endpoint requests on page 515](#).

Creating Microsoft™ .NET transport configurations

You can manually create a Microsoft™ .NET transport configuration to describe the transport settings for service requests that use the Windows™ Communication Foundation (WCF) protocol.

Before you begin

If you are using SOAP security, ensure that the environment is configured with the correct libraries and configuration files.

Certificates and libraries required by the Microsoft™ client proxy must be installed on the computer, including Microsoft™ .NET libraries.

You must link a modified version of the Microsoft™ client proxy configuration file of the WCF service (by default `client.exe.config`) to the Microsoft™ .NET transport configuration. You must rename the file to `soaclient.exe.config` and edit it as described in the following procedure.



Tip: You can create a Microsoft™ .NET transport configuration automatically by importing the Microsoft™ .NET WSDL file. In this case, you must still manually edit the Microsoft™ .NET transport configuration to point to the modified `soaclient.exe.config` file as described in the following procedure. For more information, see [Sending service requests with WSDL files on page 508](#)

About this task

The product supports testing WCF services that use the following bindings:

- BasicHttpBinding
- WsHttpBinding

- NetMsMqBinding for 1-way calls only
- WSFederationHttpBinding
- WS2007FederationHttpBinding
- NetTcpBinding
- Custom bindings that do not integrate custom extensions in the channel, serialization of the message, transport, and security



Note: The following WCF services are not supported:

- Transaction and scopes
- Duplex mode requests, such as callbacks or 2-way services based on the Microsoft™ Message Queuing (MS-MQ) transport



Only for IBM AppScan users: To use Generic Service Client with IBM Appscan to test a WCF application, add the following code to the WCF configuration file:

```
<system.diagnostics> <trace autoflush="true" />
  <sources> <source name="System.Net"
maxdatasize="1048576"><listeners><add
name="System.Net"/></listeners></source> <source
name="System.Net.Cache"><listeners><add
name="System.Net"/></listeners></source> <source
name="System.Net.Http"><listeners><add name="System.Net
"/></listeners></source> <source
name="System.Net.Sockets"><listeners><add
name="System.Net"/></listeners></source> <source
name="System.Net.WebSockets"><listeners><add
name="System.Net"/></listeners></source> </sources>
<sharedListeners> <add
name="System.Net"
type="IBM.ServiceModel.Soa.Extension.tools.TrafficTraceListener,
Soa-Behavior-Library"
initializeData="" />
</sharedListeners> <switches> <add name="System.Net"
value="All"/> <add name="System.Net.Cache"
value="All"/> <add name="System.Net.Http"
value="All"/> <add name="System.Net.Sockets"
value="All"/> <add name="System.Net.WebSockets"
value="All"/> </switches></system.diagnostics>
```

IBM Appscan expects only HTTP requests in WCF. The following HTTP bindings are supported:

- BasicHttpBinding
- Custombinding above standard httpTransport
- WsHttpBinding
- WsFederationHttpBinding
- WS2007FederationHttpBinding



Also, the following patterns are supported:

- Action value (mandatory)
- Reply Action value (mandatory)
- Protection level

1. Create a modified `soaclient.exe.config` file by completing the following steps:

- Create a copy of `client.exe.config` (or `proxy_client_name.config`) file from the Microsoft™ .NET project and rename the copy to `soaclient.exe.config`.
- Edit the `soaclient.exe.config` file to use the version of Microsoft™ .NET that the product supports, as specified on the following line:

```
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
```


- Edit the `soaclient.exe.config` file so that the endpoints in the configuration file point to the client contract of the product, as specified on the following line:

```
contract="IBM.ServiceModel.Soa.Extension.Stub.IStubTest"
```

- Import the modified `soaclient.exe.config` file into the workspace.

Result

After you create the `soaclient.exe.config` file, you can skip the following steps and import the WSDL file to automatically create a Microsoft™ .NET transport configuration based on the information provided by the WSDL. For more information, see [Sending service requests with WSDL files on page 508](#).

- Click the **Generic service client** toolbar button () to open the generic service client and click the **Transport** tab.
- On the **Transport Configurations** page, click **Create a Microsoft .NET configuration**.
- Type a name for the new transport configuration and specify the following options:

Location of soaclient.exe.config

Specify the location of the `soaclient.exe.config` file. You must create this file manually by copying and editing the `client.exe.config` file from the Microsoft™ .NET service.

User authentication

If the service requires authentication, select **User Authentication** and type the user name and password to access the service.

Endpoint protection

By default, the transport configuration uses the endpoint protection level that is described in the `soaclient.exe.config` file. Use this setting to specify a different **Protection level**:

- **Signature**: Select this option to digitally sign requests.
- **Encryption and Signature**: Select this option to digitally sign and encrypt requests.

Advanced properties

Use this table to list the request and response actions by order of the methods in the WSDL file. Click **Add** to specify the name and value of request and response actions that are required by the service. This table is generated automatically when you import a Microsoft™ .NET WSDL file.

5. Click **OK** to create the transport configuration.

What to do next

After you create the configuration, you can use it with any service call that uses the Microsoft™ .NET transport protocol. You can use the **Configurations** list in the generic service client to edit existing configurations or to create duplicate configurations.

Creating SSL configurations

You can create a Secure Sockets Layer (SSL) configuration that describes the settings for a service request that uses SSL certification mechanisms. SSL configurations can be associated with any service request that uses the HTTP or IBM® WebSphere® MQ transport protocols.


Before you begin

If you are using SSL, ensure that you have valid certificate keystore files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files. See [Configuring the environment for SOAP security on page 228](#) for more information.


About this task

If you have to use different mutual SSL authentications for virtual testers in a test, you can create a dataset that stores all of the trust aliases names. In the test editor, in the **SSL Configuration** tab, you add a SSL configuration and associate it with the dataset. When a schedule is run, the SSL configuration is applied to each virtual tester.

1. Click the **Generic service client**  toolbar push button to open the generic service client, and click the **Transport** tab.
2. Either open an existing HTTP or WebSphere® MQ transport configuration, or create a new one, and then click **Configure SSL**.
3. Click

Rename



to rename the default SSL configuration or **New**  to create one.

4. Specify the following settings for the SSL configuration.

Server Authentication

This section describes how the client trusts the server.

Always trust server

Select this option if no authentication is required or to ignore server certificates so that all servers are trusted. If you are using single authentication and you want to accept trusted servers only, then disable this option and specify a truststore that contains the trusted server certificates.

Client truststore

When you are using single authentication, the client truststore contains the certificates of all trusted servers. Click **Browse** to specify a KS, JKS, or JCEKS file containing valid certificates of the trusted servers.

Password

If the client truststore file is encrypted, type the password required to access the file.

Mutual Authentication

This section describes how the server trusts the client in addition to server authentication.

Use client-side certificate

If you are using double authentication, select this option to specify a keystore containing the client certificate. This certificate allows the server to authenticate the client.

Client certificate keystore

Click **Browse** to specify a KS, JKS, or JCEKS file containing a valid certificate that authenticates the client.

Password

If the client truststore file is encrypted, type the password required to access the file.

Select trust alias for Mutual Authentication

Select an alias to be used for the SSL configuration. There could be multiple aliases in a keystore for different security certificates. Choose an appropriate alias for a user. You can also use dataset to store aliases that you can apply to virtual users at run time.



Note: You can copy the contents from an SSL configuration into another SSL configuration by using

Copy  and **Paste**  in the SSL editor.

5. Click **OK** to create the configuration, and close the SSL editor.

What to do next

When the SSL configuration is created, you can use the SSL configuration with any service request that uses SSL certification. You can use the SSL editor to edit existing configurations.

Sending service requests with WSDL files

You can send requests to services based on SOAP, Java Messaging Service (JMS), WebSphere® MQ, and Microsoft™ .NET that use a Web Service Description Language (WSDL) file to specify the contents of the service request.

Before you begin

Ensure that you have a valid WSDL file, which is accessible either on the file system, in the workspace, at a specific URL, or in an IBM® WebSphere® Service Registry and Repository or a Universal Description Discovery and Integration (UDDI) repository.

Ensure that the WSDL files use the correct syntax for the test environment. The generic service client might not work with some WSDL files.

If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before sending the request. For more information, see [Creating SSL configurations on page 506](#).

If the service uses SOAP security for encryption, signature, or other security algorithms, you must first configure the environment with the correct libraries and configuration files, and then create a WSDL security profile. For more information, see [Configuring the environment for SOAP security on page 228](#) and [Creating security profiles for WSDL files on page 353](#).

To import a WSDL file from a secured site that requires mutual authentication, you must have the Keystore file in the workspace.



About this task

When you create a call from a WSDL file, the call is configured automatically with any SOAP, JMS, WebSphere® MQ, or Microsoft™ .NET endpoints that are available in the WSDL file. Select the corresponding transport configuration on the **Transport** page of the request.






Note: For the specific requirements related to Microsoft™ .NET support, see [Creating Microsoft .NET transport configurations on page 503](#).

To send a service request based on a WSDL file:

1. Click the **Open the Generic Service Client** toolbar button  and select the **Requests** page.
2. Click **Add**  and select the method to add a WSDL file or click the corresponding shortcut button on the main page.

Choose from:

- Click **Add WSDL from Workspace** to add a WSDL file from the local workspace.
- Click **Add WSDL from File System** to add a WSDL file from the file system.
- Click **Add WSDL from URL** to download and import an online WSDL from the web.

- Click **Add WSDL from WSRR** to add a WSDL from WebSphere® Service Registry and Repository. Enter the URL of the WebSphere® Service Registry and Repository and click **Connect**. You can click **Search**  to browse the contents of the repository.
- Click **Add WSDL from UDDI** to add a WSDL from a Universal Description Discovery and Integration (UDDI) repository. Enter the URL of the UDDI and click **Connect**. You can click **Filter**  and **Search**  to browse the contents of the repository.



Note: If you are importing the WSDL file from a secured site that requires certificate authentication, click **Import Properties** and, for **Keystore**, select the keystore file that contains the certificate to be provided to the server, and for the **Keystore password**, type the password.

3. Click **OK**.

Result

The WSDL file is added to the **Request Library**.

4. In the **Request Library**, expand the WSDL file, binding, and operation, and then select the call element.

Result

The generic service client shows three steps: **Edit Data**, **Invoke** and **View Response**. The details for the call are displayed under the **Edit Data** step.

5. On the **Message** page, use the Form, Tree, or Source views to edit the contents of the request. Each view shows a different format of the same data. To add or remove XML elements in the Form or Tree view, click **Schema > Validate and Assist** to comply with an XML Schema Definition (XSD) specified in the schema catalog.
6. On the **Transport** page, specify the transport configuration for the request. The transport information from the WSDL file is imported automatically into the transport configuration.

For Microsoft™ .NET, select the corresponding transport configuration and specify the location of the `soaclient.exe.config` file. You must create this file manually. For details, see [Creating Microsoft .NET transport configurations on page 503](#).



Note: If you are using IBM® Security AppScan®, only the HTTP and .Net transport protocols are available.

7. On the **Request Stack** page, specify whether to override the security or processing algorithms that are applied to the outgoing request for the WSDL file.

Click **Show Response Stack** to add a **Response Stack** page to edit the security or processing algorithms for incoming responses.




Note: These settings apply only to the current request. If you want to edit the request or response stack for all requests that use the current WSDL file, click **Edit WSDL Security** to open the **WSDL Security Editor**.

8. When you are ready to send the service request, click **Invoke**.

Result

The generic service client sends the request and displays the message return under the **View Response** step.

What to do next

Successful requests are recorded and added to the **Request History** list. If you are using HCL OneTest™ Performance, you can create a service test by clicking the **Generate Test Suite** button ().

Sending HTTP endpoint requests



You can send requests to services that use an HTTP endpoint.

Before you begin

If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before sending the request. For more information, see [Creating SSL configurations on page 506](#).

If the service uses SOAP security for encryption, signature, or other security algorithms, you must first configure the environment with the correct libraries and configuration files, and then create a security profile for the WSDL file. For more information, see [Configuring the environment for SOAP security on page 228](#) and [Creating security profiles for WSDL files on page 353](#)

To send a request to an HTTP service:

1. Click the **Open the Generic Service Client** toolbar button  and select the **Requests** page.
2. Click the **Add** icon  and click a type of request that you want to send or in Request Library, right-click **EndPoints** and select a type of request that you want to send.
3. In the **Configure Protocol** window, select **HTTP** and specify the HTTP transport configuration. If necessary, click **New** to create an HTTP transport configuration for the call.

To send the HTTP/2 requests, in the **Create HTTP Protocol configuration** window, click the **Activate** check box. Before capturing the HTTP/2 traffic, configure the computer. See [Preparing to record a test for the HTTP/2 service on page 235](#) for instructions.

4. Type the URL of the call, the HTTP method and version, and specify any header or cookie properties. Click the **Rest mode** check box to split the URL into resource and parameters.
5. Click **Next**.
6. On the **Select Root Element** page, if the service uses a specific XML Schema Definition (XSD), select one from the list or click **Browse** to import the XSD file, and then, select the root element for the request. If no XSD is available for the service, select **No Schema**.
7. Click **Finish**.

Result

The request is added to the **Endpoints** section of the **Request Library**.

8. In the **Request Library**, select the request element.

Result

The generic service client shows three steps: **Edit Request**, **Invoke**, and **View Response**. The details for the request are displayed under the **Edit Request** step.

9. Based on the request selected in Step 2, on the **Message** page, use the **Form**, **Tree**, or **Source** views to edit the contents of the request.

Each view shows a different format of the same data. To add or remove XML elements in the **Form** or **Tree** view, click **Schema > Validate and Assist** to comply with an XSD specified in the schema catalog.

10. On the **Attachments** page, specify any file attachments to send with the request.
To add an attachment, click **Add** and follow the wizard to attach a file with the request.
11. On the **Transport** page, if necessary, change the transport configuration to be used by the request.
To create and edit transport and security configurations, use the **Transport** tab.
12. If you selected SOAP XML request in step 2, on the **Request Stack** page, specify whether you want to override the security or processing algorithms that are applied to the outgoing request for the WSDL file.
To add a **Response Stack** page to edit the security or processing algorithms for incoming responses, click **Show Response Stack**.



Note: These settings apply only to the current request. To edit the request or response stack for all requests that use the current WSDL file, click **Edit WSDL Security** to open the **WSDL Security Editor**.

13. When you are ready, click **Invoke** to send the service request.

Result

The generic service client sends the request and displays the message return under the **View Response** step.

What to do next

Successful requests are recorded and added to the **Request History** list. If you are using HCL OneTest™ Performance, you can create a service test by clicking the **Generate Test Suite** button ().

Sending a JMS endpoint request

You can send requests to services that use a Java™ Messaging Service (JMS) endpoint.

Before you begin

If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before sending the request. For more information, see [Creating SSL configurations on page 506](#).

To send a request to a JMS service:

1. Click the **Open the Generic Service Client** toolbar button () and select the **Requests** page.
2. Click **Add** () and click a type of request that you want to send or in Request Library, right-click **EndPoints** and select a type of request that you want to send.
3. In the **Configure Protocol** window, select **JMS** and specify the JMS transport configuration.
If necessary, click **New** to create an JMS transport configuration for the call.
4. Click **Add** to specify any properties that are to be sent with the call.
5. Click **Next**.

- On the **Select Root Element** page, if the service uses a specific XML Schema Definition (XSD), select one from the list or click **Browse** to import the XSD file, and then, select the root element for the call.
If no XSD is available for the service, select **No Schema**.

- Click **Finish**.

Result

The request is added to the **Endpoints** section of the **Request Library**.

- In the **Request Library**, select the request element.

Result

The generic service client shows three steps: **Edit Request**, **Invoke**, and **View Response**. The details for the request are displayed under the **Edit Request** step.

- Based on the request selected in Step 2, on the **Message** page, use the **Form**, **Tree**, or **Source** views to edit the contents of the request.

Each view shows a different format of the same data. To add or remove XML elements in the **Form** or **Tree** view, click **Schema > Validate and Assist** to comply with an XSD specified in the schema catalog.

- On the **Transport** page, if necessary, change the transport configuration to be used by the request.

To create and edit transport and security configurations, use the **Transport** tab.

- If you selected SOAP XML request in step 2, on the **Request Stack** page, specify whether you want to override the security or processing algorithms that are applied to the outgoing request for the WSDL file.

To add a **Response Stack** page to edit the security or processing algorithms for incoming responses, click **Show Response Stack**.



Note: These settings apply only to the current request. To edit the request or response stack for all requests that use the current WSDL file, click **Edit WSDL Security** to open the **WSDL Security Editor**.

- When you are ready, click **Invoke** to send the service request.

Result

The generic service client sends the request and displays the message return under the **View Response** step.

What to do next

Successful requests are recorded and added to the **Request History** list. If you are using HCL OneTest™ Performance , you can create a service test by clicking the **Generate Test Suite** button (.

Sending a WebSphere® MQ endpoint request

You can invoke calls to services that use a WebSphere® MQ endpoint.


Before you begin

If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before sending the request. For more information, see [Creating SSL configurations on page 506](#).

If the service uses SOAP security for encryption, signature, or other security algorithms, you must first configure the environment with the correct libraries and configuration files, and then create a security profile for the WSDL file. For

more information, see [Configuring the environment for SOAP security on page 228](#) and [Creating security profiles for WSDL files on page 353](#).

To send a request to an WebSphere® MQ service:

1. Click the **Open the Generic Service Client** toolbar button () and select the **Requests** page.
2. Click **Add** (+) and click a type of request that you want to send or in Request Library, right-click **EndPoints** and select a type of request that you want to send
3. In the **Configure Protocol** window, select **WebSphere MQ** and specify the WebSphere® MQ transport configuration.
If necessary, click **New** to create an WebSphere® MQ transport configuration for the call. For more information about creating a new WebSphere MQ transport configuration, see [Creating a WebSphere MQ transport configuration on page 500](#).
4. Specify the SOAP action.
If the service requires that you override the header specified in the WebSphere® MQ transport configuration, select **Override MQ protocol configuration values** and specify the correct details.
5. Click **Next**.
6. On the **Select Root Element** page, if the service uses a specific XML Schema Definition (XSD), select one from the list or click **Browse** to import the XSD file, and then, select the root element for the request.
If no XSD is available for the service, select **No Schema**.
7. Click **Finish**.

Result

The request is added to the **Endpoints** section of the **Request Library**.

8. In the **Request Library**, select the request element.

Result

The generic service client shows three steps: **Edit Request**, **Invoke**, and **View Response**. The details for the request are displayed under the **Edit Request** step.

9. Based on the request selected in Step 2, on the **Message** page, use the **Form**, **Tree**, or **Source** views to edit the contents of the request.
Each view shows a different format of the same data. To add or remove XML elements in the **Form** or **Tree** view, click **Schema > Validate and Assist** to comply with an XSD specified in the schema catalog.
10. On the **Transport** page, if necessary, change the transport configuration to be used by the request.
To create and edit transport and security configurations, use the **Transport** tab.
11. If you selected SOAP XML request in step 2, on the **Request Stack** page, specify whether you want to override the security or processing algorithms that are applied to the outgoing request for the WSDL file.
To add a **Response Stack** page to edit the security or processing algorithms for incoming responses, click **Show Response Stack**.




Note: These settings apply only to the current request. To edit the request or response stack for all requests that use the current WSDL file, click **Edit WSDL Security** to open the **WSDL Security Editor**.

12. When you are ready, click **Invoke** to send the service request.

Result

The generic service client sends the request and displays the message return under the **View Response** step.

What to do next

Successful requests are recorded and added to the **Request History** list. If you are using HCL OneTest™ Performance , you can create a service test by clicking the **Generate Test Suite** button ().

Sending OData endpoint batch requests

To test services that use OData protocol, you can send requests in a batch. The request contains HTTP operations such as GET, POST, and PUT to manage data in the service.



Before you begin

You must have sent individual requests through Generic Service Client (GSC).

About this task

When you send requests in a batch, you can group a set of operations into one HTTP request. You can start a batch request from GSC or from a service test. To initiate a batch request from a service test in the Test editor, select multiple requests to include in a batch, right-click and select **\$batch odata requests**.

To initiate a batch request from GSC, complete the following steps:

1. Click the **Open the Generic Service Client** toolbar button  and select the **Requests** page.
2. Click the **Add** icon  and click a type of request that you want to send or in Request Library, right-click **EndPoints** and select **Send a Batch Request**.
3. In the **ODATA batch information** page, select the OData version that your application supports.
4. To set HTTP headers, ensure that the **Set ODATA batch request http headers** radio button is selected.

If needed, you can change the headers on the next page of the wizard.

5. To group appropriate requests into change sets, select the **ODATA batch with changesets** radio button.
6. In **Selection of calls to batch**, select the requests to include in the batch.
If you initiated the batch request from the service test, the requests are already selected.
7. Click **Next**.
8. In the **Configure Protocol** window, select **HTTP** and specify the HTTP transport configuration.
If necessary, click **New** to create an HTTP transport configuration for the call.

To send the HTTP/2 requests, in the **Create HTTP Protocol configuration** window, click the **Activate** check box. Before capturing the HTTP/2 traffic, configure the computer. See [Preparing to record a test for the HTTP/2 service on page 235](#) for instructions.

9. Click **Finish**.

Result

The request is added to the **Endpoints** section of the **Request Library**.

10. In the **Request Library**, select the request element.

Result

The generic service client shows three steps: **Edit Request**, **Invoke**, and **View Response**. The details for the request are displayed under the **Edit Request** step.

11. On the **Transport** page, if necessary, change the transport configuration to be used by the request.
To create and edit transport and security configurations, use the **Transport** tab.
12. When you are ready, click **Invoke** to send the service request.

Result

The generic service client sends the request and displays the message return under the **View Response** step.



Sending WebSphere Java MQ endpoint requests

You can send requests to services that use a WebSphere Java MQ endpoint.

Before you begin

If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before sending the request. For more information, see [Creating SSL configurations on page 506](#).

To send a request to a Java MQ service:

1. Click **Open the Generic Service Client**  and select the **Requests** page.
2. Click **Add**  or in Request Library, right-click **EndPoints** and select a type of request to send.
3. In the Configure Protocol window, select **WebSphere Java MQ** and specify the transport configuration. If necessary, create the transport configuration for the call by clicking **New** (see [Creating a WebSphere Java MQ transport configuration on page 502](#)).
4. Complete the following information in the **General** tab:



Learn more about the UI elements in the General tab:

Queue

Name of the queue as defined on the WebSphere MQ server.

Message type

The types of messages are these:

- *Datagram* means that the message does not require a reply.
- *Request* means that the message requires a reply.
- *Reply* means that the message is a reply to an earlier request message.
- *Report* means that the message is reporting on some expected or unexpected occurrence, usually related to some other message. An example is a request message that contained data that was not valid.

Message Persistence

This value indicates whether the message is persistent or not. If the message is persistent, it survives the system failures and restarts of the queue manager. If the



message is not persistent, it survives a restart if it is present on a queue having the NPMCLASS(HIGH) attribute. However, even with the NPMCLASS(HIGH) attribute a message does not survive a QMGR class. Nonpersistent messages on queues having the NPMCLASS(NORMAL) attribute are discarded at queue manager restart, even if the message is found on the auxiliary storage during the restart procedure.

Dynamic Reply

Select this check box for the WebSphere MQ server to dynamically create a temporary queue as a reply. If this check box is not selected, the message in Reply Queue is used.

Reply Queue

This is the name of the message queue to which the application that issued the get request for the message should send the reply and report messages.

Reply Manager

This is the name of the queue manager on which the reply-to queue is defined.

Additional properties

Specify the additional properties for the queues.

5. **Optional:** If necessary, complete the following information on the **Config** tab:



Learn more about the UI elements in the Config tab:

Message Priority

This is the priority of the message. The lowest priority is 0.

Encoding

This is the numeric encoding of numeric data in the message. This value does not apply to numeric data in the MQMD structure itself.

Expiry Interval

This is the period of time, in tenths of a second, after which the message becomes eligible to be discarded if it has not already been removed from the target queue. The expiry interval is set by the application that put the message.

Character set

This is the character set identifier of the character data in the application message data.

6. **Optional:** In the **Report** tab, select the report messages to receive.
7. **Optional:** If necessary, complete the following information in the **Context** tab:



Learn more about the UI elements in the Context tab:

Application Identity Data

This information is defined by the application suite. Use it to provide information about the message or its originator.

Application Origin Data

This information is defined by the application suite. Use it to provide additional information about the origin of the message.

Accounting Token

This information is needed by the application to appropriately charge for the work that is done as a result of the message.

User ID

This is the user identifier of the application that originated the message.

8. **Optional:** In the **Identifiers** tab, for the messages that require binary input, specify the ID in the string format in the second column. The first column is filled automatically in the hexadecimal format.
9. **Optional:** In the **Segmentation** tab, select the segment of the message and click **Next**.
10. This step is not applicable for a Text request. On the Select Root Element page, if the service uses a specific XML Schema Definition (XSD), select one from the list. If the XSD element is not listed, click **Browse** to import the XSD file, and select the root element for the request. If no XSD is available for the service, select **No Schema**.
11. Click **Finish**. The request is added to the **Endpoints** section of the Request Library.
12. In the **Request Library**, select the request element.

Result

The generic service client shows three steps: **Edit Request**, **Invoke**, and **View Response**. The details for the request are displayed under the **Edit Request** step.

13. Based on the request selected in Step 2, on the **Message** page, use the **Form**, **Tree**, or **Source** views to edit the contents of the request.
Each view shows a different format of the same data. To add or remove XML elements in the **Form** or **Tree** view, click **Schema > Validate and Assist** to comply with an XSD specified in the schema catalog.
14. On the **Transport** page, if necessary, change the transport configuration to be used by the request.
To create and edit transport and security configurations, use the **Transport** tab.
15. If you selected SOAP XML request in step 2, on the **Request Stack** page, specify whether you want to override the security or processing algorithms that are applied to the outgoing request for the WSDL file.
To add a **Response Stack** page to edit the security or processing algorithms for incoming responses, click **Show Response Stack**.




Note: These settings apply only to the current request. To edit the request or response stack for all requests that use the current WSDL file, click **Edit WSDL Security** to open the **WSDL Security Editor**.

16. When you are ready, click **Invoke** to send the service request.

Result

The generic service client sends the request and displays the message return under the **View Response** step.

What to do next

Successful requests are recorded and added to the **Request History** list. If you are using HCL OneTest™ Performance, you can create a service test by clicking the **Generate Test Suite** button (.

Testing all operations in a WSDL file

You can use the generic service client to rapidly send requests to a service using all the operations in a Web Services Description Language (WSDL) file. The calls are generated with default values based on the type of data.


Before you begin

Ensure that you have a valid WSDL file. Ensure that the WSDL files use the correct syntax for the test environment. The generic service client might not work with some Web Services Description Language (WSDL) files.




If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before invoking the call. See [Creating SSL configurations on page 506](#) for details.

If the service uses SOAP security for encryption, signature, or other security algorithms, you must first configure the environment with the correct libraries and configuration files, and then create a security profile for the WSDL. See [Configuring the environment for SOAP security on page 228](#) and [Creating security profiles for WSDL files on page 353](#) for details.

Calls will be generated for each operation in the WSDL file using the default values for each type. For example, strings will use the default value `str`. You can change the default values in the **XML Default Values** preferences.

1. Open the generic service client and click the **Requests** tab, and then, click  **Add a WSDL file**.
2. In the Add WSDL Files window, select an existing WSDL or import a WSDL with one of the following methods:

Choose from:

- Click **Import from File** to import a WSDL file from the file system.
- Click **Import from URL** to download and import an online WSDL from the web.
- Click **Import from WSRR** to import a WSDL from an IBM® WebSphere® Service Registry and Repository (WSRR). Enter the URL of the WSRR and click **Connect**. You can click  **Search** to browse the contents of the repository.
- Click **Import from UDDI** to import a WSDL from a Universal Description Discovery and Integration (UDDI) repository. Enter the URL of the UDDI and click **Connect**. You can click  **Filter** and  **Search** to browse the contents of the repository.

3. Click **OK**.


Result

The WSDL is added to the **Call Library**.

4. In the Call Library, right-click the WSDL and select **Test WSDL Methods**.

The call is automatically configured with any SOAP or JMS endpoints that are available in the WSDL.

What to do next

Successful calls are recorded and added to the **Request History** list. If you are using HCL OneTest™ Performance or IBM® Rational® Service Tester for SOA Quality, you can click the **Generate Test Suite**  button to create a service test.

Viewing message content

The **Raw Transaction Data** view displays the raw XML, text, or binary content of any service request or response that is selected in the generic service client.

About this task

The **Raw Transaction Data** view displays plain text, XML, or binary data, depending on the type of the message content.

To view text, XML, or binary message content:

1. In the generic service client, click the **View** menu, and select **Raw Transaction Data**.
If you are using HCL OneTest™ Performance, click **Window > Show View > Other > Generic Service Client > Raw Transaction Data**.
2. Select a service request or response.
If you are using HCL OneTest™ Performance or Rational® Service Tester for SOA Quality, this view is also linked to the selected request or response in service tests, service stubs, or in the test log.
3. Depending on the nature of the message content, the following actions are available:

Text mode

When a plain text element is displayed, you can select and copy text. Click **Colorize Text** to enable or disable text colorization for HTML.

XML mode

When an XML element is displayed, you can select and copy text. Click **Colorize Text** to enable or disable text colorization for XML. Click **Enable XML Pretty Serialization** to improve readability by adding line breaks and indentation to the XML content.

If the XML content is modified by a request or response stack or by the WSDL security editor, the **Stack Contents** pane displays the list of steps in the stack. You can select each step to view the changes to the XML content. You can also select one or two steps and click **Compare Steps** to open a comparison window.

Binary mode

When a binary element is displayed, you can switch between **Binary** and **Raw-ASCII** views. Right-click the binary view to perform the following actions:

- **Select**: Opens the **Select** window, where you can select binary data by string or by specifying the number of characters to select. When a portion of binary data is selected, you can copy it to the clipboard.
- **Go to Offset**: Opens the **Go to Offset** window, where you can move to bytes at a particular offset.
- **Find**: Opens the **Find** window, where you can search for and replace binary data in a number of formats.
- **Encodings**: Select the encoding to use for displaying binary data in the text column.

Synchronizing a remote WSDL file

For web services that make their Web Services Description Language (WSDL) file available from a URL, you might have to ensure that the WSDL that you work with is always up to date. By synchronizing the WSDL, you ensure that the local copy of the WSDL in your workspace is regularly synchronized with the remote WSDL.


Before you begin

Ensure that you have a valid WSDL file. Ensure that the WSDLs use the correct syntax for the test environment. The product might not work with some WSDL files.

WSDL synchronization only works with remote WSDLs that are imported from a URL.

The WSDL synchronization runs either when the workbench is started or after a specified period. If the remote WSDL changes, the local copy of the WSDL is updated. Depending on the changes, a merge is performed and any service requests that use the WSDL are updated. If the changes to the WSDL cannot be automatically applied to the service requests, for example if an operation is removed or renamed or if the XML structure of the service request is changed, the test is marked with an error.

To import a synchronized remote WSDL:

1. Open the generic service client, click the **Requests** tab, and then, click **Add a WSDL file** .
2. In the **Add WSDL Files** window, click **Import from URL** to download and import a remote WSDL from the web.
3. On the **Import WSDL from URL** page, type the URL of the remote WSDL.
If you are connecting through a proxy or a corporate firewall, click **Proxy properties** to specify your network settings.
4. In the **Synchronization policy** area, specify whether and when to synchronize WSDLs:
Choose from:
 - Select **Never** if you do not want the remote WSDL to be updated.
 - Select **On session launch** to synchronize the WSDL each time you start the workbench.
 - Select **Every** to specify a synchronization period in days.
5. Click **OK**.

Result

The WSDL is added to the **Call Library**.

What to do next

After the WSDL is imported, you can change the synchronization settings by right-clicking the WSDL in the generic service client **Call Library** or in the test navigator. Then select **WSDL Synchronization**. The **WSDL Synchronization** window also displays the date of the latest synchronization.

Related information

[Sending service requests with WSDL files on page 508](#)

[Testing all operations in a WSDL file on page 518](#)

Synchronizing a local WSDL file with GSC

If you edit a local WSDL source file, the Generic Service Client (GSC) should display the changes in the UI. You must keep the GSC up-to-date with the WSDL changes to ensure that you test the latest service request.

About this task

When you set GSC to automatically pick the WSDL changes, the GSC calls are fully re-created. This means that when you make some changes to WSDL, there might be some content that you did not change, however, was dependent on the changed content. Therefore, when you use this preference, the whole structure of the GSC calls is re-created.

The Request History view in GSC shows the changes occurred to the WSDL file.

To apply the local WSDL changes in GSC:

1. Click **Window > Preferences > Generic Service Client**.
2. Select the **Apply WSDL changes to GSC** check box.

Adding static XML headers to a service request

You can add static XML headers to service requests to ensure compliance with WS-Addressing, WS-ReliableMessaging, and WS-Coordination specifications as well as other predefined standards.

About this task

Static XML headers are compliant with the web service specifications for service-oriented architecture (SOA). Checks are performed to ensure that the XML headers are valid.

To add a static XML header to a request:

1. Open a service request in the generic service client. The location of the XML header depends on the product that you are using:

Choose from:

- For IBM® Security AppScan®, click the **Request Stack** tab and in the algorithm stack for the request, click **Add > Static XML Headers**.
 - For HCL OneTest™ Performance, IBM® Rational® Service Tester for SOA Quality, or other products, click the **Message** tab and click **Form**.
2. On the **Header** bar, click **Add (+)** to open the menu.
 3. Select the web service specification for the request to be comply with, or click **More** to open a detailed list of specifications.

Result

The XML structure of the header is created.

4. Edit the header as required.

Some elements require completion or content to be specified. XML elements that are invalid or require attention are marked with a warning or an error symbol.

Related information

[Editing WSDL security profiles on page 352](#)

[Adding WS-Addressing to a security configuration on page 366](#)

Opening file attachments

When a service sends a file attachment with the response, you must import it as a resource to open the attachment.

Before you begin

Ensure that you have specified an editor to view the attachment type in. Click **Window > Preferences > General > Editors > File associations**, and specify the editor.

1. Open the message return, and click the **Attachment** tab.

File attachments are listed with a default name, a MIME type, and a contents ID.
2. Select the line for the attachment that you want to open, and click **Open**.
3. In the **Create Resource** window, type a name for the resource, and select a location where it will be imported, and click **OK**.

Ensure that the name of the resource includes a file extension that is compatible with the MIME type of the attachment.

What to do next

After the attachment has been imported, you can click on **Open** again to open the file in the corresponding editor.

Emulating workloads

You emulate a workload by creating a schedule and adding user groups/rate runner groups, tests, and other elements to it.

Schedule overview

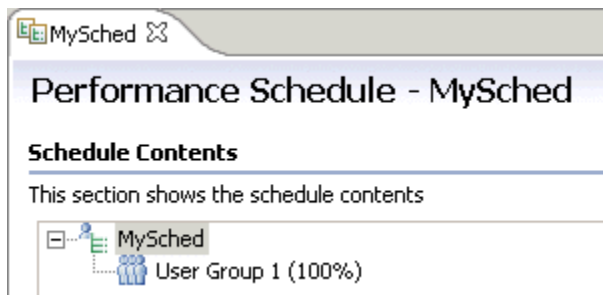
A schedule is the "engine" that runs a test. However, schedules are much more than simple vehicles for running tests. You can design or emulate the real-life workload by creating various groups and dividing the load across different remote agents that generate load on the application under test. A schedule can be as simple as one virtual user or one iteration running one test, or as complicated as hundreds of virtual users or iteration rates in different groups, each running different tests at different times.

You can create a VU Schedule or a Rate Schedule. The VU Schedule is used to add virtual users to generate the load on the application under test. The Rate Schedule is used to ascertain the rate at which a task can be achieved in a specific time frame. The Rate Schedule can be used on agents that were purchased only with PVU-based licenses.

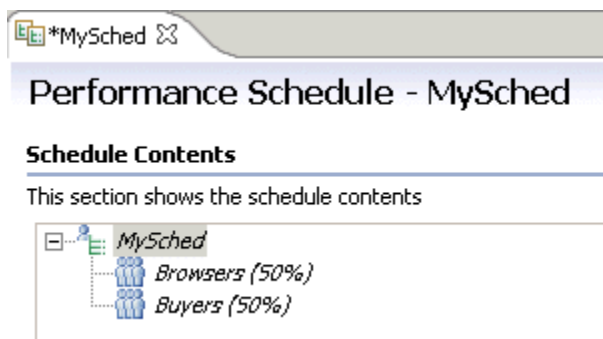
You can use a schedule to control tests in the following ways:

- Group tests under groups, to emulate the actions of different types of users or rates.
- Set the order in which tests run: sequentially, randomly, or in a weighted order.
- Set the number of times that each test runs.
- Run tests at a certain rate
- Run tests for a certain time, and increase or decrease virtual users or rate during the run

When you first create a schedule, it is displayed with one group, as shown below. You add more groups, tests, and other items to the schedule to emulate a workload.

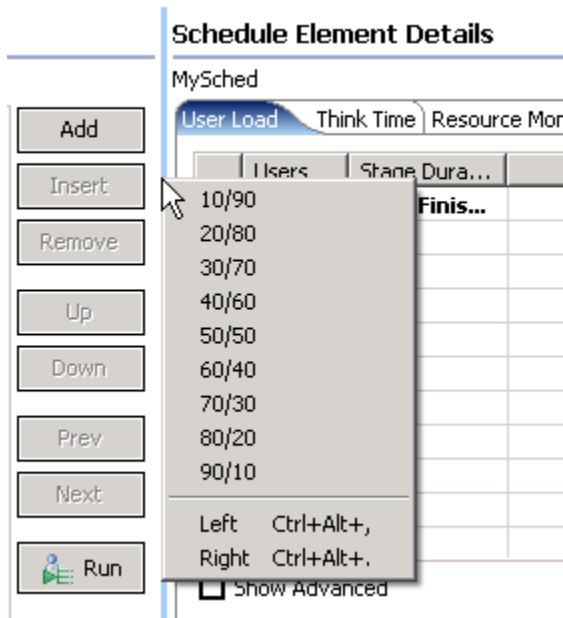


When you add items to a schedule, they appear in italic type, as shown below. The italic type changes to regular type after you save the schedule.



To resize the schedule window, do one of the following:

- Click Ctrl+Alt+> or Ctrl+Alt+< to enlarge or reduce the window.
- Hover at the left side of the Schedule Element Details area. When you see a vertical blue line, right click the line and select a size ratio from the menu.



The new size remains the next time you open the window.

Creating a VU Schedule

By creating a VU Schedule, you can accurately emulate the actions of individual users.

1. In the Test Navigator view, right-click the project, and then click **New > VU Schedule**.
2. Type the name of the VU Schedule, and then click **Finish**.

Result

A new VU Schedule that contains one user group is displayed.

3. Add user groups and set the locations (agent computers) on which each user group will run. Although you can run user groups from your workbench computer, doing so affects the accuracy of your tests.
 - a. To add user groups: Right-click the VU Schedule, and then click **Add > User Group**.
For more information, see [Adding a user group to VU Schedule on page 534](#).
 - b. To set the locations for the user groups: Click a user group, click the **Locations** tab, and then select **Run this group on the following locations**.
For more information, see [Running a user group at a remote location on page 538](#).
4. Set the loops for the tests (or other schedule elements) to use: Right-click the user group to contain the loop, and click **Add > Loop**.

Loops are used to run many iterations of a test, to run tests at a set rate, and to run tests in stages, which is discussed later. For more information, see [Repeating tests in a schedule on page 560](#) and [Running tests at a set rate on page 563](#).

5. Add selectors and their weights: Right-click the schedule element to contain the selector, and click **Add > Random Selector**.

Selectors are used to run a series of tests in random order, thus emulating the varied actions of real users, instead of running each test within a user group sequentially. The weight that you assign each selector determines the statistical probability that its child element is selected during an iteration. For more information, see [Running tests in random order on page 564](#).

6. Add tests to each user group: Right-click the schedule element to contain the test, and click **Add > Test**. For more information, see [Adding a test to a schedule on page 556](#).
7. Set the stages for the VU Schedule. Each stage lasts for a specific amount of time and contains a specific number of users. By setting stages, you can model workloads that reflect real-world usage over time. Putting the tests in a stage in an infinite loop prevents virtual users from finishing before the stage ends. To set a stage:
 - a. Open the VU Schedule, and click the **User Load** tab.

Result

- b. On the **User Load** page, click **Add**.
Enter the number of users in the stage and the duration of the stage.
- c. Click **Window > Preferences > Test > Test Reports**, and verify that **Launch Compare report when staged run completes** is selected.
This selection automatically generates a report that compares each stage.

For detailed information on stages, see [Setting user loads on page 527](#).

8. Add other schedule elements to refine the schedule structure: Right-click a schedule element, and click **Insert** (adds the new element before the selection) or **Add** (adds the new element after the selection).

Element	Purpose	For more information
Synchronization point	Used for coordinating the activities in a schedule, such as forcing virtual users to wait at a specific point	Synchronizing users on page 540
Delay	Used to emulate user actions accurately; for example, a user might delay before placing an order	Delaying virtual users or actions on page 540
Comment	Used for your notes and comments regarding the schedule element	

9. Set the VU Schedule options:

Tab name	Typical setting	For more information
Resource monitoring	<p>Select Enable resource monitoring to enable resource monitoring.</p> <p>You can capture resource monitoring data from these sources:</p> <ul style="list-style-type: none"> ◦ Apache HTTP Server Managed Beans ◦ Apache Tomcat Managed Beans ◦ IBM® DB2® monitoring ◦ IBM® Tivoli® monitoring ◦ IBM® WebSphere® Performance Monitoring Infrastructure ◦ JBoss Application Server Managed Beans ◦ Java™ Virtual Machine Managed Beans ◦ Oracle Database monitoring ◦ Oracle WebLogic Server Managed Beans ◦ SAP NetWeaver Managed Beans ◦ The rstatd daemon (UNIX™) ◦ Simple Network Management Protocol (SNMP) agents ◦ Windows™ Performance Monitor 	<p>Enabling Resource Monitoring from the workbench on page 576</p> <p>Enabling Resource Monitoring on Windows Vista, Windows 7, and Windows Server 2008</p>
Response time breakdown	<p>Select Enable collection of response time data to enable response time breakdown.</p> <p>You can collect response time breakdown data from HTTP or SOA tests.</p>	<p>Enabling response time breakdown collection on page 593</p> <p>Enabling response time breakdown collection on Windows Vista, Windows 7, and Windows Server 2008 on page 593</p>

Tab name	Typical setting	For more information
Think time	Use the options on this page to increase, decrease, or randomize the think time. The default setting is to use the recorded think time.	Think time overview on page 549
Statistics log level	Typically, keep the default settings. If you are running a long test, change the sampling rate from the default 5 seconds to a larger interval.	Setting the statistics displayed during a run on page 597
Test log level	Typically, keep the default setting of Primary test actions . You must have at least this level of logging to create a Page Percentile report and to see page title verification points that you have set.	Setting the data that the test log collects on page 599
Problem definition log level	Change the default settings only when requested to do so by Support.	Setting the problem determination level for schedules on page 604
Advanced (at the bottom of the Schedule Element Details area)	Click Edit Options to set protocol-specific options that apply to all tests in the schedule. Setting protocol-specific options for a schedule is similar to setting protocol-specific options for a user group.	Emulating slower network traffic on page 537 Running long duration Citrix tests on page 610

What to do next

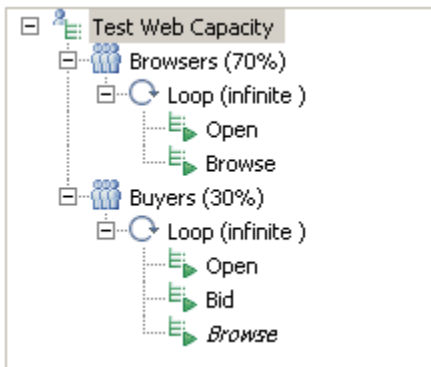
After you create a VU Schedule that describes the behavior for your software system, run it against successive builds of the application under test or with an increasing number of virtual users. Then analyze the results that are reported.

Setting user loads

By setting stages, you can model workloads over time and change the number of users that perform certain tasks to reflect real-world usage. You can vary the user load and collect performance metrics for each stage independently, which means that a single run can more efficiently accomplish the work of multiple runs that require shutting down and restarting users. Each stage, which lasts a specific amount of time and contains a specific number of users, defines a different load.

About this task

When a VU Schedule contains stages, you can place the tests in the schedule in an infinite loop, as shown in the following figure. This setting prevents virtual users from finishing the stage before the allotted time.



You can also use the **Percentage of users allowed to exit during execution** option to specify the number of users that can stop during a stage without stopping the stage or the entire test run.

To add stages to a VU Schedule:

1. In the Test Navigator, browse to the schedule and double-click it.

Result

By default, the User Load option contains one stage with five users that run until finished. The following figure shows the default User Load option.

Schedule Element Details

rec_schedule

User Load
Think Time
Resource Monitoring
Statistics
»»5

	Users	Stage Duration	Chan	
	5	Until Finished	0	<div style="background-color: #e0e0e0; padding: 2px; margin-bottom: 2px;">Add...</div> <div style="background-color: #e0e0e0; padding: 2px; margin-bottom: 2px;">Edit...</div> <div style="background-color: #e0e0e0; padding: 2px; margin-bottom: 2px;">Remove</div> <div style="background-color: #e0e0e0; padding: 2px; margin-bottom: 2px;">Up</div> <div style="background-color: #e0e0e0; padding: 2px;">Down</div>

Show Advanced

Time limit for a user to respond to a stop request:

Seconds

Percentage of users allowed to exit during execution:

User Load Preview

The graph shows a constant load of 5 users over a 10-minute period. The y-axis is labeled 'Users' with ticks at 0, 2, and 4. The x-axis is labeled 'Run duration [min]' with ticks at 0, 3, 6, and 9. A horizontal red line is drawn at the 5 user level from x=0 to x=10.

2. In the User Load section, click **Add**.
3. In the **Create User Stage** window, enter the information for a stage, and click **OK**.

Option	Description
Number of users	Enter the total number of users in the stage. This is not the number of users to add to or to remove from those currently running; it is the total number of active users at this stage.
Stage Duration	Enter the length of time (and the time units) for the stage to run. After the Number of users setting is achieved, the users will run for up to this amount of time. When the time expires, the users continue to run if they are needed for the next stage, or, if not, they are stopped.
Rate of Change	<p>Specify the amount of time to delay, when changing the number of users, between adding or removing each user.</p> <p>Adding or removing all users over a time period changes the users in a uniform random distribution over the time specified for changing users, which is the time before the settle and the stage begin. This slight variance closely emulates human behavior.</p> <p>Adding or removing one user every time unit adds the same delay for each user. Although this option does not emulate human behavior as closely as the first option, it is useful when you must adhere to a certain rate because of limitations of the system under test, such as the time it takes for a user to log on to the system.</p>
Settle Time	<p>After the desired user population has been reached, a system might still experience a period of flux in reaction to the change in user population. Setting a settle time allows the system to re-establish its steady-state equilibrium so that it can accurately reflect the user population.</p> <p>The Stage Duration starts after the settle time expires. The settle time is not part of the stage duration and the settle-time metrics are not included in the Compare report, which is generated at the end of the run. However, settle time does affect how long a</p>

Option	Description
	<p>VU Schedule runs, because it adds time to the beginning of each stage. And, although the Compare report does not include the settle-time metrics, these metrics are collected and you can include them by changing the time range of the report.</p> <p>If your system does not have significant flux or if the stage is long enough that the flux comprises only a minor part of it, you might not need a settle time.</p>

4. In the User Load section, modify the stages as necessary:
 - a. Click **Up** or **Down** to change the order of the rows.
 - b. Double-click a row to modify it.
5. Enter the **Time limit for a user to respond to a stop request** value.

If a stage contains fewer virtual users than its predecessor, the excess users are asked to stop. This value gives a stopped virtual user extra time to complete its current action (such as an HTTP request). If the virtual user cannot complete its action before the time limit expires, it is forced to stop. Note that a long time limit might delay the next stage.
6. Enter a value for **Percentage of users allowed to exit during execution** to specify the percentage of users that can stop during a stage of a test run. The default is 0%, which means if any users stop during a stage, the entire test ends after that stage completes. If you enter a value, the test run can continue to the next stage even if some users stop running. You can specify a value from 0 to 100 with fractions up to one decimal place. Examples of valid percentages include 0.5%, 3%, and 99.1%.
7. To stop the run after a specific number of successive failed stages, select the **Exit run for failing requirements** check box and specify a value in **Number of failing stages in a row**. If, at the end of a completed stage, that stage has failed, and if such stage failures happen successively for the specified number of times, the VU Schedule will stop.
8. Examine the **User Load Preview** section to verify that the stages are set correctly. The red line segments indicate that the total number of users has been achieved for the stage and the settle time, if one is specified, has ended.

Result

The following figure illustrates a VU Schedule with two 16-minute stages. The second stage has a 4-minute change rate and a 4-minute settle time:

Schedule Element Details

rec_schedule

User Load Think Time Resource Monitoring Statistics Variable Initializat »4

	Users	Stage Duration	Change Rate	
	50	16 Minutes	0	Add...
	200	16 Minutes	All/4 Minutes	Edit...
				Remove
				Up
				Down

Show Advanced

Time limit for a user to respond to a stop request: 30 Seconds

Percentage of users allowed to exit during execution: 0

User Load Preview

The graph shows the number of users over a 40-minute period. The y-axis is labeled 'Users' with a scale from 0 to 200. The x-axis is labeled 'Run duration [min]' with a scale from 0 to 40. The user count starts at 50 at time 0 and remains constant until 16 minutes. At 16 minutes, the user count increases linearly to 200 by 20 minutes. From 20 minutes to 40 minutes, the user count remains constant at 200.

What to do next

You can display a Compare report, which compares the time ranges of each stage, when the run is complete. This report provides a quick side-by-side analysis of how the system under test performs under various user loads. To display a Compare report, right-click the test results; then click **Compare All Time Ranges**.

To display a Compare report automatically at the end of each staged run, click **Window > Preferences > Test > Test Reports**, and select **Launch Compare report when staged run completes**.

User group overview

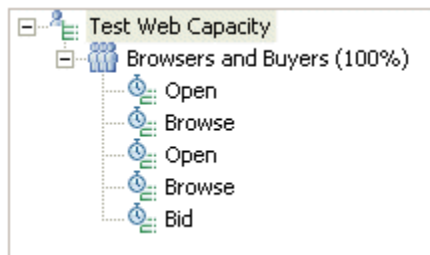
User groups enable you to group tests in a logical order.

With user groups, you can control test execution in several ways:

- **Group tests by characteristics.** For example, you could have two user groups—a Buyers group and a Browsers group—that represent the types of users on your system.
- **Influence the order in which tests are run.** When you run a VU Schedule, the first test in each user group runs—in parallel, not sequentially. After the first test in a user group is completed, the second test runs, then the third, and so on.

Example

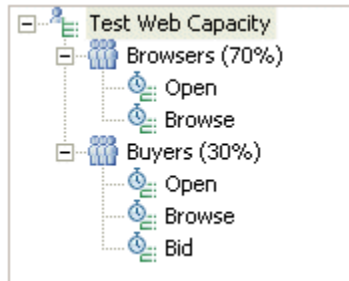
The following VU Schedule contains one user group.



If you run this VU Schedule with 10 users, they are assigned to the only user group—Browsers and Buyers. When the run starts, the 10 virtual users start running the first test in parallel. As soon as one test is finished, a virtual user moves to the second test. Thus, you have 10 virtual users, all starting at the same time and running each test sequentially. This does not give you much control over the run.

Example

The following VU Schedule contains the same tests in the same order, but they are divided between two user groups. Conceptually, this VU Schedule is easier to understand, because the user's tasks are grouped logically—the Browsers browse, and the Buyers browse and then bid on a product. But, even more important, this one gives a more accurate representation of the types of users on your system, because each user group contains tests that represent the actions that they do, and the proportions of the user groups (70% and 30%) represent the proportions of the users on your system.




If you run this VU Schedule with 10 users, seven are assigned to the Browsers group, and three are assigned to the Buyers group. When the run starts, the seven Browsers and the three Buyers start in parallel. Thus, you have seven Browsers, each running two tests sequentially, and three Buyers, each running three tests sequentially.

Adding a user group to VU Schedule

By defining user groups, you can group related tests and run the tests in parallel. Tests belonging to different user groups run in parallel.

1. In the Test Navigator, double-click the VU Schedule.
2. Right-click the name of VU Schedule, and then click **Add > User Group**.
3. In **Group name**, type a descriptive name for the user group.
4. Under **Group size**, select **Absolute** or **Percentage**, and type the number of users or a percentage of users in the group.

Option	Description
Absolute	Specifies a static number of virtual users. Type the maximum number of virtual users that you want to be able to run. For example, if you type 50, you can run up to 50 virtual users each time you run a schedule. Typically, you create an Absolute user group only if the group does not add a workload. For example, if one test prepared a website for use and another test restored the site to its initial state, each test would be in an Absolute user group that contains one user.
Percentage	Specifies a dynamic number of users. Type the percentage of the workload that the user group represents. Typically, you assign user groups a percentage, rather than an absolute number. For example, perhaps 70.0% of your users browse your website, and 30.0% order an item from your website. Set up two user groups in this proportion. Then, at the schedule level, type the initial number of users to run. You can also add users during the run. The schedule

Option	Description
	distributes the users among the dynamic user groups according to the percentages you specify.  Tip: You can specify fractional percentages.

5. Under **Locations**, select **Run this group on the local computer** or **Run this group on the following locations**.

Option	Description
Run this group on the local computer	The user group runs on your computer. Use this option if the workload is small or if you are testing the VU Schedule.
Run this group on the following locations	Typically, you run user groups on remote computers. When user groups run on remote computers, the workbench activity on the local computer does not affect the ability to apply load. Run user groups at remote locations in these cases: <ul style="list-style-type: none"> ◦ When a large number of virtual users are running and the local computer does not have enough processor or memory resources to support this load. You can conserve resources by running the users on different locations, so that fewer users run on each computer. ◦ When a test requires specific client libraries or software. The user group that contains this test must run on a computer that has the libraries or software installed.

6. To declare a remote location:

a. Click **Add > Add New**.

Result

The **Add New** wizard opens. On the first page of the wizard, you can specify general properties for the remote location.

b. In **Hostname**, type the IP address or the fully qualified host name of the remote computer.

c. In **Name**, type a descriptive name for the remote computer.

d. In **Deployment Directory**, enter a fully qualified pathname of the directory, which is in the remote computer to store the test assets.



Note: The environment variables such as %TEMP% are not supported in the **Deployment Directory** field.

- e. In **Operating System**, select the operating system of the remote computer, and then click **Next**.
- f. Specify the IP aliasing properties for this location. To make it appear as though each virtual user has its own IP address, click **Enable IP Aliasing**.
- g. To use IP addresses from all network interfaces at the remote location, click **Use IP addresses from all network interfaces**.
- h. To use addresses from a subset of network interfaces, click **Only use IP addresses from the following network interfaces**.



Tip: Click **Add** to add the name of an interface and **Edit** to change the interface name. Specify network interfaces separated by commas, for example, `eth0, eth1`. If you do not use this form, the connection attempt fails.

- i. Click **Next**. On the third page of this wizard, you can specify file locations.
- j. In **File name**, type the name of the file to contain information about this computer, and then click **Next**.



Note: The data stored in the file includes information such as the host name and deployment directory. You can change this information later by opening the Test Navigator and double-clicking the file.

7. To add an already declared location:
 - a. Click **Add > Add Existing**.
 - b. In the **Select Location** window, select the computer on which the user group will run, and then click **OK**.

Example

The following schedule shows two user groups. Browsers represent 70.0% of the users, and Buyers represent the remaining 30.0%:



What to do next

After you have added user groups to the VU Schedule, add the tests that each user group will run.

Adjusting user groups

You can adjust multiple user groups simultaneously to distribute a load across groups and to set the groups to run on the workbench computer. Typically, when you run VU Schedule, user groups run on remote agent computers, not on the local workbench computer.

1. In the Test Navigator, double-click the VU Schedule.
2. Under **Schedule Contents**, select the groups to adjust.
3. Under **User Group Details**, adjust the group sizes or set all groups to run on the workbench computer.

Option	Description
Percentage	Click to distribute the load for the user group based on percentage.
Absolute	Click to distribute the load for the selected user group, based on absolute numbers.

Emulating slower network traffic

You can emulate various WAN connection speeds used for HTTP traffic to determine its effect on response times and throughput.

About this task

You can set a user group to emulate the speed at which HTTP data is sent and received. By delaying the network uploads and downloads to emulate a slower network connection, the user group mirrors real-world interaction with production servers.

To emulate line speed:

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.

2. In the schedule, click the user group for which you want to define a line speed.
3. In the Schedule Element Details area, click the **Options** tab.
4. To declare the line speed value:
 - a. Click **Enable line speed control**.
 - b. Select the actual line speed, or select **Custom values** to enter a value.
 - c. Select the required line speed, or select **Custom values** to enter a value, which must be lower than the actual line speed.

Example

If a user group employs multiple agent computers (locations) that have different real line speed values, to obtain the highest possible accuracy, we recommend that you divide the user group into smaller user groups so that each user

group contains agents with the same actual line speed value. However, even if the actual line speeds are different, if they are both much larger than the required line speed, then the inaccuracy will be small, and may even be tolerable.

The following example illustrates this difference. The actual line speeds are different (100 Mbps and 10 Mbps) but are both much larger than the desired line speed of 56 Kbps:

The computed line speed delay for a 5000 byte response where the actual line speed is 100 Mbps and the desired line speed is 56 Kbps is 706 ms. The computed line speed delay for a 5000 byte response where the actual line speed is 10 Mbps and the desired line speed is 56 Kbps is 702 ms. In this example, the inaccuracy is only 4 ms.

Running a user group at a remote location

You can run a user group at a remote location (also called an agent computer), rather than on your local computer, to prevent your workbench activity from affecting the ability to apply load.

Before you begin

Before you run a user group at a remote location, verify that:

- HCL OneTest™ Performance Agent is installed on the remote computer. The agent is configured and connected to the HCL OneTest™ Performance workbench.
- Firewall is disabled on the workbench computer or configured to allow incoming connections on the port number 7080.
- A reasonable number of virtual users will run at the remote location. When you assign a user group to a remote location, do not overload the remote computer (agent). If you exceed the number of virtual users that the remote computer can run, the performance measurements of the server will be skewed because they will be affected by the performance of the computer. The test results will reflect the load of the computer more than the load of the server. For best results on a computer with a 1 GHz processor and 1 GB of RAM, do not exceed 1000 concurrent virtual users.

About this task

Generally, you should run user groups at a remote locations. You *must* run a user group at a remote location in these cases:

- When a large number of virtual users are running and the local computer does not have enough processor or memory resources to support this load. You can conserve resources by running the users on different locations, so that fewer users run on each computer.
- When a test requires specific client libraries or software. The user group that contains this test must run on a computer that has the libraries or software installed.

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.

2. In the schedule, click the user group that you want to run on a different computer.
3. In the Schedule Element Details area, click **Run this group on the following locations**.

4. To declare a remote location:

- a. Click **Add > Add New**.

Result

The **Add New** wizard opens. On the first page of the wizard, you can specify general properties for the remote location.

- b. In **Hostname**, type the IP address or the fully qualified host name of the remote computer.
- c. In **Name**, type a descriptive name for the remote computer.
- d. In **Deployment Directory**, enter a fully qualified pathname of the directory, which is in the remote computer to store the test assets.



Note: The environment variables such as `%TEMP%` are not supported in the **Deployment Directory** field.

- e. In **Operating System**, select the operating system of the remote computer, and then click **Next**.
- f. Specify the IP aliasing properties for this location. To make it appear as though each virtual user has its own IP address, click **Enable IP Aliasing**.
- g. To use IP addresses from all network interfaces at the remote location, click **Use IP addresses from all network interfaces**.
- h. To use addresses from a subset of network interfaces, click **Only use IP addresses from the following network interfaces**.



Tip: Click **Add** to add the name of an interface and **Edit** to change the interface name. Specify network interfaces separated by commas, for example, `eth0, eth1`. If you do not use this form, the connection attempt fails.

- i. Click **Next**. On the third page of this wizard, you can specify file locations.
- j. In **File name**, type the name of the file to contain information about this computer, and then click **Next**.



Note: The data stored in the file includes information such as the host name and deployment directory. You can change this information later by opening the Test Navigator and double-clicking the file.

5. To add an already declared location:

- a. Click **Add > Add Existing**.
- b. In the **Select Location** window, select the computer on which the user group will run, and then click **OK**.

Delaying virtual users or actions

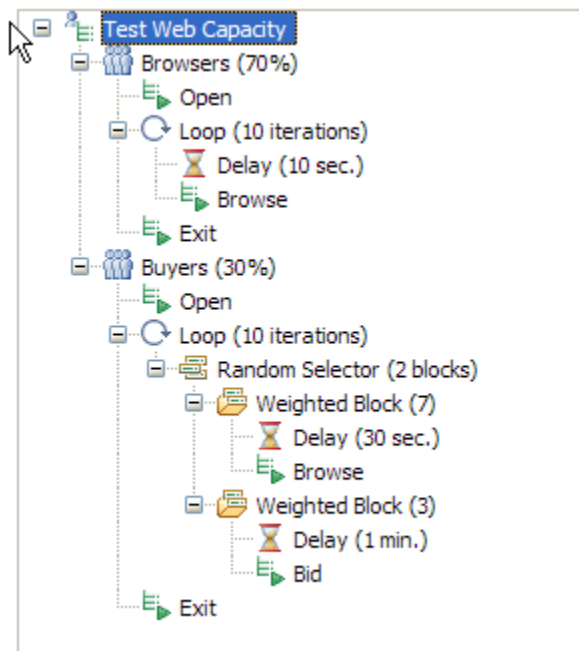
By adding a delay to a schedule or a compound test, you can emulate user actions more accurately.

1. In the Test Navigator, browse to the schedule or compound test and double-click it.
2. Right-click the schedule or test element to be delayed, and then click **Add > Delay**.
3. In the Element Details area, type the length and the time units of the delay.

Example

After you add a delay, you generally add the schedule or test elements that the delay controls. The elements are at the same level as the delay—they are not children of the delay.

The following schedule shows three delays. The Browsers delay 10 seconds between each browse action. The Buyers delay 30 seconds between each Browse action and one minute before they decide to bid on an item.



Synchronizing users

Inserting a synchronization point enables you to coordinate the activities of a number of virtual users by pausing and resuming activities. You can synchronize all virtual users at the beginning of a schedule and stagger the release times so that the users do not overload the system. Synchronization points are also useful in stress testing.

About this task

You can insert a synchronization point into a schedule or a test. The advantage of inserting a synchronization point into a schedule is that the synchronization point is more visible than in a test. Also, you can define release options and select a timeout for a synchronization point set in a schedule.

Synchronization points within loops are not reset. In other words, once a synchronization point has been released (in the first iteration of a loop) it stays released for all further iterations.

To insert a synchronization point into a schedule:

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.

2. Right-click the element just below the place that you want to add the synchronization point, and then click

Insert > Synchronization point.

3. Enter a name for the synchronization point, or select the name of an existing synchronization point to modify its release type.

Result

The synchronization point opens in the schedule for you to set its attributes.

Schedule Element Details

Synchronization Point: staggered with default timeout

Sync. Point name: [Change...](#)

Release Type

Together

Restart time:

Staggered

Minimum time:

Maximum time:

Timeout:

4. Set the release type; that is, whether you want the users to be released at the same time or at staggered times. Use the **Together** release type when you are performing a stress test on the system. Use the **Staggered** release type when you want the users released in such a way that they will not overwhelm the system.

Option	Description
Together	<p>Releases all users at once from a synchronization point. The default restart time is 0, which means that when the last user reaches the synchronization point, all users are released together immediately.</p> <p>To delay the users, enter a number in the Restart Time field. For example, if you set the restart time to 4 seconds, after all of the users reach the synchronization point (or the timeout occurs), they wait 4 seconds, and then they are all released.</p>

Option	Description
Staggered	<p>Releases the users one by one from a synchronization point.</p> <p>The amount of time that each user waits to be released is chosen at random and is uniformly distributed within the range that you set in the Minimum time and the Maximum time fields.</p> <p>For example, if the Minimum time is 1 second and the Maximum time is 4 seconds, after the users reach the synchronization point (or the timeout occurs), each user waits between 1 and 4 seconds after being released. All users are distributed randomly between 1 and 4 seconds.</p>

- Set the timeout period. The timeout period begins when the first virtual user reaches the synchronization point. If all the users that are associated with the synchronization point do not reach it when the timeout period ends, any users at the synchronization point are released. A timeout of 0 means that there is no timeout.

Setting a timeout is useful, because one user might encounter a problem that prevents him from reaching the synchronization point. You do not want to hold up all users because of a problem with one user.

A user reaching a synchronization point after a timeout is not held. However, the user is delayed if the **Minimum time** and **Maximum time** are set.

Creating a Rate Schedule

By creating a Rate Schedule, you can model the different behaviors of how the application is accessed and measure the rate accuracy.

About this task

The Rate Schedule can be run only on agent locations.

When you run a Rate Schedule, it tries to achieve the desired rate with the specified number of clients. However, if the actual rate falls below the specified threshold level, the product will automatically add more number of clients within the purview of the maximum number of clients to reach the desired rate. You can modify the threshold level based on the need of the tests at **Window > Preferences > Test > Test Execution > Actual rate threshold (percentage)**.

- Click **File > New > Rate Schedule** or from the Test Navigator view, right-click the project and click **New > Rate Schedule**.
- Select a project for the Rate Schedule, specify the name of the Rate Schedule, and click **Next**.
- Specify the number of stages and the number of rate runner groups for the Rate Schedule.

The number of stages and rate runner groups is determined by the amount of load you want to generate and the model of the workload.

4. Click **Finish**.

The Rate Schedule is created.

5. To make changes to the stages in the Rate Schedule, for the **Load** category in the Rate Schedule Details section, click **Add**, **Edit**, or **Remove** buttons.
6. To add tests to the Rate Runner Group, select it and click **AddTest** and select a test.
7. To add a new Rate Runner Group, select the Rate Schedule and click **Add > Rate Runner Group**. For more information, see [Adding Rate Runner Groups on page 545](#).
8. To specify the agents to generate the load, select the **Agents** category in the Rate Schedule Details section and add agent locations. For more information, see [Adding Agent Locations on page 547](#).
9. **Optional:** Set the loops for the tests (or other schedule elements) to use: Right-click the Rate Runner Group to contain the loop, and click **Add > Loop**. Loops are used to run many iterations of a test, to run tests at a set rate, and to run tests in stages.
10. Add selectors and their weights: Right-click the schedule element to contain the selector, and click **Add > Random Selector**. Selectors are used to run a series of tests in random order, therefore emulating the varied actions of real users, instead of running each test within a Rate Runner Group sequentially. The weight that you assign each selector determines the statistical probability that its child element is selected during an iteration. For more information, see [Running tests in random order on page 564](#).
11. Add other schedule elements to refine the schedule structure: Right-click a schedule element, and click **Insert** (adds the new element before the selection) or **Add** (adds the new element after the selection).

Some of the elements to add to the Rate Schedule:

Element	Purpose	More information
Delay	Used to emulate actions accurately; for example, a transaction might be delayed before placing an order.	Delaying actions on page 540
Comment	Used for your notes and comments regarding the schedule element.	
Transaction	Used to group certain actions in a transaction.	Adding transactions on page 566

12. Save the Rate Schedule.

Setting rate load

By setting stages, you can model workloads over time and change the number of transactions that perform certain tasks to reflect real-world usage. You can divide the load into stages and collect performance metrics for each stage independently, which means that a single run can more efficiently accomplish the work of multiple runs. Each stage,

which lasts a specific amount of time and contains a specific number of transactions, defines a different load. Each load is used to derive a certain rate of transactions in a given time.

About this task

You can add load only to the **Rate Schedule Details** section. When you add the load, they are automatically displayed in the **Rate Runner Group Details** section. Managing the rate at which the load is to be run is specified in the **Rate Runner Group Details** section. You can change the rate at the time of the run too.

1. In the Test Navigator, browse to the rate schedule and double-click it. In the Rate Schedule editor that opens, the **Load** category displays one stage that runs for 10 minutes.
2. In Rate Schedule Details section, click **Add**.
3. Specify the duration of the stage and the settle time. The time required for the system to stabilize in between reaching the peak load and starting another stage is called settle time.
4. Click **OK**.

Result

The new stage is added to the Rate Schedule and is displayed to all the Rate Runner Groups.

5. In **Time limit to respond to a stop request**, specify a duration value.
If a stage contains fewer iterations than its predecessor, the excess iterations in the previous stage are asked to stop. This duration value gives a stopped iteration extra time to complete its current action (such as an HTTP request). If the iteration cannot complete its action before the time limit expires, it is forced to stop. Note that a long time limit might delay the next stage.
6. **Optional:** To update the stage details, click a Rate Runner Group, select a stage from the table, and click the **Show Advanced** check box. You can also click the **Edit** button and update the following options.

Iteration rate

Specify the rate at which a transaction needs to be executed.

Distribution

Specify the frequency rate at which the rate generator should run.

Change Rate

Use this option to ramp up or ramp down the iteration rate to the desired level.

Min Clients

Specify the minimum number of clients to be used to achieve the desired rate of execution. This option is typically used by an advanced user if the default number of clients do not meet the desired rate.

Max Clients

Specify the maximum number of clients to be used to achieve the desired rate of execution.

7. Save the Rate Schedule.

Rate Runner group overview

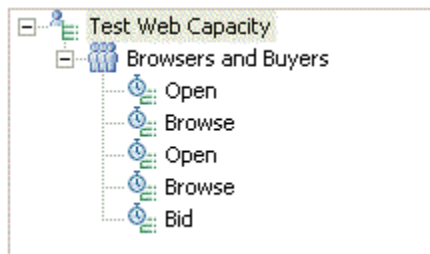
With the Rate Runner group, you can group tests in a logical order and run them in parallel.

You can control test execution in Rate Runner groups in several ways:

- **Group tests by characteristics:** For example, you could have two groups such as Buyers group and Browsers group. These groups represent the types of users on your system.
- **Influence the order in which tests are run:** When you run the Rate Schedule, the first test in each group runs in parallel, and not in sequence. After the first test in the group is completed, the second test runs, then the third, and so on.

Example

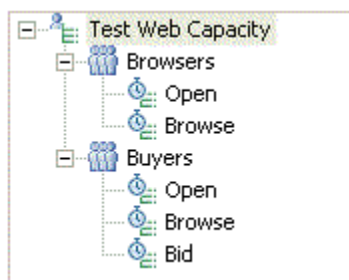
The following Rate Schedule contains one user group.



If you run this Rate Schedule with 10 iterations, they are assigned to the only Rate Runner group: Browsers and Buyers. When the run starts, the 10 iterations start running the first test in parallel. As soon as one test is finished, an iteration moves to the second test. Thus, you have 10 iterations, all starting simultaneously and running each test sequentially. This does not give you much control over the run.

Example

The following Rate Schedule contains the same tests in the same order, but they are divided between two Rate Runner groups. Conceptually, this Rate Schedule is easier to understand, because the user's tasks are grouped logically as Browsers browse and Buyers browse and then bid on a product.



Adding Rate Runner Groups

By defining Rate Runner Groups, you can group related tests. The tests within a group run in a sequence and the tests across the groups are run in parallel.

About this task

For example, there are two Rate Runner Groups, such as Rate Runner Group A and Rate Runner Group B. Each group contains one test. When you run the Rate Schedule, both the tests run in parallel. If there are more tests in the groups, those tests will run after the first test completes.

1. From the Test Navigator view, double-click the **Rate Schedule**.
2. In the **Rate Schedule** editor, select the schedule and click **Add > Rate Runner Group**.
3. In the **Group name**, type a descriptive name for the user group.
4. In the table, click **Show Advanced** check box and click **Edit**.
5. You can modify the following options for the Rate Runner Group:

Iteration Rate

Specify the rate at which a transaction needs to be executed.

Distribution

Specify the frequency rate at which the rate generator should run.

Change Rate

Click this option to set a delay between starting and stopping each iteration. The **All iterations over** option produces a slight variance that is representative of human behavior. The **One iteration every** option adds the same delay to each iteration, which prevents many iterations from performing an action simultaneously if the system under test has limitations.

Min Clients

Specify the minimum number of clients to be used to achieve the desired rate of execution. This option is typically used by an advanced user if the default number of clients do not meet the desired rate.

Max Clients

Specify the maximum number of clients to be used to achieve the desired rate of execution.

6. Define the scope of the tests in the Rate Runner Group.

Define Locally: Click this button to use the tests from the Rate Schedule.

Use compound test: Click this button to define the group from the compound test. You can click **Browse** to add an existing compound test to the Rate Runner group or click **Create** to create a compound test. The definition of the compound test is then applied to the Rate Runner group.

7. In the **Options** tab, define the protocol-specific options.

Select **Override think time options** to specify a think time behavior for the current group.

Use the recorded think time

Select to play back a test at the same rate that it was recorded. This option has no effect on the think time.

Specify a fixed think time

Each think time is exactly the same value that you specify. Although this does not emulate users accurately, it is useful if you want to play a test back quickly.

Increase/decrease the think time by a percentage

Type a percentage in the **Think time scale**. Each think time is multiplied by that percentage. A value of 100 causes no change in think times. A value of 200 doubles the think times, therefore the schedule plays back half as fast as it was recorded. A value of 50 reduces the think times by half, therefore the schedule plays back twice as fast. A value of 0 indicates no delays.

Vary the think time by a random percentage

Each think time is randomly generated within the upper and lower bounds of the percentages that you supply. The percentage is based on the recorded think time. For example, if you select a **Lower limit** of 10 and an **Upper limit** of 90, the think times will be between 10 % and 90 % of the original recorded think time. The random time is uniformly distributed within this range.

Limit think times to a maximum value

Setting a maximum think time is useful with tests that emulate actual think times. By setting a maximum, you do not have to search for and edit each long think time within a test, if, for example, you are interrupted during recording. No think time used will be greater than the maximum limit you set, even if you have chosen to vary the think time by a percentage that would exceed this maximum. To restore the original think times, clear this check box.

8. Click the **Edit Options** button to edit certain options that apply to specific test extensions.
9. In the **Variable Initialization** tab, create or select the existing variables to be used by all the tests in a Rate Runner group. For more information, see [Assigning variables to schedule and groups on page 557](#)
10. Save the schedule.

Run a Rate Runner group at a remote location

To generate good amount of load, you need computers with enough processing capability and throughput. It is a good practice to install the workbench on your computer and the agents on remote computers. The agents on the remote computers help generate the load.

Before you begin

Before you run a user group at a remote location, verify that:

- HCL OneTest™ Performance Agent is installed on the remote computer. The agent is configured and connected to the HCL OneTest™ Performance.
- One transaction rate per second is equivalent to one Virtual Tester. From 9.2.0.1, to run a Rate Schedule with more than 5 transactions per second, you must have appropriate VT Pack licenses.
- Firewall is disabled on the workbench computer or configured to allow incoming connections on the port number 7080.

- A reasonable number of iterations will run at the remote location. When you assign a Rate Runner group to a remote location, do not overload the remote computer (agent). If you exceed the number of iterations that the remote computer can run, the performance measurements of the server will be skewed because they will be affected by the performance of the computer. The test results will reflect the load of the computer more than the load of the server. For best results on a computer with a 1 GHz processor and 1 GB of RAM, do not exceed 1000 concurrent iterations.

About this task

Generally, you should run the groups at a remote locations. You *must* run a Rate Runner group at a remote location in these cases:

- When a large number of iterations are running and the local computer does not have enough processor or memory resources to support this load. You can conserve resources by running the iterations on different locations, so that reasonable number of iterations run on each computer.
- When a test requires specific client libraries or software. The Rate Runner group that contains this test must run on a computer that has the libraries or software installed.

1. From the Test Navigator view, double-click the Rate Schedule
2. Select the name of the Rate Schedule and from the Rate Schedule Details section, select the **Agents** category. You can also select the agents at the Rate Runner Group level. Select a Rate Runner Group and click the **Agents** tab.



Note: When the agents are declared for both Rate Runner Group and Rate Schedule, the Rate Runner Group agents take precedence.

3. To declare a remote location:

- a. Click **Add > Add New**.

Result

The **Add New** wizard opens. On the first page of the wizard, you can specify general properties for the remote location.


- b. In **Hostname**, type the IP address or the fully qualified host name of the remote computer.
- c. In **Name**, type a descriptive name for the remote computer.
- d. In **Deployment Directory**, enter a fully qualified pathname of the directory, which is in the remote computer to store the test assets.




Note: The environment variables such as `%TEMP%` are not supported in the **Deployment Directory** field.

- e. In **Operating System**, select the operating system of the remote computer, and then click **Next**.

- f. Specify the IP aliasing properties for this location. To make it appear as though each virtual user has its own IP address, click **Enable IP Aliasing**.
- g. To use IP addresses from all network interfaces at the remote location, click **Use IP addresses from all network interfaces**.
- h. To use addresses from a subset of network interfaces, click **Only use IP addresses from the following network interfaces**.

 **Tip:** Click **Add** to add the name of an interface and **Edit** to change the interface name. Specify network interfaces separated by commas, for example, `eth0, eth1`. If you do not use this form, the connection attempt fails.

- i. Click **Next**. On the third page of this wizard, you can specify file locations.
- j. In **File name**, type the name of the file to contain information about this computer, and then click **Next**.

 **Note:** The data stored in the file includes information such as the host name and deployment directory. You can change this information later by opening the Test Navigator and double-clicking the file.

- 4. To add an already declared location:
 - a. Click **Add > Add Existing**.
 - b. In the **Select Location** window, select the computer on which the user group will run, and then click **OK**.
- 5. Save the schedule.

Think time overview

Think time is a delay in the processing of a request to reproduce the time that a human would take to read or examine the data that is displayed from a previous user action. Think time is calculated from the time that a request is received (that is, the display is complete on the monitor) until the time that the user clicks a key or link to perform an action.

Setting think time behavior in schedules

You can increase, decrease, or randomize think time in your tests, or you can play it back exactly as recorded.

To set the think time in a schedule:

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.
2. You can set the think time behavior for an entire schedule or you can override the think time behavior for any specific user group.

Choose from:

- To set the think time behavior for the entire schedule, in the **Schedule Contents** area, click the name of the schedule and click the **Think Time** tab.
 - To override the think time behavior for a specific user group, in the **Schedule Contents** area, click the name of the user group, click the **Options** tab, and select the **Override think time options** check box.
3. Set the think time behavior to one of the following options:

Option	Description
Use the recorded think time.	This option does not affect the think time. The time that it takes for a test to play back is the same as the time that it took to record it. So, for example, if you were interrupted for five minutes during recording, the same five-minute think time occurs when you run the test.
Specify a fixed think time.	Each virtual user's think time is exactly the same value: the value that you type. Although this option does not emulate users accurately, it is useful if you want to play a test back quickly.
Increase/decrease the think time by a percentage.	In the Think time scale field, specify a percentage by which each virtual user's think time is multiplied. A value of 100 indicates no change in think time. A value of 200 doubles the think times, so that the schedule plays back half as fast as it was recorded. A value of 50 reduces the think times by half, so that the schedule plays back twice as fast. A value of 0 indicates no delays at all.
Vary the think time by a random percentage.	Each virtual user's think time is randomly generated within the upper and lower bounds of the percentages that you supply. The percentage is based on the recorded think time. For example, if you select a lower limit of 10 and an upper limit of 90, the think times are between 10 percent and 90 percent of the original recorded think time. The random time is distributed uniformly within this range.

4. To set a maximum think time, select the **Limit think times to a maximum value** check box and specify a value.
5. Save the schedule.

Limiting think times in schedules

You can speed up playback by defining a maximum value for the think times of virtual users in schedules.

About this task

Setting a maximum think time is useful with tests that mimic the actual user's think times. For example, if you are interrupted when you record a test, you do not have to record the test again. Instead, you can set a maximum think time. By setting a maximum, you can truncate all think times that exceed the specified value, without having to search for and edit each long think time. No think time used will be greater than the maximum limit you set, even if you have chosen to vary the think time by a percentage that would exceed this maximum.

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.

2. In the Schedule Contents area, click the name of the schedule.
3. Click the **Think Time** tab and select the **Limit think times to a maximum value** check box.
4. In **Maximum think time**, type a number and select a time unit.

What to do next

To restore the original think times, clear the **Limit think times to a maximum value** check box.

Limiting think times in tests

You can speed up playback and the time that debugging tests requires by defining a maximum value for the think time of a single user test playback.

About this task

Limiting think time is especially useful when you are debugging a test. By setting a maximum, you can truncate all think times that exceed the specified value, without having to search for and edit each long think time. No think time will be greater than the maximum you set.



Note: This maximum applies to running individual tests. The limit does not apply to running tests in schedules. When a test is run as part of a schedule, the maximum think time value is ignored.

1. In the Test Navigator, click **Window > Preferences > Test > Test Execution**.

Result

The **Test Execution** window opens.

2. The default value for maximum think time is 2 seconds. To change the maximum think time value, in **Max think time (seconds)**, change the number. If you do not want to limit the amount of think time during test playback, enter a large number.
3. Click **Apply**, and then click **OK**.

What to do next

To restore the original think time default settings, click **Restore Defaults**.

Working with agents

If you have a significant workload to test, typically a single computer might not be able to process the load efficiently. You need to distribute the load across multiple computers, also called HCL OneTest™ Performance agents. The agents are installed on computers to generate the load on the application.

You create VU Schedule to generate user load and you create Rate Schedule to generate transaction load on the agent locations. You use the HCL OneTest™ Performance workbench to distribute the load. When you distribute the load among agent computers, carefully consider the load that each agent computer can take efficiently.

To view the health of the agent computers, see [Agents Health Report on page 807](#).

HCL OneTest™ Performance agent, also known as load generation agent, is used to generate load for the application under test by creating simulated connections.

When you install HCL OneTest™ Performance Agent, you can specify the host name of the workbench to poll. After the installation, the agent automatically connects to the workbench. If you want to share the agent with multiple workbenches, you can either manually add them to the `majordomo.config` file at `installationDir/Majordomo` or use the **Share Agent with New Workbench** option that is mentioned in the [Checking status of agents on page 553](#) topic.

When you run a schedule with multiple agents, an agent might be lost, especially during the long load test run. Losing an agent is not common and occurs during some extreme cases such as when computer's memory is exhausted. When an agent is lost, by default, the schedule is stopped. When the schedule is stopped in this manner, you must fix the reason of agent loss or add more agents before running the schedule. To continue to run the schedule without the lost agent, in the Schedule editor, click the **Advanced** tab and clear the **Loss of an agent halts execution** check box.

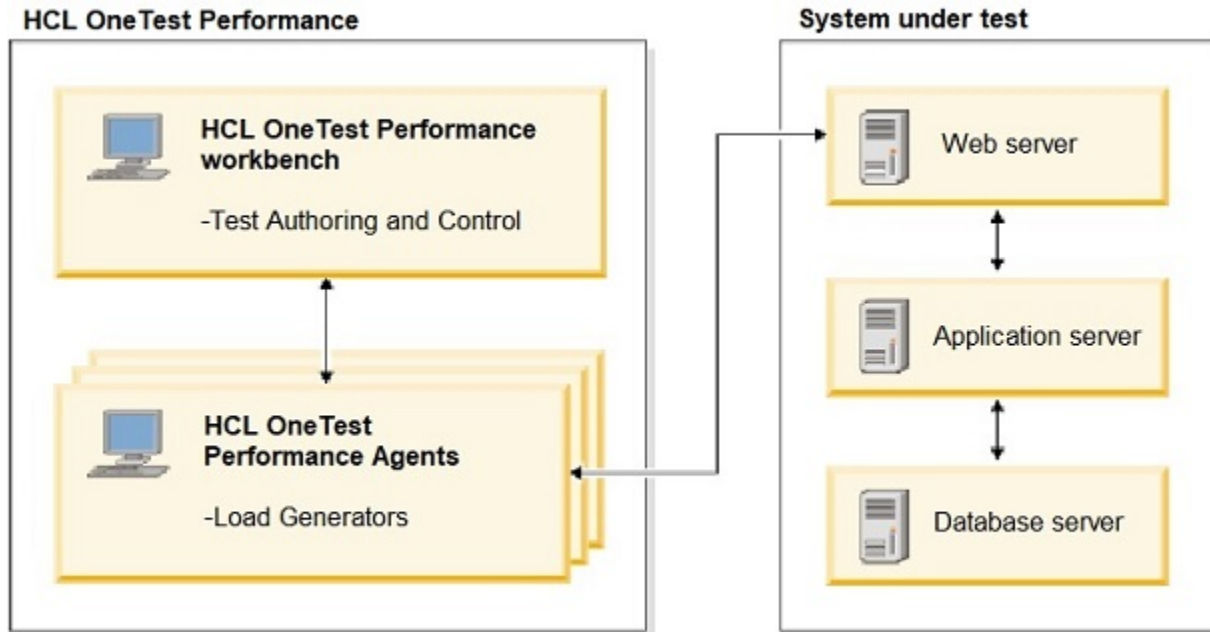


Note:

- You can install HCL OneTest™ Performance and HCL OneTest™ Performance Agent on any of the supported platforms that are mentioned in the download document at https://hclpnpsupport.hcltech.com/kb_view.do?sysparm_article=KB0069531. It is not necessary that both are installed on the same platforms or the bitness of the platform must be same.
- If you are creating load on the HCL OneTest™ Performance Agent computer, ensure that, in the workbench, you point to the agent computers.
- If you are using HCL OneTest™ Performance workbench 8.3 or later, you must use HCL OneTest™ Performance Agent 8.3 or later. Also, it is recommended to use the same release version for both.

The HCL OneTest™ Performance workbench automatically deploys test assets that are required for execution to participating agents. For more information about the deployment directory, see [Managing deployment directory on page 555](#).

A typical setup of HCL OneTest™ Performance workbench, HCL OneTest™ Performance agents, and the system under test is shown here:



Related information

[Troubleshooting performance testing on page 908](#)

[Configuring port numbers for agents on page 554](#)

Checking the status of agents

Before running a schedule or test, you can check whether the agents are active and connected to the workbench.

Before you begin

Ensure that the HCL OneTest™ Performance agent is running and points to the correct workbench. When installing the agent, ensure to specify the correct non-secure port number where the workbench listens for the agent. You can view or change the non-secure port number of the current workbench at **Windows > Preferences > Test > Server**.

About this task

An agent can be in one of the following statuses:

Agent status	Description
Ready	Majordomo is running on an agent computer and is in regular (default 5 seconds) contact with a workbench.
Busy	Agent is participating in the schedule execution.
Lost Contact	Agent has not contacted the workbench for the past 10 seconds.

If you want an agent to take direction from more than one workbench, you can share the agent with those workbenches. Also, if you no longer wish for an agent to contact a workbench you can disconnect it.



Note: The `majordomo.config` file located at `installationDir/Majordomo` in the HCL OneTest™ Performance agent computer contains the host name and the port number of the workbench the agent polls for work to do. By default, the agent polls for work at the interval of every 5 seconds.

1. To check the status of agents, on the toolbar, click .

Result

The **Agent Status** window lists the agents that are connected to the workbench, the license mode, operating system, architecture, and status of the agents.

2. To share an agent with other workbenches, in the **Agent Status** window, select an agent and click **Share Agent with New Workbench**.
3. Specify the host name and port number of the new workbench and click **OK**.
4. To disconnect an agent from the workbench, select an agent and click **Disconnect Agent from this Workbench**.

To disconnect the agent from all the workbenches, you must perform the disconnection from each workbench.

Configuring port numbers for agents

If any service, such as an application server, on the workbench computer uses the default port numbers, you can change the port numbers so that agents can communicate through the new port numbers.

About this task

By default, the port numbers for unsecured and secured ports are 7080 and 7443. The HCL OneTest™ Performance agents poll the workbench using the non-secure port number. If you want to encrypt the communication between the workbench and agents, select **Workbench and agent communication is encrypted using TLS/SSL**. HCL OneTest™ Performance then uses the secure port number for communicating with agents. The workbench internally sends the secure port number to the agents via messages exchanged over the non-secure port number.



Note:

- If you change the non-secure port number, you must configure the agents to poll the new port number. In the HCL OneTest™ Performance computer, open the `majordomo.config` file located at `installationDir/Majordomo` and update the port number.
- If two instances of a workbench are opened on one computer, first workbench gets access to port 7080 to communicate to an agent. For the another workbench to access the agent, you must use a different port number.
- To work with a dataset, you must ensure that two consecutive port numbers are available. One of the ports is used to communicate between the workbench and agent, and the other is for the dataset server. For example, you must ensure that the port number 7081 is open for the dataset server if the



default port number 7080 is used to communicate between the workbench and agent. Similarly, if you use any different port number for communication between the workbench and agent, you must ensure that the next port number is available for the dataset server.

1. In HCL OneTest™ Performance, click **Windows > Preferences > Test > Server**.
2. Specify the new port numbers and click **OK**.

What to do next

You must now update the `majordomo.config` file for the agents to poll the new port numbers.

Managing the deployment directory for agents

You can choose to delete or keep the deployment directory for agents after schedule execution. By default, HCL OneTest™ Performance does not delete the deployment directory.

About this task

The deployment directory stores the files that are required for test execution. If you do not delete the deployment directory, the next time that you run a schedule, HCL OneTest™ Performance only retrieves the test assets that it does not have or that have changed since the last execution. This mechanism reduces the launch time for subsequent test executions.

However, if you have many schedules and the accumulation of files in the deployment directory might create a problem, you can choose to delete the deployment directory on the agent computers. The deletion of the deployment directory depends on the successful completion of a schedule. A schedule execution is deemed successful when all the agents reach a state of inactivity with no active users or when the schedule execution reaches the end of the last stage and then stops.



Note: If the run is stopped manually, the run encounters a severe error, or communication between the agents and workbench stops, the deployment directory is not deleted even if you specify to delete it.

To delete the deployment directory on the agent computers after schedule execution, click **Windows > Preferences > Test > Server** and select the **Delete deployment directory on the agent after execution** check box.

Configuring the high-resolution timer

You can configure agent computers to use a high-resolution timer when collecting performance data. The high-resolution timer ensures that all measurements are precise to within 1 millisecond.

Before you begin

The high-resolution timer is used by default on the Linux™ operating system. You do not need to configure the high-resolution timer on the Linux™ operating system. For the Microsoft™ Windows™, AIX®, and z/OS® operating systems, the high-resolution timer is not used by default.

Enabling the high-resolution timer can increase processor usage on Windows™ agent computers. Typically, processor usage increases by 10% to 25%, but the increase can vary based on workload. Enable the high-resolution timer on Windows™ agent computers if you are measuring response times less than 15 milliseconds. Enable the high-resolution timer if you are measuring response times of less than 150 milliseconds and need 1 millisecond precision instead of 15 millisecond precision. If you enable the high-resolution timer on Windows™ agent computers, monitor the processor usage to determine whether the trade-off of higher processor usage is acceptable.

To enable the high-resolution timer on an agent computer:

1. In the Test Navigator, open the location that represents the agent computer.
2. Click the **General Properties** tab.
3. On the **General Properties** page, click **Add** to create a property for the selected location.
4. In the **New Property** window, create a property entry:
 - a. In **Property Name**, type `RPT_VMARGS`.
 - b. In **Operator**, select `=`.
 - c. In **Property Value**, type `-DrptNanoTime`.

If you need to set multiple RPT_VMARGS values for a location, place them in the same property entry and separate them with a space. Do not use multiple property entries to set multiple RPT_VMARGS values for a location.

5. Click **OK**, and then save the location.

Result

The high-resolution timer is enabled for the location.

Adding a test to a schedule

By adding a test to a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule., you can emulate the action of an individual user.

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.

2. Right-click the schedule element that will contain the test, and then click **Add > Test**.
3. In the **Select Tests** window, expand the project name to display the test that you want to add.
4. Click the name of the test, and then click **OK**.

Result

The test is displayed in the schedule.

Adding must run tests

In a schedule, you can use the **Finally** block to specify tests that must be run after the main workload is completed, when the last stage duration is expired, or a schedule is stopped manually.

About this task

A schedule can contain many user groups. A user group can contain only one Finally block. A Finally block can contain many tests.

1. From the Test Navigator, open a schedule.
2. Select a user group and click **Add > Finally**.
3. Select the **Finally** block and click **Add > Test Invocation**.
4. Select the tests that you want to add and click **OK**.

Assigning variables to schedule and groups

In addition to assigning variables at the test level, you can assign variables at the A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule. level and User or Rate Runner group level. When you assign variables at the schedule level, all the tests and groups in the schedule can use the variable initial values, if they have the same variable names.

About this task

When you initialize variables at the schedule level, all the groups in the schedule use the variable initial values, except those for which a specific value is defined. If the same variable initial value is defined at the group level and schedule level, precedence is given to the group level and then to the schedule level. If the visibility of a test variable is set to **This test only**, the test does not use the value that is defined at the group level or schedule level.

If a variable is initialized at various places such as test, compound test, schedule, or user group, the product uses the following order to initialize the value of the variable when running the test. The variable set in the variable table of the compound test editor takes the highest precedence followed by others:

1. Compound test setting in the variable table UI
2. Compound test specified in a var file
3. User group setting in the variable table UI
4. User group specified in a var file
5. Schedule specified setting in the variable table UI
6. Schedule specified in a var file
7. Command line



Note: You must select **All tests for this user** from the **Visible in** drop-down list to take the precedence of variable initialization.

1. In the **Test Navigator** view, double-click a schedule to open it.
2. To assign a variable at the schedule level or a group level, in the Schedule Contents area, select a schedule or a group.
3. In the **Schedule Elements Details** area, for a schedule select the **Variable Initialization** category, or at the group level, click the **Variable Initialization** tab, and click **Add**.
4. Type a variable name and its initial value.

If you have already defined the variables at the test level and want to reuse them, click **Select existing variables**.

5. **Optional:** To use the variables from an existing file, click the **Use variable initial values file** check box, browse for the file, and click **Finish**.
6. Click **Options > Save**.

Defining requirements in schedules

You can define performance requirements to specify the acceptable thresholds for the performance parameters in a schedule. The performance requirements that you define can also be used to validate the service-level agreements.

About this task

You can define both the performance and functional requirements in the schedules. The verdict of the schedule is computed based on the requirements defined in the schedule. You can view the verdict in the Requirements report. You can add the requirement for counters, which you can generate, by using the custom code in a schedule. After the test run is initiated, you can view the information about the counters graphically when the test run starts the custom code.

1. Find the schedule you want and double-click it.

Result

The schedule is displayed.

2. Perform the following steps in the **VU Schedule Details** pane:
 - a. Select **Requirements** from the **Category** field.
 - b. Select **Enable Requirements** check box.
3. Enter a name for a requirement in the **Name** field.
Alternatively, you can click **Use Defaults** to use the default name for a requirement.
4. Perform the following steps, to set an *Operator* and *Value* for the requirements you defined:
 - a. Click the **Operator** field to display the list of mathematical operators and select an operator for the requirement from the list.
 - b. Enter a value for the requirement in the **Value** field.
The value you enter is selected as the standard value for the requirement by default.
 - c. Clear the **Standard** check box or let the selection remain as is based on the following options:
 - Clear the selection – This action enables the value of the selected requirement to be a supplemental value.
 - Retain the selection – This action enables the value of the selected requirement to be a standard value.
5. Perform the following steps to add the counter information generated by using the custom code to a requirement:

- a. Expand the **Custom** section and double-click the row.

Result

The **Add Custom Requirement** window is displayed.

- b. Enter the path that you used in the custom code in the **Counter path** field.

For example, if you specify the counter as `getValueCounter("New Counter","Test","Test Value")` in the custom code, then you must enter the counter path as `/New Counter/Test/Test Value`.

- c. Select the **Component** from the list, and then click **OK**.

6. To remove the selected requirement, select the requirement, and then click **Clear**.



Note: The requirement is disabled and can be redefined.

Exemple

You can define the performance requirements in a schedule or in a test if your protocol supports it. When you define a requirement in a test, the requirement is defined individually for each test element even if you select multiple test elements. When you define a requirement in a schedule, the requirement is applied to the aggregate of test elements.

For example, assume that you select every page in a test and define the following as a requirement: `Page Response Time(Average)` must be less than 5 seconds.

This means that if one page in the test has a response time of 6 seconds, the requirement on that page fails. If the other pages have a response time of less than 5 seconds, the requirement is pass.

For example, in a schedule you can define the following requirement: `Response Time For All Pages (Average)` as less than 5 seconds. This measures the average response time for all the pages. If one page has a response time of 30 seconds and if there are 7 pages that have a lower response time such that the average response time is less than 5 seconds, then the page with a response time of 30 seconds also passes the requirement.

What to do next

After you defined the requirements in a VU Schedule, you can run the requirement against the application under test, and then analyze the results that are reported.

Related information

[Defining requirements in tests on page 285](#)

[Reports and counters on page 805](#)

[Creating custom Java code on page 657](#)

[Adding custom counters to reports on page 678](#)

Repeating tests in a schedule

By adding a loop to a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule., you can repeat a test for a number of iterations and set the rate for running a test. If the loop contains a synchronization point, the synchronization point is released after the first iteration of the loop and stays released for all further iterations.

About this task

A schedule that contains only user groups and tests runs each test in a user group sequentially. Loops provide more sophisticated control than running a simple sequence of consecutive tests.

You can set a loop within a schedule or a test. The following table shows the advantages of both methods:

Loop location	Results
Schedule	<p>Loops in schedules are easy to locate and modify.</p> <p>Loops in schedules close the server connection at the end of each iteration and reopen it at the beginning of the next iteration. This action models the behavior of a user closing and reopening a browser.</p> <p>Use this method to run a test at a set rate. For more information, see Running tests at a set rate on page 563.</p>
Test	<p>Loops in tests can be more granular, and thus provide a higher level of control.</p> <p>Loops in tests reuse the server connection during each loop iteration.</p> <p>Use this method, with loops that have high iteration counts, to stress test a server.</p>

To add a loop to a schedule:

1. In the Test Navigator, browse to the schedule and double-click it.

Result
The schedule opens.
2. Click the item that you want to be the parent of the loop, and then click **Add > Loop**.
3. In the Schedule Element Details area, type the number of iterations for the loop to repeat.

Option	Description
Count-based	Runs for the number of iterations that you select.
Time-based	Runs at least for the time that you specify. The loop always finishes the iteration. For example, if you select a time of 1 second and a loop takes 10 seconds to run, the loop finishes one iteration, and then checks the time.

Option	Description
Infinite	Runs until the schedule duration ends. Use this option to gather performance data over time: set an infinite loop, and then set the schedule to stop after a specific time.

4. To maintain a set transaction rate for all schedule items that are children of this loop:

Example

- a. Select **Control the rate of iterations**.
- b. In the **Iteration rate** field, type a number, and select a time unit.
This sets the actual rate.
- c. Select or clear the **Randomly vary the delay between iterations** check box. Selecting this check box causes the delay to vary slightly. This option models users more accurately, because the iterations are spread out randomly over a certain period of time.

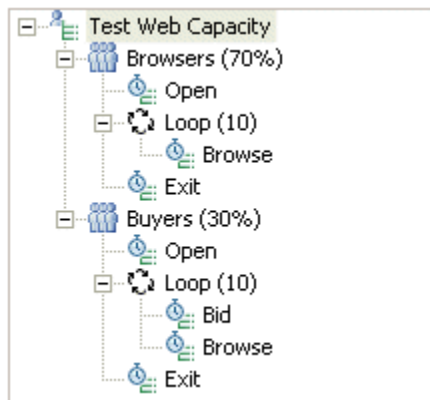


Note: Statistically, the **Randomly vary the delay between iterations** option sets delay amounts at random from a negative exponential distribution with the same mean as the fixed delay value. The negative exponential distribution has a long "tail," which means that a very small number of delays will have very large values. Therefore, make sure that the application you are testing is not negatively affected by long periods of inactivity (such as a timeout that disconnects the user).

- d. Select or clear the **Delay before the first iteration of the loop** check box. Selecting this check box staggers the first delay in each iteration so that you get a realistic mix at the first iteration.

Example

The following schedule contains two loops. Note that the Browse and Bid tests, which are controlled by the loop, must be children of the loop.



Creating rate generators in user groups

A rate generator is a workload container that specifies the number of tasks that the virtual testers run in a given time period. For example, you might be testing an Order Entry group that completes 10 forms every hour, or you might be testing a web server that you want to be able to support 100 hits every minute. Use a rate generator to model this time-based behavior.

Before you begin

Create a schedule and ensure that user groups have tests.

About this task

If the rate generator does not meet the target rate, it could mean that there are performance issues with the application or there are insufficient virtual users or agents to meet the target rate. All rate generators are coordinated, which means they have a built-in synchronization point and all virtual testers work together to generate the workload. You can add a rate generator to a user group in a schedule or to a compound test.

Some of the important points about the rate generator are as follows:

- You cannot have elements such as loop, finally, rate generator, or transaction as a parent to the rate generator.
 - The same rate generator, as identified by name, can appear in more than one User Group in a schedule.
 - Rate generators can only be added to percentage user groups. A percentage user group containing a rate generator may not be modified to be a fixed user group.
1. From the Test Navigator view, double-click a Schedule to open.
 2. Select a user group and click **Add > Rate Generator**.
 3. Specify a name to the rate generator.
 4. Set the duration of the rate generator.
 - To specify the number of iterations that the rate generator should run, select **Count-based**.
 - To specify the minimum time for the generator to run, select **Time-based**. The generator always finishes the iteration. For example, if you select a time of 1 second and a rate generator takes 10 seconds to run, the generator finishes one iteration, and then checks the time.
 - To continue running the rate generator until it is manually stopped by you or when the last schedule stage duration expires, select **Infinite**.
 5. The **Uninterruptible iteration** option modifies the behavior of the rate generator if schedule execution is stopping. Select this check box to continue running the rate generator until the current iteration completes. If a timeout to stop schedule execution occurs before an iteration completes, the rate generator stops.
 6. Set the rate type of the rate generator.
 - To ensure that all of the virtual users in the schedule maintain the desired transaction rate irrespective of the user load, select **Total Rate**.
 - To adjust the transaction rate proportionally to the number of virtual users specified in a stage of a schedule, select **User Rate**. For example, to compare a workload at 100 virtual users a minute, 200

virtual users a minute, and 300 virtual users a minute, set the stages with that many virtual users. User rate is dynamic and is adjusting at stages, whether users are ramping up or down or are added manually.

7. In **Iteration rate**, specify the number of times the rate generator runs in a specified time period.
8. In **Variance**, specify a percentage value that is used to determine the range when the transactions start.

For example, the transaction rate is 4 every minute (that is, 1 transaction for every 15-second interval). If you select a variance of 20%, your transaction has a 3-second window on each side of that 15-second interval, because 20% of 15 seconds is 3 seconds. Therefore, the first transaction starts at 12–18 seconds. The second transaction starts 15 seconds (plus or minus 3 seconds) after the first transaction starts. If the first transaction starts at 12 seconds, the second transaction would start at 24 to 30 seconds. However, if the first transaction starts at 18 seconds, the second transaction would start at 30 to 36 seconds.

Because each transaction starts randomly within the range that you specify, it is normal for transactions to run at a rate that is faster or slower than the rate that you selected for short periods of time. For example, if a transaction starts every 12 seconds for a minute, the rate for that initial interval is 5 every minute and not the rate of 4 every minute that you selected. Over time, however, the transaction rate averages out to 4 every minute.

9. In **Distribution**, specify the frequency rate at which the rate generator should run.
 - Select **Constant** for the workload of the rate generator to occur exactly at the rate you specify. For example, if the iteration rate is 4 every minute, the workload starts at 15 seconds, 30 seconds, 45 seconds, and 60 seconds, which is exactly 4 every minute, evenly spaced, with a 15-second interval.
 - Select **Uniform** when the time between each workload is not constant. However, the workload that occurred over time averages out to the rate that you specified. The time between the start of each workload is chosen randomly with a uniform distribution within the selected range as specified in **Variance**.
 - Select **Negative Exponential** to emulate the spike of activity followed by a dull period that is typical of user behavior. So, if the rate is 4 every minute, the probability that the workload starts immediately is high, but decreases over time. HCL OneTest™ Performance maintains the desired average rate.
10. Add the test under the Rate Generator and save the schedule. After the schedule run completes, in addition to the Performance Report, the Rate Generator report is available.

Related reference

[Rate Generator report on page 813](#)

Running tests at a set rate

To run a test at a set rate, you add a loop to the schedule to control the iteration rate, and then add tests to the loop. The tests, which are children of the loop, are controlled by the loop. If the loop contains a synchronization point, the synchronization point is released after the first iteration of the loop and stays released for all further iterations.

To add a loop that controls the iteration rate for running tests:

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.

2. Click the element that will be the parent of the loop, and then click **Add > Loop**.
3. In the **Schedule Element Details** area, type the number of iterations that the loop will repeat.
4. To continue executing the loop even after a request to stop the schedule execution is issued, select the **Uninterruptible iteration** check box.

The schedule execution stops after the current loop iteration is complete. This check box is available only for schedules.

5. To maintain a set transaction rate for all schedule items that are children of this loop, select the **Control the rate of iteration** check box.

6. At **Iteration rate**, type a number and select a time unit.

This sets the actual rate.

7. Select or clear the **Randomly vary the delay between iterations** check box. Selecting this check box causes the delay to vary slightly. This option models your users more accurately because rather than delaying iterations at fixed intervals, the delay amounts are varied randomly while maintaining the same average iteration rate.



Note: Statistically, the **Randomly vary the delay between iterations** option sets delay amounts at random from a negative exponential distribution with the same mean as the fixed delay value. The negative exponential distribution has a long "tail," which means that a very small number of delays will have very large values. Therefore, make sure that the application you are testing is not negatively affected by long periods of inactivity (such as a timeout that disconnects the user).

8. Select or clear the **Delay before the first iteration of the loop** check box. Selecting this check box staggers the first delay in each iteration, so that you get a realistic mix at the first iteration.

What to do next

After you have added the loop, you add the schedule items, usually tests, that the loop controls.

Running tests in random order

A schedule that contains only user groups and tests will run each test in a user group sequentially. By adding a random selector to a schedule, you can repeat a series of tests in random order, thus emulating the varied actions of real users.

To add a random selector to a schedule:

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.

2. Click the name of the schedule element to contain the random selector, and then click **Add > Random Selector**.

- In the Schedule Element Details area, add the number of iterations to loop.

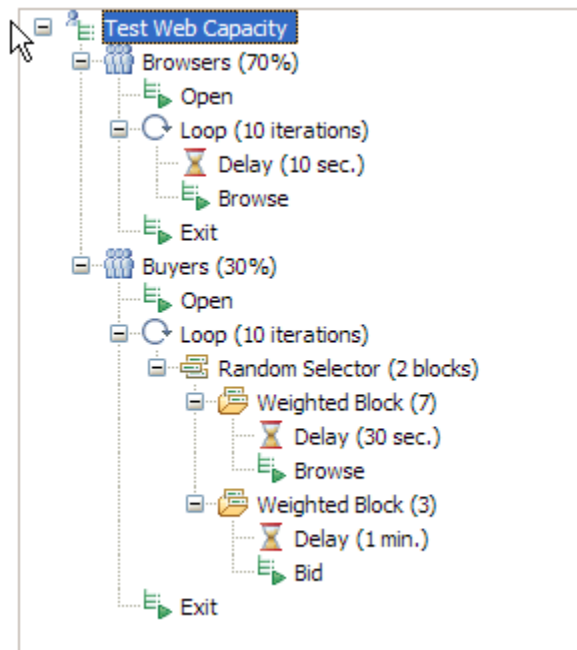
Result

Think of yourself as a "random selector." You are blindfolded, and you have a bucket that contains 10 red balls and 10 green balls. You have a 50% chance of picking a red ball, and a 50% chance of picking a green ball. You pick a ball randomly—it is red. You then replace the ball in the bucket. Every time you pick a ball, you have a 50% chance of getting a red ball. Because the ball is replaced after each selection, the bucket always contains 10 red balls and 10 green balls. It is even possible (but unlikely) that you will pick a red ball every time.

- Set the weight of the random selector. The weight determines the statistical probability that a specific element will be selected.
 - Right-click the random selector and click **Add > Weighted Block**.
 - In the **Weight** field, type an integer.
This integer shows the relative proportion that each test runs.

Result

Assume that a random selector contains two tests: Browse and Bid. You assign Browse a weight of 7 and Bid a weight of 3. Each time the loop is processed, Browse has a 70% chance of being selected, and Bid has a 30% chance of being selected.



Example

When a selector contains many different weights, you can mathematically determine the likelihood that a block will be executed. To do this, add the weights together and divide the weight for each block by that total.

For example, assume a selector contains six blocks set to the following weight:

- two blocks set to a weight of 1
- one block set to a weight of 2
- two blocks set to a weight of 5
- one block set to a weight of 9

The total of the weights is: $1 + 1 + 2 + 5 + 5 + 9 = 23$. Therefore, the statistical likelihood of selection is:

Weight of block	Likelihood of block being selected
1 (two blocks)	$1/23 = 0.0435$, or about 4.35% (for each block)
2	$2/23 = 0.0870$, or about 8.70%
5 (two blocks)	$5/23 = 0.2174$, or about 21.74% (for each block)
9	$9/23 = 0.3913$, or about 39.13%

Note that a higher weight increases the likelihood, but does not guarantee, that a block will be executed. Some variation might occur. For example, if you run a test 23 times, you cannot predict that the first and second blocks will execute exactly once, the third block exactly twice, the fourth and fifth blocks exactly five times, and the sixth block exactly nine times. However, the more times that the blocks are executed, the more accurate this prediction is.

Adding a transaction to a schedule

A *transaction* is a specific group of test elements whose performance you are interested in. When viewing the test results, you can view performance data about any transactions that you have added.

About this task

You can add a transaction to a test or to a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule.. When specifying error conditions for a transaction, you can set the behavior of the condition to affect the health of the transaction. The Transaction Health tab of the Transaction report displays the health of the transactions in a test.

To put a group of schedule elements into a transaction:

1. In the Test Navigator, browse to the schedule and double-click it.

Result
The schedule opens.
2. In the schedule, select the test elements to group together. Use Shift+click to select multiple contiguous elements; use Control+click to select multiple noncontiguous elements; each noncontiguous element is put in a separate transaction.
3. Click **Add** (to place the transaction after the selected element) or **Insert** (to place the transaction immediately before the selected element or block), and click **Transaction**.
4. In the **Schedule Element Details** area, give the transaction a meaningful name. This is useful in the Transactions report, which lists transactions by name.

What to do next

You can now use the **Add** or **Up** and **Down** buttons to add the tests or elements to the transaction.

Emulating network traffic from multiple hosts

By default, when you run a schedule, each virtual user has the same IP address. However, you can make each virtual user appear as though it is running on its own host. To do this, you configure IP aliases on the host computer, and enable IP aliasing in the schedule. When you run the schedule, the network traffic will appear to be generated by multiple hosts.

About this task

To avoid IP conflicts, the IPs have to be legitimate, available within the same subnet, and blocked for distribution.

Configuring IP aliases for a remote Windows™ location

To make it appear that a virtual user has its own IP address during a schedule run, configure IP aliases for each Windows™ remote location.

1. Click **Start > Settings > Control Panel > Network Connection**.
2. Open the network interface that you want to associate the IP aliases with. In most cases this is the Local Area Connection. Click **Properties**.
3. Scroll down to **Internet Protocol (TCP/IP)**, and click **Properties**.
4. You must be using static IP addresses to create IP aliases on this host. Therefore, confirm that **Use the following IP address** is selected, and then click **Advanced**.
5. Create the IP aliases:
 - a. Click **Add** in the **IP Addresses** area of the IP Settings page to specify the IP address of the new alias. Make sure that the address is valid for the network and is not being used by another host.
 - b. Enter the IP address and the subnet mask of the new alias.
6. After you create the alias, click **OK** in each previous dialog box to complete the configuration.
7. Set the schedule so that the virtual users will use IP aliases during a run; for information, see [Enabling virtual users to use IP aliases on page 569](#).

Results

When you run the schedule, it will give the impression that the network traffic is being generated from multiple hosts.

What to do next

You can insert custom code into your test to retrieve the runtime IP addresses of each virtual user. For information, see [Retrieving the IP address of a virtual user on page 569](#).



Note: To add multiple IP aliases, use the netsh command, as shown in the following example:

```
netsh -c Interface ip add address name="Gig Ethernet" addr=10.10.0.21 mask=255.255.0.0
```

The ntcmds.chm file, typically located in C:\WINDOWS\help, contains more details about the netsh command. When you are finished with the IP aliases, use the following command to remove them:

```
netsh -c Interface ip delete address name="Gig Ethernet" addr=10.10.0.21
```



You can also use a batch file to add and delete the aliases.

Configuring IP aliases for a remote Linux™ location

To make it appear that a virtual user has its own IP address during a schedule run, configure IP aliases for each Linux™ location.

1. Use the `ip(8)` command to create an IP alias.

Example

The following example attaches the IP address 9.37.207.29 to the `eth0` network interface:

```
# ip address add 9.37.207.29 dev eth0
```



Note: To create a large number of aliases on a Redhat Linux™ platform, follow the instructions in the file `/etc/sysconfig/network-scripts/ifup-aliases`.

The following example deletes the alias:

```
# ip address del 9.37.207.29 dev eth0
```

2. Set the schedule so that the virtual users will use the IP aliases during a run; for information, see [Enabling virtual users to use IP aliases on page 569](#).

Results

When you run the schedule, it will give the impression that the network traffic is being generated from multiple hosts.

What to do next

You can insert custom code into your test to retrieve the runtime IP addresses of each virtual user. For information, see [Retrieving the IP address of a virtual user on page 569](#).

Configuring IP aliases for a remote AIX® location

To make it appear that a virtual user has its own IP address during a schedule run, configure IP aliases at each remote AIX® location

1. Use the `ifconfig` command to create an IP alias.

To have the alias created when the system starts, add the `ifconfig` command to the `/etc/rc.net` script.

Example

The following example creates an alias on the `en1` network interface. The alias must be defined on the same subnet as the network interface.

```
# ifconfig en1 alias 9.37.207.29 netmask 255.255.255.0 up
```

The following example deletes the alias:

```
# ifconfig en1 delete 9.37.207.29
```

2. Set the schedule so that the virtual users will use the IP aliases during a run; for information, see [Enabling virtual users to use IP aliases on page 569](#).

Results

When you run the schedule, it will give the impression that the network traffic is being generated from multiple hosts.

What to do next

You can insert custom code into your test to retrieve the runtime IP addresses of each virtual user. For information, see [Retrieving the IP address of a virtual user on page 569](#).

Enabling virtual users to use IP aliases

After you have configured aliases at on remote computers, you set the schedule so that the virtual users can use the configured IP aliases.

Before you begin

Before you can enable virtual users to use IP aliases, you must:

1. Configure the aliases at the remote location.
2. Add the remote location to the user group.

To set the schedule so that the virtual users will use the IP aliases during a run:

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.

2. Click the user group whose virtual users will use aliasing.
3. Click **Run this group on the following locations**.

Result

The list of locations shows whether IP aliasing is enabled at that location.

4. To change whether IP aliasing is enabled or disabled, click a row in the table, and then click **Edit**.
5. On the IP aliasing page, click **Enable IP Aliasing**.

Option	Select when
Use IP addresses from all network interfaces	You have one network interface, or you have multiple interfaces and want to use them all.
Only use IP addresses from the following network interfaces	You want to use some, but not all, network interfaces. Select the check box next to the interfaces that you want to use. Click Add to add a new network interface. Typically, you enter an interface name in the form eth0 (Windows™ and Linux™) or en0 (AIX®).

Monitoring resource data

Resource Monitoring is used to capture data, such as processor or memory usage, while running a test schedule. It can provide a comprehensive view of a system under test, to help identify issues. You can monitor data sources from a local or cloud schedule, or from a Service.

To use Resource Monitoring, you must enable Resource Monitoring in the test schedule and add data sources. There are two options to monitor data sources from the schedule editor:

- Monitoring data sources that are available in the local schedule or the cloud schedule, you must enable resource monitoring from the workbench to use the data sources of the local or cloud schedule.
- Monitoring data sources from a Service, you must enable this option to connect the workbench with the server on which the Resource Monitoring Service is installed.

By default, this is HCL OneTest™ Server's URL that is entered in the preferences but it can be changed for another URL if the data sources must be monitored from another server.

Enablement of Resource Monitoring services for a schedule

You can enable Resource Monitoring service in HCL OneTest™ Performance for a performance schedule to monitor the resources in a schedule and capture the performance statistics of such resources during the schedule run.

Prerequisites

You must have completed the following tasks:

- Installed HCL OneTest™ Server and started it. For more details about installation, refer to [Installing the software](#).
- Configured the Resource Monitoring sources in HCL OneTest™ Server for your project. For more information about Resource Monitoring sources, refer to [Resource Monitoring Sources](#).
- In HCL OneTest™ Performance, you must have specified the HCL OneTest™ Server URL in **Window > Preferences > Test > HCL OneTest™ Server** and project in **Window > Preferences > Test > HCL OneTest™ Server** **Resource Monitoring > Project to get sources from** to get the Resource Monitoring sources.
- Generated an offline user token in HCL OneTest™ Server.



Note: In HCL OneTest™ Performance, you must have used the same token in **Window > Preferences > Test > HCL OneTest™ Server** **Test Connection** to establish the connection with HCL OneTest™ Server. For more information about generating an offline token, refer to [Generating an offline token](#).

Overview

When you connect HCL OneTest™ Performance to HCL OneTest™ Server, the Resource Monitoring sources that you added in your HCL OneTest™ Server project are displayed in HCL OneTest™ Performance.



Note: You must have created a project and added Resource Monitoring sources to your project in HCL OneTest™ Server.

You can configure HCL OneTest™ Performance to access HCL OneTest™ Server by using an offline user token.

You must generate the offline user token from HCL OneTest™ Server. You can then use the offline user token on the **Preferences** window in HCL OneTest™ Performance so that a connection is established between HCL OneTest™ Performance and HCL OneTest™ Server.

After the connection is established with HCL OneTest™ Server, you can monitor the following Resource Monitoring sources in HCL OneTest™ Performance:

- Apache httpd server
- NGINX server
- Java Virtual Machine
- Windows Performance Host
- Linux Performance Host
- Docker Host
- OpenMetrics exporter
- Prometheus server

When you enable Resource Monitoring service for a schedule, you can add the Resource Monitoring sources by using any of the following methods:

- [Adding Resource Monitoring sources to a performance schedule on page 571](#)
- [Adding Resource Monitoring sources to a performance schedule by using labels on page 574](#)

After you add the Resource Monitoring sources to the performance schedule that is available in your HCL OneTest™ Performance project, you can run the schedule from HCL OneTest™ Performance.

If you want to run a performance schedule from the HCL OneTest™ Performance command line, you can use the `overridermLabels` command to enable the Resource Monitoring service. See [Running a schedule from a command line on page 627](#).

Related reference

[Service Performance report on page 837](#)

Adding Resource Monitoring sources to a performance schedule

You can add the Resource Monitoring sources to a performance schedule. After you add the Resource Monitoring sources, you can run the schedule from HCL OneTest™ Performance.

Before you begin

You must have configured the Resource Monitoring sources in HCL OneTest™ Server for your project.

1. Find a schedule from the **Test Navigator** pane and double-click it.

A schedule can be a VU schedule or a Rate schedule.

Result

The schedule is displayed in a schedule editor.

2. Perform the following steps in the *Schedule Details* section of the schedule editor:
 - a. Select the **Resource Monitoring from Service** option from the **Category** drop-down list.
 - b. Select the **Enable resource monitoring from service** check box.

Result

The Resource Monitoring options related to the **Resource Monitoring from Service** category are displayed.

3. Select the **Collect from the following sources** option.

You can add the available Resource Monitoring sources in the schedule editor by performing the following steps:

- a. Click **Add/Remove**.

Result

The Resource Monitoring sources are displayed in the **Sources from the Resource Monitoring Service** dialog box.



Note: Only the Resource Monitoring sources that you added on the **Resource Monitoring Service** page in your HCL OneTest™ Server project are available for selection.



Tip: You can also click **Refresh** to fetch the Resource Monitoring sources in the **Sources from the Resource Monitoring Service** dialog box.

- b. Select the Resource Monitoring sources that you want to add to the schedule.
- c. Click **Finish**.

Result

The selected Resource Monitoring sources are added to the Data Source table.

- d. Set the polling time for the Resource Monitoring sources by performing the following steps:

- i. Click the **Polling Time** option that corresponds to the Resource Monitoring source that you added, and then click the ellipsis icon.

The **Set Time** dialog box is displayed.

- ii. Enter a time duration in the **Polling interval** dialog box and set a time unit.

The default polling time is 5 seconds.

- iii. Click **OK**.

The polling time is a time interval at which time the values for the Resource Monitoring source are collected when the schedule runs. For example, if you set the polling time as 5 seconds, data from the Resource Monitoring source is collected at every 5 seconds.

4. Select the **Ignore invalid resources when executing the schedule** check box to suppress any error message about the Resource Monitoring sources and continue running the schedule during the schedule run.



Note: The errors can occur if the Resource Monitoring sources are unreachable or unavailable. After the schedule run completes, you can view the logs and verify the error messages.

5. Save the schedule.

Results

You have enabled Resource Monitoring service for a performance schedule, and then added the Resource Monitoring sources to the performance schedule in HCL OneTest™ Performance.

What to do next

You can add a test to the schedule and run the schedule in HCL OneTest™ Performance. After the run completes, you can view the details of the Resource Monitoring sources in the **Performance Report** as follows:

- The **Resources** page displays the following information:
 - Resource monitoring sources that were monitored during the schedule run.
 - All resource counters for those Resource Monitoring sources that were monitored during the schedule run.
 - Unavailable Resource Monitoring sources that were unreachable or unavailable during the schedule run.
- The **Performance Summary** table under the **Resources graph** lists the following information:
 - The type of Resource Monitoring sources.
 - The most recent values of their corresponding resource counters that were monitored during the schedule run.

Adding Resource Monitoring sources to a performance schedule by using labels

You can add the Resource Monitoring sources to a performance schedule in HCL OneTest™ Performance by using labels that are created in HCL OneTest™ Server. After you add the Resource Monitoring sources, you can run the schedule from HCL OneTest™ Performance.

Before you begin

You must have added labels for the Resource Monitoring sources in your HCL OneTest™ Server project. See [Add Resource Monitoring labels](#).



Important: Note down the labels that you added to the Resource Monitoring sources in your HCL OneTest™ Server project because you must use these labels for adding the Resource Monitoring sources to the performance schedule in HCL OneTest™ Performance.



Note: When you run the same schedule from HCL OneTest™ Server, the same labels are displayed in the **Results** page for the schedule. Open the schedule and go to the **Details** card to view the resource monitoring labels for the schedule that you run.

About this task

After you add the Resource Monitoring sources by using labels to a performance schedule, you can run the schedule from HCL OneTest™ Performance. After the schedule run completes, you can view the labels and the Resource Monitoring sources associated with those labels in **Performance Report > Resources** page. The **Resources** page also displays the Resource Monitoring sources that were unreachable or unavailable during the schedule run.

1. Find a schedule from the **Test Navigator** pane and double-click it.

A schedule can be a VU schedule or a Rate schedule.

Result

The schedule is displayed in a schedule editor.

2. Perform the following steps in the *Schedule Details* section of the schedule editor:
 - a. Select the **Resource Monitoring from Service** option from the **Category** drop-down list.
 - b. Select the **Enable resource monitoring from service** check box.

Result

The Resource Monitoring options related to the **Resource Monitoring from Service** category are displayed.

3. Select the **Collect from sources matching at least one of the following labels** option.

You can add the available Resource Monitoring sources by using labels in the schedule editor by performing the following steps:

- a. Click **Add** to add the labels of the Resource Monitoring source.

Result

The **Add Label** dialog box is displayed.



Note: The labels that you added for the Resource Monitoring sources in your HCL OneTest™ Server project are available for selection. You can also add a label in a performance schedule, and then use the same label for a Resource Monitoring source in your HCL OneTest™ Server project before running the schedule.

- b. Enter or select a label in the **Enter a label** field.

- c. Click **OK**.

Result

The label is added to the Data Source table.



Note: You can add multiple labels to a performance schedule if required.

Optionally, you can edit or delete the labels that you added if required. Select the label from the Data Source table. You can then click **Edit** to edit an existing label. If you want to remove an existing label that you do not want to use in the schedule, select the label, and then click **Remove**.

- d. Set the polling time for the Resource Monitoring sources by performing the following steps:
 - i. Enter a time duration in the **Polling Time** box.
 - ii. Set a time unit.

The default polling time is 5 seconds.

The polling time is a time interval at which time the values for the Resource Monitoring source are collected when the schedule runs. For example, if you set the polling time as 5 seconds, data from the Resource Monitoring source is collected at every 5 seconds.

4. Select the **Ignore invalid resources when executing the schedule** check box to suppress any error message about the Resource Monitoring sources and continue running the schedule during the schedule run.



Note: The errors can occur if the Resource Monitoring sources are unreachable or unavailable. After the schedule run completes, you can view the logs and verify the error messages.

5. Save the schedule.

Results

You have enabled Resource Monitoring service for a performance schedule, and then added the Resource Monitoring sources by using labels to the performance schedule in HCL OneTest™ Performance.

What to do next

You can add a test to the schedule and run the schedule in HCL OneTest™ Performance. After the run completes, you can view the labels and the Resource Monitoring sources in the **Performance Report** as follows:

- The **Resources** page displays the following information:
 - Labels and Resource Monitoring sources associated with those labels that were monitored during the schedule run.
 - All resource counters for those Resource Monitoring sources that were monitored during the schedule run.
 - Unavailable Resource Monitoring sources that were unreachable or unavailable during the schedule run.
- The **Performance Summary** table under the **Resources graph** lists the following information:
 - The type of Resource Monitoring sources.
 - The most recent values of their corresponding resource counters that were monitored during the schedule run.

Enabling Resource Monitoring from the workbench

You can enable Resource Monitoring from the workbench to capture system resource data such as processor or memory usage.

Before you begin

To capture accurate resource monitoring data, you must ensure that the clocks on all computers are synchronized.



Note: If you do not synchronize the clocks on the workbench and on all of the computers involved in a test, resource counters will be displayed inaccurately (with respect to time) in the reports. (There are a number of tools that are available at no cost on the web to help you accomplish synchronization.)

1. Open a schedule in the editor.
2. In the **Performance Schedule Details** area, select the **Resource Monitoring** option in the **Category** drop-down list.
3. Select the **Enable resource monitoring** check box.

Result

This activates the Data Source table.

4. If this is a new schedule, the Data Source table is empty. If the resource monitoring data sources are available in the local schedule and you create the cloud schedule from it, the data sources are automatically added to the cloud schedule. Clicking **Remove** does not delete the data source from the file system; it merely removes it from this view. Other test schedules or applications might still use the data source.

5. If you have existing locations in your workspace, you can click **Add** to add and configure them. If you do not configure the existing location, you are warned in the Data Source table that it is `Not Configured`.
6. Select **Ignore invalid resources when executing the schedule** to suppress error messages about resource monitoring data sources. These errors can occur if the data sources are unreachable or invalid. If you select this option, you must view logs to see the error messages.

Results

You have enabled Resource Monitoring from the workbench.

What to do next

You must specify the data sources. Configuration changes that you make for a particular data source are stored with that location. This configuration storage means that you have to set up a data source only once. If you export a schedule, it contains the data source configuration information. This data might include potentially sensitive information, such as stored passwords.

Adding sources for resource monitoring data

If you enable resource monitoring, you must specify the sources of resource monitoring data. A cloud schedule can contain resource monitoring locations that can be in the local premises, in the cloud location, or in both the places.

1. Open a schedule for editing.
2. At the bottom of the **Resource Monitoring** page, click **Add**. You might need to scroll down in the view.

Result

The **Create and manage configurations** wizard opens.

3. Do one of these steps:

Choose from:

- To create a resource monitoring location, click **Create new resource monitoring location**.
 - To add an existing location or to create a resource monitoring location that is based on an existing location, click **Create or add a resource monitoring location from an existing source**.
4. Type an IP address or the fully qualified host name. This address or host name is for the node to monitor, not the Tivoli Enterprise™ Monitoring Server. The IP address can be of a computer that is in the cloud.

Result

You can choose the types of data sources.

5. Select from these data sources:

Choose from:

- **Apache HTTP Server Monitoring**
- **Apache Tomcat Application Server Monitoring**
- **IBM DB2 Monitoring**
- **IBM Tivoli Monitoring**
- **IBM WebSphere PMI Monitoring**
- **JBoss Application Server Monitoring**
- **JVM Monitoring**
- **Oracle Database Monitoring**

- **Oracle WebLogic Server Monitoring**
- **SAP NetWeaver Web Application Server Monitoring**
- **SNMP Monitoring**
- **UNIX rstatd monitor**
- **Windows Performance Monitor**

You can select multiple types of data sources, but you must configure each one separately. You cannot collect Windows™ Performance Monitoring data from a computer that is running the Linux™ operating system. Also, you cannot collect Windows™ Performance Monitoring data if your workbench is running the Linux™ operating system. In other words, to collect Windows™ Performance Monitoring data, the workbench and the node to monitor must both be running the Windows™ operating system.

What to do next

Configure the data sources.

Adding Apache HTTP Server sources

To capture resource monitoring data from Apache HTTP Server, you must configure the data source.

Before you begin

Enable Status Support on the Apache HTTP Server by modifying the httpd.conf file. Refer http://httpd.apache.org/docs/2.2/mod/mod_status.html for details.

Then, restart Apache HTTP Server.

1. On the **Location** page, specify the connection and authentication information for the server that runs Apache HTTP Server. If you have to change the port that is used to communicate with the server, change the information in **Connection**. Typically, your Apache HTTP Server system administrator specifies this information. The port number must match the port that you specified when configuring Java™ Management Extensions on the server.
2. If administrative security is enabled on the server, select **Administrative security enabled**.
 - a. Type the user ID and password for Apache HTTP Server in **Authentication**.
 - b. Select **Save Password** to save your password locally. If you do not save your password, you might be prompted for it (depending on the host system configuration) when you edit the configured location or when you run test schedules that use the location.
3. On the **Resource** page, select the type of data to capture. The tree view shows the counter groups and counters that are available from the application server. Be selective; monitoring all possible resource data requires substantial amounts of memory. Hold your mouse pointer over a counter to see details about what the counter measures.
4. Configure the options on the **Options** page.

- a. Click **Reset counters at start of run (client side reset only)** to set the counters to 0 at the start of a schedule run. This does not change the actual values of the counters on the application server. Instead, the recorded values of the counters are corrected to start at 0.
 - b. In **Polling Interval**, specify in seconds the interval for collecting resource data. For example, if you accept the default setting of 5 seconds, counter information is collected at 5-second intervals from the specified host during the schedule run.
 - c. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.
5. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
 6. Click **Next**.
 7. Specify a location and name and click **Finish**.

Adding Apache Tomcat sources

To capture resource monitoring data from Apache Tomcat, you must configure the data source.

Before you begin

The Java™ Management Extensions must be configured on the server that is running Apache Tomcat. After installing Apache Tomcat on Microsoft™ Windows™, run the **Configure Tomcat** shortcut. Click the **Java** tab, and then edit the **Java Options**. Following are sample Java™ options:

```
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=8880
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
```

On Linux™, set the environment variable `CATALINA_OPTS` in either `TOMCAT_HOME/bin/setenv.sh` or `TOMCAT_HOME/bin/catalina.sh` file. For example,

```
export CATALINA_OPTS="-Dcom.sun.management.jmxremote=true
-Dcom.sun.management.jmxremote.port=8686
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
-Djava.rmi.server.hostname=your.server.domain.com"
```

Then, restart Apache Tomcat.

1. On the **Location** page, specify the connection and authentication information for the server that runs Apache Tomcat. If you need to change the port that is used to communicate with the server, change the information in **Connection**. Typically, your Tomcat system administrator specifies this information. The port number must match the port that you specified when configuring Java™ Management Extensions on the server.
2. If administrative security is enabled on the server, select **Administrative security enabled**.

- a. Type the user ID and password for Apache Tomcat in **Authentication**.
 - b. Select **Save Password** to save your password locally. If you do not save your password, you might be prompted for it (depending on the host system configuration) when you edit the configured location or when you run test schedules that use the location.
3. On the **Resource** page, select the type of data to capture. The tree view shows the counter groups and counters that are available from the application server. Be selective; monitoring all possible resource data requires substantial amounts of memory. Hold your mouse pointer over a counter to see details about what that counter measures.
 4. Configure the options on the **Options** page.
 - a. In **Polling Interval** specify in seconds the interval for collecting resource data. For example, if you accept the default of 5 seconds, counter information is collected at 5-second intervals from the specified host during the schedule run.
 - b. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.
 5. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
 6. Click **Next**.
 7. Specify a location and name and click **Finish**.

Adding IBM® DB2® sources

To capture resource monitoring data from IBM® DB2®, you must configure the data source.

Before you begin

Your DB2® administrator must turn on the snapshot monitor switches for the resource monitoring data to collect. If the monitor switches are off, none of the resource counter groups will contain counters. For more information on the DB2® snapshot monitor, see these topics in the DB2® documentation: [System monitor switches](#) and [Snapshot monitor](#).

To configure the DB2® data source:

1. On the **Location** page, specify the connection and authentication information for the instance of DB2® to use to capture resource monitoring data. If you need to change the port that is used to communicate with the computer that is running DB2®, change the information in **Connection**. Typically, your DB2® system administrator specifies this information.
2. Under **Database Settings**, type a name in **Database Name** and partition number in **Partition number**.
If the instance of DB2® to monitor uses the Database Partitioning Feature (DPF), then change the partition number to the partition to monitor. Otherwise, leave the partition number set to the default, 0.
3. Type the user ID and password for DB2® in **Authentication**.
To collect resource monitoring data from DB2®, the specified user account must have SYSADM, SYSCTRL, SYSMANT, or SYSMON authority.

- a. Select **Save Password** to save your password locally. If you do not save your password, you might be prompted for it (depending on the host system configuration) when you edit the configured location or when you run test schedules that use the location.
4. **Optional:** On the **Monitoring Status** page, click **Check Monitoring Status** to determine the state of the monitor switches on the DB2® server.

The states of the following monitor switches are checked:

DFT_MON_BUFPOOL

Buffer pool switch

DFT_MON_LOCK

Lock switch

DFT_MON_SORT

Sort switch

DFT_MON_STMT

Statement switch

DFT_MON_TABLE

Table switch

DFT_MON_TIMESTAMP

Timestamp switch

DFT_MON_UOW

Unit of work switch

5. On the **Resource** page, select the type of data to capture. The tree view shows the database server and its counter groups and counters. Be selective; monitoring all possible resource data requires substantial amounts of memory. Clear the **Show only selected counters** check box to see all available counters. Hold your mouse pointer over a counter to see details about what that counter measures.
6. Configure the option on the **Options** page.
 - a. In **Polling Interval** specify in seconds the interval, for collecting resource data. For example, if you accept the default of 5 seconds, counter information is collected at 5-second intervals from the specified host during the schedule run.
 - b. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.
7. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
8. Click **Next**.
9. Specify a location and name and click **Finish**.

Adding IBM® Tivoli® Monitoring sources

To capture resource monitoring data from an IBM® Tivoli Enterprise™ Monitoring Server, you must configure the data source.

To configure the IBM® Tivoli® Monitoring source:

1. On the Tivoli Enterprise™ Monitoring Server page, specify the monitoring server that you want to use to capture resource monitoring data.
 - a. Type the IP address or the fully qualified host name of the monitoring server in the **Host** field on the Tivoli Enterprise™ Monitoring Server page. This is different from the **Host** field at the top of the **Create and manage configurations** wizard.
 - b. Type the user ID and password for the monitoring server in **Authentication**.
 - c. Change the **Connection** information if needed. Typically, your Tivoli® system administrator will specify this information.
 - d. Select **Save Password** to save your password locally. If you do not save your password, you might be prompted for it (depending on the host system configuration) when editing the configured location or when running test schedules that use the location.

Result

After you have specified the monitoring server, you can choose resources to capture. If the host is not managed by the monitoring server, you will see an error message.

2. On the Resource page, select the type of data that you want to capture. The tree view shows the host and all of its available IBM® Tivoli® Monitoring agents, and their respective counter groups and counters. Be selective; monitoring all possible resource data requires substantial amounts of memory. Clear the **Show only selected counters** check box to see all available counters. Hold your mouse pointer over a counter to see details about what that counter measures.
3. Configure time intervals on the **Options** page.
 - a. Type the **Polling Interval** in seconds, for collecting resource data. For example, if you accept the default of 5 seconds, counter information will be collected at 5-second intervals from the specified host during the schedule run.
 - b. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.
4. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
5. Click **Next**.
6. Specify a location and name and click **Finish**.

Adding IBM® WebSphere® Performance Monitoring Infrastructure sources

To capture resource monitoring data from the IBM® WebSphere® Performance Monitoring Infrastructure, you must configure the data source.

To configure the IBM® WebSphere® Performance Monitoring Infrastructure data source:

1. On the Location page, specify the connection and authentication information for the instance of WebSphere® Application Server to use to capture resource monitoring data. If you need to change the SOAP port used to communicate with the computer that is running WebSphere® Application Server, change the **Connection** information. Typically, your WebSphere® system administrator will specify this information.
2. If administrative security is enabled on the computer that is running WebSphere® Application Server, select **Administrative security enabled**.
 - a. Type the user ID and password for WebSphere® Application Server in **Authentication**.
 - b. Select **Save Password** to save your password locally. If you do not save your password, you might be prompted for it (depending on the host system configuration) when you edit the configured location or when you run test schedules that use the location.
3. On the Resource page, select the type of data to capture. The tree view shows the application server and its counter groups and counters. Be selective; monitoring all possible resource data requires substantial amounts of memory. Clear the **Show only selected counters** check box to see all available counters. Hold your mouse pointer over a counter to see details about what that counter measures.
4. Configure the options on the Options page.
 - a. Click **Reset counters at start of run (client side reset only)** to set the counters to 0 at the start of a schedule run.
This does not change the actual values of the counters on the application server. Instead, the recorded values of the counters are corrected to start at 0. For example, if you monitor the ServletRequestCount counter, it starts at 0 instead of its previous value.
 - b. Type the **Polling Interval** in seconds, for collecting resource data. For example, if you accept the default of 5 seconds, counter information will be collected at 5-second intervals from the specified host during the schedule run.
 - c. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.
5. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
6. Click **Next**.
7. Specify a location and name and click **Finish**.

Adding Java™ Virtual Machine sources

To capture resource monitoring data from a Java™ Virtual Machine (JVM), you must configure the data source.

Before you begin

The Java™ Management Extensions must be configured on the server that is running the JVM. Following are sample arguments to pass to the JVM:

```
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=8880
```

```
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
```

1. On the **Location** page, specify the connection and authentication information for the computer that is running the JVM. If you need to change the port that is used to communicate with the server, change the information in **Connection**. The port number must match the port that you specified when configuring Java™ Management Extensions on the server.
2. If administrative security is enabled on the JVM computer, select **Administrative security enabled**.
 - a. Type the user ID and password in **Authentication**.
 - b. Select **Save Password** to save your password locally. If you do not save your password, you might be prompted for it (depending on the host system configuration) when you edit the configured location or when you run test schedules that use the location.
3. On the **Resource** page, select the type of data to capture. The tree view shows the counter groups and counters that are available from the computer running the JVM. Be selective; monitoring all possible resource data requires substantial amounts of memory. Hold your mouse pointer over a counter to see details about what that counter measures.
4. Configure the option on the **Options** page.
 - a. In **Polling Interval** specify in seconds the interval for collecting resource data. For example, if you accept the default of 5 seconds, counter information is collected at 5-second intervals from the specified host during the schedule run.
 - b. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.
5. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
6. Click **Next**.
7. Specify a location and name and click **Finish**.

Adding JBoss Application Server sources

To capture resource monitoring data from JBoss Application Server, you must configure the data source.

Before you begin

Before you can capture resource monitoring data from JBoss Application Server, you must specify the path to the JBoss client file, `jbossall-client.jar` on the **JBoss Client Preferences** page. The JBoss client file is provided with JBoss, not with HCL OneTest™ Performance. The version of the JBoss client file must match the version of JBoss Application Server running on the server under test.



Note: HCL OneTest™ Performance supports JBoss Application Server version 5.x.

To configure the JBoss Application Server data source:

1. On the Location page, specify the connection and authentication information for the instance of JBoss Application Server to use to capture resource monitoring data. If you need to change the SOAP port used to communicate with the computer that is running JBoss Application Server, change the **Connection** information. Typically, your JBoss system administrator specifies this information.
2. If administrative security is enabled on the computer that is running JBoss Application Server, select **Administrative security enabled**.
 - a. Type the user ID and password for JBoss Application Server in **Authentication**.
 - b. Select **Save Password** to save your password locally. If you do not save your password, you might be prompted for it (depending on the host system configuration) when you edit the configured location or when you run test schedules that use the location.
3. On the Resource page, select the type of data to capture. The tree view shows the application server and its counter groups and counters. Be selective; monitoring all possible resource data requires substantial amounts of memory. Clear the **Show only selected counters** check box to see all available counters. Hold your mouse pointer over a counter to see details about what that counter measures.
4. Configure the option on the **Options** page.
 - a. Type the **Polling Interval** in seconds, for collecting resource data. For example, if you accept the default of 5 seconds, counter information will be collected at 5-second intervals from the specified host during the schedule run.
 - b. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.
5. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
6. Click **Next**.
7. Specify a location and name and click **Finish**.

Adding Microsoft™ Windows™ Performance Monitor sources

To capture resource monitoring data from Microsoft™ Windows™ Performance Monitor, you must configure the data source.

Before you begin

The Windows Performance Monitor option is enabled only for Microsoft Windows computers. If you are monitoring a non-Windows computer, this option is disabled.

To configure the Windows™ Performance Monitor data source:

1. On the Location page, type the user ID, password, and domain. The domain is optional, required only if you need to perform cross-domain authentication.



Note: The user ID must correspond to a user that is a member of the Performance Monitor Users group or the Administrators group on the computer from which you want to collect resource monitoring data. If the user is not in the Performance Monitor Users group or the Administrators



group on the computer that you want to monitor, no Windows™ Performance Monitor data is collected.

2. Select **Save Password** to save your password locally. If you do not save your password, you might be prompted for it (depending on the host system configuration) when editing the configured location or when running test schedules that use the location.
3. On the Resource page, select the type of data that you want to capture. The tree view shows the host and all of its respective counter groups and counters. To see all the available counters, clear the **Show only selected counters** check box.
Be selective; monitoring all possible resource data requires substantial amounts of memory. Hold your mouse pointer over a counter to see details about what that counter measures.
4. Configure time interval on the **Options** page.
 - a. Type the **Polling Interval** in seconds, for collecting resource data. For example, if you accept the default of 5 seconds, counter information will be collected at 5-second intervals from the specified host during the schedule run.
 - b. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.
5. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
6. Click **Next**.
7. Specify a location and name and click **Finish**.

What to do next



Note: The host that you want to monitor must be accessible through the Windows™ network. Resource monitoring data is collected using the net use command to establish a connection to remote computers. File and printer sharing must be enabled on the computer running the Windows™ operating system. Simple File Sharing, where remote connections are processed as access by the Guest user, must be disabled. In addition, system policies must not be set so that remote connections are processed as access by the Guest user. For example, if the `Network security: Sharing and security model for local accounts` policy is set to `Guest only - local user authenticate as Guest` and the `Accounts: Guest account status` policy is set to `Enabled`, then remote connections are processed as access by the Guest user. In that case, resource monitoring data is not collected.

Typically, if you are able to connect to a shared hard disk drive on the remote host from the workbench, then you will also be able to collect resource monitoring data from Windows™ Performance Monitor on the remote host. If file and printer sharing is not enabled on the remote host and you attempt to set up resource monitoring in a schedule, the following message is displayed:

```
IWAY0241E The host name IP_address is either not a known host or is not a Windows host.
```

To enable Windows™ file and printer sharing:



1. Open the Network Connections Control Panel.
2. Right-click the **Local Area Connection** (or the currently active network adapter).
3. In the menu, click **Properties**.
4. Select **File and Printer Sharing for Microsoft Networks** in **This connection uses the following items**.
5. Click **OK**.

Adding Oracle Database sources

To capture resource monitoring data from Oracle Database, you must configure the data source.

Before you begin

Before you can capture resource monitoring data from Oracle Database, you must specify the path to the Oracle Database client file, `ojdbc6.jar` on the **Oracle Database Client Preferences** page. The Oracle Database client file is provided with Oracle Database, not with HCL OneTest™ Performance. The version of the Oracle Database client file must match the version of Oracle Database that is running on the server under test.

1. On the **Location** page, specify the connection and authentication information for the server that runs Oracle Database. If you have to change the port that is used to communicate with the server, change the information in **Connection**. Typically, your Oracle Database system administrator specifies this information.
2. In **Database Name**, type the name of the database to monitor.
3. On the **Resource** page, select the type of data to capture. The tree view shows the counter groups and counters that are available from the application server. Be selective; monitoring all possible resource data requires substantial amounts of memory. Hold your mouse pointer over a counter to see details about what that counter measures.
4. Configure the options on the **Options** page.
 - a. In **Polling Interval**, specify in seconds the interval for collecting resource data. For example, if you accept the default setting of 5 seconds, counter information is collected at 5-second intervals from the specified host during the schedule run.
 - b. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.
 - c. Clear the **Keep cursors open during runtime** check box to allow database cursors to close while tests run. By default, **Keep cursors open during runtime** is selected, which prevents database cursors from being closed while tests run.
5. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
6. Click **Next**.
7. Specify a location and name and click **Finish**.

Adding Oracle WebLogic Server sources

To capture resource monitoring data from Oracle WebLogic Server, you must configure the data source.

Before you begin

To capture resource monitoring data from Oracle WebLogic Server, you must specify the path to the client files:

- wljmxclient.jar
- wljmsclient.jar
- wlclient.jar

These Oracle WebLogic client files are provided with Oracle WebLogic, not with HCL OneTest™ Performance, so you must copy the client files from the server to a folder on the workbench computer. In HCL OneTest™ Performance, click **Window > Preferences > Test > Performance Resource Monitoring > Oracle WebLogic Client Preferences** and specify the path to the folder containing the client files. The versions of the client files must match the version of Oracle WebLogic Server running on the server under test. By default, for Oracle WebLogic Server 10g Release 3 (10.3), the client files are installed in the `c:\bea\wlserver_10.3\server\lib\` folder.

About this task

The resource monitoring for Oracle WebLogic server works with the Internet Inter-ORB Protocol (IIOP). If you use JMX protocol in Oracle WebLogic Server, you can use the JVM monitoring option.

1. In the Schedule editor, select the **Resource Monitoring** category, and click **Add**.
2. On the **Location** page, for a new resource monitoring location, specify the connection and authentication information for the computer that runs the Administration Server instance of Oracle WebLogic Server for the domain. If you need to change the port that is used to communicate with the Administration Server for the domain, change the information in **Connection**. Typically, your WebLogic system administrator specifies this information.
An Oracle WebLogic domain consists of one or more Oracle WebLogic Server instances. One server is the Administration Server. If you want to collect resource monitoring from an Oracle WebLogic domain that consisting of multiple servers, the specified host must be the Administration Server. The Administration Server is also known as the domain controller.
3. If administrative security is enabled on the Administration Server, select **Administrative security enabled**.
 - a. Type the user ID and password for Oracle WebLogic Server in **Authentication**.
 - b. Select **Save Password** to save your password locally. If you do not save your password, you might be prompted for it (depending on the host system configuration) when you edit the configured location or when you run test schedules that use the location.
4. On the **Resource** page, select the type of data to capture. The tree view shows the application servers that the Administration Server manages and their counter groups and counters. Be selective; monitoring all possible resource data requires substantial amounts of memory. Clear the **Show only selected counters** check box to see all available counters. Hold your mouse pointer over a counter to see details about what that counter measures.
5. Configure the options on the **Options** page.
 - a. In **Polling Interval** specify in seconds the interval for collecting resource data. For example, if you accept the default of 5 seconds, counter information is collected at 5-second intervals from the specified host during the schedule run.
 - b. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.

6. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
7. Click **Next**.
8. Specify a location and name and click **Finish**.

Adding SAP NetWeaver sources

To capture resource monitoring data from SAP NetWeaver, you must configure the data source.

Before you begin

Before you can capture resource monitoring data from SAP NetWeaver, you must specify the path to the client files:

- `com_sap_pj_jmx.jar`
- `exception.jar`
- `logging.jar`
- `sapj2eeclient.jar`

Specify the path to the client files on the **SAP NetWeaver Web Application Server Client Preferences** page. The SAP NetWeaver client files are provided with SAP NetWeaver, not with HCL OneTest™ Performance. The versions of the client files must match the version of SAP NetWeaver running on the server under test. Copy the client files to the workbench computer, and then specify the path to the files.

The resource monitoring feature was tested on SAP NetWeaver 7.0 (2004s). Resource monitoring from other versions of SAP NetWeaver might not be supported.

1. On the **Location** page, specify the connection and authentication information for the instance of SAP NetWeaver to use to capture resource monitoring data. If you need to change the port for communicating with the computer that is running SAP NetWeaver, change the **Connection** information. Typically, your SAP NetWeaver system administrator specifies this information. Use the P4 port to communicate with the computer that is running SAP NetWeaver. The default value of the P4 port is 50004. For more information about port numbers, see the SAP NetWeaver documentation.
2. Ensure that **Administrative security enabled** is selected.
 - a. Type the user ID and password for SAP NetWeaver in **Authentication**.
To capture resource monitoring data from SAP NetWeaver, you must use an account with administrator rights.
 - b. Select **Save Password** to save your password locally. If you do not save your password, you might be prompted for it (depending on the host system configuration) when you edit the configured location or when you run test schedules that use the location.
3. On the **Resource** page, select the type of data to capture. The tree view shows the application server and its counter groups and counters. Be selective; monitoring all possible resource data requires substantial amounts of memory. Clear the **Show only selected counters** check box to see all available counters. Hold your mouse pointer over a counter to see details about what that counter measures.

4. Configure the options on the **Options** page. Type the **Polling Interval** in seconds, for collecting resource data. For example, if you accept the default setting of 5 seconds, counter information will be collected at 5-second intervals from the specified host during the schedule run.
5. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
6. Click **Next**.
7. Specify a location and name and click **Finish**.

Adding Simple Network Management Protocol sources

To capture resource monitoring data from a Simple Network Management Protocol (SNMP) agent, you must configure the data source.

1. On the **Location** page, specify the connection and authentication information for the SNMP agent.
2. In **MIB path**, type or browse to the management information base (MIB) file. Typically, MIB files are supplied by manufacturers of devices that support SNMP.
3. On the **Resource** page, select the type of data to capture. The tree view shows the counter groups and counters that are available from the application server. Be selective; monitoring all possible resource data requires substantial amounts of memory. Hold your mouse pointer over a counter to see details about what that counter measures.
4. Configure the options on the **Options** page.
 - a. Click **Reset counters at start of run (client side reset only)** to set the counters to 0 at the start of a schedule run.
This setting does not change the actual values of the counters on the device that runs the SNMP agent. Instead, the recorded values of the counters are corrected to start at 0.
 - b. In **Polling Interval**, specify in seconds the interval for collecting resource data. For example, if you accept the default setting of 5 seconds, counter information is collected at 5-second intervals from the specified host during the schedule run.
 - c. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.
5. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
6. Click **Next**.
7. Specify a location and name and click **Finish**.

Adding UNIX™ rstatd sources

To capture resource monitoring data from the UNIX™ rstatd daemon, you must configure the data source. UNIX™ rstatd is bundled with most Linux™ distributions.

Before you begin

To collect resource monitoring data from UNIX™ rstatd, the portmapper service must be enabled on the host computer. To determine whether the portmapper service is enabled, type the following on the command line of the host computer:

```
rpcinfo -p localhost
```

The portmapper service is listed in the output of the rpcinfo command. If it is not, contact your system administrator for help with installing or configuring the portmapper service.

1. On the **Resource Monitoring** tab of the schedule editor, click **Enable resource monitoring**, and then click the **Add** button.
2. Click **Next** and type the hostname of the computer that you want to monitor.
3. In Data Sources, click **UNIX rstatd monitor**, and on the **Locations** tab, specify the connection parameters:
 - a. Click **UDP** or **TCP** protocol.
 - b. To use a different port number, click **Use port** and type the port number.
4. On the **Resource** tab, select the type of data that you want to capture. The tree view shows all available performance counters, with a default set of counters preselected. To see all available counters, clear the **Show only selected counters** check box.
Be selective; monitoring all possible resource data requires substantial amounts of memory. Hold your mouse pointer over a counter to see details about what that counter measures.
5. Configure time interval on the **Options** page.
 - a. In **Polling Interval**, type a time value in seconds for collecting resource data. For example, if you accept the default of 5 seconds, counter information will be collected at 5-second intervals from the specified host during the schedule run.
 - b. In **Timeout Interval**, type a time value in seconds. If the resource monitoring host does not respond within this amount of time during a schedule run, an error is logged.
6. To collect the resource monitoring data for the system under test that is in the cloud, in the **Cloud Options** tab, click the **Location in Public Cloud** check box.
7. Click **Next**.
8. Specify a location and name and click **Finish**.

Resource Monitoring Service

When you apply load to a system under test, the system's resources are consumed increasingly. If the capacity of the resources does not match the load, you will notice performance degradation in the results. With the Resource Monitoring Service, you can continually observe the health of the system's resources.

To monitor a remote system under test, you can install an agent on that system. For testing, you would need many machines. For example, you might have one machine with the application server, another machine with the database server, and some machines to apply the user load. Due to network or firewall issues, sometimes, it becomes difficult for multiple machines to connect to each other. Resource Monitoring Agents are installed on the target machines so that they can establish a connection with HCL OneTest™ Server to gather resource statistics of the target host.

The agent always initiates the connection with the Resource Monitoring Service. Also, to monitor the resources, you must first add them to the Resource Monitoring Service web UI. For example, if you want to view the throughput, requests rate, and CPU usage of the Apache server, you must add it as a data source. For more information about Resource Monitoring Agents and Services, see [HCL OneTest™ Server documentation](#).

Starting Resource Monitoring Service

You must access the Resource Monitoring web UI to perform the resource monitoring operations such as adding the data sources, counters, and viewing the live performance statistics.

Before you begin

You must have installed HCL OneTest™ Server. For more information about installing the software, see [HCL OneTest™ Server documentation](#).

1. Go to the HCL OneTest™ Server URL.

Result

For example, `https://server.ip.nip.io`.

2. Enter your user name and password, and then click **Login**.

Monitoring response time breakdown

You can use response time breakdown to see statistics on any page element that is captured while running a test schedule or imported from historical data.

Response time breakdown is a type of application monitoring that shows how much time was spent in each part of the system under test as the system was exercised. The response time breakdown view is associated with a page element (URL) from a particular execution of a test or schedule. This view shows the "insides" of the system under test, because the data collection mechanisms are on the systems under test, not the load drivers. Response time breakdown provides information down to the Java™ EE method level for applications that are running on IBM® WebSphere® Application Server or BEA WebLogic Server, as well as calls from Java™ EE methods to plain old Java™ objects (POJOs).

You can collect response time breakdown data from HTTP and SOA tests. Response time breakdown does not apply to other protocols, such as SAP.

Typically, you capture response time breakdown in real time in development, or test, environments, rather than production environments. To capture response time breakdown data, you must enable it in a test or schedule and configure the amount of data to be captured.

The data collection infrastructure collects response time breakdown data. Each application server on which the application runs and from which you want to collect data must have the data collection infrastructure installed and running. In addition, each application server must be configured, or *instrumented*, to use the data collection infrastructure. See the installation guide to learn more about installing the data collection infrastructure.

HCL OneTest™ Performance provides limited ability to collect response time breakdown data. You can collect response time breakdown data for up to four processor cores on one application tier. For enhanced response

time breakdown collection, contact your account representative to learn more about IBM® Rational® Application Performance Analyzer.



Note: Rational® Application Performance Analyzer is a version of HCL OneTest™ Performance that is licensed for extended response time breakdown data collection. To install, configure, or use Rational® Application Performance Analyzer, follow the documentation for HCL OneTest™ Performance.

Enabling response time breakdown collection

You can enable response time breakdown collection to see how much time is spent in each part of the application as it runs.

Before you begin

To collect response time breakdown, the data collection infrastructure must be installed, configured, and running on all computers that are used in the distributed application under test. To learn how to install and configure the data collection infrastructure, see the Installation guide [Installation of HCL OneTest Performance on page 87](#) and [Configuring the data collection infrastructure on page 107](#). If you enable response time breakdown collection for a test and the remote computers are not running the data collection infrastructure, the following error is displayed:

```
IWAY0159E The data collection infrastructure does not appear to be running on hostname. Please ensure that it is running and try again.
```

1. Open a schedule in the editor.
2. In the **Schedule Element Details** area, click the **Response Time Breakdown** tab.
3. Select **Enable collection of response time data**.

Result

This activates the test list, the location list, and **Options**.

4. Click **Add**.

Choose from:

- To add a new response time breakdown location, select **Add New**.
- To add an existing response time breakdown location, select **Add Existing**.

Add locations for every application server from which to collect response time breakdown information.

Results

You have enabled response time breakdown data collection.

What to do next

Set logging detail levels.

Enabling response time breakdown collection on Windows™ Vista, Windows™ 7, and Windows™ Server 2008

Microsoft™ Windows™ Vista, Windows™ 7, and Windows™ Server 2008 include security features that are not in previous versions of Windows™. You must adjust the security settings of these operating systems to collect response

time breakdown data. The default security settings do not allow response time breakdown collection. By default, the firewall blocks inbound connection attempts. You must create an inbound connection rule to allow the HCL OneTest™ Performance Agent to connect to the computer.

Before you begin

To collect response time breakdown, the data collection infrastructure must be installed, configured, and running on all computers that are used in the distributed application under test. To learn how to install the data collection infrastructure, see the Installation of Rational Service Tester for SOA Quality [Installation guide on page 87](#). If you enable response time breakdown collection for a test and the remote computers are not running the data collection infrastructure, the following error is displayed: `IWAY0159E The data collection infrastructure does not appear to be running on hostname. Please ensure that it is running and try again.`

1. Open the Windows™ Administrative Tools Control Panel, and click **Local Security Policy**.

Result

The **Local Security Policy** window opens.

2. Expand **Windows Firewall with Advanced Security**.
3. Expand **Windows Firewall with Advanced Security - Local Group Policy Object**.
4. Select **Inbound Rules**. Right-click the pane on the right, and select **New Rule**; then complete these steps:



- a. On the **Rule Type** page, select **Port**; then click **Next**.
 - b. On the **Protocol and Port** page, select **TCP**, and enter these ports: `10002, 10003, 10004, 10005, 10006`; then click **Next**.
 - c. On the **Action** page, ensure that **Allow the Connection** is selected; then click **Next**.
 - d. On the **Profile** page, select all profiles; then click **Next**.
 - e. On the **Name** page, type a name, for example, `Remote Agent Controller`, and click **Finish**.
5. Right-click the new rule, and select **Enable Rule**.

Setting logging levels

To limit the amount of response time breakdown data collected, adjust logging levels.

Before you begin

To set logging levels, you need to have a schedule where you have enabled response time breakdown data collection.

1. Open a schedule in the editor.
2. In the Schedule Element Details area, click the **Response Time Breakdown** tab.

3. Choose a **Detail level** of **Low**, **Medium**, or **High**.
4. If you set the detail level to **High** or **Medium**, also click **Only sample information from a subset of users** to prevent the log from getting too large.
 - a. Click **Fixed number of users**, and type a number to specify that the given number of users from each group is sampled. Unless you have specific reasons to collect data from multiple users, select **Fixed number of users**, and specify one user per user group.
 - b. You can also click **Percentage of users**, and specify a percentage. That percentage is sampled from each user group, but at least one user is sampled from each user group.

What to do next

Now you can run schedules and capture response time breakdown data.

Related information

[Filtering POJO packages, methods, and classes on page 596](#)

Enabling response time breakdown collection for specific page elements

You can enable response time breakdown for specific pages or page elements to see how much time is spent in each part of the application as the schedule runs. Enabling response time breakdown collection for only certain page elements can be useful if you want to minimize the amount of collected data. Enabling response time breakdown collection for a whole test or schedule can require substantial amounts of memory.

Before you begin

To collect response time breakdown, the data collection infrastructure must be installed, configured, and running on all computers that are used in the distributed application under test. To learn how to install and configure the data collection infrastructure, see the [installation guide on page 107](#) and [Configuring the data collection infrastructure on page 107](#). If you enable response time breakdown collection for a test and the application servers are not running the data collection infrastructure, the following error is displayed: `IWAY0159E The data collection infrastructure does not appear to be running on hostname. Please ensure that it is running and try again.`

1. Open a test in the editor.
2. Expand the list under **Test Contents** to display the pages or page elements of interest.
3. Select the pages or page elements to collect response time breakdown data for. Under **Test Element Details**, select **Enable response time breakdown**. You might need to scroll down in the test editor view to display the **Enable response time breakdown** check box.

The **Enable response time breakdown** check box is displayed only for elements that support response time breakdown data collection. The specific elements that support response time breakdown vary depending on the protocol (HTTP, SAP, Citrix, and so on).



Note: Restricting the scope of the response time breakdown collection improves performance and memory utilization. Enable response time breakdown at as fine-grained a level as possible. Enabling



response time breakdown at too broad a scope can greatly increase the time spent in areas with a large number of Java™ EE interactions.

Result

You are prompted to select an existing response time breakdown location or to add a new response time breakdown location.

- To add other response time breakdown locations, click **Add** on the **Advanced** page.

Choose from:

- To add a new response time breakdown location, select **Add New**.
- To add an existing response time breakdown location, select **Add Existing**.

Add locations for every server from which to collect response time breakdown information.

Results

You have enabled response time breakdown data collection for the specified page elements.

Filtering POJO packages, methods, and classes

You can filter selected plain old Java™ object (POJO) packages, methods, and classes from response time breakdown collection.

Before you begin

Create a test or schedule that is enabled for response time breakdown collection.

- Open the location that you used for response time breakdown collection.
- Click the **General Properties** tab.
- Click **Add** to create a new property.
- In the **New Property** window, type `RTB_POJO_EXCLUDE` for the **Property Name**.
- In **Property Value**, type a semicolon-separated list of the POJO packages, methods, and classes to filter. Use an asterisk (*) for a wildcard character.

Result

When you run the test again, the POJO methods and classes that you selected are filtered from response time breakdown collection.

Example

Assume that the `RTB_POJO_EXCLUDE` property has this property

value:`com.ibm._js*;com.ibm.websphere.samples.plantsbywebspherewar.*;com.ibm.websphere.samples.plantsbywebsphereejb.Util.debug.`

All packages with names that start with `com.ibm._js`, the

`com.ibm.websphere.samples.plantsbywebspherewar` package, and the

`com.ibm.websphere.samples.plantsbywebsphereejb.Util.debug` method are excluded from response time breakdown results.

Setting log and statistic levels

Within a schedule, you set the size and sampling rate of the test log and the problem determination log, as well as the statistics that are displayed during a run.

Setting the statistics displayed during a run

You can set the type of data that you see during a run, the sampling rate for that data, and whether data is collected from all users or a representative sample.

To set the level of statistics logging:

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.

2. In the Schedule Contents area, click the name of the schedule.
3. On the Statistics page, set **Statistics log level** to one of the following options:

Option	Description
None	No statistics are displayed during the run, and any report that depends on statistics is not generated. At the end of the run, you see only a Summary report that contains three items: the time the run took, whether the results were on the local computer (or, if a remote location, which one), and the status of the run, which is <code>complete</code> .
Schedule Actions	Select this option if you are interested only in the number of users. Schedule actions report the number of active and completed users in the run.
Primary Test Actions	Select this option to limit the processing required by the workbench. Primary test actions include all schedule actions plus: <ul style="list-style-type: none"> ◦ For HTTP tests, HTTP page-related actions (attempts, hits, and verification points). ◦ For SAP tests, SAP screens. ◦ For Citrix tests, all keyboard and mouse actions. ◦ For socket tests, this option does not apply.
Secondary Test Actions	Select this option to limit the processing required by the workbench. Secondary test actions include all primary test actions plus HTTP page element-related actions.

Option	Description
	<ul style="list-style-type: none"> ◦ For HTTP tests, HTTP page element–related actions. ◦ For SAP tests, SAP screen element–related actions. ◦ For Citrix tests tests, statistics are identical to Primary Test Actions. ◦ For socket Send and Receive actions, the exchanged data is also available in the test log, by means of attachments.
All	<p>For HTTP, SAP, and Citrix tests, provides statistics for all actions.</p> <p>For socket Send and Receive actions, the exchanged data is also available in the test log, by means of attachments.</p>

4. In **Statistics sample interval**, type a number and select a time unit.

When you run a schedule, the reports show such information as response time during a specific interval, the frequency of requests being transferred during an interval, and average response trend during an interval. You set this interval here.

5. To set a sampling rate, select **Only sample information from a subset of users**, then select one of the following options.

The number or the percentage that you specify is applied to each user group. If you are running user groups at remote locations, the number or percentage that you select is distributed evenly among the remote locations.

Option	Description
Fixed number of users	<p>The number is applied to each user group. Assume that a schedule contains two user groups. One group contains four users, and another group contains 1000 users. If you specify “2” for this option, two users are sampled from each group.</p>
Percentage of users	<p>The percentage is applied to each user group, but at least one user will be sampled from each group. Assume that a schedule contains two user groups. One group contains four users, and another group contains 1000 users. If sampling rate is set to 10%, one user is sampled from the first group, and 100 users are sampled from the second group. Similarly, if sampling rate is set to 25%, one user is sampled from the</p>

Option	Description
	first group, and 250 users are sampled from the second group.

6. Typically, you should select **Only store All Hosts statistics**.

Selecting this option reduces the amount of statistical data stored, thus enabling you to test a larger load over a longer period of time with significantly less memory usage. Although you will not be able to analyze data from each computer that adds to your test, this data is generally not of interest.

However, if you are running a test over different WANs—and if you are interested in seeing the data from each remote computer—you should clear this box.

Setting the data that the test log collects

The test log shows the events that occurred during a run. By setting the level of information that is collected for a schedule run, you can control whether you receive individual response-time statistics for Page Percentile reports and information about verification points. You can set the level of detail for each type of event: errors, warnings, and other events.

About this task

The level of information collection directly affects log sizes. Depending on the setting that you select, the logs can become quite large. By limiting the log level and collecting the information from a representative sample of users, you can decrease your log size and still have sufficient information for analysis.

For example, if you are debugging a test, you might set all three **What to Log** fields to **All** or **Action Details**. These settings produce large test logs, especially if your tests are long or you are running a large number of users. Large test logs, in turn, increase the test log transfer time, and might even cause your computer to run out of disk space or the agent computer to run out of memory. To reduce transfer times and the likelihood of running out of disk space, sample information from a very small subset of users; smaller even than the default of 5 users per user group. A fixed sampling rate samples the same number of virtual users from each group. A percentage sampling rate samples a percentage of virtual users from each group, but guarantees that at least one user is sampled from a group.

To set the amount of information collected in the test log and the rate of sampling:

1. In the Test Navigator, browse to the schedule, and double-click it.

Result

The schedule opens.

2. In the **Schedule Contents** area, click the name of the schedule.
3. On the Test Log page, select the types of events that you want to collect under **What to Log**. You can collect errors only, errors and warnings, or all events. In other words, **Also show warnings** and **And also show all other types** are unavailable until you select **Show errors and failures**. Similarly, **And also show all other types** is unavailable until you select **Also show warnings**. If none of the **What to Log** check boxes are selected, no test log events are collected.

4. For each type of event, set the **Log Level** to one of the following options:

Option	Description
<p>Schedule Actions</p>	<p>Collects events that correspond to actions executed in the schedule:</p> <ul style="list-style-type: none"> ◦ The overall schedule verdict. The verdict can be one of these values: <ul style="list-style-type: none"> ▪ Pass indicates that all verification points matched or received the expected response. For example, a response code verification point is set to <code>PASS</code> when the recorded response code is received during playback. If your test does not contain verification points, <code>PASS</code> means that all primary requests in the test were successful. ▪ Fail indicates that at least one verification point did not match the expected response or that the expected response was not received. ▪ Error indicates one of the following results: a primary request was not successfully sent to the server, no response was received from the server for a primary request, or the primary request response was incomplete or could not be parsed. ◦ The start and stop time of the schedule, each user group, each virtual user, and each test invocation. ◦ The start and stop time of each loop iteration, if loops are set in the schedule. ◦ The start and stop time of each selector, if selectors are set.
<p>Primary Test Actions</p>	<p>Typically, you set data collection at this level. Primary test actions include schedule actions, plus the following actions:</p> <ul style="list-style-type: none"> ◦ Test verdict, test start, and test stop events. ◦ Loop iteration start and loop iteration stop events, if loops are present in the test.

Option	Description
	<ul style="list-style-type: none"> ◦ Transaction start and stop events if transactions are present in the test. ◦ For HTTP tests, Page title verification points. With this option you can see any page title verification points that you have set. The following events are collected: <ul style="list-style-type: none"> ▪ The page verdict. You see a page verdict only if a connection problem occurs or if you have set verification points. Any failures or errors are rolled up to the test verdict level. ▪ The start and stop time of each page. ▪ The start and stop time of each loop, and the number of iterations of each loop, if you have set loops within a page. ▪ The start and stop time of each transaction, and the duration of each transaction, if you have set page-level transactions in your test. ◦ For SAP tests, SAP screen information, such as SAP screen title verification points. ◦ For Citrix tests, connection elements, window events, and image synchronizations ◦ For socket tests, connect, send, receive, and close elements.
Secondary Test Actions	<p>Secondary test actions include primary test actions, plus this information:</p> <ul style="list-style-type: none"> ◦ For HTTP tests, request-level events. To collect information about response code or response size verification points that you have set, set data collection at this level of detail or greater. <ul style="list-style-type: none"> ▪ The time that the first byte and last byte were sent. ▪ The time that the first byte and last byte were received. ▪ The character set of the response data.

Option	Description
	<ul style="list-style-type: none"> ▪ Expected and actual values of page-level verification points that you have defined. ▪ HTTP think events. ▪ The start and stop time of each transaction, and the duration of each transaction, if you have set request-level transactions in your test. ◦ For SAP tests, SAP element information (primarily Set Property or Call Method actions). ◦ For Citrix tests, synchronization points, delays, text elements, and logoff elements. ◦ For socket tests, this option does not apply.
Action Details	<p>Action details include secondary test actions, plus this information:</p> <ul style="list-style-type: none"> ◦ For HTTP tests, request and response data; for example, HTTP headers and any request data. ◦ For SAP tests, think time information. ◦ For Citrix tests, think time information, mouse actions, and keyboard actions. ◦ For socket tests, this option does not apply.
All	<p>For HTTP, SAP, and Citrix tests, All and Action Details provide the same information.</p> <p>For socket send and receive actions, the exchanged data is also available in the test log, by means of attachments.</p>

5. To set a sampling rate, select **Only sample information from a subset of users**.

The number or percentage that you select is applied to each user group. If you are running user groups at remote locations (that is, on agent computers), the number or percentage that you select is distributed evenly among each location.

Option	Description
Fixed number of users	<p>The number is applied to each user group. Assume that a schedule contains two user groups. One group contains four users, and another group contains</p>

Option	Description
	1000 users. If you specify "2" for this option, two users are sampled from each group.
Percentage of users	The percentage is applied to each user group, but at least one user will be sampled from each group. Assume that a schedule contains two user groups. One group contains four users, and another group contains 1000 users. If sampling rate is set to 10%, one user is sampled from the first group, and 100 users are sampled from the second group. Similarly, if sampling rate is set to 25%, one user is sampled from the first group, and 250 users are sampled from the second group.

6. If you want to log health failure events irrespective of the standard **Test Log** settings, enter a value in **Number of health failure events to log (regardless of logging level)**. By default, this option is set to 50 and is useful to limit the number of events in the Test Log for playbacks that are expected to generate many failures.

Option	Description
Fixed number of users	The number is applied to each user group. Assume that a schedule contains two user groups. One group contains four users, and another group contains 1000 users. If you specify "2" for this option, two users are sampled from each group.
Percentage of users	The percentage is applied to each user group, but at least one user will be sampled from each group. Assume that a schedule contains two user groups. One group contains four users, and another group contains 1000 users. If sampling rate is set to 10%, one user is sampled from the first group, and 100 users are sampled from the second group. Similarly, if sampling rate is set to 25%, one user is sampled from the first group, and 250 users are sampled from the second group.

With the default settings for the **Test Log**, where **Show errors and failures** option is set to **All**, the value specified in this field is relevant only if you are sampling for a subset of users. If you are not sampling for a subset of users, all the errors or failures will be logged regardless of the value in this field.

If you clear the **Show errors and failures** or select the **Only sample information from a subset of users**, then you can use **Number of health failure events to log (regardless of logging level)** option to control how many

failures appear in the test log. You can further refine which failure events are logged by setting the **Affects page health** to **No** in the **Error Handling** page.

Perform the following sub-steps to minimize the logging of failure events:

- a. Clear all the **What to Log** options.
- b. Set the **Number of health failure events to log (regardless of logging level)** to 0.
- c. Disable the Execution Event Console by clearing the **When schedules start** check box in the **Execution Event Console** settings because failures are logged if they are reported in the **Execution Event Console** page.

Exemple

The default setting, to log all errors and warnings, as well as primary test actions, fits most purposes. However, you can log any type of information, from no information to all information from all users, although neither is a typical situation.

- To see only errors and warnings, set the first two **What to Log** check boxes to **All** and clear the third check box, **And also show all other types**, to avoid logging successful events.
- To check a schedule's structure, when you are not interested in the test execution results, set all three **What to Log** boxes to **Schedule Actions**.

Both choices, as well as the default setting, will limit the size of the test log and reduce the total time to run the schedule by significantly shortening the test log transfer time at the end of a test.

If you are debugging a test, you might set all three **What to Log** fields to **All** or **Action Details**. These settings produce large test logs, especially if your tests are long or you are running a large number of users. Large test logs, in turn, increase the test log transfer time, and might even cause your computer to run out of disk space.

Setting the problem determination level for schedules

You can set the level of information that is saved in the problem determination log during a run. By default, only warnings and severe errors are logged. Typically, you change this log level only when requested to do so by the Support person.

About this task

The problem determination logs contain internal information about the playback engine. These logs are particularly useful for debugging problems such as Kerberos authentication, SSL negotiation, and resource constraints on an agent. The log files are named `CommonBaseEvents00.log` and are located in the deployment directory. For example, if you play back a schedule on an agent and set `C:\Agent` as the deployment directory, the problem determination log files are in a directory similar to `C:\Agent\deployment_root\\A1E14699848784C00D2DEB73763646462\CommonBaseEvents00.log`. If a large amount of log information is generated, multiple `CommonBaseEvents` files are created.

To set the level of problem-determination logging and the sampling rate:

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.

2. In the Schedule Contents area, click the name of the schedule.
3. On the Problem Determination page, set **Problem determination log level** to one of the following options:

Option	Description
All, Finest, Finer, Fine	Set these options only if you are requested to do so by technical support.
Config	Logs static configuration messages. Configuration messages, which include hardware specifications or system profiles, require no corrective action.
Info	Logs informational messages. Informational messages, which include system state, require no corrective action.
Warning	Logs warning messages. This is the default setting. Warning messages, which might indicate potential problems, require no corrective action.
Severe	Logs critical and unrecoverable errors. Critical and unrecoverable messages interrupt normal program execution, and require corrective action.
None	Turns logging off.

4. To set a sampling rate, select **Only sample information from a subset of users**.

The number or the percentage that you select is applied to each user group. If you are running user groups from remote locations, the number or percentage that you select is distributed evenly among the remote locations.

Option	Description
Fixed number of users	The number is applied to each user group. Assume that a schedule contains two user groups. One group contains four users, and another group contains 1000 users. If you specify "2" for this option, two users are sampled from each group.
Percentage of users	The percentage is applied to each user group, but at least one user will be sampled from each group. Assume that a schedule contains two user groups. One group contains four users, and another group contains 1000 users. If sampling rate is set to 10%, one

Option	Description
	user is sampled from the first group, and 100 users are sampled from the second group. Similarly, if sampling rate is set to 25%, one user is sampled from the first group, and 250 users are sampled from the second group.

Results

When a user group runs on your local computer, the problem determination logs are in the `deployment_root` directory in your workspace. When a user group runs at a remote location, which is the typical use case, each remote location has a deployment directory, which you define and which is listed on the Locations page for that user group.



Note: Common Base Event XML logs from remote locations use Universal Coordinated Time (UTC), also called Zulu time, which is likely different from your local time. For example, Eastern Standard Time (EST) is 5 hours behind UTC time.

What to do next

To view the problem determination log, open the log file in an XML editor. Select the log whose timestamp matches that of the problem run. The most recent log has the suffix `00.log`.

Setting problem determination level for tests

You can set the level of information that is saved in the problem determination log during a run. By default, only warnings and severe errors are logged. Typically, you change this log level only when requested to do so by the Support person.

About this task

The problem determination logs contain internal information about the playback engine. These logs are particularly useful for debugging problems such as Kerberos authentication, SSL negotiation, and resource constraints on an agent. The log files are named `CommonBaseEvents00.log` and are located in the deployment directory. For example, if you play back a schedule on an agent and set `C:\Agent` as the deployment directory, the problem determination log files are in a directory similar to `C:\Agent\deployment_root\<UserName>\A1E14699848784C00D2DEB73763646462\CommonBaseEvents00.log`. If a large amount of log information is generated, multiple `CommonBaseEvents` files are created.

1. Open the test for which you want to set the problem determination log level.
2. Select the root node and from the **Test Details** section, select **Problem Determination**.
3. On the Problem Determination page, set **Problem determination log level** to one of the following options:

Option	Description
All, Finest, Finer, Fine	Set these options only if you are requested to do so by technical support.
Config	Logs static configuration messages. Configuration messages, which include hardware specifications or system profiles, require no corrective action.
Info	Logs informational messages. Informational messages, which include system state, require no corrective action.
Warning	Logs warning messages. This is the default setting. Warning messages, which might indicate potential problems, require no corrective action.
Severe	Logs critical and unrecoverable errors. Critical and unrecoverable messages interrupt normal program execution, and require corrective action.
None	Turns logging off.

4. Save the test.

Chapter 8. Test Execution Specialist Guide

This guide describes tasks that you can perform on schedules, test execution with custom code, and Extending HCL OneTest™ Performance to support other protocols. This guide is intended for testers or test execution specialists.

Running schedules

After you have added the user groups, tests, and other items to a schedule, and you are satisfied that it represents a realistic workload, you run the schedule.

Running a local schedule or test

You can run a test locally or a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule. on remote locations with a default launch configuration.

Before you begin

To play back tests against the applications that require client authentication such as Digital Certificates, Smart Card, or Kerberos, you must provide the appropriate authentication before playing back the test.

- To play back a test with a digital certificate, see [Playing back a test with a digital certificate on page 266](#).
- To play back a test that require smart card authentication, click the **Security** tab in the Test editor, and provide the same smart card certificate alias and PIN that you used for the recording. See the Smart card authentication topic for more information.
- To play back a test that require Kerberos authentication, see [Generating tests that use Kerberos on page 270](#).

About this task

When you run a schedule or test in this way, HCL OneTest™ Performance automatically sets up a simple launch configuration. A test runs on the local computer, with one user. A schedule runs with the user groups or Rate Runner groups and the locations that you have set. However, the execution results have a default name (the same as the schedule or test, with a suffix) and are stored in a default location.

The Rate Schedule can be run only on agent locations.

When you run a schedule with multiple agents, an agent might be lost, especially during the long load test run. Losing an agent is not common and occurs during some extreme cases such as when computer's memory is exhausted.

When an agent is lost, by default, the schedule is stopped. When the schedule is stopped in this manner, you must fix the reason of agent loss or add more agents before running the schedule. To continue to run the schedule without the lost agent, in the Schedule editor, click the **Advanced** tab and clear the **Loss of an agent halts execution** check box.

Typically, the agents divide the load among themselves. So, running a schedule without the lost agent might give unpredictable results. If you use a segmented dataset and if you run a schedule without the lost agent, the data is not redistributed among the surviving agents. Also, if the schedule has multiple stages, by default, the load is distributed among the surviving agents at the next stage. But, if the **Replace lost users in current stage** check box is selected,

then the load is distributed evenly among the surviving agents in the current stage. If the check box is cleared and a percentage of users or clients are allowed to exit during stage execution, the load is distributed among the surviving agents in the next stage. Loss of an agent in a schedule run is logged in the Performance Report.

To stop a test gracefully without causing incomplete page hits, select the **Active actions are allowed to complete if stop requested** check box at **Window > Preferences > Test > Test Execution**.

To receive email notification for the status of the run, specify the email properties in **Window > Preferences > Test > Test Execution**.

1. In the Test Navigator, expand the project until you locate the schedule or test.
2. Right-click the schedule or test, and then click **Run As > Performance Schedule** or **Run As > Test**.



Note: If you run an HTTP schedule on a remote Macintosh computer, the test fails. The cipher suite that is used for recording must be available in Oracle JDK on the Macintosh computer. For example, you can use `TLS_RSA_WITH_AES_128_CBC_SHA` on Macintosh.

Results

After you run a test or a schedule, the Performance Test Runs view opens. In this view, you can add comments about the selected result and view the settings that were used to run the schedule. To add comments, in the lower-left panel of the Performance Test Runs view, click **User Comments**. The comments that you enter are displayed on the Summary page of performance reports. To view the settings that were used for a schedule run, click **Schedule Settings**. The Performance Test Runs View Schedule Settings page displays and shows the statistics and test log settings that were used for the run.



Note: When you record a test that includes a file download, the file is not physically saved to disk. However, you can confirm that the file was retrieved from the server by looking in the response of the request that asked for the file. One method to locate the request for large downloaded files is to look for a request with a large response size.

What to do next

You can configure a schedule or test. A typical reason for setting up a configuration is to control where the execution results are stored. For more information, see [Setting a launch configuration on page 621](#).

Running a long run mode SAP GUI test

When running a SAP GUI test that could last for many hours and could use up the operating system resources, you can choose to run the test with the Long Run Mode process. It is an external process that restarts automatically after the specified number of SAP sessions are over. So, tests of longer duration tests are more likely to finish.

About this task

The following guidelines improve the success of long duration tests:

- Use agent computers with at least 2GB of RAM and 10GB of free disk space, running the same version of the HCL OneTest™ Performance or HCL OneTest™ Performance Agent.
 - Disable antivirus software, screen savers, and automatic updaters on the agent computers during the test.
 - Keep individual tests short by having loops of not more than 10 iterations within each test. Get the desired long run duration by looping within the schedule. Do not exceed 20 or 30 virtual testers for each agent computer with a think time of more than several seconds.
 - Use tests with a minimal number of verification points.
 - Do not use the Signature theme. Disable all animations in the SAP GUI.
 - Do not use the mouse during the test. Any mouse click could be interpreted by a hidden SAP GUI window, which could cause the test to fail.
 - Ensure that the Screen Throughput counter remains low (approximately 1 every second for each agent).
 - Before playing back a test, in SAP Connection Details editor, click **Test Connection** to test the connection to the SAP GUI server.
1. In the schedule, select a User Group that includes a SAP GUI test.
 2. In the User Group Details, click the **Options** tab and click **Edit Options**.
 3. In **Long Run Mode process renewal level**, select one of the following options:
 - **Schedule**: Starts the Long Run Mode process at the schedule level. All of the SAP sessions are managed by one process.
 - **User Group**: Starts the Long Run Mode process at the beginning of each user group.
 - **Virtual User**: Starts the Long Run Mode process for each virtual user.
 - **Long Run Mode off**: Does not use the Long Run Mode process. In this mode, the SAP GUI is directly called by HCL OneTest™ Performance, thereby increasing the resource consumption of the operating system.
 4. In **Number of SAP session starts per process**, specify the number of SAP sessions to run. The Long Run Mode process runs one session at a time. For example, if you specified 100 sessions for Virtual User, the Long Run Mode process takes one session at a time, and when the 100 sessions are run, a new process starts.
 5. Click **OK** and save the schedule.

Related information

[Evaluating results on page 773](#)

Running long duration Citrix tests

When tests exceed many hours, resource consumption issues can cause problems for the Citrix clients. The long run mode increases the reliability of long duration tests with the Citrix protocol by running the test using multiple processes.

Before you begin

Ensure that the test plays back reliably, with no errors, when run as a single test.

For Citrix tests, a new process is created for each virtual tester.

The following guidelines improve the success of long duration tests:

- Use agent computers with at least 2GB of RAM and 10GB of free disk space, running the same version of the HCL OneTest™ Performance or HCL OneTest™ Performance Agent. Disable antivirus software, screen savers, and automatic updaters. Avoid using the agent computers during the test.
- Keep individual tests short by avoiding loops of more than 10 iterations within each test, and achieve the desired long run duration by looping within the schedule. Do not exceed 20 or 30 virtual testers per agent computer with a think time above several seconds.
- Use tests with a minimal number of verification points.

To enable long run mode:

1. In the Test Navigator, browse to the schedule and double-click it.

Result

The schedule opens.

2. In the schedule, click the user group for which you want to enable the long run mode.
3. In the Schedule Element Details area, click the **Options** tab and click **Edit Options**.
4. Select the **Enable long run mode** check box and click **OK**.
5. Save the schedule.

Testing with Docker images

HCL OneTest™ Performance, HCL OneTest™ UI, and HCL OneTest™ Performance Agents are available for download as Docker images. You can use them to fulfill the continuous testing aspects of your DevOps lifecycle.

You must use only floating licenses for the product and VT-pack when playing back tests using Docker. These licenses should be hosted on a server that can be accessed by the workbench.

Running tests with containerized agents

When you have a local workbench, instead of installing the agents on different machines and locations, you can deploy the containerized agents to generate the load.

Before you begin

You must have configured the Docker container. See [Configuring Docker containers on page 113](#).

About this task

Typically, when the agents are installed, you specify the workbench host name and port number to establish the connection with the workbench. If you use containerized agents, they are already installed. Therefore, you specify the connection information during the run.



Note: The version number of the container images and the desktop products must match. If you have previous version of the container image, uninstall it and install the latest version.


To uninstall the image, you must stop the container by running the `docker stop "CONTAINER ID"` command, and then run the `docker rmi -f "image ID"` command to uninstall the image.

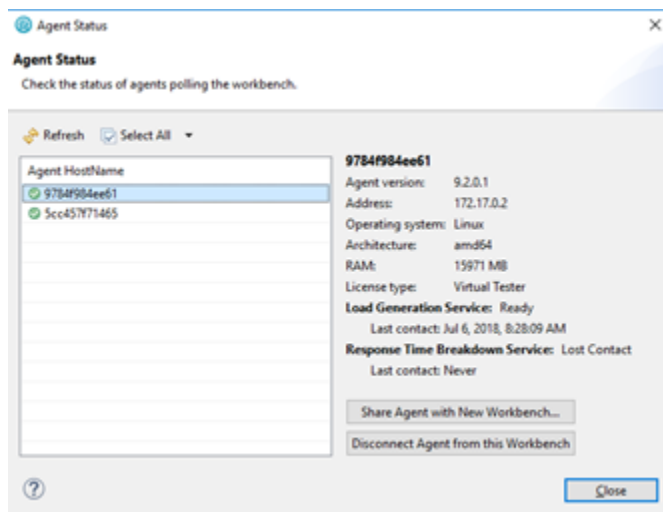
1. Start the container instance of the agent by running the following command:

```
$ docker run -dit --rm -e MASTER_NAME=Workbench_name or IP -e MASTER_PORT=port_number -e AGENT_NAME=Agent_name -e AGENT_IP=IP_address imageName: imageVersion
```

Table 3. Description of parameters

Command	Description
-dit	Specifies that the agent container runs in the background.
--rm	Specifies to clean up the container and remove the file system when the container exits.
MASTER_NAME	Specifies the IP or host name of the workbench.
MASTER_PORT	Specifies the port number of the workbench. If you use the default port number of 7080, this command is optional.
AGENT_NAME	Optional: Specifies the name of the agent that report to the workbench.
AGENT_IP	Optional: Specifies the IP address of the agent that report to the workbench.
imageName:image-Version	Specifies the name and version of the image.

2. Click the **Agent Status** icon  from the product to verify the two container agents are polling the workbench.





Note: The agent host names should match the IDs of the containers running in Docker. Make a note of the IP address of the each agent since they must be used when creating the agent locations.

3. In the schedule editor, create a new location test asset for each container agent so that the selected **User Group** runs on two agent locations.
4. Run the schedule.

The deployment step could result in the schedule remaining in the “*Launching*” state for several minutes.

Running automated tests with containerized workbench and agents from Docker

To simplify the deployment piece of Continuous Testing, you can use built Docker images to deploy the workbench and the agents and start testing in no time. You need not install the workbench and the agents on different machines. You can deploy the Docker images and use Docker commands to play back tests.

Before you begin

You must have configured the Docker container. See [Configuring Docker containers on page 113](#).

You must already have exported the test assets to a location from where Docker can import them. For information about exporting the test assets, see [Copying test assets with dependencies on page 396](#).



Note: The version number of the container images and the desktop products must match. If you have previous version of the container image, uninstall it and install the current version. To uninstall the image, use these commands:

1. Stop the container by running

```
docker stop "CONTAINER ID"
```

2. Uninstall the image by running

```
docker rmi -f "image ID"
```

1. To run the test without using any agents, start the container:

```
$ docker run --rm -e HCL_ONETEST_LICENSING_URL=<URL> HCL_ONETEST_LICENSING_ID=<server_ID>
-v hostTestAssets:/containerTestAssets -v hostImportedData:/containerImportedData imageName:imageVersion
cmdline -workspace /containerImportedData/workspace -project projectName -schedule testName -results
autoResults -stdout -exportlog /containerPathExtracted/testlog.txt
```

Table 4.

Command	Description
--rm	Removes the container after the run completes.

Command	Description
-e	Sets environment variables.
HCL_ONETEST_LICENSING_URL=<URL>	Specifies the URL of the license server, usually, <code>https://hclsoftware.compliance.flexnet-operations.com</code> .
HCL_ONETEST_LICENSING_ID=<ID>	Specifies the cloud license server ID. If you are using a local license server, do not use this variable. The floating license for the product and VT-packs must be on the license server.
<i>hostTestAssets:/containerTestAssets</i>	Specifies the folder location on the host machine and the container containing the compressed test assets (Zip format). Use both the locations to map one or more shared volumes to transfer data such as test assets, logs, and execution results between the host and the container.
<i>hostImportedData:/containerImportedData</i>	Specifies the workspace location on the host machine and the container containing the test assets that are not compressed. Results from the test execution are saved to the directory you specify on the host machine.
TEST_IMPORT_PATH=<PATH>	Specifies the location of the compressed test assets to be imported into the container. The location path is on the container side and not the host. For example, <code>/containerTestAssets/archiveName.zip</code> . The volume and path names are user defined and should be consistent.
imageName:imageVersion	Specifies the name of the image and its version to run.
cmdline	Specifies the existing command line arguments to define the location of the workspace, project name, test or schedule name, results file name, and the location of the exported logs.

Result

After the test run completes, check the *hostImportedData* on the host machine to view the exported log.

- To run the tests on containerized agents, load the agent images into the Docker repository:

```
tar --wildcards --to-command='docker load' -xzf <workbenchImageName> 'images/*'
```

For example, the workbench image name could be `hcl-onetest-<versionNumber>.tar.gz`.

Result

When the image is loaded, the following message is displayed - Loaded image: `imageFileName:versionNumber`

3. Specify the agent details in the workbench. You can do this in one of the two ways:
 - Specify all of the details in the local workbench and export the test assets to the directory that will be used by the containers to choose the tests.
 - Bring up the workbench UI in the container to specify the agents in the schedule editor. To bring up the workbench UI in the container, install and configure an X11 server such as Xming on your host machine and specify `-e DISPLAY=<IP>:0.0` parameter in the docker run command. The IP is the IP address of your host machine. For information about how to install and configure Xming, see its product documentation.



Note:

- If you use 'localhost' instead of the IP address with the default Docker settings, the container will forward the display to itself. If your IP is assigned via DHCP it is liable to change and you will need to update your container's environment variable accordingly.
- Do not use underscores in the agent names.

Name	Hostname	Directory	IP Aliasing
<input checked="" type="checkbox"/> hpt-agen...	hpt-agent1	/tmp/deployment	
<input checked="" type="checkbox"/> hpt-agen...	hpt-agent2	/tmp/deployment	

4. To map a User Group with a specific agent, assign a static IP to the agent. To assign an IP to the agent, use the **NODE_IP** parameter along with the docker run command.



Note: Ensure that this IP matches with the IP specified for the Location asset in the workbench.

5. Initiate the test runs against the agents in one of the following ways:

a. To facilitate a run where container agents will automatically connect the workbench container to run a schedule, install the [Docker Compose](#) tool.

- i. Create a `docker-compose.yml` file that specifies similar parameters as mentioned in step 5.
- ii. To run the tests, in the command prompt, navigate to the directory containing the yml file and run:

```
docker-compose up
```

In addition to the parameters in step 5, you must also specify the following two parameters in the yml file:

- **MASTER_NAME:** Specify the name of the workbench container.
- **AGENT_NAME:** Specify the name of the agent. The agent name defined in the schedule must match with the name of the agent container.



Sample compose file:

```
#SIMPLE DOCKER COMPOSE FILE/TEMPLATE
#BE SURE TO REPLACE ANY PROJECT-SPECIFIC NAMES/PATHS AND LICENSING VARIABLES WITH
YOUR OWN VALUES
version: '2'
services:
  agent1:
    image: <agentImageName>:<imageVersion>
    environment:
      - MASTER_NAME=<workbenchImageName>
      - AGENT_NAME=<agentImageName>

  agent2:
    image: <agentImageName>:<imageVersion>
    environment:
      - MASTER_NAME=<workbenchImageName>
      - AGENT_NAME=<agentImageName-2>

  workbench:
    image: <workbenchImageName>:<imageVersion>
    entrypoint: cmdline -workspace /runData/workspaceJuly10 -project
921proj -schedule Schedules/agentSched -results autoResults -stdout
-exportlog /runData/agentSchedLogJuly10.txt
    ports:
      - "7080:7080"
      - "7443:7443"
    volumes:
      - C:\Tests:/Tests
      - C:\runData:/runData
    environment:
```



```
- HCL_ONETEST_LICENSING_URL=<URL> HCL_ONETEST_LICENSING_ID=<ID>
- TEST_IMPORT_PATH=/Tests/agentProj.zip
```



Note: Docker Compose is included with some versions of Docker. The tool automates some network configurations and makes it easier to coordinate multiple containers. To check whether you have it, run `docker-compose --version`.

- b. Start the agent containers by passing the following command as many times as you want the number of agents for the run. To start four agents, pass the command four times.

```
docker run -it -e MASTER_NAME=IP_ADDRESS -e MASTER_PORT=PortNumber -e AGENT_NAME=NameofAgent
-e AGENT_IP=AgentIP imageName:imageVersion
```

Table 5. Description of parameters

Command	Description
-dit	Specifies that the agent container runs in the background.
-e	Sets environment variables.
-rm	Specifies to clean up the container and remove the file system when the container exits.
MASTER_NAME	Specifies the IP or host name of the workbench.
MASTER_PORT	Specifies the port number of the workbench. If you use the default port number of 7080, this command is optional.
AGENT_NAME	Specifies the name of the agent. When there are multiple agents running the test, the agent names helps you in identifying the results the agent is associated with.
AGENT_IP	Specifies the IP address of the agent.
imageName:imageVersion	Specifies the name and version of the image.

6. Verify whether the schedule has completed successfully. If you used an option such as **-exportlog** to output results to the shared volume, check the corresponding directory on your host machine that was mapped to `hostImportedData` to retrieve the exported data.

What to do next

If the test or schedule has completed successfully, the agent will likely be running. You might have to explicitly stop the agent by running

```
docker stop containerID
```

If you used Docker Compose tool to run the tests, you can stop the agents when the workbench container exits by running

```
docker-compose up --abort-on-container-exit
```

Related information

[Testing with containerized agents on page 611](#)

Adjusting delays in HTTP tests

You can configure HTTP tests to use client-side processing delays. Client-side processing delays wait for the first character or last character that is received in a response for a previous request in order to better emulate the work done on the client computer. You can also scale the recorded delays in HTTP tests to change the rate at which a test runs.

Configuring HTTP client delays

You can configure delays for HTTP requests to emulate client-side processing delays. Applications that use client-side Javascript, such as Web 2.0 applications, often incur significant delays due to processing done on the client. You can emulate this client-side processing in HTTP tests. Running an HTTP test too quickly can cause unexpectedly low page response times to be reported, and can generate excessive load on the server under test.

About this task

To remove request delays from response times for all the tests, click **Window > Preferences > Test > Test Generation > HTTP Test Generation** and select the **Remove HTTP request delays from response times**.

To remove request delays from response times for a specific test, in the Test editor select **HTTP Options** and clear the **Remove HTTP request delays from response times** check box.

To configure HTTP client delays:

1. In the Test Navigator, browse to the test and double-click it.

Result

The test opens.

2. Click the name of the test.
3. Click **HTTP Options**.
4. On the **HTTP options** page, select **Enable new client processing delays**.
5. Under **Test Contents**, select the request on which to enable client processing delays.
6. Under **Test Element Details**, click the **Advanced** tab.
7. Under **Client Processing Delay**, click **Request**.

Result

A test editor window opens, listing the previous requests in the test.

8. Select the request to wait for, and then click **OK**.

9. For **Release when**, select **First Character Received** or **Last Character Received**.
10. Type any **Additional delay** to add in milliseconds.

Adjusting client delays for all tests

To ensure consistency in client processing delays for all the tests in a user group, starting from 9.1.1.1, you can override the client processing delays of the tests and set it in the schedule.

About this task

To remove request delays from response times for all the tests, click **Window > Preferences > Test > Test Generation > HTTP Test Generation** and select the **Remove HTTP request delays from response times**.

To remove request delays from response times for a specific test, in the Test editor select **HTTP Options** and clear the **Remove HTTP request delays from response times** check box.

1. In the schedule editor, select the user group for which you want to set consistent client delay.
2. Click the **Options** tab and click **Edit Options**.
3. Select the **Override Test Client Delay** check box, adjust the HTTP requests delay, and click **OK**.

Overriding the HTTP connection timeout value

By default, HTTP page connections use a timeout value of 190 seconds, which might not be sufficient for some applications. You can override the page connection timeout value for specific pages in a test.

To configure the HTTP connection timeout:

1. In the **Test Navigator** view, browse to the test and double-click it.

Result

The test opens.
2. Under **Test Contents**, select the request on which to enable client processing delays.
3. Under **Test Element Details**, click the **Advanced** tab.
4. Under **Timeout**, select **Override timeout value** and specify the timeout delay. You can specify a numeric value for the timeout delay or use a dynamic value from data sources such as dataset, variables, and data correlation.

Overriding WebSocket response timeout value

By default, the timeout value for WebSocket responses is 240 seconds at the test level. This value might be insufficient or in excess for different applications. You can customize the timeout value for specific responses.

1. In the **Test Navigator** view, browse to the test and double-click it. The test opens.
2. Under **Test Contents**, select the request on which to enable client processing delays.
3. Under **WebSocket Response Details**, click the **Advanced** tab.
4. Under **Timeout**, select **Timeout activated** and specify the timeout delay. You can specify a numeric value for the timeout delay or use a dynamic value from data sources such as dataset, variables, and data correlation.

Related information

[Recording a WebSocket test on page 189](#)

Playing back HTTP tests faster than the recorded rate

If the client computer used for recording an HTTP test was slower than required, or if you want to emulate a faster client computer, you can increase the playback rate without altering the actual values in the recorded test by reducing the client delays proportionately. Similarly, you can slow down the client by increasing the client delays.

About this task

Each request in a recorded test includes a programmatically calculated delay before the request is issued. This delay is a statistical emulation of user behavior. To see a delay in a test, click a request, and examine the **Test Element**

Details area: Delay:

You can scale the delay in these requests to change the rate that a test runs. This scaling occurs at the test level.

To scale the delays:

1. In the Test Navigator, browse to the test and double-click it.

Result

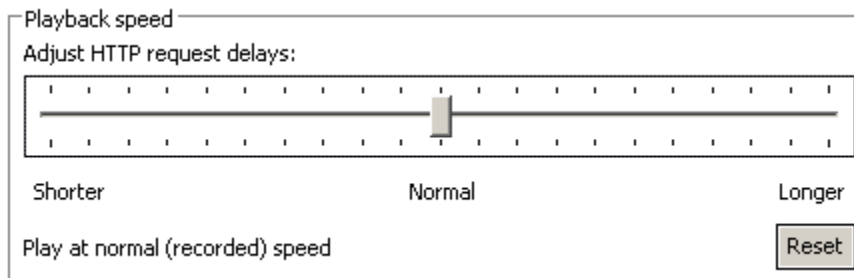
The test opens.

2. Click the name of the test.
3. In the **Test Element Details** area, select a scaling percentage. Move the slider to scale the speed at which the HTTP requests are sent. You can specify a range from no delays to twice the recorded length.

This scale is applied to all requests in the test.



Note: If you increase playback speed dramatically, requests might occur out of order. To correct this problem, decrease playback speed until the test runs correctly.



What to do next

You can also set a maximum HTTP delay. Click **Window > Preferences > Test > Test Generation > HTTP Test Generation**. Select the **Protocol** tab, and enter a value for **Maximum Request Delay**. Although requests larger than this value are truncated in the generated test, the recorded test still contains the original values.

Setting a launch configuration

Instead of using the default launch configuration, you can specify the file name for the execution results, the name of the folder for the execution results, and, for a test, the number of users.

About this task

You generally run a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule. by right-clicking it and selecting **Run > Run VU Schedule** VU Schedule or **Run > Run Rate Schedule** A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule.. However, you should set a launch configuration when:

- You want to specify a name for the execution results, or you want them in a separate folder.
- You plan to run a test outside of a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule., and you want to run the test with more than one user.
- You want the launch configuration to appear in your toolbar menu.
- You want the launch configuration to be available to other users.

To set a launch configuration:

1. In the Test Navigator, expand the project until you locate the schedule or test.
2. Right-click the schedule or test, and then click **Run As > Run configuration**.
If the Perspectives page is displayed, keep the defaults.
3. In the **Configurations** area on the left, click **VU Schedule** or A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule., and then click **New**.

Result

A test configuration, initially named `New_configuration`, is created. Typically, you supply a configuration name that is similar to the schedule name.

At this point, you can run the schedule if you click **Run**. However, you will not have created a meaningful configuration.

4. Click the **Test Logs** tab and check the default settings. To change the default settings, clear the **Use defaults** check box and type a file name for the execution results. The product appends a time stamp to this name. To overwrite the file each time that you run the configuration, select the **Override existing test log** check box.
5. Click the **Common** tab to inspect or modify your run preferences.
6. In **Save as**, select one of the following options:

Option	Description
Local	This launch configuration is stored in your work-space, and it is not visible to other users.
Shared	Other users have access to the launch configuration; you are asked where to store it.

7. For **Display in favorites menu**, select one or more of the following options:

Option	Description
Run	The configuration is displayed in your Run toolbar menu. If you select a toolbar menu at all, this is the logical choice for a schedule or test.
Debug	The launch configuration is displayed in your Debug toolbar menu.
Profile	The configuration is displayed in your Profile toolbar menu.

8. Verify that **Launch in background** is selected. If you do not run the configuration in the background, you cannot do anything in Eclipse until it finishes running the configuration.
9. Click **Apply**, and then click **Run** to run the configured schedule or test, or click **Close** to save the configuration and run it later.

Running a configured schedule

If you do not use the default launch configuration, you can configure the schedule and then run it.

Before you begin

You must configure the schedule before you run it. For more information, see [Setting a launch configuration on page 621](#).

1. In the Test Navigator, expand the project until you locate the schedule.
2. Right-click the schedule, and then click **Run > Run**.
3. In the Configurations area on the left, click **Test Schedule**, and then click the name of the schedule to run.
4. Click **Run**.

Results

While the schedule is running, the reports are updated in real time, and you can see the changes.

Configuring multiple host names for a location

You can run several locations on the same computer by configuring multiple host names for a location. This configuration affects all tests running at that location; all tests will run with the configured port.

To configure multiple host names for a location:

1. Open the hosts file, which maps IP addresses to hosts, with an ASCII editor.
On Windows™, the hosts file is in `C:\Windows\system32\drivers\etc\hosts`. On Linux™, the hosts file is in `\etc\hosts`.
2. At the end of the hosts file, add your IP mappings. Use one IP address, but map it to two (or more) logical host names.

Example

For example, you could add map the IP address 123.4.5.6. to two logical hosts, as show in the bottom two lines:

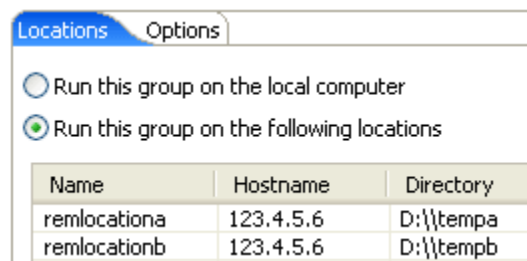
Result

```
# Copyright (c) 1993-1999 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com           # source server
#       38.25.63.10      x.acme.com              # x client host
127.0.0.1       localhost
123.4.5.6       remlocationa
123.4.5.6       remlocationb
```

3. Create two deployment locations that have names identical to the names you added in the hosts file:
 - a. Open the schedule that contains the user group that you want to run on multiple hosts.
 - b. Open the user group, and click the **Location** tab.
 - c. Click **Add > Add New**, and enter the location data. Make sure the locations have different directories (in this example, they are tempa and tempb).

Example

Result



Name	Hostname	Directory
remlocationa	123.4.5.6	D:\tempa
remlocationb	123.4.5.6	D:\tempb

- d. Click **Finish**.

Automating tests

You can run a schedule from the command line. You can also set preferences to export results after the run completes from the command line or from the workbench. Together, these features let you run tests and analyze results without opening the workbench. You can even write scripts to process the exported results.

Creating a command-line config file

Starting from V10.0.2, you can create command line config file from the product, which you can use while running tests or schedules from the command-line interface and Maven.

Before you begin

You must have performed the following tasks:

- Created test assets in a workspace.
- Installed Maven if you are running tests or schedules from the Maven build.

For information about creating tests or schedules and installing Maven, see related links.

About this task

Previously, you created the config file manually by adding parameters to it for running the tests or schedules by using the config file from the command line. Now, you can create a command-line config file from the product by right-clicking the test asset. The required parameters are automatically assigned, and you can specify any optional parameters, while creating the config file. You can use this config file to run the tests or schedules from the command-line interface and Maven plug-in that is provided with the product as part of Maven build.

1. In the Test Navigator, browse and select the test or schedule.
2. Right-click the test or schedule , and then click **Create command line config file**.
3. In the **Create New Config File** window, enter a name for the new configuration file and then click **Next**.
4. Perform the following sub-steps in the **Command Line Arguments** window:
 - a. Select the format of the config file from the following options:
 - **Regular** – Use this format to run tests or schedules from the command-line interface.
 - **Maven** – Use this format to run tests or schedules from the Maven build.
 - b. If you want to add more parameters to a config file, specify the values in the fields from the available configuration options.
5. Click **Finish**.

Results

The **Config file created** dialog box displays the location of the config file.

What to do next

You must complete the following steps:

1. Close the product.
2. Run a test or schedules by using the config file either from the command-line interface or from the Maven build.

Related information

[Creating tests on page 180](#)

[Testing with Maven on page 174](#)

[Running a test or schedule from a command line on page 627](#)

Activating secure storage of dataset passwords for test automation

Starting from 9.2.0.1, you can store the encrypted dataset passwords in the Eclipse secure storage location on the computer. Now, when you run the tests from the command line, the product automatically uses the password and completes the test run. Prior to 9.2.0.1, you could not run the tests from the command line with encrypted datasets.

1. Click **Window > Preferences > Test > Test Execution > Automation Security**.
2. Select the **Activate Secure Storage Support for Encrypted Datasets** check box.
The password is stored in the Eclipse secure storage. Do not share the computer's login credentials with others.
3. To add the encrypted datasets, click **Add**, select the encrypted dataset, and click **OK**.
You will be prompted to enter the dataset password that you used when encrypting the dataset.
4. Enter the password and click **OK**.

Results

When you run the tests from the command line, the test runs will complete successfully without the need to specify the password. If another user runs the same tests with encrypted datasets, the dataset password must be entered for the tests to run successfully.

Exporting report counters automatically

You can change the test preferences so that report counters are automatically exported at the end of a run. This option is useful when you run a schedule from the command line because you can automatically export results without opening the workbench.

To automatically export report counters to a CSV file when a test or schedule is complete:

1. Open the Preferences page. Click **Windows > Preferences**.
2. Open the Export Reports page. In the **Preferences** window, expand **Test** and **Performance Test Reports**, and then select **Export Reports**.
3. In the Export Reports window, select the options as follows:

Option	Description
Export reports when run completes from	Select Command line or Workbench . If you select Command line , you can also select Print simple CSV reports to command line to display the exported data on the command line (standard output) as well as

Option	Description
<p>Simple (counters as lines, time ranges as columns)</p>	<p>export it to the CSV file. The file is displayed after the command line run has completed.</p> <p>To export a simple report in the CSV format, select this check box.</p> <p>List All Time Ranges - Select this check box to include data from all the time ranges. The Entire Run time range is included by default.</p> <p>Include per instance counters: Select this check box to include counters for all the page elements.</p> <p>Export each agent separately - Select this check box to export data for each agent to a separate CSV file.</p> <p>One file per agent - Select this check box to export data for each agent (location) to different sections in a single CSV file.</p>
<p>Full (time intervals as lines, counters as columns)</p>	<p>To export a comprehensive report that includes the result name, node name, and time ranges, select this check box. Typically, you do not include these details unless you are exporting customized reports that include counters from specific test runs.</p> <p>Split output if column exceeds - Because there will be a lot of data, select this check box to create multiple CSV files if the number of columns exceed the specified limit.</p> <p>Include per instance counters - Select this check box to include counters for all the page elements.</p> <p>Export each agent separately - Select this check box to export data for each agent (location) to different sections in the same CSV file.</p> <p>One file per agent - Select this check box to export data for each agent to a separate CSV file.</p>
<p>HTML report</p>	<p>Select this check box to export full report data in the HTML format.</p>

Option	Description
Executive Summary report	Select this check box to export the report in the HTML format. This report summarizes the state of the test or schedule run and display the report on only one HTML page. It can be printed.
Select reports to export	<p>Expand the tree, if necessary, to display the type of report to export. If you select more than one report, each report is exported to a separate CSV file in the Test Runs directory.</p> <p>Show Report Ids - Select this check box to view the ID of each report. The IDs are used when exporting the specific reports from command line.</p>

4. Click **Apply**.

Results

The CSV file is called `results_file_name.report_name.csv`. It contains metadata about the test run, a blank line, and then lists each counter and its last value. Each counter is on a separate line.

Running a test or schedule from a command line

You can run a test or schedule without opening the product by using the command-line interface.

Before you begin

You must have set the **Web Reports** preferences in the desktop client to view the status of the test run from the command line. See [Accessing reports remotely on page 800](#).

1. Navigate to the directory that contains the `cmdline.bat` and `cmdline.sh` files. On Windows™ operating systems, this directory is typically found as `productInstallationDirectory/cmdline`. For example, `C:\Program Files\HCL\HCLOneTest\cmdline`.
2. Issue the following command:

Example

```
cmdline -workspace workspace_full_path -project proj_rel_path -eclipsehome eclipse_full_path
-plugin plugin_full_path -schedule sched_rel_path -suite suite_rel_path -importzip file_full_path.zip
-varfile variable_file_full_path -servicename service -serviceargs service_args -configfile
file_full_path -results result_file -overwrite {"true" | "false"} -quiet -users nn -vmargs JVM_args
-rate RateRunnerGroupName=iterationNumber/duration, iterationNumber/duration-duration
Stage1=durationOfStage; Stage2=durationOfStage -publish serverURL#project.name=projectName
-publish_for {ALL,PASS,FAIL,ERROR, INCONCLUSIVE} -exportlog log_full_path -exportstats local_dir_path
-timerange "all, 5 Users, 10 Users" -exportstatshtml local_dir_path -compare "result_path1, result_path2"
```

-exportstatreportlist *stats_list* **-execsummary** *local_dir_path* **-execsummaryreport** *reportID* **-usercomments** *"any user comment"* **-publishreports** *"FUNCTIONAL, MOBILE_WEBUI, STATS, TESTLOG"* **-stdout** **-swapdatsets** *existing_dataset_file_path:new_dataset_file-path* **-history** *jaeger,testlog,null* **-overridermLabels** *"label name 1,label name 2"*

**Note:**


- The workspace is locked after you issue the command. To check the progress of the test or schedule during the run, invoke another workspace and open the project through that workspace.
- On Linux operating system, the command must start with `cmdline.sh`.
- The command line does not provide a way to specify the secure storage password for resource monitoring. You must provide the password in the workbench and ensure that it is stored and persisted in the schedule before you execute the schedule from the command line.





If a value contains spaces, enclose the value in quotation marks. To see the online help for this command while you are in the directory that contains the `.bat` file, type `cmdline -help`.


The following table explains each option:


-work-space	Required. The complete path to the Eclipse workspace.
-project	Required. The path, including the file name of the project relative to the workspace.
-eclipse-home	Optional. The complete path to the directory that contains <code>eclipse.exe</code> .
-plugins	Optional. The complete path to the folder that contains the plugins. Typically, on Windows operating systems, this folder is located at <code>C:\Program Files\HCL\HCLIMShared\plugins</code> . Required. This option is required only if the folder is at a different location.
-schedule	Optional. However, in a command, it is mandatory to use one of the following options: <ul style="list-style-type: none"> ◦ <code>-suite</code> ◦ <code>-schedule</code> ◦ <code>-servicename</code> <p>You must not use the <code>-schedule</code> option along with the other options. The path includes the file name of the schedule to run relative to the project.</p> <p>Starting from V9.2.1.1, you can execute multiple schedules simultaneously.</p> <p>For example, -schedule <code>sch1:sch2:sch3</code></p>
-suite	Optional. However, in a command, it is mandatory to use one of the following options:

	<ul style="list-style-type: none"> ◦ -suite ◦ -schedule ◦ -servicename <p>You must not use the -suite option along with the other options. The path includes the file name of the suite to run relative to the project.</p> <p>Starting from V9.2.1.1, you can execute multiple tests simultaneously.</p> <p>For example, -suite test1:test2:test3.</p>
-importzip	<p>Optional. To import the project as test assets with dependencies into your workspace, use the -importzip option. This command is available from V9.2.1.1 and later.</p> <p>You can execute test assets from the imported zip file, but you must specify the -importzip {complete path where the zip file is stored on your computer} option along with the -schedule or -suite options.</p> <p>For example, C:\User\Desktop\test1.zip</p>
-varfile	<p>Optional. You can use this option to specify the complete path to the XML file that contains the variable name and value pairs.</p>
-service-name	<p>Optional. However, in a command, it is mandatory to use one of the following options:</p> <ul style="list-style-type: none"> ◦ -suite ◦ -schedule ◦ -servicename <p>You must not use the -servicename option along with the other options. The path includes the file name of the service to run relative to the project. Instead of running a performance test, the specified service is run when it is available.</p>
-serviceargs	<p>Optional. You can use this option to specify a series of arguments to pass to the service specified.</p> <p>For example, -serviceargs "myserviceparm1 myserviceparm1value"</p> <p>The values are in quotation marks as they contain spaces.</p>
-config-file	<p>Optional. You can use this option to specify the complete path to a file that contains the parameters for a test or schedule run. Each parameter must be on a single line. To create a configuration file, you must use an editor that does not wrap lines. Any parameters, whether required or optional, can be set in the configuration file. The command line parameters override the values in this file.</p>

	 Notes: <ul style="list-style-type: none"> ◦ If you are creating a config file manually, the file must be in the UTF-8 format. You must not use quotation marks in this file even for values that contain spaces. ◦ You can create command line config file from the product, which you can use while running tests or schedules from the command-line interface or Maven. For more information about how to create a command line config file from the product, see related links.
-results	<p>Optional. You can use this option to specify the name of the results file. The default result file name is the test or schedule name with a time stamp appended. You must specify a folder name that is relative to the project to store the test results.</p> <p>For example, -results <i>folder/resultname</i></p>
-over-write	<p>Optional. Determines whether a results file with the same name is overwritten. The default value, <i>false</i>, indicates that the new results file is created. If the value is <i>true</i>, the file is overwritten and retains the same file name. You must use double quotes "" for values <i>true</i> or <i>false</i>.</p>
-quiet	<p>Optional. Turns off any message output from the launcher and returns to the command shell when the run or the attempt is complete.</p>
-users	<p>Optional. Overrides the default number of virtual users in the run. For a schedule, the default is the number of users specified in the schedule editor. For a test, the default is one user. This option creates a new copy of the schedule that contains the specified number of users.</p>
-vmargs	<p>Optional. To specify the Java™ maximum heap size for the Java™ process that controls the command line playback, use the -vmargs option with the <i>-Xmx</i> argument.</p> <p>For example, when you use -vmargs <i>-Xmx4096m</i>, specify a maximum heap size of 4096m. This method is similar to specifying <i>-Xmx4096m</i> in the <i>eclipse.ini</i> file for the workbench when playing back the test from the user interface.</p>
-rate	<p>Optional. You can use this argument to specify a rate that you want to achieve for a workload in the Rate Runner group. For example, -rate <i>"Rate Runner Group 1=1/s, 3/m; Rate Runner Group 2=5/s, 10/s"</i>.</p> <p>Here, <i>Rate Runner Group 1</i> is the name of the Rate Runner group that has two stages. The desired rate for the first state is one iteration per second and the rate for the second stage is three iterations per minute.</p>

	 Note: The Rate Runner group name must match with the name in the Rate Schedule.
-duration	<p>Optional. You can use this argument to specify the duration of the stages in the Rate Schedule.</p> <p>For example, -duration <i>Stage1=10s; Stage2=3m</i></p>  Note: The stage number specified must exist in the Rate Schedule.
-publish	<p>Optional. You can use this parameter to publish test results to the HCL OneTest™ Server. You can use the following options along with the -publish parameter:</p> <ul style="list-style-type: none"> ◦ <i>serverURL#project.name=projectName</i>: You can use this option to specify the project name. If the project name is not specified, the name of the current project is used.  Note: If you provide the server and the project details under Window > Preferences > Test > HCL OneTest Server in the product, and if you use <i>serverURL#project.name=projectName</i> along with the -publish parameter, the server details in the command-line interface takes precedence over the product preferences.
	<ul style="list-style-type: none"> ◦ no: You can use this option if you do not want to publish test results after the run. This option is useful if the workbench preferences are set to publish the results, but you do not want to publish them.  Important: You must provide the offline user token for the server by using the HCL_ONETEST_OFFLINE_TOKEN environment variable before you use the -publish parameter in the command-line interface.
-publish_for	<p>Optional. You can use this option to publish the test results based on the completion status of the tests:</p> <ul style="list-style-type: none"> ◦ ALL - This is the default option. You can use this option to publish test results for any text execution verdict. ◦ PASS - You can use this option to publish test results for the tests that have passed. ◦ FAIL - You can use this option to publish test results for the tests that have failed. ◦ ERROR - You can use this option to publish test results for the tests that included errors. ◦ INCONCLUSIVE - You can use this option to publish test results for the tests that were inconclusive. <p>You can add multiple parameters separated by a comma.</p>
-export-log	<p>Optional. You can use this parameter to specify the file directory path to store the exported HTTP test log.</p>

	<p>Starting from V10.0.1, by using the -exportlog parameter, you can provide multiple parameter entries when running multiple tests. You must use a colon to separate the parameter entries.</p> <p>For example: -exportlog <i>c:/logexport.txt:c:/secondlogexport.txt</i></p> <p>If there are multiple -suite parameter entries with a single -exportlog parameter entry, then the -exportlog parameter generates the appropriate number of test logs by appending 0, 1, 2, and so on to the -exportlog parameter entry name.</p> <p>For example: -suite <i>"sampletest1:sampletest2:sampletest3"</i> -exportlog <i>c:/logexport.txt</i> The command generates the following test logs:</p> <ul style="list-style-type: none"> ◦ logexport_0.txt ◦ logexport_1.txt ◦ logexport.txt <p>The last test log generated has the same name as that of the initial -exportlog entry.</p> <p> Note: If there are multiple -suite and -exportlog parameter entries, the number of -suite entries must match with the number of -exportlog entries. Otherwise, the following error message is displayed:</p> <pre>Error, number of -suite and -exportlog entries do not match.</pre>
-export-stats	Optional. You can use this option to export reports in comma-separated values (CSV) format, with the file name derived from the report name. This directory can be relative to the project or a directory on your file system. If the -exportstatreportlist option is not specified, the reports specified on the Export Reports page of the Performance Test Report preferences are exported.
-timerange	Optional. You can use this option along with -exportstats , -exportstatshtml , and -execsummary to export test results within one or more time ranges. The value is the time range that you specify in the schedule. <p>For example, <i>"all, 5 Users,10 Users"</i>. You must separate time ranges with a comma and use double quotation marks (") when there is space in a time range.</p>
-export-statshtml	Optional. When you want to export web analytic results, you can use this option. The results are exported in the specified directory. You can then analyze the results on a web browser without using the test workbench.
-compare	You can use this argument along with -exportstatshtml and -execsummary to export the result in compare mode. The value can be paths to the runs and are relative to the workspace. You must separate the paths by a comma.

-export-statereportlist	<p>Optional. You can use this option to specify a comma-separated list of report IDs along with -exportstats or -exportstatshtml to list the reports that you want to export in place of the default reports, or the reports selected under Preferences. To view this setting, navigate to Window > Preferences > Test > Performance Test Reports > Export Reports.</p> <p>To copy the report IDs list into your command line, navigate to Window > Preferences > Test > Performance Test Reports > Export Reports. Under Select reports to export, select the required reports, and click Copy ID to clipboard. You can then paste the clipboard content on to your command line editor</p>
-exec-summary	<p>Optional. You can use this option to export all of the reports for the test run in a printable format, also known as an executive summary, to the local computer. You must specify the path to store the executive summary.</p>
-exec-summaryreport	<p>Optional. You can use this option to export a specific report as an executive summary for the test run to the local computer. You must specify the ID of the report to export.</p> <p>For example, to export an HTTP performance report, specify <code>http</code>. You must use this option along with -execsummary.</p> <p>To copy the report IDs list into your command line, navigate to Window > Preferences > Test > Performance Test Reports > Export Reports. Under Select reports to export, select the required reports, and click Copy ID to clipboard. You can then paste the clipboard content on to your command line editor</p>
-user-comments	<p>Optional. You can add text within double quotation mark (") to display it in the User Comments row of the report.</p> <p> Note: You can use the file <code>CommandLine.exe</code> to run the command to add comments in a language that might not support Unicode characters on Windows operating system.</p>
-publishreports	<p>Optional. You can use this option to publish test results in HCL OneTest™ Server. The parameters that you can use with it are the following:</p> <ul style="list-style-type: none"> ◦ FUNCTIONAL ◦ MOBILE_WEBUI ◦ STATS ◦ TESTLOG <p>For example, -publishreports "STATS, TESTLOG"</p> <p>You must prefix with "!" to publish all the reports except the specified one.</p> <p>For example, -publishreports "! TESTLOG"</p>

	<p>All the reports except the TESTLOG report is published to HCL OneTest™ Server after executing the command.</p>
-stdout	<p>Optional. You can use this option to display the information about the test or schedule on the command line.</p> <p>After you run a test or schedule from the command line, the following outputs are displayed to give you the overall information of the test or schedule:</p> <ul style="list-style-type: none"> ◦ --VERDICT: The verdict of the test or schedule. ◦ --REMOTE_RESULT: The URL of the result published to HCL OneTest™ Server. ◦ --REMOTE_RESULT_UI: The URL of the result published to HCL OneTest™ Server and can be opened in a browser to analyze the result. ◦ --LOCAL_RESULT: The path of the result saved locally. <p>For example, -workspace <i>workspace_full_path</i> -project <i>proj_rel_path</i> -schedule <i>sched_rel_path</i> -publish<i>publish_url</i> -stdout</p>
-swap-datasets	<p>Optional. Use this option to replace dataset values during a test or schedule. If a test or schedule is associated with a dataset, you can replace the dataset at run time while initiating the run from the command line.</p> <p>You must ensure that both original and new datasets are in the same workspace and have the same column names. You must also include the path to the dataset when you run the -swap-datasets command.</p> <p>For example, -swapidatsets <i>/project_name/ds_path/ds_filename.csv:/project_name/ds_path/new_ds_filename.csv</i></p> <p>You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon.</p> <p>For example, -swapidatsets <i>/project_name1/ds_path/ds_filename.csv:/project_name1/ds_path/new_ds_filename.csv;/project_name2/ds_path/ds_filename.csv:/project_name2/ds_path/new_ds_filename.csv</i></p>
-history	<p>Use this command when you want to view a record of all events that occurred during a <i>test</i> or <i>schedule</i> run. However, you must use the command suffixed with any of the following options:</p> <ul style="list-style-type: none"> ◦ jaeger: To send test logs to the Jaeger UI during the <i>test</i> or <i>schedule</i> run. ◦ testlog: To send test logs as traditional test logs in HCL OneTest™ Performance during the <i>test</i> or <i>schedule</i> run. ◦ null: To send no test logs either to the Jaeger UI or HCL OneTest™ Performance during the <i>test</i> or <i>schedule</i> run. <p>For example:</p>

```
-workspace workspace_full_path -project proj_rel_path
-suite suite_rel_path -stdout -history comma delimited list of modes
```

```
-workspace C:/Users/HCL/hclonetest/test_ws -project Project1
-suite test1.testsuite -stdout -history jaeger
```



Note: You can add multiple options separated by a comma to send test logs during the `test` or `schedule` run to HCL OneTest™ Performance and the Jaeger UI.

For example:

```
-workspace C:/Users/HCL/hclonetest/test_ws -project Project1
-suite test1.testsuite -stdout -history jaeger, testlog
```

For more information about how to view test logs in the Jaeger UI and HCL OneTest™ Performance, see related links.

- override- ermlabels

Optional. By using the `-overriderm1abels` command, you can control the Resource Monitoring sources that are required to collect in a performance schedule during the schedule run.

You can use this command if you want to perform any of the following actions:

- To enable the **Resource Monitoring from Service** option for a performance schedule if the **Resource Monitoring from Service** option is not enabled from the schedule editor in HCL OneTest™ Performance.
- To ignore Resource Monitoring sources that were set in the performance schedule and to change for a label matching mode.
- To replace an existing set of Resource Monitoring labels that were set in the performance schedule and run the schedule with a new set of Resource Monitoring labels.



Note: You must add the Resource Monitoring labels to the Resource Monitoring sources on the **Resource Monitoring** page in your HCL OneTest™ Server project. You can use these labels for adding the Resource Monitoring sources to run the performance schedule through the command line interface.

The command accepts a comma-separated list of labels.

For example, if you have added a label in HCL OneTest™ Server for a Resource Monitoring source as `rm1`, then run the following command to collect data from the source as follows:

```
-workspace workspace_full_path -project proj_rel_path
-suite suite_rel_path -overriderm1abels "rm1"
```



Note: You can add multiple labels to a performance schedule separated by a `comma` to collect data from the multiple sources during the schedule run. For example:



```
-workspace workspace_full_path -project proj_rel_path
-suite suite_rel_path -overridermLabels "rm1,rm2,rm3"
```

If your label contains a `comma (,)`, then when running the `-overridermLabels` command, you must replace the **single comma** with the **double comma** in the label.

For example, if you have added a label to a Resource Monitoring source as `(rm1, test)`, then you must run the following command to collect data from source as follows:

```
-
workspace workspace_full_path -project proj_rel_path -suite suite_rel_path -overridermLabels "rm
1,,test"
```

To stop the test run, you can open another command prompt window and use one of the following options with the cmdline option:

Com- mand	Description
-sto- prun	Optional. Stops the test run after the specified number of seconds. The block is executed, and the test log is transferred before stopping the run. You must use the -workspace command and specify the location of the workspace.
- aban- don- run	Optional. Stops the test run immediately. You must use the -workspace command and specify the location of the workspace.



Note: Messages are displayed to indicate when the test or schedule is launched and when it is completed unless you include the `-quiet` option.

Exemple

Examples of the commands for running tests from the command line

You can run tests from the command line either by using a configuration file or by directly specifying the path of the test in the command. Each command-line option must be followed by an appropriate value.

The contents of a sample configuration file, `config_file1` are as follows:

```
workspace=D:\My Workspace
eclipsehome=C:\Program Files\HCL\HCLOneTest
plugins=C:\Program Files\HCL\HCLIMShared\plugins
project=myProject
suite=mytestsuite
```

To run tests from the command line by using the sample config file `config_file1` you must use the following command:

cmdline -configfile <config file path>

For example:

cmdline -configfile E:\Workspace1\Project1\Tests\config_file1.txt

To run the tests from the command line without using a configuration file, you must specify the path of the tests along with the command as follows:

cmdline <path of the test>

For example:

cmdline -workspace "D:\My Workspace" -eclipsehome "C:\Program Files\HCL\HCLOneTest" -plugins "C:\Program Files\HCL\HCLOneTest\plugins" -project myProject -suite mytestsuite

The -workspace command-line option is followed by a value that contains a space. If the value contains space, then you must enclose the value, D:\My Workspace within quotes. Otherwise, you can provide the value without quotes.

What to do next

After you run the test or schedule, you may want to export the results for further analysis. For more information, see [Exporting report counters automatically on page](#) .

Related information

[Viewing test logs in Jaeger on page 167](#)

[Creating a reference or field reference on page 463](#)

[Viewing test logs on page 799](#)

Controlling cache sizes

If you use an infinite loop and the number of cached responses in a test increases exponentially, you can set a limit to cache for a user group in the schedule.

About this task

When the cache limit is reached, the least-recently accessed cached entry is released to accommodate a new entry. Also, when a test follows another test in the schedule, you can clear the cache before a test starts.

1. To clear the cache before a test starts, from the Test Navigator, open a test.
2. Click the **HTTP Options** tab and select the **Clear page cache when the test starts** check box.
3. To set a limit to the number of cache entries, in the Test Navigator, navigate to a schedule and double-click it to open it.
4. Click the user group for which you want set the cache limit.
5. Click the **Options** tab and then click **Edit Options**.
6. Select the **Set cache size limit** check box and, in the **Maximum cache size** field, type a numeric value.

This value indicates the number of entries allowed for a user.

7. Click **OK** and save the schedule.

Increasing memory allocation

The virtual users that access your web server require memory to prepare requests, send requests, and receive responses. Because the amount of memory is not automatically set on remote computers, you might receive an out-of-memory error. To correct this situation, increase the memory allocation for that computer.

About this task

If you receive an out-of-memory error when you run a test or schedule, override the default amount of memory that is allocated for that computer. To do this, set the RPT_VMARGS property, which overrides RPT_DEFAULT_MEMORY_SIZE. After the first successful execution, HCL OneTest™ Performance automatically sets value for RPT_DEFAULT_MEMORY_SIZE, which represents the maximum heap that will be specified by HCL OneTest™ Performance in subsequent executions.



Note: Ensure there is at least one successful execution after all locations are created so RPT_DEFAULT_MEMORY_SIZE exists.



Tip:


If you see out-of-memory issues, it is a good practice to first check the `javacore*` file. You can also look at the results and verify that the server is responding correctly because many times errors can lead to excessive resource consumption. You can also monitor memory usage with Task Manager or other tools at varying user load levels such as 10, 50, 100, 500 or 1000 users and use that data to make an estimate of the memory needs per virtual user and then project memory requirements for larger user loads. In some cases the best solution is to add another agent.

HCL OneTest™ Performance sets heap size for RPT_DEFAULT_MEMORY_SIZE based on the bit-type of the JRE:

- For 32-bit Java Runtime Environment (JREs), HCL OneTest™ Performance sets 70% of the size of physical memory to RPT_DEFAULT_MEMORY_SIZE. Typically, the maximum limit is set to 1200m.
- For 64-bit JREs, some workloads might perform better with a lesser heap size than 70% of physical memory up to a maximum of 12000m.

To increase the memory allocation on a remote computer:

1. In the Test Navigator (from your local computer), expand the project until you find the deployment location that you want to change.

Deployment locations are represented by the  icon.

2. Right-click the deployment location, and then click **Open**.
3. Under **Property Groups**, click the **General Properties** link, and then click **Add**.

4. In the New Property window:

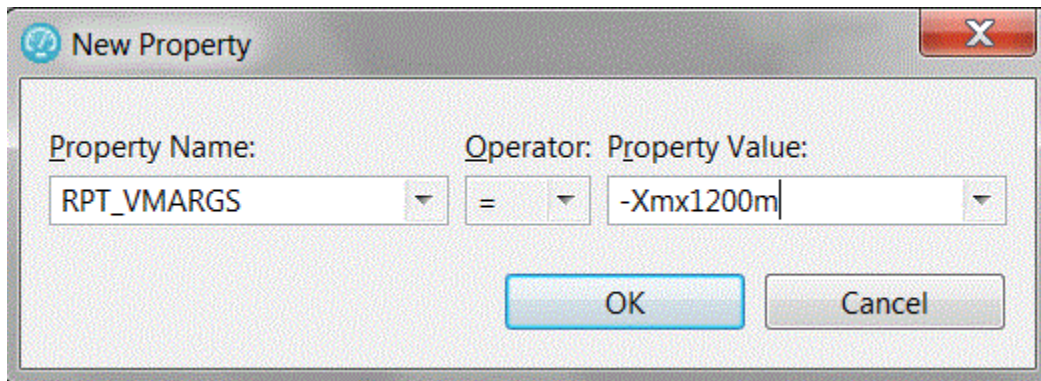
- a. In the **Property Name** field, type `RPT_VMARGS`.
- b. In the **Operator** field, confirm that the operator is `=`.
- c. In the **Property Value** field, type `-Xmxnnnm`, where `nnnn` is the amount of memory, in megabytes, and then click **OK**.

Example

If you need to set multiple `RPT_VMARGS` values for a location, place them in the same property entry and separate them with a space. Do not use multiple property entries to set multiple `RPT_VMARGS` values for a location.

Result

The following **New Property** window sets maximum heap to 1200 megabytes:



i Tip: It is a good practice is to monitor memory usage with Task Manager or other tools at varying user load levels such as 10, 50, 100, 500 or 1000 users and use that data to make an estimate of the memory needs per virtual user and then project memory requirements for larger user loads. In some cases the best solution is to add another agent.

What to do next

If you have increased the available memory and you still receive out-of-memory errors, add more remote computers for your user groups. For information about how to do this, see [Running a user group at a remote location on page 538](#).

Controlling the test runs from web analytics report

When a schedule is in the running state, you can perform the following actions from the web analytics reports to cater to your need.

Changing the number of virtual users during a run

If the number of virtual users that was defined initially is incorrect, you can correct the number of users, and apply the change to one stage or to all of the remaining stages in the schedule.

1. From the test editor, click **Run Test**. The execution report is opened either in the internal or external browser, and the state of the run is shown as Running.
2. During the run, click **Running > Change Users Number**.
3. In **Change number of users** dialog box, select one of the following options:

Option	Description
Add users	<p>If a schedule contains only percentage groups, the virtual users are added by proportion. For example, assume that your schedule contains three user groups, assigned at 20%, 30%, and 50%. If you add 10 virtual users, two added are to the first group, three are added to the second group, and five are added to the third group.</p> <p>If a schedule contains both absolute and percentage groups, the absolute groups are assigned first. For example, assume that your schedule has one user group that is fixed at 10 users, and only one virtual user is running in that group. You add 100 virtual users. Nine virtual users are added to the absolute group, and the remaining virtual users are apportioned among the percentage groups.</p>
Remove users	Users are removed proportionately from user groups according to each user group's percentage value. The time at which users are asked to stop is controlled by the Change rate that you set in the schedule.
Apply to all remaining stages	Click to apply the change to all remaining stages in the schedule run, and clear to apply the change only to the current stage.

4. Click **Change**.
The number of users is changed for the specified stages.

Changing the rate during a run

Typically, you specify the desired rate for the Rate Runner group in the Rate Schedule. However, due to various reasons, you might want to change the rate when the Rate Schedule is running.

About this task

For example, you apply the load iteratively. Therefore, in the Rate Schedule, you specify a moderate rate. If the run is progressing steadily and there is scope to apply more load, you can increase the rate. When you change the rate, the

changed rate is applied to the remainder of the current stage. The next stage will automatically select the rate defined in the schedule editor.

To change the rate during a run:

1. From the Test editor, click **Run Rate Schedule** . The execution report is opened either in the internal or external browser, and the state of the run is shown as Running.
2. During the run, click **Running > Change Rate**.

Result

The **Change Rate** dialog box displays all the Rate Runner groups that are defined in the Rate Schedule.

3. To specify another rate for the Rate Runner group, in **Target Rate**, specify the rate.

Releasing virtual users from synchronization points

To record the response time of the system under test at different points, you can release virtual users either all together or in staggered intervals. You can release virtual users from synchronization points during a run.

Before you begin

Add synchronization points to a schedule. See [Synchronizing users on page 540](#)

1. From the test editor, click **Run Test**. The execution report is opened either in the internal or external browser, and the state of the run is shown as Running.
2. During the run, click **Running > Manage Synchronization Points**.
3. Select the synchronization points to release, and click **Release**.



Note: Typically, the virtual user wait time is based on the time that the last user arrives at the synchronization point. However, if a virtual user arrives after you manually release a synchronization point, the user wait time is instead based on the time at which the synchronization point was released.

Changing the stage duration during a run

Increase or decrease the duration of the current stage during a schedule run. You can change the duration of a stage that is set to run for a specified time, but not for a stage that is set to run until the work is complete. You can change the duration of a stage when the status is `Running`, but not when the status is `Ramping Or Settle Time`.

Before you begin

1. From the test editor, click **Run Test**. The execution report is opened either in the internal or external browser, and the state of the run is shown as Running.
2. During the run, click **Running > Change current stage duration**.
3. In the **Change Stage Duration** window, type the new stage duration in **Run for specified period of time**. Type a value that is longer than the amount of time that has elapsed for the current stage. Use the list to change the time units.

4. Click **Change**.

The duration of the current stage changes.

Changing the log level during a run

You can change the log level to determine problems during a run. By default, only warnings and severe errors are logged. Typically, you change this level only when requested to do so by IBM® Software Support.

About this task

Although the test log provides general information about problems that occur during a run, you might need to investigate certain problems further by examining a detailed trace of the run. In general, change the problem determination level only when asked to by technical support. However, under certain conditions, you yourself might want to change the problem determination level. For example, if problems occur when a run reaches a certain number of users, you might increase the level to **Config**, which is the most detailed level that you will generally use.

To change the log level during a run:

1. From the test editor, click **Run Test**. The execution report launches either in the internal or external browser, and the state of the run is shown as Running.
2. During the run, click **Running > Change Log Level**.
3. To change the log level, select any of the following options:

Option	Description
All, Finest, Finer, Fine	Set these options only if you are requested to do so by technical support.
Config	Logs static configuration messages. Configuration messages, which include hardware specifications or system profiles, require no corrective action.
Info	Logs informational messages. Informational messages, which include system state, require no corrective action.
Warning	Logs warning messages. This is the default setting. Warning messages, which might indicate potential problems, require no corrective action.
Severe	Logs critical and unrecoverable errors. Critical and unrecoverable messages interrupt normal program execution, and require corrective action.
None	Turns logging off.

4. Click **Change**.

Results

When a user group runs on your local computer, the problem determination logs are in the `deployment_root` directory in your workspace. When a user group runs at a remote location, which is the typical use case, each remote location has a deployment directory, which you define and which is listed on the Locations page for that user group.



Note: Common Base Event XML logs from remote locations use Universal Coordinated Time (UTC), also called Zulu time, which is likely different from your local time. For example, Eastern Standard Time (EST) is 5 hours behind UTC time.

What to do next

To view the problem determination log, open the log file in an XML editor. Select the log whose timestamp matches that of the problem run. The most recent log has the suffix `00.log`.

Stopping test runs

You can stop a test run before it is complete. For example, you might detect serious problems in the run and not want to wait for it to finish. When stopping the test run, you can choose to save the results and the test log of the run.

About this task

To stop a test run on the local computer, use this command `cmdline.bat -stoprun -workspace "c:\myWorkspace"`.

To stop a test run from a remote computer, you can send the REST API command HTTP POST to `http://hostNameOrIP:7878/executioncontrol/stoptestrun` with POST data `{\"btnExecFinally\":true, \"btnResultCollection\":true, \"timeout\":30, \"timeoutScale\": \"sec\"}`.

For instance, you can use Curl to send the POST command.

```
curl "http://hostNameOrIP:7878/executioncontrol/stoptestrun" -d {"btnExecFinally":true, \"btnResultCollection\":true, \"timeout\":30, \"timeoutScale\": \"sec\"}
```

Table 6. POST arguments

Argument	Description
<code>btnExecFinally</code>	Indicates whether to run the Finally block in the schedule.
<code>btnResultCollection</code>	Indicates whether to collect the results for the test.
<code>timeoutScale</code>	Indicates the units of time such as "milli", "sec", "min", "hour".

If you do not require test results and logs, abandon the test by clicking **Running > Abandon Test Run** or by using the command `cmdline.bat -abandonrun -workspace "c:\myWorkspace"`.

1. During a test run, click **Running > Stop Test Run**.
2. In the **Timeout** field, type a number and select a time unit.

If you are running HTTP tests and want the results, consider selecting a duration that is long enough for a page to return.

3. **Optional:** To collect the results and the test log until the time the test ran, select **Collect test results and history**. Typically, the partially run report is useful for debugging specific issues. If you do not select the check box, the report is not generated.
4. **Optional:** To stop the execution of the test in the Finally block, clear **Execute Finally block**.
5. Click **Stop**.

The test stops after the timeout.

Debugging HTTP tests

If a test does not behave as expected during playback, you can use the protocol data and test log to assist in debugging the test.

HTTP debugging overview

If a test is not behaving as expected, you can use the **Protocol Data** view to debug the test. This view can be useful after you record a test, after you make changes to an existing test by adding datasets or data correlation, or after you make changes to the system under test.

The **Protocol Data** view displays data that was recorded or played back. The **Protocol Data** view can also display data in real time as tests and schedules run. To see recorded data, click a test element in the test editor. To see data played back after running tests, click an element in the test log. The **Protocol Data** view updates accordingly. Substituted data is highlighted on the **Request**, **Response Headers** and **Response Content** pages when you view test log or test editor elements that use data correlation. The **Protocol Data** view contains these pages:

Request

This page displays all request data sent to a server. This includes the URL of the requested resource, request headers, cookies, and form data.

Response Headers

This page displays the response status line and header fields that are received from a server.

Response Contents

This page displays the response contents, in text form, that is received from a server.

Browser

This page attempts to display the response contents as a web browser would display the contents. Because the protocol data is used instead of data from a live web server, playback might be successful even though the **Browser** page might not render the contents exactly as a web browser would. Pages that might not display correctly include those that have resources cached on a web server and those that use Javascript and framesets extensively.

Event Log

This page displays event summary information for each HTTP page in the test. After a run, when you select a particular user from the test log, the page displays information about that user.

During a run, the page displays real-time information if real-time protocol data support is enabled. To enable this support, click **Window > Preferences > Test > HTTP Protocol Data View**, and then select **Enable real-time protocol data support**.

Debugging HTTP Data Correlation Errors

Data correlation errors are very common and sometimes complex to fix. A dedicated Eclipse-based perspective can guide you to debug data correlation errors.

About this task

Using the perspective, you can fix the following data correlation issues:

- Missing reference for substitutions for the substitution sites. The product tries to locate values greater than seven characters in length and is alphanumeric.
- Missing substitution site in HTTP headers for values greater than seven character in length and is alphanumeric. The value to be substituted will be highlighted.
- Invalid regular expression.

The content in the different views of the perspective is connected to each other. So, when you select a request in the Test view, the log event corresponding to that request is displayed in the Test Log view. The Compare tab in the Protocol Data view displays the comparison between the selected request in the Test and the Log views. The Event Log selects the proper page event. The behavior is same if you select an event in the Test Log. The Test, the Event Log, and the Compare tab will update themselves to match that event. All tabs work in conjunction with each other.

1. In the Test Navigator view, right-click a result that contains data correlation errors and click **Debug HTTP Data Correlation Errors**.

Result

The HTTP Test Debug perspective displays the first error.

2. Use the **Compare** tab in the Protocol Data view to compare the requests or responses of the Test and Test Log.
3. Use the **Problems** tab to view the recommended fix. Click **Fix** to fix the error. You can navigate between the errors by clicking **Previous** and **Next**.

Watching a virtual user during a test run

The **Browser** page in the **Protocol Data** view displays browser results during an individual HTTP test run. The **Browser** page displays each page in real time.

Before you begin

Verify that real-time support is enabled. Click **Window > Preferences > Test > HTTP Protocol Data View**

1. Verify that **Enable real-time protocol data support** is selected.
2. Select which page you want the Protocol Data view to display by default. The Browser page renders the visual data, and the Event Log page displays a table of page titles, response times, verification point failures, and unexpected response codes. You can switch between the pages when you watch the user.

1. In the Test Navigator, browse to an HTTP test and double-click it.

Result

The test opens.

2. Click **Run > Run As > Test**.
3. In the **Protocol Data** view, click the **Browser** tab.

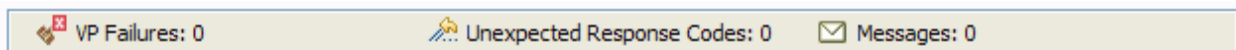
Result

The HTTP pages are displayed. Because the protocol data is used, the **Browser** page might not render the contents exactly as a web browser would render the contents. But even if an HTTP page does not display correctly, you can still use the information on the **Request**, **Response Headers**, **Response Content**, and **Event Log** pages to help you debug the test.

4. Use the playback buttons in the **Protocol Data** view to control the real-time browser display. The playback buttons control only the **Protocol Data** view. They do not pause or stop test execution. If you pause the real-time browser display and then click **Play**, the page displayed in the Browser window is the next page loaded by the playback engine. Use the **Back** and **Forward** buttons to control the display of pages that have already been loaded. If a problem occurs during playback, pause the test and use the **Back** and **Forward** buttons to move to the page where the problem occurred. The **Request**, **Response Headers**, and **Response Content** pages show the information for the primary request.

Results

The information on each page is updated in real time as you run the test. The **Event Log** page shows the response time, verification point failures, unexpected response codes, and messages for each request. All of the other pages in the Protocol Data view display an Event Log summary bar while tests are running. The Event Log summary bar shows the number of verification point failures, unexpected response codes, and messages for the current page request.



An unexpected response code is defined as a response code not in the 200 or 300 range. Response codes outside the 200 and 300 range that were recorded or that are represented in a response code verification point are not considered unexpected. For example, a response code of 404, "Not Found," is an unexpected response code, unless there is an associated response code verification point. If a 404 response code occurred while recording and then during playback the response code is also 404, this is not considered unexpected. Messages include data correlation failures and custom code messages.

All pages in the **Protocol Data** view are active and updated while a test is running. Any of the events in the Event Log summary bar might indicate playback failures that require further investigation. After a test runs, you can also view the test log to debug the test. To learn more about the test log, see [Viewing test logs on page](#) .

Watching a virtual user during a schedule run


During a schedule run, you can select any virtual user and watch that user's real-time browser. Watching the real-time browser lets you investigate the status of individual virtual users while they are running. You can determine whether a run is valid despite the occurrence of individual virtual user failures.

About this task

Watching a virtual user during a run is useful in the following situations:

- It enables you to quickly verify that virtual users are receiving the data that you expect. Assume that you want to verify that virtual users use different dataset data during a schedule run. You can select a virtual user during the run and see the real-time data for that user. You can then select another virtual user and see that user's data.
- It confirms whether a schedule run is valid despite the occurrence of failures. Assume that you are running a schedule that takes several hours to complete. Intermittent failures occur during the run. To find out whether these failures are significant enough to interrupt the run, you can investigate the activities of virtual users that are running at each location (agent computer). You can decide whether to allow the schedule to run to completion.
- It enables you to investigate errors during a run. Assume that you have confirmed that a schedule is running successfully, despite the errors of a few virtual users. However, you want to find the specific virtual users with verification point failures and see what each virtual user is doing when the failure occurs. To do so, start monitoring different virtual users until you find one who is experiencing failures. The Verification Point report is helpful for narrowing down which user groups are experiencing failures. When the virtual user has been identified and monitoring has started, use the Event Log page to see the error details for this user.

To watch a virtual user during a schedule run:

1. Run the schedule: Right-click the schedule, and then click **Run > Run Schedule**.
2. In the Protocol Data view, click  (Watch Virtual User icon).
3. In the **Select Virtual User to Watch** window, select the user group that contains the user.
4. Select the virtual user number in the user group or at a particular location (agent) on which the user group runs, and click **OK**.







Option	Description
Specify a user inside this user group	<p>The window lists the ranges of active users in the group. Specify a user number within the listed ranges.</p> <p>This option is useful when you want to watch any user, or a particular user, in a specific user group. This might be the case if you have run the schedule before, examined the test log, and know the number of the virtual user that you want to see.</p>

Option	Description
	For example, a schedule that uses a dataset of user names might run correctly for the first 10 users but issue verification point failures for the remaining users. In this case, you watch user number 11.
Indexed user on location	<p>Select this option to see a user running at a specific location. Enter an index between 1 and the number of users at that location. This number is mapped to a virtual user number, which is displayed in the title of the Event Log tab.</p> <p>You enter an indexed number rather than a specific user number because not only are the actual user numbers spread out between locations (User 1 might be at Location A, Users 2 and 3 at Location B, and so on) but also a user's location can vary from run to run (User 3 might run on Location A during one run and Location B during the next run).</p>

5. In the Protocol Data view, click the **Browser** tab.

Only pages loaded while watching the virtual user are displayed; pages that have been loaded before live rendering began are not available. However, the data that you see is also available in the test log after the run completes. This data is always available, regardless of the test log preferences that you set in the schedule.

6. Click an icon for the virtual user.

Icon	Description
 (Pause)  (Play)  (Back)  (Forward)	Navigation actions, which let you move among pages.
 (Stop Watching)	Stop monitoring the current user. Clicking this icon does not stop the user from running.
 (Watch Virtual User)	Change from one user to another.

You can even add virtual users during the run and watch the added users. For more information, see [Changing the number of virtual users during a run on page 639](#).

Replaying a virtual user after a run completes

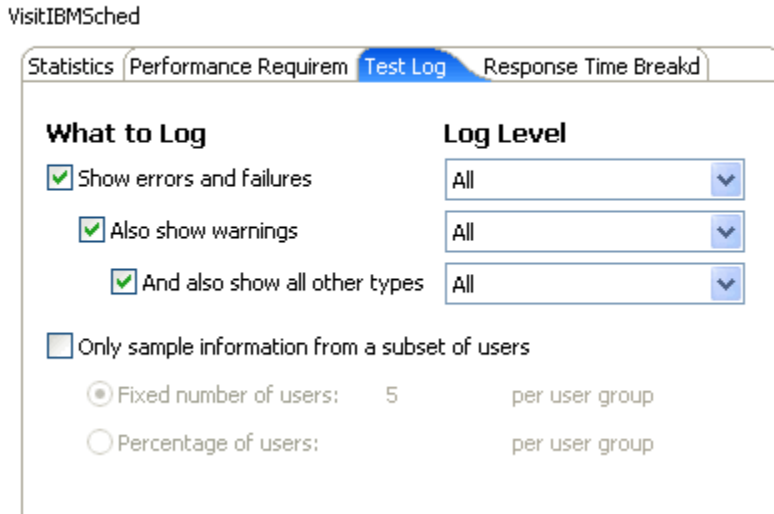
After you create a schedule, run it with a small number of users and watch their activity to verify that the schedule is behaving as expected. You can compare the visual data that is rendered as well as the events that are logged for each user.



1. Open the schedule, click the **Test Log** tab, and set each **Log Level** to **All**.

This setting makes sure that the virtual user that you select will have complete test log data.

Example

Schedule Element Details



2. Verify that the schedule contains a small number of virtual users.
 - a. Open the schedule, click the **User Load** tab and set a small number of users.
Setting a small number of users prevents the log from becoming unwieldy, which can occur when the log level is **All**.
 - b. If you are running a fixed number of users, select each user group and set the numbers so that the total is equal to that in the **User Load** tab.
3. Save the schedule and click **Run** to run the schedule with the limited number of users.
4. After the schedule completes, open the test log: Right-click the schedule run, and select **Display test log**.
5. In the test log, click the **Events** tab, expand the test log to display the virtual users, and select a virtual user to watch.
6. In the Protocol Data view, click **Replay** ().
The virtual user's pages, which were loaded during the run, are redisplayed. Each page pauses the number of seconds that you set in the **Replay Delay** preference (**Window > Preferences > Test > HTTP Protocol Data View**).
7. To stop replaying, click **Stop Replay** (.

Viewing the playback summary of a virtual user

The **Event Log** page in the **Protocol Data** view provides a summary of what happened during an HTTP test run. A complete summary is always available for a test. The information available for a schedule depends on its logging level.

About this task

The amount of information that you see in the Event Log depends on the amount of information that is collected by the test log. When you run a test, all logging is automatically enabled, therefore the Event Log contains complete details. When you run a schedule, however, you select a **Log Level** setting, and the amount of detail that the Event Log contains depends on this setting. For more information, see [Setting the data that the test log collects on page 599](#).

If your schedule run is not large, select the setting **All** for all types of events. For large schedule runs, do not use the **All** setting, because of the time and space required to transfer large amounts of log detail to your computer after a run is completed. Instead, define a special schedule for debugging tests with the log levels set to **All**. Limit this schedule to a single test (or a small number of tests) and to a small number of user groups or loop iterations. After you are satisfied that a test is performing correctly, you can move it to a schedule that emulates a realistic workload.

To view the **Event Log** information:

1. In the Test Navigator, open a schedule or test.
2. Run the schedule or test: Click **Run > Run As**, and select **Performance Test** or **Performance Schedule**.
3. In the **Protocol Data** view, click the **Event Log** tab.
 - If you are running a test, the Event Log page is immediately populated.
 - If you are running a schedule, wait until the run is complete. Right-click on the report and select **Display test log**. Select a virtual user in the test log. The Event Log page shows the summary of events for that user.

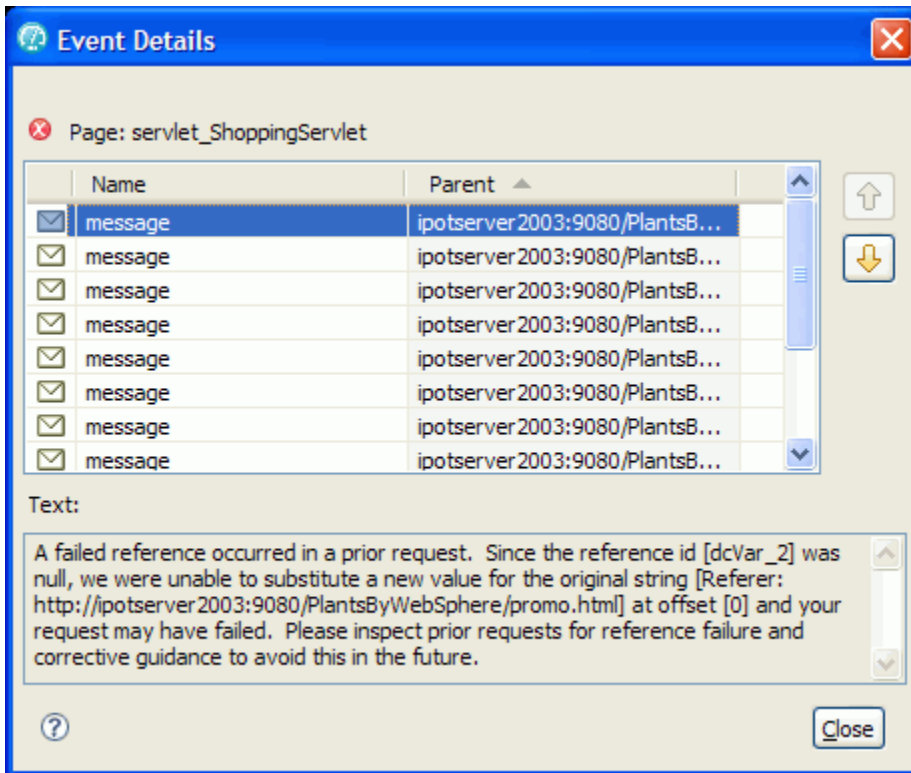
Result

The first column has an arrow to indicate the page that is currently displayed in the other **Protocol Data** view windows: **Request**, **Response Headers**, **Response Content**, and **Browser**.

The second column shows the status of the page:

Symbol	Meaning
Green check	All requests for the page are successful and that there are no warnings.
Red X	The page contains requests with failed verdicts or unexpected response codes. Typically, failed verdicts are verification point failures. Unexpected response codes are codes outside the 200 and 300 range that were not recorded as such or that are not represented in associated verification points.
Yellow warning symbol	The page contains messages but no failure verdicts or unexpected response codes.

4. Double-click an entry in the **Event Log** to open the **Event Details** window. The **Event Details** window shows a list of events for the page that is highlighted in the **Event Log**. The **Name** and **Parent** columns show the name and parent request for each event. The **Text** field shows details for each event.

Result

Note: The **Go To** menu at the top of the **Event Details** window lets you jump to the corresponding event in the test or the test log. This is useful for obtaining more contextual information about the error shown in the table.

5. Leave the **Event Details** window open, and click other pages in the **Event Log**, or use the **Forward** and **Back** buttons to navigate to other pages in the **Event Log**.

Result

The **Event Details** window is updated to show the events for the page that is selected in the **Event Log**.

What to do next

The information in the **Event Details** window supplements the information in the test log. To see more information about a particular event, view the test log after the test finishes running. With the test log, you can see an event in context. The test log also provides links to the recorded test.

Inspecting HTTP test logs in the Protocol Data view

To verify that a test is performing as you intend, use the Protocol Data view, which displays the HTML details that were generated during a schedule run. If problems occur in a test run, you can also compare the data retrieved during the run with the recorded data.

Before you begin

Set the detail level. The amount of detail that you can see in the Protocol Data view depends on the **Test Log** settings in the schedule; see [Setting the data that the test log collects on page 599](#). If you plan to use the Protocol Data view and your schedule run is not large, select the setting **All** for all types of events. For large schedule runs, do not use the **All** setting, because of the time and space required to transfer large amounts of log detail to your computer after a run is completed. You might create a special schedule for Protocol Data view runs with the log levels set to **All**. Limit this schedule to a single test (or a small number of tests) and to a small number of user groups or loop iterations. After you are satisfied that a test is performing correctly, you can move it to a schedule that emulates a realistic workload.

1. In the Test Navigator, right-click the results that you want to inspect, and click **Display Test Log**. The results have the same name as the test or schedule with a time stamp appended.
2. In the editing area, click the **Protocol Data** tab to open the view.



Tip: If you cannot locate the **Protocol Data** tab, click **Window > Show View > Protocol Data**.

3. In the Protocol Data view, click the tab for the type of contents or view that you want to display: **Request**, **Response Headers**, **Response Contents**, or **Browser**, or **Event Log**. Substituted data is highlighted on the **Request**, **Response Headers** and **Response Content** pages when you view test log or test editor elements that use data correlation.
4. In the Events hierarchy area of the test log, click the line that contains the detail that you want to view.

Result

The detail is displayed in the Protocol Data view.



Note: User groups, virtual users, and some HTTP requests are processed in parallel. The test log reflects the order of execution, rather than the recording order or the order of user groups in schedules. Therefore, the order of page requests in the test might be different from the order in the test log, and the order of user groups in the test log might be different from the order in the schedule. However, the order of pages in a test and the order of tests inside a user group are the same in the test log as in the corresponding test and schedule.

What to do next

If you have problems during playback, you can compare the data that you recorded with the data retrieved during the run. For information on displaying the recorded data, see [Viewing a test in the Protocol Data view on page 309](#).

Managing HTTP information in the Protocol Data view

Several actions can be performed on the text displayed in the **Protocol Data** view. You can save the text from the **Protocol Data** view pages to a text file for use in other applications. You can search for text in the protocol data. You can compare the data retrieved during a run with the recorded data from an HTTP test. Typically, you compare a request or response from the test log to its corresponding data in the recorded test. You can also compare requests or responses from different virtual users in one test log.

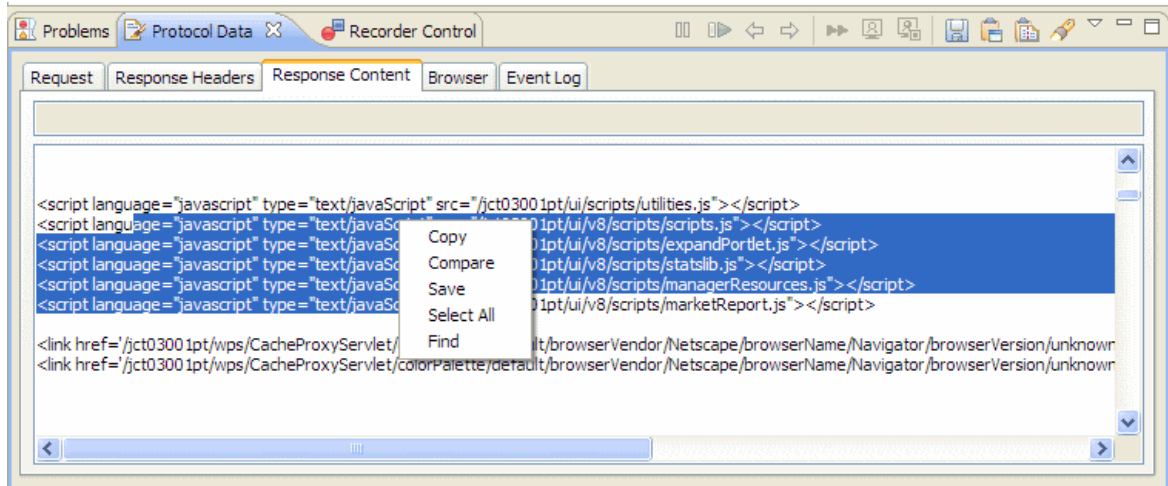
To export HTTP data from the Protocol Data view to a text file:

1. In the Test Navigator, right-click the results that you want to export, and click **Display Test Log**. The results have the same name as the test or schedule with a timestamp appended.
2. In the editing area, click the **Protocol Data** tab to open the view.
3. Click the button that corresponds to the data that you want to export: **Request**, **Request Headers**, or **Response Content**.

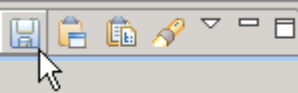


You cannot export data from the **Browser** page.


4. Optional:

To export only a portion of the text, select the text, and click the appropriate option from the menu:



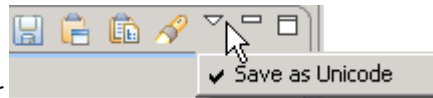
5. Select one of the following options from the toolbar:

Option	Description
 <p data-bbox="444 1325 675 1350">Save to protocol data</p>	<p data-bbox="855 1188 1406 1255">Saves the entire text on the page to a text file within or outside of your workspace.</p>
 <p data-bbox="464 1514 656 1539">Copy to clipboard</p>	<p data-bbox="855 1377 1390 1402">Copies the entire text on the page to the clipboard.</p>
 <p data-bbox="444 1696 675 1722">Compare to clipboard</p>	<p data-bbox="855 1566 1419 1839">Compares the entire text on the page to the contents of the clipboard. To compare the playback data with the recorded data, view the protocol data in the test log and select Copy to clipboard. Then, use the element link in the details section of the test log to navigate directly to the associated recorded element, and select Compare to clipboard.</p>

Option	Description
 <p data-bbox="532 394 586 422">Find</p>	<p data-bbox="857 268 1401 338">Searches the data in that test log page. You cannot replace data in the test log.</p>

Result

The text is saved in the default (locale-specific) encoding that is set on your computer. Keep this default locale setting unless your text contains non-Latin characters.



6. **Optional:** Click the menu on the toolbar to save the file in Unicode.

Debugging Citrix tests

The Citrix dashboard is an optional panel that displays detailed information and control commands for each virtual user during the run of a schedule. This is useful for debugging your tests and allows you to pause, interact, resume, or stop the execution of individual virtual user sessions.

Enabling and disabling the Citrix monitoring panel

With the optional Citrix monitoring panel, you can monitor detailed information during the run of a schedule and debug your tests. To use the Citrix monitoring panel, you must enable it in the schedule.

Before you begin

The Citrix monitoring panel is available only during the run of a schedule. Enabling the option requires that you create a location for it in the project.

The Citrix monitoring panel uses resources; therefore use the monitoring panel only for debugging and test development. For actual performance testing, disable the panel.

To enable the Citrix monitoring panel in a schedule:

1. Open a schedule that contains a user group with at least one Citrix test.
2. Select the user group, and on the **Schedule Element Details** pane, click the **Options** tab, and then click **Edit Options**.

Result

The **Protocol-specific Options** window opens.

3. If several types of tests are available in the user group, click the **Citrix Options** tab.
4. Select **Enable monitoring panel**.
5. **Optional:** Select **Enable log file generation** if you want to keep a log file of the debug session.

After the run, you can locate the log file in the file system, in the `deployment_root` directory of the workspace directory.
6. Click **OK**, and save the schedule.

Results

During the next run of the schedule, the Citrix session window displays the monitoring panel, with which you can debug and control the progress of each virtual tester.

Related reference

[Citrix monitoring panel reference on page 1189](#)

Related information

[Debugging tests with the Citrix monitoring panel on page 655](#)

Debugging tests with the Citrix monitoring panel

When enabled, the Citrix monitoring panel provides the ability to pause your tests during a run and to provide manual input. It also provides a comparison view to compare expected window events with the actual window events received during the test. This capability can be useful when you are debugging your tests.

Before you begin

To display the Citrix monitoring panel during VU Schedule runs, first enable the panel. See [Enabling and disabling the Citrix monitoring panel on page 654](#) for more information.

The Citrix monitoring panel uses resources; therefore, use the panel only for debugging and test development. For actual performance testing, disable the panel so that the results are accurate.

1. With the monitoring panel option enabled, run the VU Schedule.
A window opens with the virtual users running Citrix clients on multiple pages. Click a tab to display the corresponding virtual user.
2. Click the **Monitoring** tab to display the monitoring panel.
The monitoring panel displays the following information:

Monitoring Panel

This panel displays information about the execution of each virtual user.

Pool Name

Displays the name of the virtual user pool. There is one pool per location and user group.

Active Virtual Users

Displays the number of virtual users currently active. This value is updated permanently during the run.

User Action Rate

Displays the number of Citrix user key or mouse actions that were simulated during the last 5 second interval.

Total Elapsed Time

Displays the total time elapsed since the start of the schedule run.

Current® Action

Displays the last user action executed in the test.

Timeouts

Displays the number of synchronization timeouts for the virtual user. The color represents the status of the timeout:

- Green: ok.
- Yellow: a timeout occurred on a conditional synchronization.
- Red: a timeout occurred on a mandatory synchronization.

Elapsed Time

Displays the time elapsed since the start of the virtual user run.

Status

Displays the execution status of the virtual user.

3. Select a virtual user and click one of the following buttons to interact with the execution of the test.

Go To

Click to display the Citrix session of the selected virtual user.

Pause or Play

Click to pause or resume the execution of the selected virtual user. You can also pause the execution by setting breakpoints in the test.

Step

When the test is on pause, click to execute each user input action in the test, step by step. To pause test execution, you can either click the **Pause** button or set breakpoints in the test. Click **Play** to resume the test.

Interact

When the test is on pause, click to allow manual actions in the virtual user session. Use this feature if a test fails to synchronize or gets stuck in an unexpected state. To pause test execution, you can either click the **Pause** button or set breakpoints in the test. Click **Play** again to resume the test execution at the point where it was paused.

Stop

Click to stop the execution of the selected virtual user. When all virtual users are stopped, the schedule ends.

4. When you have finished interacting with the Citrix session, click **Play** to resume the execution.

Related reference

[Citrix monitoring panel reference on page 1189](#)

Related information

[Enabling and disabling the Citrix monitoring panel on page 654](#)

Setting Citrix breakpoints

When the Citrix monitoring panel is enabled, you can define breakpoints in the test to pause the running of a user-input action. This capability is useful when you are debugging a Citrix test.

Before you begin

For the breakpoints to have any effect, the test must be running in a schedule with the Citrix monitoring panel enabled. See [Enabling and disabling the Citrix monitoring panel on page 654](#) for more information.

Breakpoints can be defined on these user-input test elements: mouse actions, key actions, text inputs, and logoff elements. When a breakpoint is encountered, the test pauses before the user input element is run.

1. Open a Citrix test in the test editor and select a user input element.
2. In the **Test Element Details** area, select **Stop test execution on this element when the monitoring panel is enabled**.

Alternatively, you can right-click the user input element, and select **Toggle Breakpoint**.

3. Add the test to a schedule and run the schedule with the monitoring dashboard enabled.

Result

During the run, the virtual user pauses at the breakpoint.

4. When the test is paused, click **Interact** to perform manual actions inside the Citrix session or **Step** to run the test step by step.
5. When you have finished, click **Play** to resume the test run.

If there are multiple breakpoints in the test, the run resumes until the next breakpoint is encountered.

Extending test execution with custom code

You can extend how you run your tests by writing custom Java™ code and calling the code from the test. You can also specify that results from the tests that are affected by your custom code be included in reports.

Creating custom Java™ code

Custom code uses references in the test as input and returns modified values to the test. Use the `ICustomCode2` interface to create custom code and the `ITestExecutionServices` interface to extend test execution. These interfaces are contained in the `com.ibm.rational.test.lt.kernel.services` package.

About this task



Note: When you use the `ITestExecutionServices` interface in your custom code to report test results, the results for the custom code are displayed in the test log. If you log custom verification point verdicts, these are reflected in the overall schedule verdict.

Custom code input values can be located in references or field references. You can also pass a text string as an argument to custom code. References that are used as input to custom code must be included in the same test as the custom code. In the test, the reference must precede the code that it affects. Verify that the test contains the references that are required for customized inputs to your code. For details about creating references and field references, see [Creating a reference or field reference on page 463](#).

If your custom code uses external JAR files, you might need to change the Java™ build path. In some cases, you can avoid changing the build path manually by running the test before adding your custom code to it. The first time a test runs, classes and libraries that are required for compilation are added to the build path. For example, you can import Test and Performance Tools Platform (TPTP) classes that are required to create custom events in the test log if the test, to which you have added your custom code, has run previously. However, if the test has never been run, import errors occur because the classes are not named in the build path for the project until the test has run.

If your code uses external resources, for example, an SQL database or a product that manages customer relationships, you must configure the custom code to work on every computer on which your test runs.

Custom code is saved in the `src` folder of the project that contains the test that calls the code. By default, custom code is located in a package named `test` in the `src` folder.

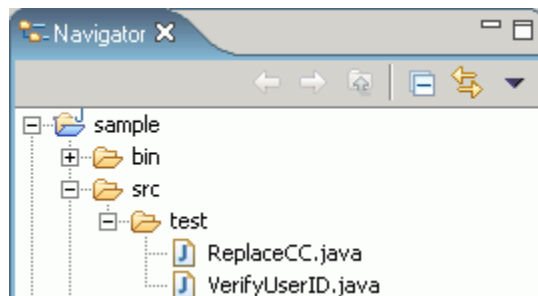
You can reuse a custom code package for tests that are located in multiple projects. The projects must be in one workspace. To reuse custom code across projects, use the project name before the custom code package. For

Test Element Details

Class name:

example,

The following example shows the standard Navigator view of two custom code classes. (The Test Navigator does not display Java™ source files.)



When you add the `ReplaceCC.java` and `VerifyYUserID.java` custom code classes to the test and return a value to the test, **Substitute** lists these two classes.

The test package also contains the generated Java™ code for tests in the project.

You can put custom code in a different package (for example, `custom`). Separate custom code from generated code, especially if you use a source-control system.

To add custom code:

1. Open the test, and select a test element.
2. Click **Add** or **Insert**, and select **Custom Code**.

Add appends the custom code to the bottom of the selected test element. **Insert** adds the custom code above the selected test element.



Note: After you add or insert custom code, the Problems view displays an error stating that the new custom code element has no Java™ file. This error message remains until you click **View Code** or **Generate Code**, to remind you that the custom code test element is not yet associated with any Java™ code.

3. Inspect the **Class name** field, and complete one of these steps:
 - If the code to call already exists, change the class name to match its name. Click **View Code** to open the code in the Java™ editor.
 - If the code does not exist, change the class name to describe the purpose of the code. Click **Generate Code** to generate a template class for logging results and to open it in the Java™ editor. If a class with this name exists, you are warned that it will be overwritten.
4. In the **Arguments** field, click **Add**.
5. In the **Custom Code** window, select all inputs that your code requires.

The **Custom Code** window lists all values in the test that can be used as inputs to your code (references or field references in the test that precede the code).

6. Click **OK**.

Result

The window closes, the selected references are added to the **Arguments** field.

7. To add text strings as inputs to your custom code, click **Text**, and then type the text string to use.
8. In the test, after your custom code, locate a value that your code returns to the test.
9. Highlight the value.
10. Right-click the highlighted value, click **Substitute**, and select the class name of your custom code.

Result

The custom code classes that you have added are listed. After you have made your selection, the value to be returned to the test is highlighted in orange, and the **Used by** table is updated with this information.

What to do next

Custom code is not displayed in the **Test Navigator** view. To view custom code, open the **Package Explorer** view and use the Java™ tools to identify the custom code that you added.

Test execution services interfaces and classes

You use the test execution services interfaces and classes to customize how you run tests. These interfaces and classes are located in the `com.ibm.rational.test.lt.kernel` package. Each interface and class is described briefly in this topic and in detail in the Javadoc information.

The custom code does not run on the mobile device, but from the generated Java code that is available in the test workbench. So, if you initiate the test run from the mobile device and the test script includes custom code, the custom code is not executed. To execute the custom code that is available in a mobile test script, you must initiate the run from test workbench. If you want to integrate custom code between two mobile instructions, you must split the test script. See [Splitting a test on page 300](#).

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

Interface	Description
ICustom-Code2	Defines customized Java™ code for test execution services. Use this interface to create all custom code.
ITestExecution- Services	Provides information for adding custom test execution features to tests. Replaces the IKLog interface. All the methods that were available in IKLog are contained in ITestExecutionServices, along with several newly exposed objects and interfaces. This interface is the primary interface for execution services. ITestExecutionServices contains the following interfaces: IDataArea, IARM, ILoopControl, IPDLogManager, IStatisticsManager2, ITestLogManager, ITime, and ITransaction.
IDataArea	Defines methods for storing and accessing objects in data areas. A data area is a container that holds objects. The elements of a data area are similar to program variables and are scoped to the owning container. To use objects specific to a protocol, you should use objects provided by that protocol that are stored in the protocol-specific data area.
IARM	Provides information about defining ARM (Application Response Measurement) specifications. You use this interface if your virtual users are being sampled for ARM processing.
ILoop- Control	Provides control over loops in a test or schedule. For example, you can use this interface to break loops at specific points in a test. The loop that is affected is the nearest containing loop found in either the test or the schedule.
IPDLog- Manager	Provides logging information such as problem severity, location levels, and error messages.
IStatistic- sManag- er	Provides access to performance counters in the ICustomCode2 interface (used for defining custom code). Performance counters are stored in a hierarchy of counters. Periodically, all the counter values in the hierarchy are reported to the testing workbench and collected into test run results, where they are available for use in reports and graphs. Each counter in the hierarchy has a type (defined in class <code>Stat- Type</code>). The operations that are available on a counter depend on the counter's type.

Interface	Description
ITestLog- Manager	Logs messages and verification points to the test log. Use this interface for handling error conditions, anomalies in expected data or other abstract conditions that need to be reported to users, or for comparisons or verifications whose outcome is reported to the test log. ITestLogManager can also convey informational or status messages after the completion of a test.
ITime	Defines basic time services, such as the current system time in milliseconds (adjusted so that all systems are synchronized with the schedule controller), the time the test begins, and the elapsed time from the beginning of the test.
ITransac- tion	Provides support for transactions. A collection of named transactions is maintained for each virtual user. Transactions created in custom code can be started and stopped wherever custom code can be used. These transactions can span several tests. Performance counters are kept for custom code transactions and appear in reports. An example of how you could use ITransaction is to create transactions for one virtual user but not another, to help verify responses from tests.
IEngineIn- fo	Provides information about the testing execution engine; for example, the number of virtual users running in this engine, the number of virtual users that have completed, the local directory in which test assets are deployed, and the host name of the computer on which the engine runs.
ITestInfo	Provides information about the test that is running; for example, the test name and information about the current problem determination log level for this test.
IVirtual- UserInfo	Provides information about virtual users; for example, the virtual user's name, problem determination log level, TestLog level, globally unique ID, and user group name.
IScalar	Provides methods for simple integer performance counters. It is used for counters of <code>SCALAR</code> and <code>STATIC</code> types. Use this interface to decrement and increment counters.
IStat	Defines observational performance counters. It defines the method for submitting a data point to performance counters of type <code>RATE</code> , <code>AVERAGE</code> , and <code>RANGE</code> .
IStatis- tics	Retrieves the performance counter tree associated with the current statistics processor. Stops the delivery of performance counters. Changes the priority of the statistics delivery thread.
IStatTree	Provides methods that can retrieve child counters, create the XML fragments that define counters, and set the description field of counters.
IText	Contains text-based performance counters. Performance counters that do not fit any of the other counter types can be created as type <code>TEXT</code> . <code>TEXT</code> counters are not assigned definitions, but they are collected in the test results.

Class	Description
-------	-------------

<code>Data- Area-</code>	Throws an exception whenever an attempt is made to modify a locked DataArea key.
------------------------------	--

Class	Description
Lock-	
Ex-	
cep-	
tion	
Out-	Indicates that an object created by ITestExecutionServices has been referenced outside of its intended
OfS-	scope.
cope-	
Ex-	
cep-	
tion	
Trans-	Throws an exception when a transaction is misused. The following conditions lead to a <code>TransactionException</code>
ac-	exception: attempting to start a transaction that has already been started, attempting to stop a transaction
tion-	that has not been started, and getting the start time or the elapsed time of a transaction that has not been
Ex-	started. Any operation (except <code>abort()</code>) on a transaction that has been aborted will throw a <code>TransactionExcep-</code>
cep-	<code>tion</code> exception.
tion	
Stat-	Provides a list of valid performance counter types. The performance counter types are: <code>AVERAGE</code> , <code>iAVERAGE</code> ,
Type	<code>iRANGE</code> , <code>iRATE</code> , <code>iSCALAR</code> , <code>iSTATIC</code> , <code>iSTRUCTURE</code> , <code>iTEXT</code> , <code>RANGE</code> , <code>RATE</code> , <code>SCALAR</code> , <code>STATIC</code> , <code>STRUCTURE</code> , and <code>TEXT</code> .

Reducing the performance impact of custom code

If custom code runs inside a page, it can affect that page's response time.

HTTP pages are containers of HTTP requests. On a given HTTP page, requests run in parallel across all of the connections between the agent computer and the system under test.

Page response time is the interval between *page start* and *page end*, which are defined as follows: Page start is the first timestamp associated with the client-server interaction. This interaction is either the first byte sent or the first connect of the first HTTP request. Page end is the last timestamp associated with the client-server interaction. This interaction is the last byte received of the last HTTP request to complete. Because of parallelism, the last HTTP request to complete might not be the last one listed for the page.

Typically, you should not insert custom code inside a page. While custom code that runs for only a few milliseconds should have little effect on page response time, the best practice is to place custom code outside a page. Custom code placed outside a page has no effect on page response time, and its execution time can overlap with think time delays.

Do not use custom code for data correlation if you can instead use the data correlation features built into the product. The built-in data correlation code takes advantage of requests running in parallel, whereas custom code actions do not begin until all earlier actions are completed.

You might need to place custom code inside a page to correlate a string from the response of a request inside that page to another request inside the same page. Even in this case, if you split the page into two pages, you can use the built-in data correlation features instead of custom code.

If you still want to run tests with custom code inside HTTP pages, use the Page Element report to evaluate performance. The Page Element report shows the response time and throughput for individual HTTP requests. Custom code does not affect the response time measurement of individual HTTP requests.

Related information

[Performance testing tips on page](#)

Custom code examples

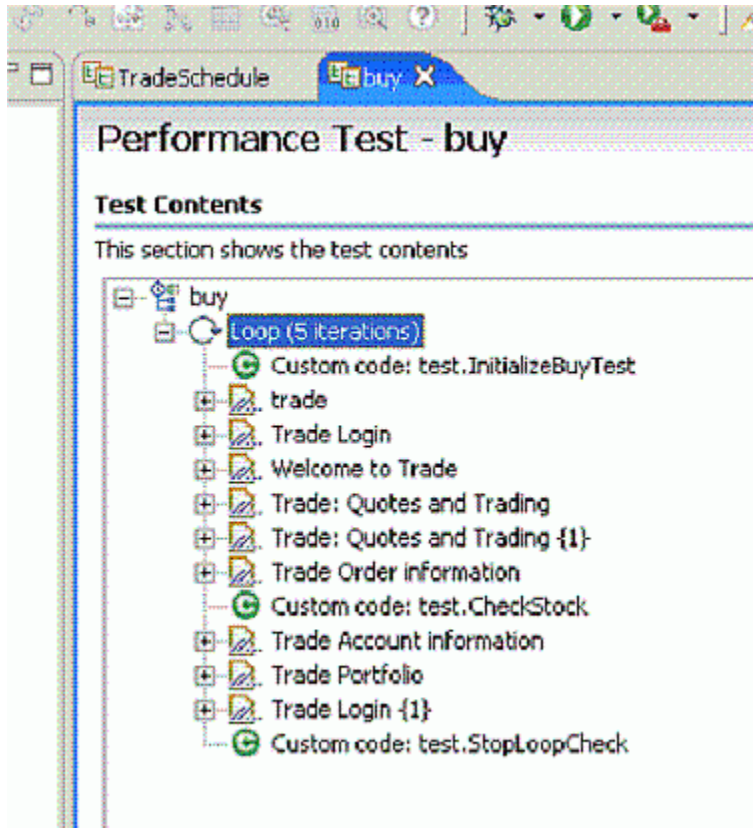
Custom code enables you to perform such tasks as managing loops, retrieving virtual user information, running external programs from tests, and customizing data correlation.

Controlling loops

This example demonstrates extending test execution by using custom code to control loops. It provides sample code that shows how you can manipulate the behavior of loops within a test to better analyze and verify test results.

This example uses a recording of a stock purchase transaction using the Trade application. The concepts shown here can be used in tests of other applications.

The test begins with a recording of a stock purchase transaction, using dataset substitution for the login IDs. The pages are wrapped in a five-iteration loop, as shown in the following figure:



Notice that among the various pages of the test, three items of custom code exist (indicated by the green circles with "C"s in them). This example explores these items of custom code.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents HCL OneTest Performance API Reference**.

The first piece of custom code, `InitializeBuyTest`, is mentioned here:

```
package customcode;

import java.util.Random;

import com.ibm.rational.test.lt.kernel.IDataArea;
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.services.IVirtualUserInfo;

/**
 * @author unknown
 */
public class InitializeBuyTest implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public InitializeBuyTest() {
```

```

}

/**
 * For description of ICustomCode2 and ITestExecutionServices interfaces,
 * see the Javadoc.. */
public String exec(ITestExecutionServices tes, String[] args) {
    // Get the test's data area and set a flag indicating that nothing
    // has failed yet. This flag will be used later to break out
    // of the schedule loop as soon as a failure is encountered.
    IDataArea dataArea = tes.findDataArea(IDataArea.TEST);
    dataArea.put("failedYet", "false");

    // Get the virtual users's data area
    IDataArea vda = tes.findDataArea(IDataArea.VIRTUALUSER);

    // Randomly select a stock to purchase from the set of s:0 to s:499.
    IVirtualUserInfo vuInfo = (IVirtualUserInfo) vda.get(IVirtualUserInfo.KEY);
    Random rand = vuInfo.getRandom();
    String stock = "s:" + Integer.toString(rand.nextInt(499));

    // Persist the name of the stock in the virtual user's data area.
    vda.put("myStock", stock);

    return stock;
}

```

This custom code is located in the method `exec()`.

First, the data area for the test is acquired to store a flag value, in this case a string of text, to be used later to stop the test loop when an error is discovered. Data stored in this way can be persisted across tests.

Then a randomly generated stock string is created. The value is stored as the variable `stock`, and is passed back as the return value for the method. This return value is used as a substitute in a request later, as shown in the following figure:

The screenshot shows a test execution tool interface. On the left, a tree view displays a test plan for 'buy' with 5 iterations. The steps include: Custom code: test.InitializeBuyTest, trade, Trade Login, Welcome to Trade, Trade: Quotes and Trading, Trade: Quotes and Trading {1}, Trade Order information, Custom code: test.CheckStock, Trade Account information, Trade Portfolio, Trade Login {1}, and Custom code: test.StopLoopCheck. The 'trade' step is expanded, showing a request with a highlighted URL: `quad2003/trade/app?action=buy&symbol=s:716&quantity=20`. On the right, the 'Request Attributes' panel shows: Version: 1.1, Method: GET, Host: quad2003, and URL: /trade/app?action=buy&symbol=s:3A716&quantity=20. The 'Data' field is empty.

The highlighted item uses a substitution (`s:3A716`), which is the value returned by the `InitializeBuyTest` custom code item. We are using custom code to drive the direction of our test.

The next lines of code in `InitializeBuyTest` use the Virtual User data area to store the name of the stock for later reference. Again, data stored in this way can persist across tests.

The second piece of custom code is called `checkStock`. Its contents are as follows (listing only the `exec()` method this time):

```
public String exec(ITestExecutionServices tes, String[] args) {

    // Get the actual and requested stock purchased.
    String actualStock = args[0].replaceAll("<B>", "");
    actualStock = actualStock.substring(0, actualStock.indexOf("<"));
    String requestedStock = args[1];

    // Set the log level to ALL.
    IDataArea dataArea = tes.findDataArea(IDataArea.TEST);
    ITestInfo testInfo = (ITestInfo)dataArea.get(ITestInfo.KEY);
    testInfo.setTestLogLevel(ITestLogManager.ALL);

    // If the log level is set to ALL, report the actual and requested stock
    // purchased.
    ITestLogManager testLogManager = tes.getTestLogManager();
    if (testLogManager.wouldReport(ITestLogManager.ALL)) {
        testLogManager.reportMessage("Actual stock purchased: "
            + actualStock + ". Requested stock: " + requestedStock
            + ".");
    }

    // If the actual and requested stock don't match, submit a FAIL verdict.
    if (testLogManager.wouldReport(ITestLogManager.ALL)) {
        if (!actualStock.equalsIgnoreCase(requestedStock)) {
            testLogManager.reportVerdict(
                "Actual and requested purchase stock do not match.",
                VerdictEvent.VERDICT_FAIL);

            // Use the test's data area to record the fact that an error has
            // occurred.
            dataArea.put("failedYet", "true");
        }
    }
    return null;
}
```

This code begins by extracting two arguments that have been passed to the code. A part of the response in the original recording is highlighted and used as a reference, as shown in the following figure.

The screenshot shows a test execution interface. On the left is a tree view with the following items:

- Trade: Quotes and Trading
- Trade: Quotes and Trading {1}
- Trade Order information
 - quad2003/trade/app?action=buy&symbol=s:716&quantit
 - Response: 200 - OK
 - Custom code: test.CheckStock
- Trade Account information
- Trade Portfolio
- Trade Login {1}
- Custom code: test.StopLoopCheck

On the right, there is a table with headers:

Date	Fri, 11 Nov 2005 00:47:13 GMT
Server	IBM_HTTP_SERVER/1.3.26.2 Apache/1.3.26 (...)
Keep-Alive	timeout=45, max=24989
Connection	Keep-Alive
Transfer-Encoding	chunked

Below the table are 'Add' and 'Edit' buttons. The 'Content:' area displays the following HTML snippet:

```
<TR>
  <TD align="left"><FONT
color="#cc0000"><B><BR>
  Order 6664145</B> to <B>buy
20.0</B> shares of <B>s:716</B> has been submitted for
processing. </FONT><BR>
<BR>
```

Some string manipulation is needed to acquire the text of interest; in this case, the name of the stock that was actually purchased. This newly created reference is then passed into `CheckStock` as an argument, as shown in the following figure:

The screenshot shows a test execution interface. On the left is a tree view with the following items:

- Trade Login
- Welcome to Trade
- Trade: Quotes and Trading
- Trade: Quotes and Trading {1}
- Trade Order information
 - quad2003/trade/app?action=buy&symbol=s:716&quantit
 - Response: 200 - OK
 - Custom code: test.CheckStock
- Trade Account information
- Trade Portfolio
- Trade Login {1}
- Custom code: test.StopLoopCheck

On the right, there is an 'Arguments' panel with the following items:

- s:716 has been submitted" - Content
- Custom code: test.InitializeBuyTest

Note that the return value of `InitializeBuyTest` is passed in as an argument as well.

The `CheckStock` custom code item uses these values to verify that the randomly chosen stock generated by `InitializeBuyTest` is actually purchased during the execution of the test.

`CheckStock` then sets the test log level, reports the actual and requested stock purchase, and raises a FAIL verdict if they do not match. `CheckStock` also stores a `true` value associated with the tag `failedYet` in the test's data area.

The third piece of custom code (`exec()` method only) is mentioned here:

```
public String exec(ITestExecutionServices tes, String[] args) {

    // Get the test log manager.
    ITestLogManager testLogManager = tes.getTestLogManager();

    // Get the test's data area and get a flag indicating to
    // see if anything has failed yet. If so, stop the loop.
    IDataArea dataArea = tes.findDataArea(IDataArea.TEST);
    String failedYet = (String) dataArea.get("failedYet");

    // Break out of the loop if an error has been encountered.
```

```

if (failedYet.equalsIgnoreCase("true")) {
    tes.getLoopControl().breakLoop();

    if (testLogManager.wouldReport(ITestLogManager.ALL)) {
        testLogManager.reportMessage("Loop stopped.");
    }
}

return null;
}

```

This code uses the test's data area to determine the user-defined value associated with the tag `failedYet`. If `failedYet` is `true`, `StopLoopCheck` breaks out of the test loop.

Retrieving the IP address of a virtual user

This example shows how to retrieve the local IP address of a virtual user. Retrieving IP addresses is particularly useful when virtual users are using IP aliases.

The following custom code retrieves the IP address that was assigned to a virtual user:

```

import java.net.InetAddress;
import com.ibm.rational.test.lt.kernel.IDataArea;
import com.ibm.rational.test.lt.kernel.services.ITestLogManager;
import com.ibm.rational.test.lt.kernel.services.IVirtualUserInfo;

public String exec(ITestExecutionServices tes, String[] args) {
    IVirtualUserInfo vui = (IVirtualUserInfo)
        tes.findDataArea(IDataArea.VIRTUALUSER).get(IVirtualUserInfo.KEY);
    ITestLogManager tlm = tes.getTestLogManager();

    if (vui != null) {
        String localAddr = null;
        InetAddress ipAddr = vui.getIPAddress();
        if (ipAddr != null)
            localAddr = ipAddr.toString();
        tlm.reportMessage("IPAlias address is " + (localAddr != null ? localAddr : "not set"));
        return localAddr;
    }
    else
        return ("Virtual User Info not found");
}

```

Printing input arguments to a file

The `PrintArgs` class prints its input arguments to the file `C:\arguments.out`. This class could be used, for example, to print a response returned by the server.

Example

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

import java.io.*;

```



```

/**
 * The PrintArgs class prints its input arguments to the file
 * C:\arguments.out. This example could be used to print a response
 * returned by the server.
 */

/**
 * @author IBM Custom Code Samples
 */

public class PrintArgs implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public PrintArgs() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        try {
            FileWriter outFile = new FileWriter("C:\\arguments.out");
            for (int i = 0; i < args.length; i++)
                outFile.write("Argument " + i + " is: " + args[i] + "\n");
            outFile.close();
        } catch (IOException e) {
            tes.getTestLogManager().reportMessage(
                "Unable to write to C:\\arguments.out");
        }
        return null;
    }
}

```

Counting the number of times that code is executed

The `CountAllIterations` class counts the number of times code is executed by all virtual users. The `CountUserIterations` class counts the number of times code is executed by an individual virtual user.

Example

The `CountAllIterations` class counts the number of times it is executed by all virtual users running in a particular JVM and returns this count as a string.

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

/**
 * The CountAllIterations class counts the number of times it is executed
 * by all virtual users running in a particular JVM and returns this count
 * as a string. If all virtual users on an agent are running in the same
 * JVM (as would typically be the case), this class will count the number of
 * times it is run on the agent.
 */

```

```

/**
 * @author IBM Custom Code Samples
 */

public class CountAllIterations implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {
    private static int numJVMLoops = 0;

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public CountAllIterations() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        return Integer.toString(++numJVMLoops);
    }
}

```

Example

The CountUserIterations class counts the number of times code is executed by an individual virtual user.

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.IDataArea;

/**
 * The CountUserIterations class counts the number of times it is executed
 * by an individual virtual user and returns this count as a string.
 */

/**
 * @author IBM Custom Code Samples
 */

public class CountUserIterations implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public CountUserIterations() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        IDataArea userDataArea = tes.findDataArea(IDataArea.VIRTUALUSER);
        final String KEY = "NumberIterationsPerUser";

        Number numPerUser = (Number)userDataArea.get(KEY);
        if (numPerUser == null) {
            numPerUser = new Number();
            userDataArea.put(KEY, numPerUser);
        }

        numPerUser.value++;
    }
}

```

```

        return Integer.toString(numPerUser.value);
    }

    private class Number {
        public int value = 0;
    }
}

```

Setting and clearing cookies for a virtual user

The `SetCookieFixedValue` class sets a `Cookie` for a virtual user, and the `ClearCookies` class clears all cookies for a virtual user.

Example

The `SetCookieFixedValue` class sets a `Cookie`, defined in the `newCookie` variable, for a virtual user just as if the server had returned a `Set-Cookie`.

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.execution.http.cookie.IHTTPVirtualUserInfo;
import com.ibm.rational.test.lt.kernel.IDataArea;

import java.text.ParseException;

/**
 * The SetCookieFixedValue class sets a Cookie, defined in the newCookie
 * variable, for a virtual user just as if the server had returned a Set-Cookie.
 */

/**
 * @author IBM Custom Code Samples
 */

public class SetCookieFixedValue implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public SetCookieFixedValue() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        String newCookie = "MyCookie=CookieValue;path=/;domain=.ibm.com";
        IDataArea dataArea = tes.findDataArea(IDataArea.VIRTUALUSER);
        IHTTPVirtualUserInfo httpInfo =
            (IHTTPVirtualUserInfo)dataArea.get(IHTTPVirtualUserInfo.KEY);

        try {
            httpInfo.getCookieCache().setCookie(newCookie);
        } catch (ParseException e) {
            tes.getTestLogManager().reportMessage("Unable to parse Cookie " +
                newCookie);
        }
    }
}

```

```

        return null;
    }
}

```

The ClearCookies class clears all Cookies for a virtual user. For information on how cookies are treated in tests and schedules, see [How loops affect the state of virtual users on page 299](#).

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.execution.http.util.CookieCacheUtil;

/**
 * The ClearCookies class clears all Cookies for a virtual user.
 */

/**
 * @author IBM Custom Code Samples
 */

public class ClearCookies implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public ClearCookies() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        CookieCacheUtil.clearCookieCache(tes);
        return null;
    }
}

```

Determining where a test is running

The ComputerSpecific class determines where a test is running

Exemple

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

import java.net.InetAddress;
import java.net.UnknownHostException;

/**
 * The ComputerSpecific class determined the hostname on which the test is
 * running, prints the hostname and IP address as a message in the test log,
 * and returns different strings based on the hostname.
 */

/**
 * @author IBM Custom Code Samples

```

```

*/
public class ComputerSpecific implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public ComputerSpecific() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        String hostName = "Unknown";
        String hostAddress = "Unknown";

        try {
            hostName = InetAddress.getLocalHost().getHostName();
            hostAddress = InetAddress.getLocalHost().getHostAddress();
        } catch (UnknownHostException e) {
            tes.getTestLogManager().reportMessage(
                "Not able to obtain host information");
            return null;
        }
        tes.getTestLogManager().reportMessage("The hostname is " + hostName +
            "; IP address is " + hostAddress);

        if (hostName.equals("host-1234"))
            return "Special";
        else
            return "Normal";
    }
}

```

Determining where a test is running

The ComputerSpecific class determines where a test is running

Example

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

import java.net.InetAddress;
import java.net.UnknownHostException;

/**
 * The ComputerSpecific class determined the hostname on which the test is
 * running, prints the hostname and IP address as a message in the test log,
 * and returns different strings based on the hostname.
 */

/**
 * @author IBM Custom Code Samples
 */

public class ComputerSpecific implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

```

```

/**
 * Instances of this will be created using the no-arg constructor.
 */
public ComputerSpecific() {
}

public String exec(ITestExecutionServices tes, String[] args) {
    String hostName = "Unknown";
    String hostAddress = "Unknown";

    try {
        hostName = InetAddress.getLocalHost().getHostName();
        hostAddress = InetAddress.getLocalHost().getHostAddress();
    } catch (UnknownHostException e) {
        tes.getTestLogManager().reportMessage(
            "Not able to obtain host information");
        return null;
    }
    tes.getTestLogManager().reportMessage("The hostname is " + hostName +
        "; IP address is " + hostAddress);

    if (hostName.equals("host-1234"))
        return "Special";
    else
        return "Normal";
}
}

```

Extracting a string or token from its input argument

The ParseResponse class extracts a string from its input argument. The ExtractToken class extracts a particular token (string) from its input argument. Both classes can be useful for handling certain types of dynamic data correlation.

Example

The ParseResponse class extracts a string from its input argument, using a regular expression for pattern matching.

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

import java.util.regex.*;

/**
 * The ParseResponse class demonstrates using Custom Code to extract a
 * string from its input argument using a regular expression for pattern
 * matching.
 *
 * In this sample, the args[0] input string is assumed to be the full
 * response from a previous request. This response contains the day's
 * headlines in a format such as:
 *
 * <a class=f href=r/d2>In the News</a><small class=m>&nbsp;<span id=nw>
 * </span></small></h2>
 * <div class=ct>
 * &#149;&nbsp;<a href=s/213231>Cooler weather moving into eastern

```

```

U.S.</a> * <br>&#149;&nbsp;<a href=s/262502>Digital camera shipments
up</a><br> *
* Given the above response, the extracted string would be:
*     Cooler weather moving into eastern U.S.
*/

/**
 * @author IBM Custom Code Samples
 */

public class ParseResponse implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public ParseResponse() {}

    public String exec(ITestExecutionServices tes, String[] args) {
        String HeadlineStr = "No Headline Available";
        String RegExpStr = ".*In the News[^;]*;[^;]*;[^;]*;<a
href=([>]*)>([<]*)<";        Pattern pattern =
Pattern.compile(RegExpStr, Pattern.DOTALL);        Matcher matcher =
pattern.matcher(args[0]);
        if (matcher.lookingAt())
            HeadlineStr = matcher.group(2);
        else
            tes.getTestLogManager().reportMessage("Input does not match
pattern.");
        return HeadlineStr;
    }
}

```

The ExtractToken class extracts a particular string from its input argument.

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

/**
 * The ExtractToken class demonstrates using Custom Code to extract a particular
 * token (string) from its input argument. This can be useful for handling
 * certain types of dynamic data correlation.
 *
 * In this sample, the args[0] input string is assumed to be comma-delimited
 * and the token of interest is the next-to-last token. For example, if
 * args[0] is:
 *     javascript:parent.selectItem('1010', '[Negative]1010', '1010', '', 'IBM',
 *     '30181', 'Rational', '1', 'null', '1', '1', '6fd8e261', 'RPT')
 * the class will return the string 6fd8e261.
 */

/**
 * @author IBM Custom Code Samples
 */

public class ExtractToken implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

```

```

public ExtractToken() {
}

public String exec(ITestExecutionServices tes, String[] args) {
    String ArgStr;
    String NextToLastStr;
    String[] Tokens = args[0].split(",");

    if (Tokens.length > 2) {
        ArgStr = Tokens[Tokens.length - 2];        // Extract next-to-last token

        // Remove enclosing ''
        NextToLastStr = ArgStr.substring(1, ArgStr.length() - 1);
    } else {
        tes.getTestLogManager().reportMessage("Could not extract value");
        NextToLastStr = null;
    }
    return NextToLastStr;
}
}

```

Retrieving the maximum JVM heap size

The JVM_Info class retrieves the maximum heap size of the JVM.

Example

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

import java.net.*;

/**
 * The JVM_Info class retrieves the maximum heap size of the JVM.
 * It writes a message in the test log with the hostname where the
 * JVM is running and the JVM's maximum heap size in megabytes.
 */

/**
 * @author IBM Custom Code Samples
 */

public class JVM_Info implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    public JVM_Info() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        Runtime rt = Runtime.getRuntime();
        long maxMB = rt.maxMemory()/(1024*1024); // maxMemory() size is in bytes
        String hostName = "Unknown";

        try {
            hostName = InetAddress.getLocalHost().getHostName();
        } catch (UnknownHostException e1) {

```



```

        tes.getTestLogManager().reportMessage("Can't get hostname");
        return null;
    }

    tes.getTestLogManager().reportMessage("JVM maximum heap size for host "
        + hostName + " is " + maxMB + " MB");
    return null;
}
}

```

Running an external program from a test

The `ExecTest` class runs a program, defined in the `execName` variable, on the system where the test is running.

Exemple

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.services.ITestLogManager;
import org.eclipse.hyades.test.common.event.VerdictEvent;

import java.io.IOException;

/**
 * The ExecTest class runs a program, defined in the execName variable,
 * on the system where the test is running.
 * The test verdict is set to PASS if the program return code is 0.
 * The test verdict is set to FAIL if the program doesn't execute or
 * if the program return code is non-zero
 * In this sample, the program is perl.exe.
 */

/**
 * @author IBM Custom Code Samples
 */

public class ExecTest implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public ExecTest() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        ITestLogManager logger = tes.getTestLogManager();
        int rtnval = 1;
        Process p = null;
        String execName = "C:/Windows/System32/perl.exe C:/Perl/true.pl";

        Runtime rt = Runtime.getRuntime();
        // Execute test
        try {
            p = rt.exec(execName);
        } catch (IOException e) {

```

```

        logger.reportMessage("Unable to run = " + execName);
        logger.reportVerdict("Execution of " + execName + " failed",
                            VerdictEvent.VERDICT_FAIL);

        return null;
    }

    // Wait for the test to complete
    try {
        rtnval = p.waitFor();
        logger.reportMessage("Process return value is " +
                            String.valueOf(rtnval));
    } catch (InterruptedException e1) {
        logger.reportMessage("Unable to wait for " + execName);
        logger.reportVerdict("WaitFor on " + execName + " failed",
                            VerdictEvent.VERDICT_FAIL);

        return null;
    }

    // Check the test return code and set the test verdict appropriately
    if (rtnval != 0)
    {
        logger.reportVerdict("Execution failed", VerdictEvent.VERDICT_FAIL);
    } else {
        logger.reportVerdict("Execution passed", VerdictEvent.VERDICT_PASS);
    }

    return null;
}
}
}

```

Adding custom counters to reports

When you want to monitor the specific requirement, you can add custom counters to performance report by using the custom code. After running tests, the results from the custom counters are automatically aggregated in the same way that the default performance testing counters.

Starting from V10.1.0, you can view and monitor the counter information generated by the custom code on a graph when the custom code starts in the test run.

After running tests, you can view the custom counter in the report. You can also view the custom counter information on a different page by creating a custom report. For more information about customizing the report, see related links.

You can add the following custom code in your test to create a custom counter in a report.

```

package test;

import org.eclipse.hyades.test.common.event.VerdictEvent;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.services.stats.CountAggregationLevel;
import com.ibm.rational.test.lt.kernel.services.stats.CounterUnits;
import com.ibm.rational.test.lt.kernel.services.stats.ICounterFolder;
import com.ibm.rational.test.lt.kernel.services.stats.ICounterRegistry;
import com.ibm.rational.test.lt.kernel.services.stats.IStatisticsManager2;
import com.ibm.rational.test.lt.kernel.services.stats.IValueCounter;

```

```

import com.ibm.rational.test.lt.kernel.services.stats.ValueAggregationLevel;

import database.DatabaseAccess;
import database.TransactionResult;

public class DatabaseStats implements com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    private static boolean registerDone;

    /**
     * This method declares the counters that will be produced during execution.
     * Declaring counters is optional, but it allows to customize some of their
     * attributes, such as the label and unit, and what level of statistical information
     * will be available in reports.
     */
    private static synchronized void registerCounters(ICounterRegistry registry) {
        if (registerDone) return;
        registry.path("Database", "Transaction", "Attempts")
            .count()
            .aggregationLevel(CountAggregationLevel.RATE_RANGE)
            .label("Started Transactions")
            .unit("transactions")
            .register();

        registry.path("Database", "Transaction", "Commits")
            .verificationPoint()
            .label("Transaction Commits VP")
            .register();

        registry.path("Database", "Transaction", "Response Time", "Network")
            .value()
            .aggregationLevel(ValueAggregationLevel.RANGE)
            .unit(CounterUnits.MILLISECONDS)
            .register();

        registry.path("Database", "Transaction", "Response Time", "Commit")
            .value()
            .aggregationLevel(ValueAggregationLevel.DISTRIBUTION)
            .unit(CounterUnits.MILLISECONDS)
            .register();

        registry.path("Database", "Error")
            .text()
            .label("Database Error Message")
            .register();
        registerDone = true;
    }

    private DatabaseAccess database = DatabaseAccess.INSTANCE;

    /**
     * This custom code adds a record in database. It produces a couple of counters,
     * such as the database transaction attempts, successes/failures, and response time.
     */
    public String exec(ITestExecutionServices tes, String[] args) {
        String product = args.length > 0 ? args[0] : "Default";
        IStatisticsManager2 mgr = tes.getStatisticsManager2();
    }
}

```

```

registerCounters(mgr.registry());

database.startTransaction();
mgr.getCountCounter("Database", "Transaction", "Attempts").increment();

database.executeQuery("INSERT INTO TABLE Purchases VALUES('" + product + "', 1000)");
TransactionResult result = database.commit();

mgr.getVerificationPointCounter("Database", "Transaction", "Commits")
    .increment(result.isSuccess() ? VerdictEvent.VERDICT_PASS : VerdictEvent.VERDICT_FAIL);
if (!result.isSuccess()) {
    mgr.getTextCounter("Database", "Error").addMeasurement(result.getErrorMessage());
}

ICounterFolder times = mgr.getFolder("Database", "Transaction", "Response Time");
times.getValueCounter("Network").addMeasurement(result.getNetworkTime());
times.getValueCounter("Commit").addMeasurement(result.getCommitTime());

IValueCounter value = tes.getStatisticsManager2().getValueCounter("MyStats", "Value");
value.addMeasurement(System.nanoTime() % 2000);

return null;
}
}

```

Related information

[Creating custom Java code on page 657](#)

[Creating custom reports on page 779](#)

Using transactions and statistics

You can use custom code to start transactions, gather additional statistics during a transaction, and stop a transaction.

The following code shows how to start a transaction. Transactions that are generated by test execution services automatically create and manage statistics.

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.services.ITransaction;

/**
 * @author IBM Custom Code Samples
 */
public class BeginTransaction implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
}

```

```

public BeginTransaction() {
}

/**
 * For Javadoc information on the ICustomCode2 and ITestExecutionServices interfaces,
 * see the 'Test execution services interfaces and classes' help topic.
 */
public String exec(ITestExecutionServices tes, String[] args) {
    // the name of the transaction could have been passed in via data correlation mechanism.
    ITransaction foo = tes.getTransaction("foo");
    foo.start();
    return null;
}
}

```

The following code shows how to gather additional statistics during a transaction.

```

package customcode;

import com.ibm.rational.test.lt.kernel.ITime;
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.statistics.IScalar;
import com.ibm.rational.test.lt.kernel.statistics.IStat;
import com.ibm.rational.test.lt.kernel.statistics.IStatTree;
import com.ibm.rational.test.lt.kernel.statistics.impl.StatType;

/**
 * @author IBM Custom Code Samples
 */
public class BodyTransaction implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public BodyTransaction() {
    }

    /**
     * For Javadoc information on the ICustomCode2 and ITestExecutionServices interfaces,
     * see the 'Test execution services interfaces and classes' help topic.
     */
    public String exec(ITestExecutionServices tes, String[] args) {
        IStatTree tranStat;
        IStatTree timeStat;
        IStatTree countStat;

        IStat timeDataStat = null; // counter for the time RANGE
        IScalar countDataStat = null; // counter for the count SCALAR

        ITime timer = tes.getTime();

        IStatTree rootStat = tes.getStatisticsManager2().getStatTree();
        if (rootStat != null) {
            // these counters set up the hierarchy
            tranStat = rootStat.getStat("Transactions", StatType.STRUCTURE);
            timeStat = tranStat.getStat("Body Time", StatType.STRUCTURE);
            countStat = tranStat.getStat("Bocy Count", StatType.STRUCTURE);
        }
    }
}

```

```

    // the name of the counters could have been passed in via data correlation mechanism
    timeDataStat = (IStat) timeStat.getStat("foo", StatType.RANGE);
    countDataStat = (IScalar) countStat.getStat("foo", StatType.SCALAR);
}

// get the start time
long startTime = timer.timeInTest();

// do the work
// whatever that work might be

// get the end time
long endTime = timer.timeInTest();

// update timeDataStat with the elapsed time
if (timeDataStat != null)
    timeDataStat.submitDataPoint(endTime - startTime);

// update the countDataStat
if (countDataStat != null)
    countDataStat.increment();

return null;
}
}

```

The following code shows how to stop a transaction.

```

package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.services.ITransaction;

/**
 * @author IBM Custom Code Samples
 */
public class EndTransaction implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public EndTransaction() {
    }

    /**
     * For Javadoc information on the ICustomCode2 and ITestExecutionServices interfaces,
     * see the 'Test execution services interfaces and classes' help topic.
     */
    public String exec(ITestExecutionServices tes, String[] args) {
        // the name of the transaction could have been passed in via data correlation mechanism.
        ITransaction foo = tes.getTransaction("foo");
        foo.stop();
        return null;
    }
}

```

```
}

```

Reporting custom verification point failures

You can use custom code to report a custom verification point failure.

The following code shows how to report a custom verification point failure.

```
package customcode;

import org.eclipse.hyades.test.common.event.VerdictEvent;
import org.eclipse.hyades.test.common.runner.model.util.Verdict;

import com.ibm.rational.test.lt.execution.core.IVerificationPoint;
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

/**
 * @author IBM Custom Code Samples
 */
public class Class implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public Class() {
    }

    /**
     * For javadoc of ICustomCode2 and ITestExecutionServices interfaces, select 'Help Contents' in the
     * Help menu and select 'Extending HCL OneTest™ Performance functionality' -> 'Extending test execution
     * with custom code'
     */
    public String exec(ITestExecutionServices tes, String[] args) {
        tes.getTestLogManager().reportVerificationPoint("CustomVP", VerdictEvent.VERDICT_FAIL);
        return null;
    }
}

```

Debugging custom code

This example demonstrates debugging custom code by adding a breakpoint. It provides sample code to add a breakpoint. This way of debugging custom code is applicable only for a schedule.

1. Start HCL OneTest™ Performance and create a performance test project `MyProject`.
2. Create an HTTP test, **MyTest**, by recording a visit to `http://<hostname>:7080/`.



Note: Before accessing the URL, ensure that HCL OneTest™ Performance is running. The URL returns an HTTP 404 error, which is expected.

Result

The screenshot shows the HCL OneTest Performance interface. On the left, the 'Test Contents' pane shows a tree view with 'MyTest' expanded to 'Test Resources' > 'kmnote' > 'kmnote:7080/' > 'Response: 404 - Not Found'. On the right, the 'Test Element Details' pane shows the response details for 'Response: 404 - Not Found'. The 'Response Data' section shows 'Status: 404' and 'Version: 1.1'. The 'Response Headers' section shows a table with 'Content-Type: text/html' and 'Content-Length: 3202'. The 'Content' section shows the HTML response body, which includes an error message: 'Error 404 - Not Found. No context on this server matched or han...'. The interface also includes a filter text input, an 'Options' dropdown, and various action buttons like 'Add', 'Insert', 'Select', 'Remove', 'Up', 'Down', 'Prev', 'Next', 'Run', and 'View'.

3. Expand the first request and click the response element.
4. In the Test Element Details section, right-click in the **Content** field and click **Create Field Reference**.
5. Type the reference name and click **OK**.
6. Click the first page, and then click **Add > Custom Code**.
7. In the **Arguments** section of Test Element Details, click **Add**.
8. Expand the data source for the search results page, select the reference name that you created in step 5, and click **Select**.
9. Click **Generate Code**.

Result

A new tab with the generated code is displayed.

10. Insert the following the code into the `exec()` method:

```
ITestLogManager history = tes.getTestLogManager();
if (args.length > 0) {
    if (args[0].indexOf("Investor Relations") != -1) {
        history.reportMessage("First page failed. Bail loop!");
        tes.getLoopControl().continueLoop();
    }
}
```

! Important:

- Fix the double quotation marks, if any, so they are straight and the compiler no longer gives warning.
- To resolve compiler warnings related to importing a class, press **Ctrl + Shift + O**.

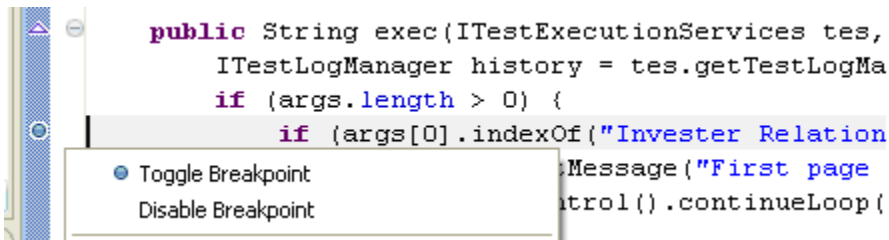
The code will look like this:


```

public String exec(ITestExecutionServices tes, String[] args) {
    ITestLogManager history = tes.getTestLogManager();
    if (args.length > 0) {
        if (args[0].indexOf("Investor Relations") != -1) {
            history.reportMessage("First page failed. Bail loop!");
            tes.getLoopControl().continueLoop();
        }
    }
    return null;
}

```

11. To set a breakpoint, click anywhere on the `args[0].indexOf` line. Move the pointer to the left-most portion of the text editor window and double-click with the pointer horizontally on the same line. A blue button is displayed in this left-most portion of the window indicating the breakpoint is set.

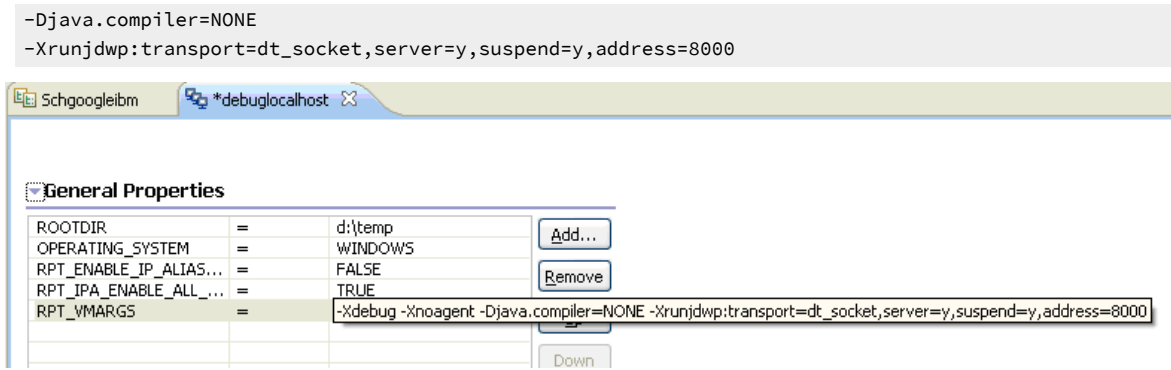


12. Save the custom code and then the test.
13. Create a new schedule, `Schtest`.
- In `Schtest`, set the number of users to run to 1.
 - Click **User Group 1** and click **Add > Test**. Select the `MyTest` test and click **OK**.
 - Click **User Group 1** and click the **Run this group on the following locations** button.
 - Click **Add > Add New**.
 - In the **New Location** window, type the following information:
 - In **Host name**, type `localhost`.
 - In **Name**, type `debuglocation`.
 - In **Deployment directory**, type `C:\mydeploy`.
 - Click **Finish**.
 - Save the schedule.
14. In the Test Navigator, right-click `debuglocation` and click **Open**.
15. Click the **General Properties** tab and click **Add**.
16. In the **Property name** field, type `RPT_VMARGS` and in the **Property value** field, add the following values each separated by a space.

```

-Xdebug
-Xnoagent

```



17. Save the location.

18. Attach the debugger to the schedule execution process.

a. Run the schedule.

Because the schedule is using **debuglocation**, it will pause at the beginning to let you attach the debugger to the execute process.

b. Click **Window > Open Perspective > Other > Debug**.

c. Click **Run > Debug Configurations**.

d. In the **Debug Configurations** window, right-click **Remote Java Application** and click **New**.

e. Click **Debug**.

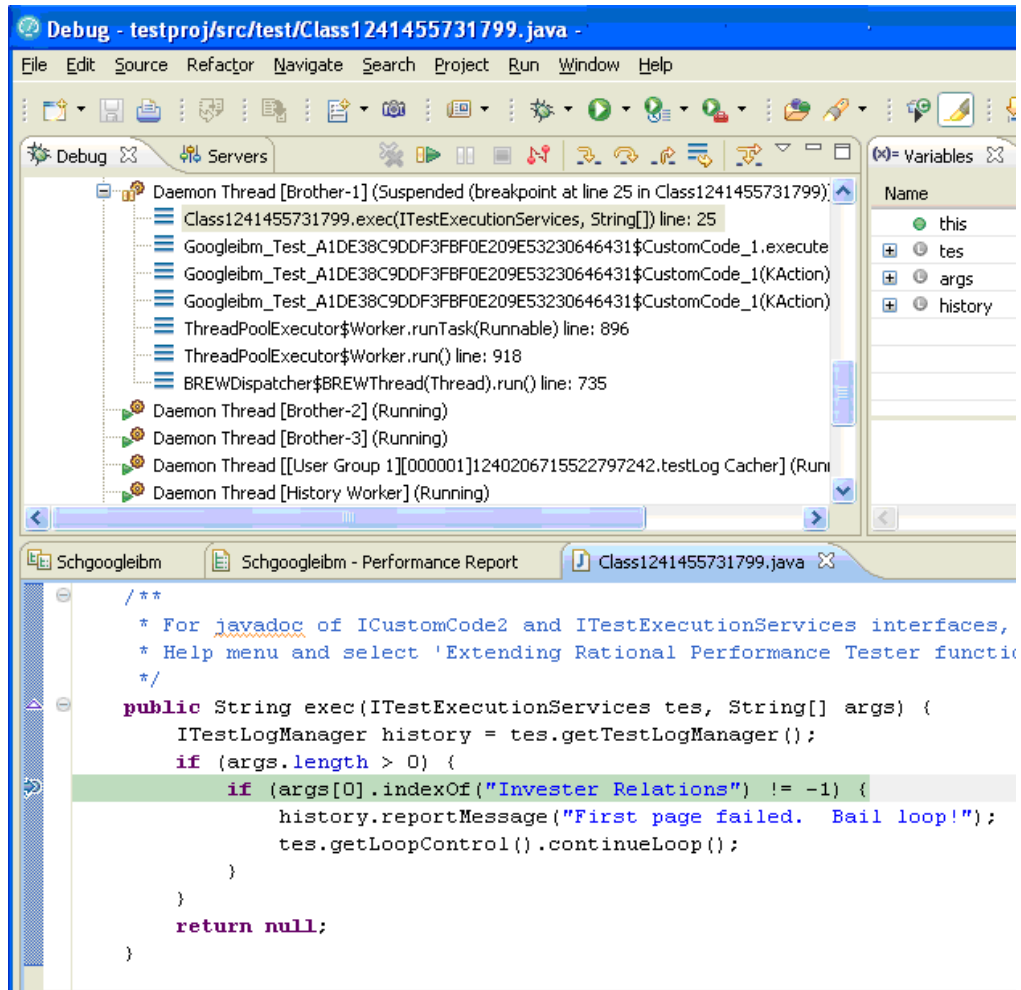
A list of running threads are displayed in the Debug window and the schedule execution pauses at the debug breakpoint.

f. If you are doing it for the first time, you might need to provide the source location to see the custom Java code. You do this by taking the following steps:

i. Click **Edit Source Lookup Path** and click **Add**.

ii. Click **Workspace Folder > OK**.

- iii. Now, expand MyProject, select the src folder, and click **OK**. The schedule run stops at the specified breakpoint.



Migrating custom code from previous versions

You can run scripts that contain custom code from previous releases and edit tests to make new calls to old or new custom code classes.

About this task

You can perform the following tasks without any additional steps:

- Run a script that contains custom code that was created in a previous release.
- Edit a test to make a new call to an old custom code class.
- Add new custom code to a test that contains old custom code.

To edit a class in existing custom code so that it can call new *TestExecutionServices* methods, type cast the *IKlog* argument in the old custom code to the *ITestExecutionServices* interface.

When you migrate the custom code from the previous versions, you must use `getStatisticsManager2()` as `getStatisticsManager()` API is deprecated from V10.1.0.

Extending HCL OneTest™ Performance to support other protocols

The HCL OneTest™ Performance Extensibility Software Development Kit (SDK) enables you to create extensions to support testing new protocols without modifying the core product code. You can create an extension that provides functionality for testing other protocols in addition to the HTTP protocol support that is standard in HCL OneTest™ Performance.

Before you begin

HCL OneTest™ Performance provides the capability to emulate multiple users applying load on a system under test. To test performance, you record a test and subsequently replicate it, and then play it back against the system being tested. Refer to the HCL OneTest™ Performance product information center for more details.

About this task

HCL OneTest™ Performance supports the HTTP protocol. The goal of the Extensibility SDK is to provide a framework that supports the development of various protocols which can be plugged into HCL OneTest™ Performance.

To develop support for testing protocols in HCL OneTest™ Performance, you must be familiar with the following areas:

- The protocol that you intend to support
- The Eclipse platform (The Eclipse documentation provides relevant information in the "JDT Programmer's Guide" topic.)
- The Test and Performance Tools Platform (TPTP).

For examples of HCL OneTest™ Performance extensions, see the following plug-ins:

- `com.ibm.rational.test.lt.sdksamples.core.socket`
- `com.ibm.rational.test.lt.sdksamples.recorder.socket`
- `com.ibm.rational.test.lt.sdksamples.models.behavior.socket`
- `com.ibm.rational.test.lt.sdksamples.codegen.socket`
- `com.ibm.rational.test.lt.sdksamples.protocol.socket`
- `com.ibm.rational.test.lt.sdksamples.testgen.socket`
- `com.ibm.rational.test.lt.sdksamples.editor.socket`
- `com.ibm.rational.test.lt.sdksamples.results.socket`
- `com.ibm.rational.test.lt.sdksamples.datacorrelation.testgen.socket`
- `com.ibm.rational.test.lt.sdksamples.datacorrelation.execution.socket`

Protocol extension structure

Before organizing your extension, note the restrictions and guidelines provided in this topic.

Restrictions

The following restrictions apply to plug-ins:

- The execution portion of the plug-in must be deployed to an agent. You cannot assume that the Software Development Workbench is available.
- The execution portion of the plug-in code cannot depend on any workbench code.
- The execution portion of the code cannot depend on the workspace because none is available on the agent.

For best results, minimize the amount of code that is deployed to the agent for execution.

Guidelines for naming plug-ins

Most base HCL OneTest™ Performance plug-ins follow this naming convention:

`<prefix>.<component>[.<subcomponent>].<protocol>`

where:

prefix: The prefix `com.ibm.rational.test.lt` is used for all load-test-specific HCL OneTest™ Performance plug-ins.

component: One of the extension components: `testgen`, `datacorrelation`, `testeditor`, `codegen`, `execution`, or `models`.

subcomponent: Some plug-ins contain subcomponents (for example, `execution.ui` contains the UI portion of execution), or they might have separate code based on their use in a different component (for example, `datacorrelation.testgen` and `datacorrelation.execution`).

protocol: For example, `http`, or `sap`. (Some plug-ins use `core` for the base plug-in.)

Using these conventions, the extension can have the following plug-ins:

- `com.ibm.rational.test.lt.recorder.protocol`
- `com.ibm.rational.test.lt.testgen.protocol`
- `com.ibm.rational.test.lt.models.protocol`
- `com.ibm.rational.test.lt.testeditor.protocol`
- `com.ibm.rational.test.lt.sdksamples.datacorrelation.testgen.protocol`
- `com.ibm.rational.test.lt.sdksamples.datacorrelation.execution.protocol`
- `com.ibm.rational.test.lt.codegen.protocol`
- `com.ibm.rational.test.lt.execution.protocol`
- `com.ibm.rational.test.lt.execution.results.protocol`

Alternatively, you can divide the code into two plug-ins, one for the workbench and one for execution. This has the advantage of deploying fewer plug-ins. For an example, refer to the Siebel extension to Performance Tester.

Using this method, you would have the following two plug-ins:

- `com.ibm.rational.test.lt.protocol`
- `com.ibm.rational.test.lt.protocol.execution`

Within these plug-ins, you can either arrange the components in one of two ways:

- As separate source folders
- As separate packages within a single source folder

Extending the test recorder

To generate a test, the data exchanged by an application and another entity such as a server must be recorded. The recorded data is then processed to generate a test.

Data is recorded in a *recording session*. In a recording session, clients are launched in parallel with recorders that capture the data that the clients and the servers (or other external entities) exchange. All data that is captured during a recording session is stored in a file.

You can contribute to the recording framework in several ways:

- By defining a new kind of captured data. Captured data is implemented as *recorder packets*. You can define a new type of recorder packet in which to store relevant information for a specific kind of captured data. The product comes with a set of recorder packet types. The packet types include proxy packets, socket packets, HTTP packets, SAP packets, and Citrix packets.
- By developing a new recorder to capture a new kind of data or to use a different recording technique. The product comes with a set of recorders. The bundled recorders include the HTTP proxy recorder, the SOCKS proxy recorder, the socket recorder, the SAP Recorder, and the Citrix recorder.
- By developing a new client whose data will be recorded. A client is not necessarily an application, but rather the ability to start an application. The product comes with a set of clients. The clients include: Application, Manual, Microsoft™ Internet Explorer, Mozilla Firefox, Apple Safari, Opera, Google Chrome, Generic Service Client, SAP client, Citrix client, and TN3270 client.
- By defining a new recorder-client binding. A recorder-client binding declares that a recorder is able to capture the data that a client sends or receives. A recorder-client binding can also define a client decorator, which configures a client so that the client can be recorded by the recorder. For example, the product comes with a decorator that can modify Internet Explorer settings so that the browser sends its traffic through a proxy recorder.

After the recorder framework has been defined, the user interface (UI) can be extended so that the new recording components can be used. The following elements are extensible:

- Packet types, recorders, and clients. You can assign specific icons to these elements.
- Wizards. You can define wizards for configuring a specific client, a specific recorder, several recorders together, or a client and several recorders together.

The recording framework has three main UI contributions:

- The **New Recording Session** wizard. With this wizard, you can select the client to use, and then select the recording technique (in other words, the recorder), and finally set the configuration details for the client and the recorder. Extensions can contribute to this wizard. For more information, see [Defining wizards for recorders and clients on page 696](#).
- The **Recording Control** view. This view shows the active recording session and the recorders and clients that are involved. This view shows a summary of captured data, and messages that the recorders and clients have issued. Extensions can contribute to labels and icons that are displayed in this view. Any user message that a recorder or a client sends is also displayed in this view.
- The **Recording Session Editor**, which opens when you double-click a recording session (.recsession) file. The editor shows the same information as the recording control view, but includes more details about the captured data. Extensions can contribute actions, labels, and icons to this editor.

Defining a new type of captured data

The data captured by a recorder must be stored in a file and consumed by a test generator to produce a test. Captured data must be defined so that it can be stored by the recorder and consumed by the test generator.

To define a new type of recorded data, complete these tasks:

- Declare a new type of recorder packet in the `plugin.xml` file, using the `com.ibm.rational.test.lt.recorder.core.recorderPacket` extension point.
- Define one or more implementation classes, in the same plug-in that implement the interface: `com.ibm.rational.test.lt.recorder.core.packet.IRecorderPacket`.

These factors apply to `IRecorderPacket` class implementations:

- You can define as many fields as required to store your data.
- Because these classes are instantiated by a recorder, they must expose a way to construct themselves.
- Because a test generator uses these classes, the classes must expose ways to retrieve information for the test generator. In other words, they must expose getter methods.
- You must make the classes serializable. Take this into account when designing the classes. The classes must not have any references to data that you do not want to include in the stored information. Fields that include references to data that you do not want stored must be declared using the `transient` keyword. If you plan to have evolutions to these classes, make sure that these evolutions are compatible with earlier versions.

Recorder packets have these common features:

- They have a type. This must be a type ID that is declared in an extension to the `com.ibm.rational.test.lt.recorder.core.recorderPacket` extension point. The implementation class must be in the same plug-in where the type ID is declared.
- They have a start time and end time. This is important for sorting captured packets, as recorders typically do not send the packets at the exact same time as they are captured. Many packets have the same time for start and end events, because they are atomically captured. Packets do not have to have different start and

end times. Packet end times must, however, be higher or equal to the start time. All timestamps must be expressed in units that the framework provides. The units must be of the highest accuracy that the system permits.

For best results, follow these practices for implementing recorder packets:

- If you have more than one type of data, define an interface that all implementation classes implement.
- Define an interface for each concrete implementation class, which exposes only read-only features of the class. The test generator requires access only to the interfaces, while the recorder requires access to the classes.
- Do not define redundant or computable fields. Because the classes are serialized, a single additional field might make the recording file much larger if many packets are stored. Declare a redundant or computable field using the `transient` keyword.

Defining a new recorder

To capture a new kind of data or an existing kind of data with a new recording technique, define a new recorder.

The only required task for a recorder is to capture data. The recorder does not start or configure a client. If you need to start a client or configure a client so that the client can be recorded by a recorder, see [Defining a new client on page 693](#) and [Defining how a recorder can record a client on page 695](#).

To define a new recorder:

- Declare a new recorder type in the `plugin.xml` file. Use the `com.ibm.rational.test.lt.recorder.core.recorder` extension point.
- Determine whether the recorder can be run locally, within the workbench, or if it must run remotely in a separate Java™ virtual machine (JVM).
- Define a delegate, which is the implementation class of the recorder. The delegate starts and stops the recorder, and captures data.
- If the recorder is remote, define a remote launcher implementation that provides details about the JVM that runs the delegate.
- Optionally, define a prerequisite validator that performs basic verification that the recorder can be run on the current computer.
- Declare which types of recorder packets are emitted by the recorder.

Follow these requirements for implementing the `IRecorderDelegate` interface:

- This interface starts and stops the recorder and provides notification about events such as "recorder started," "recorder stopped," and "packet captured." You must provide an implementation of this interface. Optionally, a recorder can support being paused and resumed. If the recorder does not support pause and resume operations, ensure that the implementations of the methods are empty.

- Most methods are asynchronous. In other words, the framework does not require that the operation be completed when the method returns. For this reason, a recorder delegate must notify the framework when an operation is complete.
- A recorder can have a configuration. The configuration is built either by using a XML file with the .reconfig extension or by using a wizard. For an example of an XML file, open an existing recording session file, and then click **File > Save recording configuration as**. The configuration is stored in a RecorderConfiguration object, which is a map of strings to various types of objects. A recorder delegate reads its configuration in its initialize() method.
- A recorder delegate is given a context in its initialize() method. This context enables the recorder to send notifications of events, record log messages, and send captured packets.
- Typically, you extend the BaseRecorderDelegate class rather than directly implement `IRecorderDelegate` interface. The base abstract class provides a basic behavior for most methods, so that you can override only those needed.
- A recorder must use the getContext().packetCaptured() method to notify the framework when it has captured data.
- When a recorder constructs a packet, the packet must be filled with a recorder ID. The recorder ID is attributed by the framework and can be retrieved using the getContext().getComponentUniqueId() method.
- When a recorder constructs a packet, the packet must be filled with time information. The time information must be expressed in a unit that the framework defines. The current time, expressed in the framework units, can be retrieved using the getContext().currentTime() method.
- A recorder delegate can send messages to the user by invoking the sendUserMessage() method or the getContext().sendMessage(new UserMessage(...)) method if the delegate does not extend the BaseRecorderDelegate class.
- A recorder delegate can be enabled to communicate with outside entities by setting or retrieving dynamic properties or by receiving messages. A recorder delegate is useful only if you develop a specific UI for the recorder or if you define a client decorator for the recorder.

Defining a new client

To produce data, a recording session must start at least one client. Several clients come with the product. To automate starting a specific application, you can define a new client.

The only task for the client is to start an application and to notify the recording framework about client life cycle events. To configure a client so that it can be recorded by a recorder, see [Defining how a recorder can record a client on page 695](#).

Sometimes there is no client to launch because the client already exists or because the client is a system that has its own life cycle. In this case, you can use the Manual client that comes with the product. The ID of the Manual client is com.ibm.rational.test.lt.recorder.core.manualClient.

To define a new type of client, complete these tasks:

- Declare a new client type in the `plugin.xml` file by using the `com.ibm.rational.test.lt.recorder.core.client` extension point.
- Determine whether the client can be run locally within the workbench, remotely in a separate JVM, or in the same JVM as the recorder to which it is bound.
- Define a delegate, which is the implementation class of the client. The delegate starts and stops the client and notifies the framework when the client is started or closed.
- If the client is remote, define a remote launcher implementation that provides details about the JVM that runs the delegate.
- Optionally, define a prerequisite validator that performs basic verification that the client can be run on the current computer.

To implement the `IClientDelegate` interface:

- You must provide an implementation of this interface. This interface starts and stops the client and sends notifications of events such as "client started" and "client stopped."
- The `start()` and `stop()` methods are asynchronous. In other words, the operation does not need to be complete when these methods return. For this reason, a client delegate must notify the framework when an operation is complete.
- The `stop()` operation is invoked only when the user clicks **Stop** in the user interface. In many cases, the user gestures in the application itself to close the application. The delegate must monitor the application and notify the framework when the application is closed.
- A client can have a configuration. The configuration is built either by using an XML file, which is a file with the `.reconfig` extension or by using a wizard. For an example of an XML file, open an existing recording session file, and then click **File > Save recording configuration as**. The configuration is stored in a `ClientConfiguration` object, which is a map of strings to various types of objects. A client delegate reads this configuration in its `initialize()` method.
- A client delegate is given a context in its `initialize()` method. This context enables the recorder to send notifications of events and record log messages.
- Typically, you extend the `BaseClientDelegate` class rather than directly implement the `IClientDelegate` class. The base abstract class provides a basic behavior for most methods. You can override specifically those that you must override.
- A client delegate can send messages to the user by calling the `sendUserMessage()` method, or by calling the `getContext().sendMessage(new UserMessage(...))` method if the delegate does not extend the `BaseClientDelegate` class.
- A client delegate can be enabled to communicate with outside entities by setting or retrieving dynamic properties, or by receiving messages. The client delegate is useful only if you develop a specific UI for the client or if you define a client decorator for the client.

If your client launches a specific process, you can extend the `com.ibm.rational.test.lt.recorder.core.clients.ProcessBuilderClientDelegate` class. In this case, you need to extend only the `initialize()` method, and then invoke setter methods to set up the command line, arguments, environment variables, and working directory.

Defining how a recorder can record a client

All recorders cannot record all clients. The framework must be notified that a recorder can record a client. In many cases, the recorder must configure a client before recording, and then undo these configuration actions when the client is closed. In these cases, you can declare a client decorator. A client decorator is a class that configures a client so that it can be recorded by a recorder.

To declare that a recorder can record a client:

- Declare a new recorder-client binding in the `plugin.xml` file by using the `com.ibm.rational.test.lt.recorder.core.recorderClientBinding` extension point.
- Specify the IDs of the recorder and the client that are compatible.
- Optionally, declare a client decorator. A client decorator is a class that modifies a client so that it can be recorded by the recorder.

To implement a client decorator, define an implementation of the `com.ibm.rational.test.lt.recorder.core.extensibility.IClientDecorator` class.

The implementation has two main methods: `decorate()` and `undecorate()`. The `decorate()` method is called when the recorder starts running, but before the client is launched. The `undecorate()` method is called after the client is closed, but before the recorder is stopped.

The decorator can interact with the recorder delegate and the client delegate by setting or getting properties from them. The methods for doing so are available in the decorator context. The client and the recorder must support the properties.

Typically, you extend the `BaseClientDecorator` class rather than directly implementing the `IClientDecorator` class. The base abstract class provides a basic behavior for most methods. With that class, you can override specifically those methods that you must override.

Running recorders and clients without a UI

To test recorders, clients, and decorators during the development process, you can run those elements before any UI components are ready to start the recorders, clients, and decorators.

You can start a recording session that includes the recorders or clients that you have developed by using a recording session configuration file. This file specifies which recorders and clients to start and the options for the recorders and clients to use.

The following file is an example of such a configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<__PT_EXTERNAL__:session xmlns:__PT_EXTERNAL__="__PT_EXTERNAL__">
  <rec:myRecorder <-- recorder configuration -->
    xmlns:rec="recorder:org.xyz.myplugin"
    option1="value1"
    option2="value2"
    id="myRecorder1"/>
  <cli:myClient <-- client configuration -->
```

```

xmlns:cli="client:org.xyz.myplugin"
optionA="valueA"
id="myClient1" />
<__PT_EXTERNAL__:binding client="myClient1" recorder="myRecorder1" />
</__PT_EXTERNAL__:session>

```

In the preceding example, replace *org.xyz.myplugin* with the name of the plug-in that defines the recorder and the client. Replace *myRecorder* with the recorder ID and *myClient* with the client ID. The *option1* and *option2* attributes can be replaced by attributes that the recorder supports. You can replace the *optionA* attribute with an attribute that the client supports. You must save the file with the *.reconfig* extension.

In the configuration file, a session node can contain as many recorder configurations and client configurations as required. All the recorders and clients that are referred to are launched together and the options from the configuration file are passed to them. All recorders and clients must have **id** attributes so the recorder and clients can be referred to in a binding node. Examples of recording session configuration files can be generated by opening an existing recording session and clicking **File > Save Recording Configuration As**.

To start a recording session from a recording configuration file, right-click the file, and then select **Start Recording Session**.

Defining wizards for recorders and clients

You can define wizards for recorders and clients.

The **New Recording Session** wizard runs in the following sequence:

1. The recording session file is selected.
2. The client is selected.
3. The recording method is selected.
4. The client wizard pages are displayed.
5. The recorder wizard pages are displayed.

Steps 4 and 5 can be combined into a single step. Steps 4 and 5 are extension contributions.

Declare a wizard for each client and each recorder that you define. A client or a recorder that has no declared wizard is not available in the user interface and can be started only with a recording configuration file or programmatically.

When the **New Recording Session** wizard is complete, the wizard produces a recording session configuration that contains a client configuration and one or more recorder configurations. The recording session configuration is used to start a recording session and to instantiate the corresponding recorders and clients.

Several types of wizards can be defined, depending on how you have defined recorders and clients:

- If you have defined only a recorder and you plan to use a client that comes with the product, define a recorder wizard.
- If you have defined one or more types of clients and one or more types of recorders, and you want to combine clients and recorders in several ways, define a wizard for each client and a wizard for each recorder.
- If you have defined one recorder type and one client type to be used together, consider defining a unique wizard that configures both of them.

Additional information about the **New Recording Session** wizard:

- The recording method selection step is displayed only if there is more than one recording method available for the client that is selected in step 2.
- Wizards that configure both recorders and clients take precedence over separate wizards for clients and recorders.

To define a client wizard:

- Declare a new client wizard in the `plugin.xml` file by using the `com.ibm.rational.test.lt.recorder.ui.clientWizard` extension point.
- Specify the client ID that the client wizard configures.
- Optionally, provide an implementation class. If you do not provide a class, the wizard has no configuration page. If you provide a class, it must extend the `com.ibm.rational.test.lt.recorder.ui.wizards.NewClientWizard` class.

About the `NewClientWizard` class implementations:

- This class extends the `JFace` class wizard, so the class must extend typical methods such as `addPages()`.
- The class is passed an empty client configuration, typed with the client ID selected by the user in step 2.
- The class sets the client configuration options, which are available using the `getClientConfiguration()` method. This configuration is typically done in the `doPerformFinish()` method.

To define a recorder wizard:

- Declare a new recorder wizard in the `plugin.xml` file by using the `com.ibm.rational.test.lt.recorder.ui.recorderClientWizard` extension point and the `recordersWizard` element.
- Specify one or more recorder IDs that the recorder wizard configures.

If the recorder wizard declares configurations for more than one recorder, examine the **performsRecorderSelection** attribute. This attribute specifies whether the wizard enables the user to choose which recorders to use or if the framework chooses which recorder to use. Depending on the conditions, the attribute then calls the wizard with the recorder ID that the user selected. In the first case, the recording method selection step is always displayed, whereas in the latter case, the recording method selection step can be skipped. In the first case, the wizard can enable several recorders, whereas in the latter case, only one

recorder is enabled. Choose the first option if you need to enable more than one recorder in one recording session.

- Provide an implementation class. This is optional only if you have associated the wizard with exactly one recorder or if the **performsRecorderSelection** attribute is `false`. If you do not provide a class, the wizard has no configuration page. If you provide a class, it must extend the `com.ibm.rational.test.lt.recorder.ui.wizards.NewRecordersWizard` class.

About `NewRecordersWizard` implementations:

- This class extends the JFace class `Wizard`, so it must extend typical methods such as `addPages()`.
- If the **performsRecorderSelection** value is `true`, the class is passed null to its `initialize()` method and it must invoke the `setRecorderConfigurations()` method from its `doPerformFinish()` method, with the recorder configurations for each recorder that must be included in the recording session.
- If the **performsRecorderSelection** value is `false`, the class is passed an empty recorder configuration, typed with the recorder ID selected in step 3, to its `initialize()` method and it must fill the empty recorder configuration with recorder options in its `doPerformFinish()` method.

To define a wizard that configures a client and one or more recorders altogether:

- Declare a new recorder and client wizard in the `plugin.xml` file by using the `com.ibm.rational.test.lt.recorder.ui.recorderClientWizard` extension point and the `combinedWizard` element.
- Specify the client ID that this wizard configures.
- Specify one or more recorder IDs that this wizard configures.
- If this wizard declares configurations for more than one recorder, examine the **performsRecorderSelection** attribute. This attribute specifies whether the wizard lets the user choose which recorders to use or if the framework chooses which recorder to use, and then invokes the wizard with the recorder ID selected by the user. In the first case, the recording method selection step is always displayed, whereas in the latter case, the recording method selection step can be skipped. In the first case, the wizard can enable several recorders, whereas in the latter case, only one recorder is enabled. Choose the first option if you need to enable more than one recorder in one recording session.
- Provide an implementation class. This is optional only if you have associated the wizard with exactly one recorder or if the **performsRecorderSelection** attribute is `false`. If you do not provide a class, the wizard has no configuration page. If you provide a class, it must extend the `com.ibm.rational.test.lt.recorder.ui.wizards.NewRecorderClientWizard` class.

About `NewRecorderClientWizard` class implementations:

- This class extends the JFace class `Wizard`, so it must extend typical methods such as `addPages()`.
- If **performsRecorderSelection** is `true`, the class is passed a null recorder configuration to its `initialize()` method and it must invoke the `setRecorderConfigurations()` method from its `doPerformFinish()` method, with the recorder configurations for each recorder that must be included in the recording session.

- If **performsRecorderSelection** is `false`, the class is passed an empty recorder configuration, typed with the recorder ID selected in step 3, to its `initialize()` method and it must fill the empty recorder configuration with recorder options in its `doPerformFinish()` method.
- The class is passed a client configuration, typed with the client ID selected in step 2, to its `initialize()` method. It must fill the client configuration with client options in its `doPerformFinish()` method.

Migrating recorder implementations from previous versions

The new recording framework includes significant improvements over the generic recorder framework (GRF) that was delivered in previous versions. The new framework introduces more flexibility for combining clients and recorders. With the framework, you can start multiple recorders and clients in one session, consolidating the recorded data in one file. In addition, the framework does not use the TPTP Agent Controller, and thus does not require recorders and clients to be run in a separate Java™ Virtual Machine. Finally, the new framework provides improved performance and better scalability in terms of the amount of recordable data and the impact on memory usage.

The API has been completely redefined to produce these improvements. Migrating existing recorder implementations to the new framework requires significant effort. The following tables summarize the changes to classes and methods in the new recording framework.

Previous class	New recording framework, version 8.2 and later	Comments
<code>com.ibm.rational.test.lt.trace.PayloadMsg</code>	<code>com.ibm.rational.test.lt.recorder.core.packet.IRecorderPacket</code> <code>com.ibm.rational.test.lt.recorder.core.packet.connection.IConnectionPacket</code>	Use Java™ serialization for serialization in this product version. If packets support connections, extend the <code>IConnectionPacket</code> interface and its subinterfaces. This extension enables filtering capabilities at test-generation time.
<code>org.eclipse.hyades.execution-recorder.remote.RecorderAgent</code>	<code>com.ibm.rational.test.lt.recorder.core.extensibility.BaseRecorderDelegate</code>	The <code>run()</code> method has been replaced with the <code>start()</code> method. The threads that this method can start are no longer monitored by the framework. The <code>setIsReady(true)</code> method has been replaced with the <code>sendStarted(enabled)</code> method. The <code>handleCommand(STOP)</code> method has been replaced with the <code>stop()</code> method. The class calls the <code>sendStopped()</code> method when the recorder has stopped. (This event was previously implicitly notified by the termination of the thread returned by the <code>run()</code>

Previous class	New recording framework, version 8.2 and later	Comments
		method.) This class is no longer required to run in a separate JVM.
org.eclipse.hyades.execution-recorder.remote.RecorderEnvironmentAdapter org.eclipse.tptp.test-provisional.recorder.framework-AbstractRecorderExecOptionsProvider org.eclipse.hyades.execution-recorder.remote.RecorderExecutableObjectAdapter	com.ibm.rational.test.lt.recorder-core.deploy.IRemoteLauncher	This class is required only if the recorder delegate must run in a separate JVM. With this version, you can add classpath entries and system properties to the launched JVM. There is no requirement for file deployment because the JVM is always run on the local computer. Recording on remote computers was never enabled in previous releases. The AbstractRecorderExecOptionsProvider.getAgentClassPath() and RecorderExecutableObjectAdapter.getAgentClassPath() classes are replaced by the delegate class declaration in the extension point.
org.eclipse.tptp.test.provisional-recorder.messages.AbstractRecorderMessageProvider	None	The IRecorderPacket auto-serialization class and the framework handle message serialization in this version. Implementing this class is no longer required.
org.eclipse.tptp.test.provisional-recorder.ui.wizards.DefaultRecWizardProvider	com.ibm.rational.test.lt.recorder.ui-wizards.NewRecorderWizard com.ibm.rational.test.lt.recorder.ui.wizards-NewRecorderClientWizard	
org.eclipse.tptp.test.provisional-recorder.framework.RecorderClientHelperAdapter	None	The framework automatically handles the step that this class completed in earlier versions.
Previous extension point	New recording framework, version 8.2 and later	Comments
org.eclipse.hyades.test.core-Recorder#Recorder.protocol	com.ibm.rational.test.lt.recorder-core.recorder#recorder.outputPacket	

Previous extension point	New recording framework, version 8.2 and later	Comments
org.eclipse.hyades.test.core- .Recorder#Recorder.id	com.ibm.rational.test.lt.recorder- .core.recorder#id	
org.eclipse.hyades.test.core- .Recorder#Recorder.name	com.ibm.rational.test.lt.recorder- .core.recorder#name	
org.eclipse.hyades.test.core- .Recorder#Recorder.recorderAgent	com.ibm.rational.test.lt.recorder- .core.recorder#recorder.delegate	
org.eclipse.hyades.test.core- .Recorder#Recorder.recorderClient- Helper	None	The framework automatically handles the step that this class handled in earlier versions.
org.eclipse.hyades.test.core- .Recorder#Recorder.recorderMessageHandlers	None	The IRecorderPacket auto-serialization class and the framework handle message serialization in this version. Implementing this class is no longer required.
org.eclipse.hyades.test.core- .Recorder#Recorder.wizardPage- Provider	com.ibm.rational.test.lt.recorder.ui- .recorderClientWizard#recordersWizard.class	
org.eclipse.hyades.test.core- .Recorder#Recorder.execOptions- Provider	com.ibm.rational.test.lt.recorder- .core.recorder#recorder.remote- Launcher	See the previous comment about the AbstractRecorderExecOptionsProvider class.
org.eclipse.hyades.test.core- .Recorder#Recorder.requiresIntermediateFile	None	This option is no longer supported. Recording sessions are always kept after recording.
org.eclipse.hyades.test.core- .Recorder#Recorder.fileExtension	None	This option is no longer supported. The recording format and extension are now always controlled by the framework in a .recsession file.
org.eclipse.hyades.test.core- .Recorder#Recorder.icon	com.ibm.rational.test.lt.recorder.ui- .recordingUIImage#recorderImage.i- con	
org.eclipse.hyades.test.core- .Recorder#Recorder.description	com.ibm.rational.test.lt.recorder.ui- .recorderClientWizard#recordersWizard.description	

Extending the test generation framework

Test generation consists of processing the recorded data and producing a test.

The recording-session file that is produced during recording provides the input for the test generation operation. Before writing a test generator, you must identify the type of data that the generator uses.

Recording and test generation are typically chained when you use the **New Test From Recording** wizard, but this relationship is not always established. The user can choose to produce only a recording, using the **New Recording Session** wizard, and can also choose to generate a test from an existing recording.

Test generation happens in the following stages and phases:

- The conversion stage is a preliminary step, during which the original packets can be filtered, sorted, aggregated, or converted to a different, typically higher-level, protocol. Extensions can contribute additional converters. For more information, see [Defining a new packet converter on page 704](#).
- The test generation stage consists of using the recorder packets that are sent from the converter stage, and then distributing the recorder packets to the appropriate test generator. The test generator then produces the corresponding model elements in the test model. Extensions can contribute new test generators for processing a new type of recorded data or producing a new type of test element. For more information, see [Defining a new test generator on page 702](#).
- The data processing phase happens after a raw test has been completely generated. During this phase data correlation and data transformation are performed.
- The test splitting phase is an optional step that runs if split points were inserted during the recording. The complete test is split into several tests.

The test generation framework also defines two wizards:

- The **Generate Test** wizard opens either when the user gestures to generate a test from an existing recording or automatically after a recording is complete. If user input is not required, the **Generate Test** wizard is not displayed, and test generation automatically follows the recording. Extensions can contribute to this wizard. For more information, see [Defining a test generator wizard on page 706](#).
- The **New Test From Recording** wizard is the highest-level wizard and combines the **New Recording Session** wizard and the **Generate Test** wizard. For more details about the recording aspects of this wizard, see [Defining wizards for recorders and clients on page 696](#). For more details about the test generation aspects of this wizard, see [Defining a test generator wizard on page 706](#).

Defining a new test generator

A test generator uses recorder packets and produces test-model elements that the test generator adds to the test model.

Before defining a test generator, you must identify the type of data that the generator can use and the type of model elements that the generator produces. The input data for a test generator can either be the raw data that

the recorders produced, or data that has been altered during the conversion stage. The output elements of a test generator are typed by *feature*, which usually corresponds to a specific protocol.

To define a new test generator:

- Declare a new test generator type in the `plugin.xml` file, using the `com.ibm.rational.test.lt.testgen.core3.testGenerator` extension point.
- Assign the test generator a unique ID and a name.
- Declare which packet types the test generator can use.
- Declare the required properties of the packet stream that is sent to the test generator. The framework includes the necessary converters in the conversion stage so these properties are verified when the packets reach the test generator. For information on defining properties by using converters, see [Defining a new packet converter on page 704](#).
- Define an implementation class that implements the `com.ibm.rational.test.lt.testgen.core.testgen.ITestGenerator` interface.

The product includes a built-in converter that produces an ordered stream of packets that is based on start time stamps of the packets. Typically, recorders produce packets that are sorted by their end time stamps. Most test generators require that input packets be ordered by their start time stamps, so the packet stream must include the **sorted** property in the required properties.

Consider these facts about `ITestGenerator` implementations:

- Typically, you extend the `com.ibm.rational.test.lt.testgen.core.testgen.BaseTestGenerator` class, which provides a basic implementation and only requires overriding the necessary methods.
- The `initialize()` method is where any options from the test generator configuration are read using the `getContext().getConfiguration()` method.
- The `process()` method is the most important. The framework calls this method for each input packet. This method creates model elements and adds the elements to the output test. Model elements must be added to the test using methods from the `ITestStack` object that is returned by the `getContext().getStack()` method.
- The framework calls the `complete()` method after all packets have been sent to the test generator. The `complete()` method can be used for performing any post-processing operations.
- Use the `getContext().logMessage()` method to report any messages from the test generator to the user, including error messages. If the message pertains to an unrecoverable error, the framework stops the test generation process.



Note: A test generator must not delay in adding elements to a test. Test generators must add elements to the test as soon as they are created. A test generator can still add data to an element after the element has been added to a test. If there are delays in adding elements to a test, generated elements might not be correctly ordered. For example, if another test generator also generates elements in a mixed protocol environment, or if the recording contains annotations, then the generated elements might not be in the correct order. If you still need to perform processing that would delay the insertion of an element into to the test (for example,



accumulating data in order to build a higher-level object), write a converter that does the processing and insert this converter before the test generator starts.

Defining a new packet converter

A packet converter transforms a stream of recorder packets. Use packet converters for adapting the raw data that recorders capture into a suitable format for the test generators to use.

A converter typically follows one of these patterns:

- A filtering converter removes input packets that do not meet a specific criterion. This converter does not modify packets, nor does the converter introduce new packets in the output stream. The product comes with a generic converter, with the `com.ibm.rational.test.lt.testgen.core3.filter` ID. You can add parameters to this converter with conditions.
- An annotator converter does not remove or modify packets in the input stream, but rather introduces additional packets in the output stream. These packets are *annotation* packets that convey additional information that are inferred from the other packets. For instance, a converter might look for session, connection, or page boundaries in a packet stream, and then add a boundary packet whenever a boundary is detected in the input stream. This assists the test generator in identifying boundaries without the need to look ahead in the packet stream.
- A reordering converter does not add, change, or remove packets from the input stream, but it outputs them in a different order. A typical example is the packet sorter that comes with the product, which outputs packets sorted by their start time stamp. The sorter ID is `com.ibm.rational.test.lt.testgen.core3.packetSorter`.
- An aggregator converter has different input and output packet types. It aggregates multiple input packets into one output packet. The converter usually translates a lower-level protocol into a higher-level protocol. For example, the product comes with a converter that transforms raw data in a byte stream that is exchanged between a client and an HTTP server into aggregated HTTP packets (request/response pairs).

To define a new converter, you must complete these procedures:

- Declare a new packet converter type in the `plugin.xml` file, using the `com.ibm.rational.test.lt.testgen.core3.packetConverter` extension point.
- Assign the packet converter a unique ID and a name.
- Declare the required properties of the packet input stream that the converter receives. The framework includes the required converters in the conversion stage, so these properties are verified when the packets reach the converter. For instance, if the converter requires the input packets to be ordered according to their start time stamps, specify the **sorted** parameter.
- Declare which properties this converter adds to the output stream or removes from the output stream, as compared to the properties of the input stream. For example, a converter might disrupt the **ordered** property of the input stream; in this case, the **sorted** parameter must be included in `removedProperties` class.

- If the converter has different input and output packet types and can be considered as a packet type converter, declare that it contributes to packet type conversions and specify the types of input and output packets that the converter produces.
- Define an implementation class that implements the `com.ibm.rational.test.It.testgen.core.conversion.IPacketConverter` interface.

Consider these facts about `IPacketConverter` implementations:

- Typically, you extend the `com.ibm.rational.test.It.testgen.core.conversion.BasePacketConverter` class, which provides a basic implementation and only requires overriding the specific methods.
- A packet converter is an `IPacketReferenceOutputStream` interface that writes to another `IPacketReferenceOutputStream` interface. In other words, a packet converter has a `writePacket()` method, which is invoked by the framework for each input packet it processes. The packet converter is responsible for invoking the `getContext().getOutputStream().writePacket()` method whenever it needs to send a packet to its output.
- A packet converter can have options. The options are available by using the `getContext().getConfiguration()` method, which is typically called in the `initialize()` method.
- A packet converter can send additional packets to its output in the `complete()` method. This method is called when there are no more input packets to use.
- Use the `getContext().logMessage()` method to report messages from the test generator to the user, including error messages. If the message pertains to an unrecoverable error, the framework stops the test generation process.

For scalability reasons, converters manipulate `IRecorderPacketReference` objects instead of `IRecorderPacket` objects. Follow these procedures to get the best results when you write packet converter code:

- An `IRecorderPacket` interface can be obtained from a reference that uses the `IRecorderPacketReference.getRecorderPacket()` method.
- A converter that must echo the same packet to its output as the one received must write the same reference instance that the converter has received.
- When a converter instantiates a new recorder packet, the converter can wrap the packet into a new reference by using the `getContext().createPacketReference()` method so that the packet can be sent as output.
- Just as recorders can produce packet attachments, converters can do so as well. To create a new attachment, use the `getContext().createPacketAttachment()` method.
- If the converter must hold a packet a long time before the packet is sent as output, the converter can unload the packet and keep only a reference to the packet. To do so, call the `unload()` method on the packet reference.
- Converters are provided with a facility for efficiently accumulating a large number of packets and atomically discarding them, or flushing them, to the output. See the `com.ibm.rational.test.It.testgen.core.store.IPacketReferenceStore` class for more information. A packet store can be created using the `getContext().createPacketStore()` method.

Generating tests without a UI

To test converters and test generators during the development process, you can run them before any UI component is ready for launching them.

You can start test generation that includes the converters and test generators that you have developed by writing a test generation configuration file. This file specifies which converters and test generators to use and options for each.

This file is an example of such a configuration file:

```
<__PT_EXTERNAL__:testGeneration xmlns:__PT_EXTERNAL__="__PT_EXTERNAL__"
  recession="/Project/MyRecording.recession"
  autoDataCorrelation="true"
  autoDataCorrelationNames="true"
  output="/Project/MyTest.testsuite">
  <cnv:packetSorter xmlns:cnv="converter:com.ibm.rational.test.lt.testgen.core3"/>
  <cnv:myConverter xmlns:cnv="converter:org.xyz.myplugin" />
  <gen:myTestGenerator xmlns:gen="generator:org.xyz.myplugin" />
</__PT_EXTERNAL__:testGeneration>
```

In the preceding example, replace *org.xyz.myplugin* with the name of the plug-in that defines the converter and the test generator. Replace *myConverter* with the converter ID, and *myTestGenerator* with the test generator ID. Save the file with the *.testgenconfig* extension.

In the configuration file, a *testGeneration* node can contain as many converter configurations and test generator configurations as required. All the referenced converters and test generators are included in the test generation process, with the options from the configuration file passed to them.

To launch test generation from a test generation configuration file, right-click the file, and then select **Generate Test**.

Defining a test-generation wizard

The **Generate Test** wizard contains four sections to guide you in defining a test generator.

The **Generate Test** wizard contains these sections:

1. Test generator selection.
2. Test files selection.
3. Data correlation options.
4. Test generator wizard.

The first page is displayed only if more than one test generator is applicable for the input recording session. The second page is displayed only if the user has not already chosen the test file before the recording.

You must declare a wizard for each test generator that you have defined. A test generator that has no declared wizard is not available in the user interface and can be launched only using a test generation configuration file, which is a file with the *.testgenconfig* extension, or programmatically.

When completed, this wizard produces a test-generation configuration that contains a set of converter and test generator configurations. This configuration is applied by the test generation framework to instantiate the appropriate

converters and test generators and to send the packet stream from the recording session to the converters and test generators.

Test-generator wizards can add pages to the **Generate Test** wizard by specifying an implementation class in the test generator wizard declaration in the `plugin.xml` file.

To define a test generator wizard:

- Declare a new test generator wizard in the `plugin.xml` file by using the `com.ibm.rational.test.lt.testgen.ui.testgenWizard` extension point.
- Specify the test generator ID that the wizard configures.
- Provide a label and icon that represents the type of test to be generated by the test generator.
- Optionally, provide an implementation class. If you do not provide a class, the wizard has no configuration page. If you provide a class, it must extend the `com.ibm.rational.test.lt.testgen.ui.wizards.NewTestGeneratorWizard` class.

Consider these facts about `NewTestGeneratorWizard` implementations:

- This class extends the JFace class wizard, so it must extend typical methods such as the `addPages()` method.
- The class is passed an empty test-generator configuration, typed with the test generator ID selected by the user in step 1.
- The class is responsible for setting the test generator configuration options, which are available by using the `getTestGeneratorConfiguration()` method. This method is typically used in the `doPerformFinish()` method.
- The wizard can get contextual information by using `getContext()` method.

Migrating test generator implementations from previous versions

The new test generation framework builds on the improvements that were made in the recording framework. These improvements include the ability to record several protocols at the same time, which in turn supports generating a single test with mixed protocols. The test generation framework also includes improvements in efficiency and scalability.

The test generation API has been completely redefined. You must refactor existing extension code to use the new framework. The following tables summarize the changes to classes and methods in the new test generation framework.

Previous class	Current® class	Comments
<code>com.ibm.rational.test.lt.testgen-core2.IC2ProtocolHandler</code>	<code>com.ibm.rational.test.lt.testgen.core-testgen.BaseTestGenerator</code>	The class no longer has to determine whether it supports a packet type. The framework sends to the test generator only packets that are declared to be supported by the test generator in the <code>plugin.xml</code> file. Previously, protocol handlers loaded mes-

Previous class	Current® class	Comments
		sages, then processed them in the process() method. In the new framework, packets are passed one-by-one to the process() method. The process() method generates model elements without delay. The complete() method is available for any post processing.
Previous extension point	Current® extension point	Comments
com.ibm.rational.test.lt.testgen-core2.protocolHandler	com.ibm.rational.test.lt.testgen-core3.testGenerator	

Contributing annotations

An annotation is an action that a user performs during a recording session. Annotations are used to document or structure the test generated from the recording.

These annotation types come with the product:

- Insert comment
- Insert screen capture
- Start transaction
- Stop transaction
- Insert Synchronization Point
- Insert Split Point
- Set page name

Extensions can define additional annotation types. To contribute an annotation type:

- Define the new annotation type and its properties.
- Contribute an annotation toolbar with an action that enables inserting the annotation.
- Process the annotation in a test generator.

Defining a new annotation type

Each annotation type has its own semantics and supports a set of properties.

To define a new annotation type, complete these procedures:

- Declare a new annotation type in the `plugin.xml` file by using the `com.ibm.rational.test.lt.recorder.core.recorderAnnotation` extension point and the `annotationType` element.
- Assign the type a unique ID. Typically the ID is in this form: `pluginName.type`.
- Assign a user-readable name to the type. The name is visible in the recording session editor.

For best results, complete these optional steps:

- Define an interface that contains a string constant with the annotation type ID defined earlier.
- In this interface, include a string constant for each property name that the annotation type supports. Specify the meaning and the type of the property in the constant Javadoc information. Property types can be those that are supported by setters and getters of the `com.ibm.rational.test.lt.recorder.core.property.AbstractConfiguration` class.

Also, define a label provider for the annotation type. A label provider returns a dynamic label that is based on the annotation properties and an image. The label and the icon are visible in the recording session editor. To define a label provider, complete these procedures:

- Declare a new annotation label provider in the `plugin.xml` file by using the `com.ibm.rational.test.lt.recorder.ui.annotationContribution` extension point and the `annotationLabelProvider` element.
- Specify the annotation type that the label provider supports.
- Define an implementation class of the label provider. The implementation class must implement the `JFace ILabelProvider` interface. Any object passed to this interface is always an instance of the `com.ibm.rational.test.lt.recorder.core.annotations.RecorderAnnotation` class.

Contributing new actions to the annotation toolbar

Typically, you produce an annotation by adding an action to the annotation toolbar. This action is represented as a toolbar button.

To contribute a new action to the annotation toolbar, complete these procedures:

- Declare a new annotation action in the `plugin.xml` file by using the `com.ibm.rational.test.lt.recorder.ui.annotationContribution` extension point and the `annotationAction` element.
- Assign a unique ID to the annotation action.
- Decide whether this action is visible by default or not. Actions that are visible by default are available in all recording sessions unless made unavailable by a client or recorder. Actions that are not visible by default are available only if a recorder or client requires the action.
- Define an implementation class for this action. The implementation class must extend the `com.ibm.rational.test.lt.recorder.ui.actions.AbstractAnnotationAction` abstract class.

Consider these facts about `AbstractAnnotationAction` implementations:

- This class extends the JFace Action class.
- In the constructor, set the name, tooltip text, and image descriptor of the action.
- Implement the run() method. This method can interact with the user. For example, this method can prompt the user for a text field. If there is any interaction with the user, the time in milliseconds spent interacting with the user must be measured.
- To create an annotation, create an instance of the `com.ibm.rational.test.lt.recorder.core.annotations.RecorderAnnotation` class, and then pass the annotation type as an argument. Set the annotation properties using the setter methods provided in this class.
- After the annotation has been built, forward the annotation by sending an `AnnotationMessage` message to the annotation recorder. This is typically a call of this form: `annotationRecorder.sendMessage(new AnnotationMessage(annotation, interactionTime));`

Generating a test-model element from a new annotation type

After annotations have been inserted into a recording session, the annotations must be processed by a test generator to translate them into a test-model element. To process annotations with a test generator, define a dedicated test generator or modify an existing test generator that you have developed.

To enable a test generator to process a new annotation type:

- Add the `com.ibm.rational.test.lt.recorder.core.recorderAnnotation` packet type to the list of supported packets of the test generator in the `plugin.xml` file. Set the **isRequired** attribute to false.
- In the `process()` method of the test generator, add code based on this template:

```
if (packet instanceof IRecorderAnnotationPacket) {
    IRecorderAnnotationPacket p = (IRecorderAnnotationPacket) packet;
    RecorderAnnotation annotation = getContext().resolveAnnotation(p);
    if (annotation == null) return true;
    if (MY_ANNOTATION_TYPE.equals(annotation.getType())) {
        // Add code to process the annotation
        return true;
    }
}
```

Advanced annotation concepts

In some rare cases, annotations can be produced by recorders or clients.

Sending annotations from a recorder or a client

Annotations can be created by recorders and clients. To create annotations, the recorder delegate or the client delegate uses the `AnnotationMessage` class. The call typically is in this form:

```
getContext().dispatchMessage(new AnnotationMessage(annotation, interactionTime));
```

Modifying the available annotation actions for a specific recorder or client

If an annotation action is available by default, you can choose to hide it when a specific recorder or client is active during the recording session. If an annotation is hidden by default, you can make it available when a specific recorder or client is active during the session.

To modify the annotation actions for specific recorders or clients, use the

`com.ibm.rational.test.lt.recorder.ui.annotationContribution` extension point and the `annotationActionFilter` element.

Managing a state for a annotation types

By default, annotations can be inserted at any time during recording. There might be cases where permitting users to insert an annotation is inappropriate. For example, in a test with “*Start Transaction*” and “*End Transaction*” annotations, “*End Transaction*” might be available only if a transaction has already been started.

To manage the state of annotation types, define an annotation state handler. An annotation state handler manages one or more annotation types and can individually enable or disable annotation insertions. For more information about annotation state handlers, refer to the

`com.ibm.rational.test.lt.recorder.core.recorderAnnotation` extension point, the `annotationStateHandler` element, and the `com.ibm.rational.test.lt.recorder.core.extensibility.AnnotationStateHandler` class.

The state handler for a specific annotation type can be retrieved from an annotation action class by using this code:

```
stateHandler = (IMyAnnotationStateHandler) getAnnotationStateHandler(myAnnotationType);
```

Extending the load test behavior model

The load test behavior model (LTBM) is a model of the behavior of a performance test in HCL OneTest™ Performance.

About this task

The load test behavior model (LTBM) was developed using Rational® Software Architect and Eclipse Modeling Framework (EMF) to model the behavior of a performance test. The LTBM also consists of code generated from this model using EMF. The LTBM is extended from the common behavior model (CBM). The schedule behavior model (SBM) that is the underlying model to the Schedule Editor in HCL OneTest™ Performance is the other model that extends the CBM. The CBM is itself built on top of the Test and Performance Tools Platform (TPTP) common behavior model (TCBM). The LTBM plug-in ID is `com.ibm.rational.test.lt.models.behavior`.

For a sample of an extension to the LTBM, see the plug-in

`com.ibm.rational.test.lt.sdk.samples.models.behavior.socket`.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

Updates to the load test behavior model

There are new features in the model and there are changes to the following two plug-ins, which the protocol behavior model plug-in depends on.

There are changes to the two plug-ins that the protocol behavior model plug-in depends on:

- `com.ibm.rational.test.common.models.behavior`
- `com.ibm.rational.test.lt.models.behavior`

com.ibm.rational.test.common.models.behavior.CBAssetMigration

To open a performance test in version 7.0 that was created in version 6.1.2, the test suite loader must identify model elements that have been modified in 7.0 and adapt to the changes. The suite loader accomplishes identification and adaptation by recognizing elements that implement the `CBAssetMigration` interface. If a model element has been modified in a given release, starting with that release, that particular model element must implement this interface.

The two methods from this interface that need to be implemented are as follows:

- `Public Boolean needMigration(CBVersion version)`: determines whether migration is needed
- `Public void migrate(CBVersion version)`: performs any required migration

The version that is passed into this method is the version of the test. `LTTestUtil.getCurrentVersion()` always returns the current version of the test suite that can be created with the installed version of the product. The methods in the `BehaviorUtil` class enable you to compare any two version objects.

com.ibm.rational.test.common.models.behavior.CBElementHost

A model element can implement this interface provided that it is a container and that it has children in the model. Some examples of core model elements that implement this interface are `CBLoop`, `LTTransaction`, and `LTTest`.

com.ibm.rational.test.lt.models.behavior.common.LTAnnotation

com.ibm.rational.test.lt.models.behavior.common.impl.LTAnnotationImpl

If a protocol model element is required to hold data that is not text, or that is large, the data being held could affect performance if it persists within the test suite model. To prevent performance issues, these types of data are stored in a file called *annotation*, which is in the test suite, but outside of the model. To use this feature, the attribute that holds the data must be created with type `LTAnnotation`. Use the APIs provided in this interface to access the data.

com.ibm.rational.test.lt.models.behavior.common.LTArmEnabled

A model element can implement this interface if this interface is required to log Application Resource Monitoring (ARM) data. ARM must be supported by the protocol being tested for this feature to work end-to-end.

Extensibility using RSA/EMF modeling

The model can now be extended by using IBM® Rational® Software Architect Eclipse Modeling Framework (EMF). A base starter model is provided along with the Load Test Behavior Model (LTBM) plug-in. You can start with this model

and add the protocol extension model elements to the base model. When generating EMF code, choose the model element that is unique to the protocol model.

Extension points for LTBM

You must register your protocol model elements with the load test behavior model (LTBM) core through extension points regardless of the way that you chose to model your protocol.

Registering a model element

The extension point for registering a model element enables the protocol extension to specify a factory class, a class providing test options, and the type of the model element handled by the protocol extension.

The extension point for registering a model element allows the protocol extension to specify:

- A factory class that contains instructions to create a model element of a given type. The factory class must implement the `ElementFactory` interface from the load test behavior model (LTBM).
- A class that provides test level options to the test for the protocol. This class should extend the `OptionImpl` class from the LTBM.
- A type of model element that is handled by this protocol extension. The element of this type should extend the `CBlockImpl` class in the LTBM. The type defaults to the fully qualified name of the model element class. If there are duplicate element types, the element loaded later is ignored and a message is logged in the error log.

Example

Sample

```
<extension
point="com.ibm.rational.test.lt.models.behavior.protocol">

<protocol
  id="com.ibm.rational.test.lt.example.protocol"
  factory="com.ibm.rational.test.lt.example.protocol.ProtocolElementFactory"
  option=" com.ibm.rational.test.lt.example.protocol.ProtocolOptions">


<element type="com.ibm.rational.test.lt.example.protocol.XModelElement"/>
<element type="com.ibm.rational.test.lt.example.protocol.YModelElement"/>

</protocol>
</extension>
```

Required attributes in a model class

A model element can contain attributes of different types, both primitive and complex.

A model element typically consists of one or more of the following types:

Attribute type	Description
Primitive, except byte or byte arrays	<ul style="list-style-type: none"> • Call the appropriate <code>setProperty()</code> method to set the property of the element. Various overrides exist for all primitive types except byte.
ByteArray or large strings	<ul style="list-style-type: none"> • To store binary data, the attribute must be of <code>LTAnnotation</code> type or one that extends it. This class has <code>setBytes()</code> and <code>getBytes()</code> methods to store and retrieve binary information. After the <code>getBytes()</code> or <code>setBytes()</code> is called, the data is maintained in memory until the test is saved. <p> Note: If this action is not wanted—and it is not in cases where the data set can be large, you can flush the data to a file by individually calling the <code>flush()</code> method on the <code>LTAnnotation</code>. Make sure that you call this method immediately after a set or get is called.</p>
Complex	<ul style="list-style-type: none"> • Call <code>setProperty(CBActionElement,CBActionElement)</code> to which you can pass the old and the new value of the attribute. All model elements extend from the <code>CBActionElement</code> class. The old value is passed so that it can be removed from the model and the new value is passed, so it can be set. If you do not have a value, then pass null. For example, if the value is changing from <i>val1</i> to <i>val2</i>, then call the <code>setProperty(val1,val2)</code> method. If you are setting the value for the first time, call the <code>setProperty(null,val1)</code> method. If you want to clear the value, call <code>setProperty(val1,null)</code>. • This is similar to the other <code>setProperty()</code> methods for the primitive type, with the added restriction that this needs the old value too. • When the model gets loaded, the <code>addReference()</code> method will be called and this method needs to be overwritten to interpret and properly assign the attribute to the object. Call the appropriate <code>setXXX()</code> method to set that attribute with the value passed in.
List	<ul style="list-style-type: none"> • The attribute should be of type <code>EList</code>. To access the list, if the element is a container, then it can implement the <code>LTElementHost</code> interface and will need to implement the <code>getElements()</code> method. This method must return the <code>EList</code>. To load the list when the children of this model element are being loaded, the <code>addReference()</code> method is called for each child and the element must put the child in the right list.

Attribute type	Description
Reference to another model element	<ul style="list-style-type: none"> • Define an attribute of the type of the element you reference. This will provide the simple getters and setters to this attribute. These getters or setters will not call <code>getProperty</code> or <code>setProperty</code>, but they will simply get or set the attribute value. • Define another attribute that will act as a proxy for this element. This element is your own class that extends <code>ProxyElement</code> from the LTBM. • When creating the reference, call the <code>setHref()</code> method of the <code>ProxyElement</code> with the ID (<code>getId()</code>) of the element that is being referenced. • To get the element that is being referenced, use the href from the <code>ProxyElement</code> (<code>getHref()</code>) to locate the element in the test. You can also use the <code>BehaviorUtil.findElement()</code> method (the test containing this element, the ID of the element being searched) to get the element.

Registration examples

The examples in this topic show you how to register strings for content verification and built-in data sources.

Registering strings for content verification

The following example shows how to register strings for content verification:

```
<extension
point="com.ibm.rational.test.lt.models.behavior.contentVPData">

<category
label="Example protocol strings"
id=" examples.protocol.strings ">

<property
type="boolean"
name="Does value exist"
value="true"
id="com.ibm.rational.test.lt.exmaple.protocol.valueExist"/>

<content
label="Joe"
id="example.protocol.strings.1"/>

<content
label="Jane"
id="example.protocol.strings.2"/>

</category>
```

```
</extension>
```

Exemple

Registering built-in data sources

A protocol extension can provide its own list of built-in data sources. A data source is used to mine data and hold it for later consumption. This is done using the following extension point.

The following example shows how to register built-in data sources:

```
<extension
point="com.ibm.rational.test.lt.models.behavior.builtInDataSource">
<dataSource typeId="protocol.buitindatasource"
className="com.ibm.rational.test.lt.example.protocol.BIDataSource">

<property name="database"/>
<property name="table"/>
<property name="column"/>

</dataSource>

</extension>
```

In the above example, a protocol extension can extract a value from a particular column in a table in a database.

Creating protocol constructs

The first step in creating protocol constructs is to identify the behavior of the protocol, how the ends, client and server, communicate through this protocol, or if there is an order in which things happen.

About this task

For example, in HTTP, the browser sends a request to the web server by using a connection to send the request and receives a response from the web server. This process suggests that at least the following model elements need to be present:

- An HTTP Request model element
- A Server Connection model element
- An HTTP Response model element

After you have the basic behavior defined, you can add additional features to the model elements.

Modeling the behavior of a protocol extension

To create a model element for your protocol, you must extend the

`com.ibm.rational.test.lt.models.behavior.impl.CBBlockImpl` class. Extending this class enables you to use all the basic functionality that is provided by this model element, including persistence into the test files and APIs to set or get properties of a model element.

About this task

There are other model elements in the load test behavior model (LTBM) that are extended from the `CBlockImpl` class and that provide various functionality. You could also extend those elements for your protocol. The LTBM provides common constructs (`com.ibm.rational.test.lt.models.behavior.common`) that can be reused by protocol extensions. These constructs are included in various packages in the LTBM. See the Javadoc information and the API topic for more details.

To create a protocol model element that is not related to any of the common constructs provided by the LTBM, extend the `CBlockImpl` class. If the model element for your protocol is a specific type of an LTBM model element, extend the implementation class for that element. For example, if a protocol extension provides a special type of a `CLoop` construct, it would extend the `CLoopImpl` class and add additional attributes to that element.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

1. Optionally, create an interface that defines the methods for the class, including the setters and getters for the attributes added by this element.
2. Create a new class that extends the `CBlockImpl` interface, and if you created an interface in step one, implement the interface.
3. In the constructor of the model element, call the method `setType(type)`, where *type* is a unique type string denoting the model element as registered using the `com.ibm.rational.test.lt.models.behavior.protocol` extension point.
4. Implement the getters and setters of attributes for this model element class.



Note: The setter must set the value attributes of the primitive data types into the underlying model using one of the overloaded `setProperty()` methods based on the type of attribute.

5. To set attributes of complex types, override the `addReference()` method.

Results

While the test is loading, the getter (at least initially) gets the value of the attribute from the underlying model by using one of the overloaded `getProperty()` methods, based on the type of the property for primitive attributes.

You can define the model element to store the value of the attributes in local attributes. However, the guideline is that the getter should get it first from the underlying model and the setter should store it in the underlying model.

Extending the classes in LTBM

You can create a new Load Test Behavior Model (LTBM) element.

Follow these steps to create a new model element:

1. Optionally create an interface that defines the methods for the class, including the setters and getters.
2. Create a new class that extends CBlockImpl interface and that optionally implements the interface defined in the above step.
3. Implement the getters and setters of attributes for this model element class.
 - a. The setter must set the value attributes of the primitive data types into the underlying model using one of the overloaded setProperty() methods.
4. Override the addReference() method to set attributes of complex types.
5. The getter, at least initially, gets the value of the attribute from the underlying model using one of the overloaded getProperty() methods based on the type of the property for primitive attributes.
6. The model element can additionally decide to store the value of the attributes in local attributes. However, the underlying principle to be followed is that the get should get it first from the underlying model and the set should store it into the underlying model.

Public APIs for LTBM

The load test behavior model (LTBM) contains multiple packages. Each package contains the interfaces that define the LTBM. In addition, each package has a corresponding implementation package that contains the implementation classes for these interfaces.

The extensions will either use or extend the implemented classes from LTBM while having access only to the methods defined in the corresponding interfaces. The following packages are part of LTBM:

- `com.ibm.rational.test.lt.models.behavior.lttest`
- `com.ibm.rational.test.lt.models.behavior.common`
- `com.ibm.rational.test.lt.models.behavior.data`
- `com.ibm.rational.test.lt.models.behavior.vps.impl`

The implementation classes are contained in the corresponding implementation packages, such as

`com.ibm.rational.test.lt.models.behavior.lttest.impl`, `com.ibm.rational.test.lt.models.behavior.common.impl`, `com.ibm.rational.test.lt.models.behavior.data.impl` and `com.ibm.rational.test.lt.models.behavior.vps.impl`.

Each of these packages also contains factory classes that enable you to create a model element from that package. For best results, use the factory classes to create model elements. For example, to create a loop, use this method: `LTTestFactory.eINSTANCE.createCBlock()`

The LTBM provides common constructs that can be reused by protocol extensions. These constructs are included in various packages in the LTBM. The Javadoc information explains each available interface in more detail.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents** HCL OneTest Performance **API Reference**.

com.ibm.rational.test.lt.models.behavior.lttest package

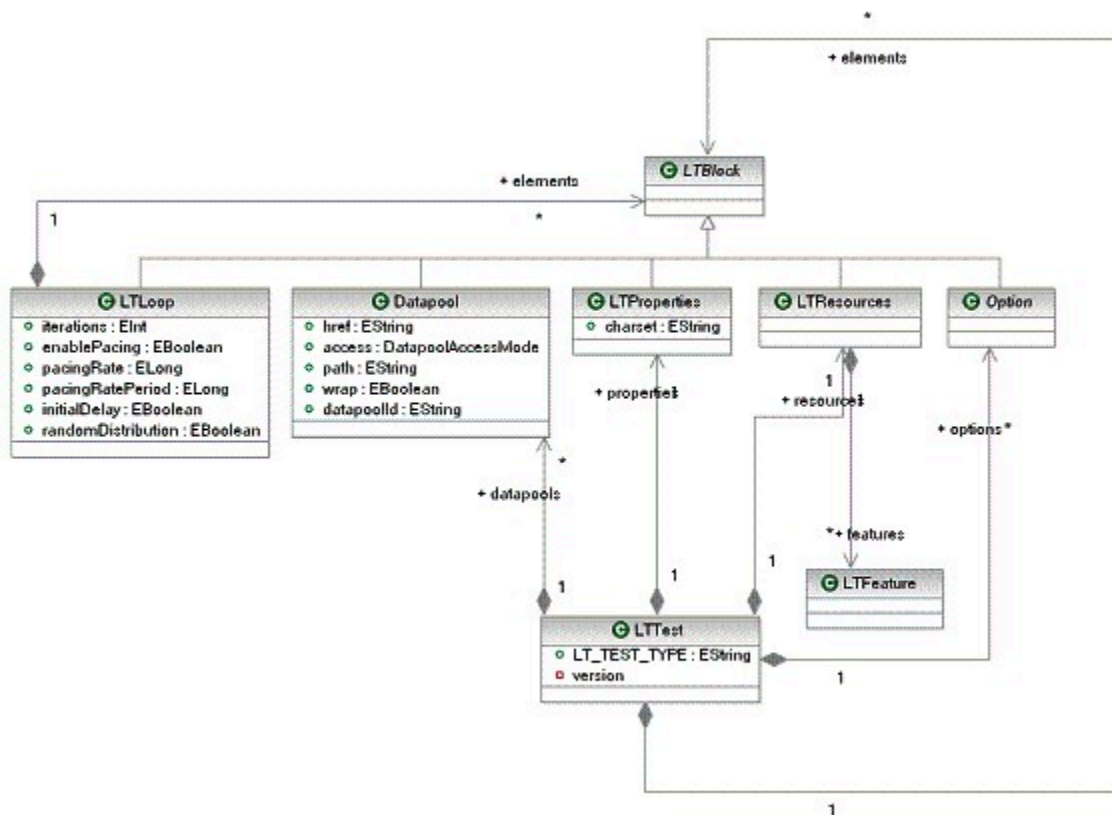
This package holds the interfaces related to the performance test object and its attributes.

The LTest interface is the root container for all model elements. The LTest interface also consists of model elements that are used to refer to outside assets, such as datasets, features, and external resources. An extension is not expected to implement or extend an LTest interface.

An LTest interface consists of the following items:

- An instance of LResources that contains features and other globally scoped items.
- A list of common and protocol based options.
- A list of datasets being used in this test, if any.
- A list of properties specific to this test.
- A list of model elements that might contain other model elements.

The following diagram shows the com.ibm.rational.test.lt.models.behavior.ltest package structure:

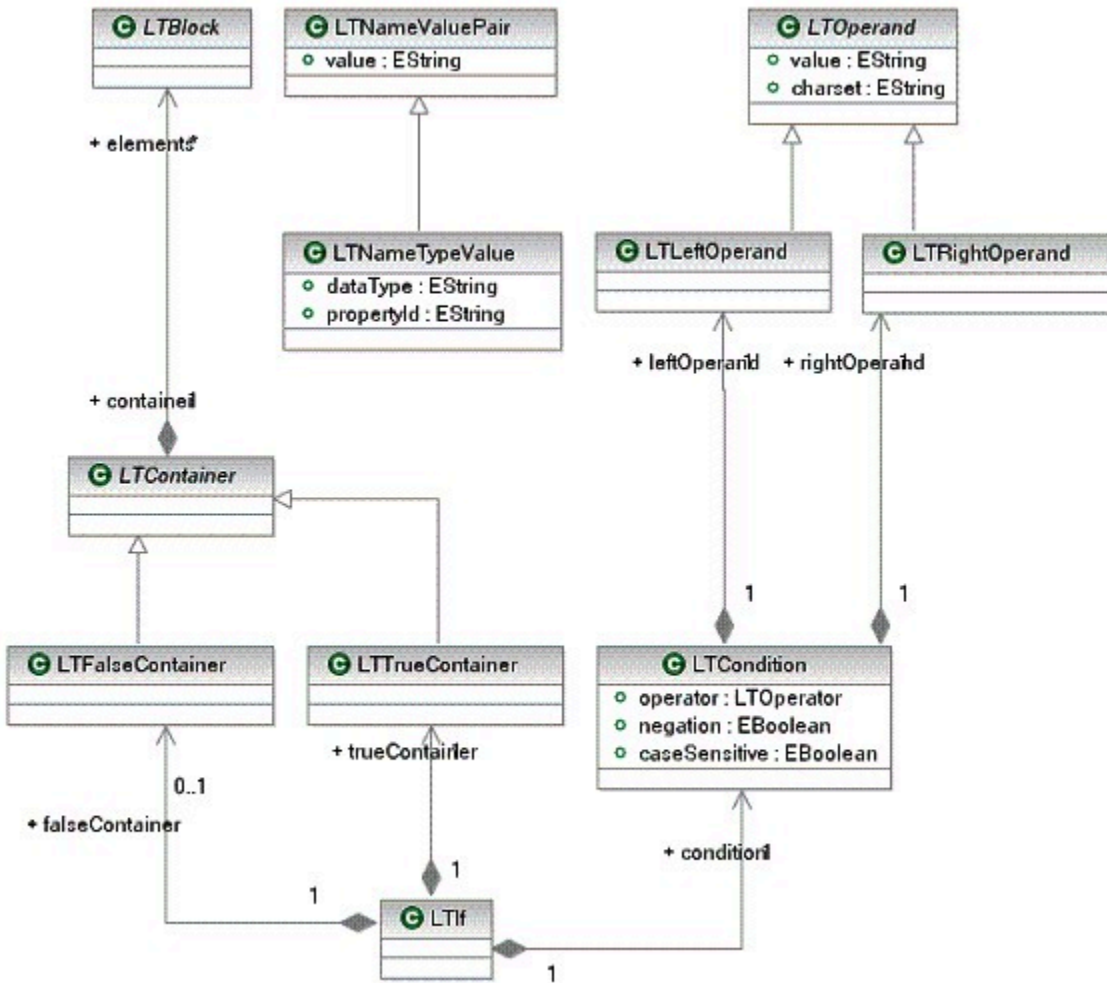


com.ibm.rational.test.lt.models.behavior.common package

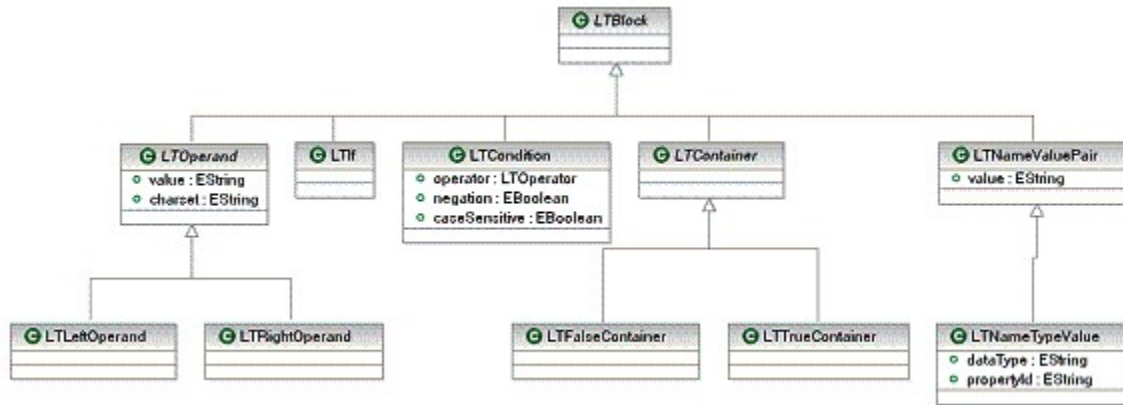
This package contains common interfaces that the extending model elements can implement to add these features to them.

The com.ibm.rational.test.it.models.behavior.common package also contains common constructs such as the IF construct. The package also contains other commonly used constructs such as LTNameValuePair, in which you can store a list of name or value pairs.

The following diagram shows the relationship between the model elements in the com.ibm.rational.test.it.models.behavior.common package:



The following diagram shows the inheritance of model elements in this package:



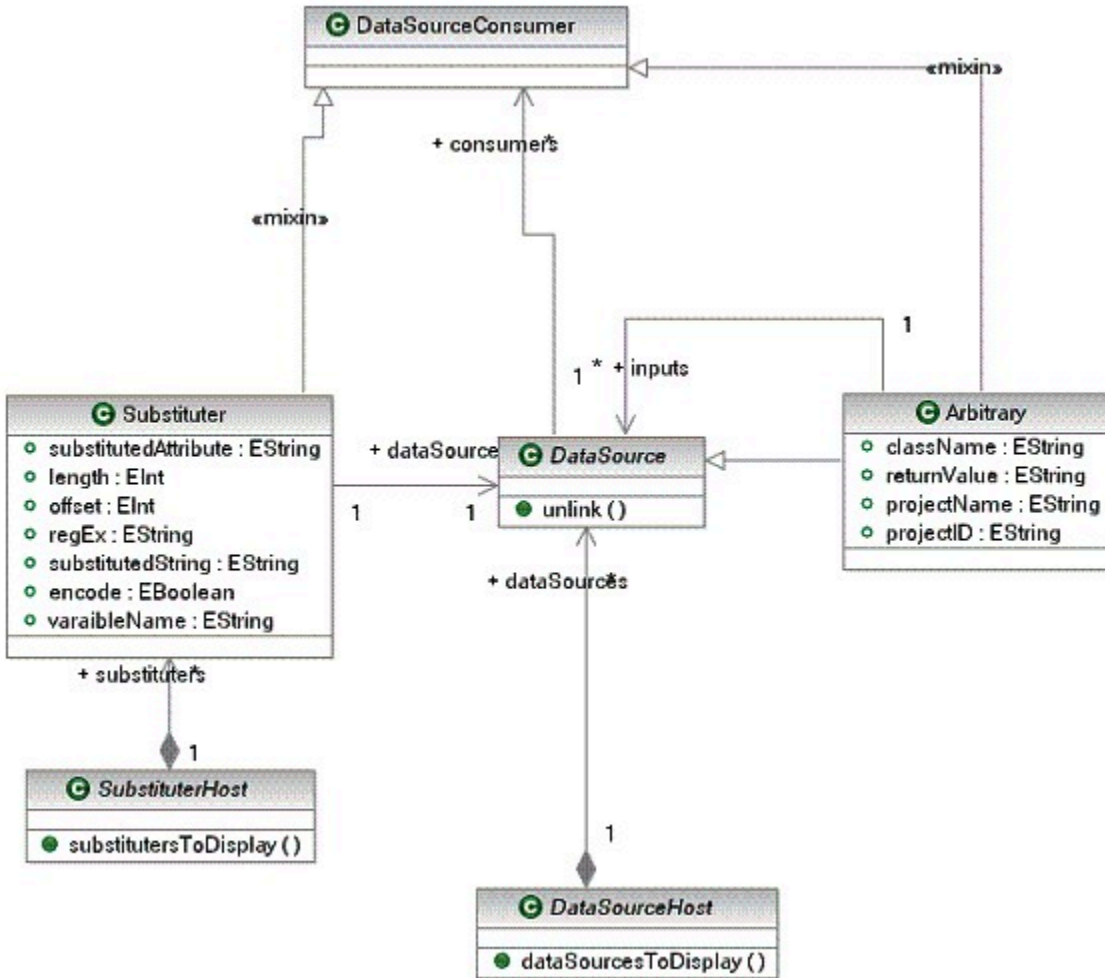
com.ibm.rational.test.lt.models.behavior.data package

This package contains classes related to data correlation and custom code.

Data correlation consists of two parts. One part is the source for the data and the other is the consumer of the data. The base class for all data sources is called DataSource and the one for consumer is Substituter.

A model element in the protocol extension that can contain data sources must be marked as a data source host by implementing the DataSourceHost interface. A model element that can contain consumers of data sources must be marked as a consumer host by implementing the DataSourceConsumer interface.

The following diagram shows the relationship between the data source and the consumer:



Data sources

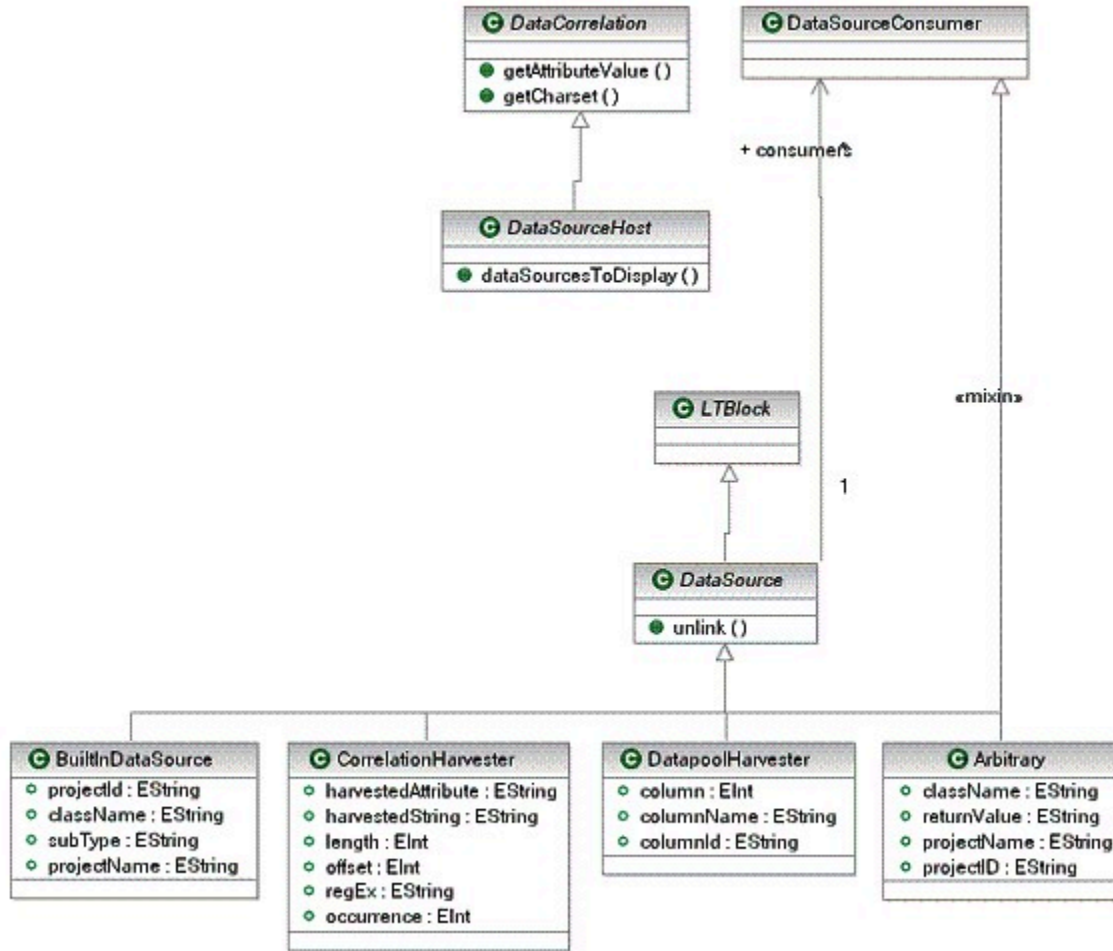
The data sources are model elements that designate data extraction and storage for consumption by other model elements. The following types of data sources are provided by the load test behavior model (LTBM):

- DatapoolHarvester - to denote columns of data extracted from a dataset.
- CorrelationHarvester - to denote a string of data to be used later on in a test.
- BuiltInDataSources - built-in functions that provide derived information (eg. Current® time).
- Arbitrary - enables users to write custom code to be inserted into a test.

For details about these classes, see the Javadoc information.

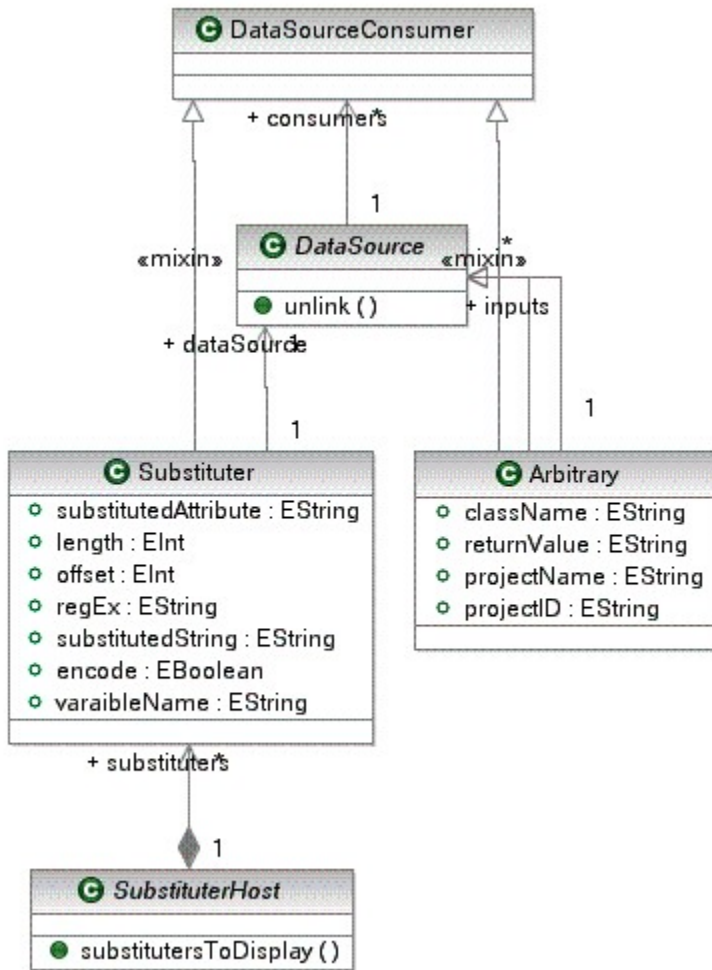
A protocol extension can provide its own set of built-in data sources using the extension point described in the "Extension points" topic.

The following diagram shows the relationship between these classes:



Substituters

The basic consumer type provided is a Substituter. The following diagram shows the Substituter structure:



The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

com.ibm.rational.test.lt.models.behavior.vps package

This package provides classes related to verification points.

The com.ibm.rational.test.lt.models.behavior.vps package contains the `VerificationPoint` interface and its corresponding implementation. This package provides the generic verification point `VPCContent`. It is used to detect the presence (or indicate the absence) of one or more defined test strings that correlate with strings, or parts of strings, that the application generates while under test. You can define the strings that the verification point should look for.

Extending data correlation

Data correlation is the process of extracting data that is returned from a server, and then sending it back to the server in a subsequent request.

About this task

There are two main parts to data correlation. One part occurs during test generation and script editing, and the other part occurs during script execution.

The test generation part of data correlation is the most significant part. There is automatic data correlation that is performed during test generation, and there is manual correlation that can be performed during script editing. There are several different things that can be done as part of manual correlation: you can substitute values into a site from a data pool, a built-in data source, or a reference that is already created.

For more details about test generation extensibility, see the "Extending the test generator" topic. For details about the Script class, see the "Extending code generation" topic.

For examples of data correlation extensions, see these plug-ins:

```
com.ibm.rational.test.lt.sdk.samples.datacorrelation.testgen.socket
```

```
com.ibm.rational.test.lt.sdk.samples.datacorrelation.execution.socket
```

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

Implementing data correlation for test generation

To implement data correlation for test generation, become familiar with these classes: `com.ibm.rational.test.lt.datacorrelation.testgen.proto.IProtoElementAdapter`, `com.ibm.rational.test.lt.datacorrelation.testgen.DataCorrelator`, and `com.ibm.rational.test.lt.datacorrelation.testgen.IDCStringLocator`.

To implement data correlation for test generation and script editing, your model elements must extend `com.ibm.rational.test.lt.models.behavior.data.DataSource` and `com.ibm.rational.test.lt.models.behavior.data.Substituter` so that you can associate data sources and substituters with your model elements. Also, when you extend those classes, make sure you implement the code to make Substituters and DataSources persist in your model. For more information, refer to the model documentation.

You need to create your own data correlation plug-in to extend the extension point `DCTestgenProto`, which is defined in the plug-in `com.ibm.rational.test.lt.datacorrelation.testgen`. To extend the extension point, your code must implement the interface `IProtoElementAdapter` which is defined in `com.ibm.rational.test.lt.datacorrelation.testgen.proto`. Then, when you extend the `DCTestgenProto` extension point, use the following items:

- ID - This is the unique ID of your extension.
- Name - The name that is meaningful for you.
- Point - `com.ibm.rational.test.lt.datacorrelation.testgen.DCTestgenProto` extension point

The other element details are:

- `class` - This is the full name of the class (including plug-in name) that implements `IProtoElementAdapter`.
- `protoType` - The type of model element that this plug-in handles, for example, in HTTP, the model element that is handled is `com.ibm.rational.test.lt.models.behavior.http.HTTPRequest`.
- `generic` - This is a boolean that can be true or false. If it is true, this means that if there are other plug-ins that handle the same `protoType`, this plug-in will be called last.
- `uniqueID` - This ID is unique among all data correlators.

After you have extended the data correlation extension point, the main data correlation engine will call your code with all the elements of the model type that you defined.

There are two parts to data correlation at testgen time. The first part is automatic data correlation which happens at testgen time. This will happen through a call to `DataCorrelator.CorrelateAll()`. When this method is called automatically by testgen, it will in turn call two main methods in your plugin. Those methods are `IProtoAdapter.findSubs()` and `IProtoAdapter.findReference()`. The `findSubs()` method should return a list of substituters that are found for the current element that is passed into your method, and null if there were no substituters found. The `findReference()` method should try to find a reference for the substituter passed into it. You start looking for references at the element before the substituter and keep working backwards in the testsuite until you find a suitable reference. If no reference is found, return null. `CorrelateAll` is used to find all correlations that you can so that the user does not have to manually find them at test editing time.

The other parts of the `IProtoAdapter` class are called at test editing time when a user wants to create a substitution site, a reference, or make a correlation between a substitution site and a reference. If you are creating your own type of Substituters or DataSources, your `makeDataSource()` and `makeSub()` methods will be used for that. You will be passed the `DCStringLocator` class with all the information about offsets and text that you need. Use this `DCStringLocator` to determine what substituter or data source you want to create and then call back into `IDataCorrelator`. Correlations, substituters, and data sources are created in the model for you. Read the documentation on the `IDataCorrelator` class to learn which methods are right for your purposes.

Implementing data correlation for execution

To implement data correlation during execution, set up your class that extends the `com.ibm.rational.test.lt.kernel.action.KAction` class to trigger a call into the data correlation engine.

Before you begin

Before you implement data correlation, become familiar with the following classes:

- `com.ibm.rational.test.lt.datacorrelation.execution.harvest.IDataHarvester`
- `com.ibm.rational.test.lt.datacorrelation.execution.sub.IDataSub`
- `IDataCorrelationVar`
- `com.ibm.rational.test.lt.datacorrelation.execution.proto.IProtoActionAdapter`

See the Javadoc information for class and method descriptions.

About this task

The `IKAction` interface defines the basic functionality that all kernel actions must implement. The `IKAction` interface is the base interface for all kernel actions. The actions can represent loops, conditions, or other code constructs.

To implement data correlation during execution, `codegen` writes your harvesters and substituters into containers. To implement data correlation at execution time, `codegen` will need to write your harvesters and substituters into containers. This is done by calling `LTTestTranslator.translateHarvesterContainer()` and `LTTestTranslator.translateSubstituterContainer()` at `codegen` time when you are translating your action. Both of these methods live in the `com.ibm.rational.test.lt.codegen.core` plugin. These containers will need to be added to your action and stored with your action. These containers must be added to and stored with your action. When your actions are being executed, they must call these substituter and datasource containers. The substituter container is called at the beginning of your action, before you send data, and the data source container is called at the end of your action, after you have received the data.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help ContentsHCL OneTest Performance API Reference**.

To implement the execution portion of data correlation:

1. You must extend an extension point in `codegen.core` to get your execution `IProtoActionAdapter` registered for playback:
 - a. Extend the extension point `DataCorrelationProtoAdapter` in the `com.ibm.rational.test.lt.codegen.core` package. To do the extension, use your plug-in that is the extension to the `com.ibm.rational.test.lt.datacorrelation.execution` plug-in that implements the `IProtoActionAdapter` interface.
 - b. Tell the `com.ibm.rational.test.lt.datacorrelation.execution` plug-in what types of `IKActions` to handle and the name of your plug-in.
 - c. When this is complete, you should see something similar to the following added near the top of your generated `.java` code:

Example

```
pa.addPA( "com.ibm.rational.test.lt.sdksamples.datacorrelation.execution.socket.SocketActionAdapter",
"com.ibm.rational.test.lt.sdksamples.protocol.socket.io.SocketSend" );
```

The first string should be the name of your class that implements the `IProtoActionAdapter`, and the second string should be the name of the `IKAction` that your `IProtoActionAdapter` class should be called for. Be very careful on the spelling, it has to be exactly right.

2. To implement the `datacorrelation.execution` plug-in extension, your plug-in must implement the `com.ibm.rational.test.lt.datacorrelation.execution.IProtoActionAdapter` interface. The main `com.ibm.rational.test.lt.datacorrelation.execution` plug-in uses the interface to call your plug-in and to get the information for the implementation.



Note: This is also the plug-in that extends the `DataCorrelationProtoAdapter` class in the `codegen` extension point.

3. To start the substitution process, in the `IDataSub.substituteData()` method, use the action and hash map as parameters.
 - a. The action that is currently active will call the `com.ibm.rational.test.lt.datacorrelation.execution.sub.IDataSub.substituteData()` method. The substitution container executes all substitution rules and put the new string values (read from data correlation variables) into the hashMap. The first value of the hashMap is the `propertyType`, and the second value is the entire new string for that `propertyType`. So, when the substitution is done, it fixes the entire string for you.
 - b. The substitute container returns the control to the `IKAction` interface.
 - c. The `IKAction` interface reads the string values from the hash map and sends them to the appropriate places.
4. To start data harvesting, the action that is currently active calls the `IDataHarvester.harvestData()` method. The action passes itself to the container.
 - a. The data harvester calls the plug-in that has extended the data correlation execution plug-in to get the string values that the harvest rules must be applied to.
 - b. The data harvester container places the harvested values into data correlation variables. These data correlation variables are then used in later substitute executions.

Extending the test editor

The performance testing software provides application programming interface (API) classes for extending the test and performance tools platform (TPTP) to write new editors and protocol extensions.

About this task

The editors in performance testing are extensions of the defined editor framework in TPTP. The editors are loaded by TPTP based on the file type, for example schedule, test, or data pool.

The performance testing editor is built of several layers: the common framework, the editor, and the protocol layers.

- The common framework layer consists mainly of abstract classes and interfaces, and some utility libraries. The common framework is initialized by TPTP editor architecture.
- The editor layer extends classes from the common framework layer to provide specific implementations for their models.
- The protocol layers can be basic and dependent. The basic layers do not depend on other protocols such as HTTP. The dependent layers are built on top of other protocols, for example Siebel on top of HTTP. The protocol layers are implemented on top of the single editor layer.

When the editor needs to be opened in a model file, TPTP determines which editor is capable of handling this particular model and loads, and initializes the specific extension.

The common framework layer provides its implementation of the handler class, the

`com.ibm.rational.common.test.editor.framework.extensions.CommonEditorExtension` class. The editor layer creates an instance of the `com.ibm.rational.common.test.editor.framework.TestEditor` class that must be extended by the editor layer. After the `TestEditor` object is created, the user interface (UI) widgets and components are created to display the model data.

The `TestEditor` class provides the bridge between TPTP (through the `CommonEditorExtension`), the model (through the `CBTest` member variable), Eclipse, and the concrete protocol code. When the editor is closed by the user or reloaded, the first instance of the `TestEditor` class is destroyed and a new one is created.

The editors support only data that comes out of the corresponding models. The editor layer of the performance testing editor is called the Load Test class. The Load Test class extends the `TestEditor` class by creating concrete `LoadTestEditor` and `CommonEditorExtension` classes through the `LoadTestEditorExtension` extension point.

For the recorder extension sample, see the plug-in `com.ibm.rational.test.lt.sdk.samples.editor.socket`.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

Migrating test editor extensibility

Changes in the performance testing editor base framework package are provided in this release to support more standardized ways of handling model element attributes.

Details area contents and layout (attribute field support)

You can now manage the layout and contents of the Details area in the test editor (schedule and test editors) that developers had to manage in earlier versions. In this version, you can manipulate the contents by using the abstraction called `AttributeField`. This construct hides much of the required behavior from developers, and enables a great deal of extensibility. Updates to `AttributeField` can coexist with the user interfaces of existing editors or protocol extensions, provided that some minor adjustments are made. Classes derived from `AttributeField` must be used whenever an attribute from a model element is displayed. Do not use the `AttributeField` for any other information presented to the user in the Details area. Instead, use regular Eclipse widgets.

Behavior

The `getXXXvalue()`, `setXXXValue()`, and `getFieldName()` classes and APIs provide a way to link model data with their user-interface representation, and at the same time hide related low-level maintenance work from the developer.

These classes are abstract. When a developer chooses one of them to display their model data, only a small number of methods need to be implemented to define the default behavior of the attribute field. If more customized behavior is needed, other methods are available for overriding. Refer to the Javadoc HTML documentation for more information.

The smallest set of methods that a developer must implement are as follows:

- `getXXXValue()`: Retrieves and returns a value from model element. `xxx` signifies a type of an attribute. For example, when extending `IntegerAttributeField`, the name of the method is `getIntegerValue()`.
- `setXXXValue()`: Sends a value obtained from the UI to the model element. The meaning of `xxx` is the same as above.
- `getFieldName()`: Returns the name of the field. Names make a field addressable for navigation.



Note: Before version 7.0, field names were optional; in this version they are mandatory.


Classes

The following table is a hierarchy of the `AttributeField` related classes that are available to editor and protocol developers. The list includes descriptions about which classes to use in specific situations.



Note: These classes have limited functionality because they must support classes and APIs from versions earlier than 7.0.

AttributeField class	Description
<code>OptionsComboField</code>	This class is used to display a set of options to a user. Options are presented in a combination box. When extending this class, you must provide an index of the option currently selected in the model element. When a user chooses a different value from the combination box, the new index is passed to the derived class, in order to update the model. The class developer needs to understand the meaning of the index in the context of the model.
<code>OptionsRadioField</code>	The same provision that applies to <code>OptionsComboField</code> applies to <code>OptionsRadioField</code> . The following exception, however, applies: options are displayed as a set of radio buttons in a group. A user must select one of the radio buttons to indicate the index of the selected option.
<code>BooleanAttributeField</code>	This class is used when the model element attribute is a Boolean value. The value is displayed as a check box. The developer must provide a Boolean value from a model element and accept a new Boolean value from the user interface to update the model element.
<code>IntegerAttributeField</code>	This class is used when the model element attribute holds an integer value. The field can represent an inte-

AttributeField class	Description
	<p>ger value in several ways. The following control types are available for representation:</p> <ul style="list-style-type: none"> • <code>StyledText</code> • <code>Spinner</code> • <code>Slider</code> • <code>Scale</code> <p> Note: This class is subject to change in the future.</p>
<code>TextAttributeField</code>	Use this field when there is text data in the model element.
<code>FilteredTextAttributeField</code>	This class extends the behavior of <code>TextAttributeField</code> by enabling condition checking and displaying alternative text (message) to the user. For example, the developer might want to filter binary data, or filter text that is too long for convenient display.
<code>DataCorrelatingTextAttrField</code>	Use this class when the text data can be either substituted, data correlated, used as a reference, or configured in any combination of these.

Porting code from `layoutProvider`

The following is a short guide to porting existing code, typically found in the `layoutProvider` class, to the new function.

Previous implementation:

```
class MyLayoutProvider extends ExtlayoutProvider
{
  layoutControls( CBActionElement element )
  {
    super.layoutControls( element ); // call super first.
    createWidgets(); // create all the UI for display
    refreshWidgets(); // call refreshLayout to populate UI
    return true; // return true is success.
  }

  createWidgets()
  {
    new StyledText();
    new Button();
  }
}
```

```

refreshControls( CBActionElement element )
{
    super.refreshControls( element ); // call super first
    // grab data from model element and apply it to UI widgets
    applyModelDataToWidgets();
    return true; // return true if success.
}

/* because the ExtLayoutProvider is SelectionListener,
this method is called when Buttons, ComboBoxes and
such are modified. */
widgetSelected( SelectionEvent event )
{
    // find the widget, get its value and apply it to model
    applyUiDataToModelElement();
    // call super to update the editor.
    super.widgetSelected();
}

/* because the ExtLayoutProvider is ModifyListener,
this method is called when StyledText is modified. */
modifyText( ModifyEvent event )
{
    // find relevant StyledText control and apply
    // its value to the model element.
    applyTextUiDataToModelElement();
    super.modifyText();
}
}

```

Current® implementation:

```

class MyLayoutProvider extends ExtlayoutProvider
{
    // class declared as internal.
    class MyTextField extends TextFieldAttribute
    {
        String getTextValue(){
            return ((MyModelElement)getSelection()).getTextAttr();
        }
        setTextView( String newVal ){
            ((MyModelElement)getSelection()).setTextAttr( newVal );
        }
        String getFieldName(){
            return MY_FIELD_NAME; // defined elsewhere
        }
    }
};

MyTextField m_fldText;
MyDataCorrelationField m_DcField; // declared outside.

layoutControls( CBActionElement element )
{
    createWidgetsAndFields(); // create all the UI for display
    updateNonFieldWidgets(); // update non-model widgets
    // always call super at the end.
    return super.layoutControls( element );
}

```



```

}

createWidgetsAndFields()
{
    // create UI widgets for displaying non-model info
    ...
    // create Fields
    m_fldText = new MyTextField( this );
    m_fldText.createLabel( ... );
    m_fldText.createControl( ... );

    // create more UI widgets for displaying non-model info

    m_DcField = new MyDataCorrelationField( this );
    m_DcField.createLabel( ... );
    m_DcField.createControl( ... );

}

refreshControls( CBActionElement element )
{
    // update NON-UI widgets only.
    applyModelDataToWidgets();
    //always call super at the end.
    return super.refreshControls( element );
}

/*You do not have to have this method unless you want
to update NON-model widgets/data. */
widgetSelected( SelectionEvent event )
{
    // find the widget and do whatever you need, but
    // do not update the model.
    applyUiDataTo_NON_ModelElement();
    // DO NOT call super to update the editor.
}

/* You do not need to have this method unless you
want to update non-model widgets/data. */
modifyText( ModifyEvent event )
{
    // find the widget and do whatever you need, but do not
    // update the model.
    applyTextUiDataToModelElement();
    // DO NOT call super to update the editor.

}

}

```

Test editor structure

The editor in HCL OneTest™ Performance is built of several layers; the common framework, the editor and the protocol layer.

The common layer defines the interfaces and extension-points, provides the API classes and interfaces. It also provides the hooks into the Eclipse user interface (UI) menus, actions, markers. The common framework layer is initialized by the TPTP editor architecture. Because test editors are in fact extensions loaded and initialized by the TPTP platform via `org.eclipse.hyades.ui.editorExtensions` extension point, one of the common editor framework's responsibilities is to provide hooks and application programming interfaces (APIs) for concrete editors implementations to communicate with TPTP. The common editor framework defines the classes that must be extended for a more specific behavior to be used by the extended editor implementations. This extension is realized by the `TestEditor` class that is extended by the Load Test Editor plugin and called `LoadTestEditor`.

The editor layer extends classes from the common framework layer to provide specific implementations for their models. The protocol layers can be basic and dependent. The basic protocols do not depend on other protocols, such as HTTP. The dependent protocols are built on top of other protocols, for example Siebel on top of HTTP. The protocol layers are implemented on top of the single editor layer. The protocol layers provide handlers for protocol-specific objects.

When the editor needs to be opened on a model file, Hyades determines which editor is capable of handling this particular model and loads and initializes specific extension. The common layer provides its implementation of the handler class, the `CommonEditorExtension` class. The common editor layer creates an instance of the `TestEditor` class that must be extended by the editor layer. After the `TestEditor` object is created, the user interface (UI) widgets and components are created to display the model data.

The `TestEditor` provides the bridge between TPTP, the model, Eclipse, and the concrete protocol code. The `CommonEditorExtension` class is used. When the editor is closed by the user or reloaded, the first instance of the `TestEditor` is destroyed and a new one is created.

The layers described above are split into separate plugins. Each plugin defines some extension points which are used by the higher-level plugins as well as by the defining plugins themselves. Additionally, there are some Java™ interfaces that must be used when writing classes.

The plugins are:

- `com.ibm.rational.test.common.editor.framework`
- `com.ibm.rational.test.lt.testeditor`
- `com.ibm.rational.test.lt.http.editor`
- `com.ibm.rational.test.lt.http.siebel`

Common editor framework

Extend performance test editing in the common editor framework. It contains the classes to extend for specific behaviors that will be used by the extended editor implementations.

Because performance testing editors are extensions loaded and initialized by the TPTP platform through the `org.eclipse.hyades.ui.editorExtensions` extension point, one of the common editor framework functions is to provide hooks and application programming interfaces (APIs) for concrete editor implementations to communicate with TPTP. The common editor framework defines the classes to be used by higher editor

implementations. One of the most widely used classes is the `TestEditor` class, which is extended by the `com.ibm.rational.test.lt.testeditor.main.LoadTestEditor` plug-in.

The `layoutProvider`, `labelProvider`, `contentProvider` and `actionHandler` extension points have been deprecated and are superseded by the `modelObjectDescriptor` extension point that combines and enhances them. The following table lists the active extension points that you can use to extend the common editor framework:

Extension point	Description
<code>modelObjectDescriptor</code>	Specifies the type of the object, the test editor, and the general name and icon for the object.
<code>testOptions</code>	Used to contribute user interface (UI) elements for displaying the protocol options in the Details area of the root element in the tree, Performance Test or Schedule .
<code>editorAddonEnabled</code>	Used to write contributions to existing user interface (UI). It can have enabled and displayed state. Works with the <code>AddonReader</code> class.
<code>searchTypeProvider</code>	Contributes a search category to the performance testing Search window. The category consists of object type and, optionally, search parameters.
<code>searchTypeOptionsContributor</code>	Contributes extra search parameters to some other search type category.
<code>preferenceContributor</code>	Provides a way to create extensible preference pages as well as to contribute UI to other preference pages. For preference pages, use the Eclipse extension point and extend the class <code>com.ibm.rational.common.test.editor.framework.kernel.EditorPreferencePage</code> . For page contributions, use this extension point and extend the <code>com.ibm.rational.common.test.editor.framework.kernel.TestPreferenceContributor</code> class.

The modelObjectDescriptor extension point

The `modelObjectDescriptor` extension point combines and enhances the deprecated `contentProvider`, `labelProvider`, `layoutProvider`, and `actionHandler` extension points.

The `modelObjectDescriptor` extension point has four attributes. The attributes specify the type of the object, the type of the test editor where the object is used, and the general name and icon for the object. The name should not have any formatting characters in it. The name with the icon is used primarily for reporting purposes.

Attribute	Description
<code>type</code>	Represents the type of the model object, returned by the <code>CBActionElement.getType()</code> method.
<code>modelElement-type</code>	The type of the model or test. The same value returned by your <code>CBTest.getType()</code> method. For performance testing, protocol extensions use <code>com.ibm.rational.test.lt.ltttest</code> .
<code>icon</code>	The image that represents your object.
<code>label</code>	The name of your object.

The `modelObjectDescriptor` extension point has four child definitions, as described in the following table:

Child	Attributes description
<code>labelProvider</code> . See <code>com.ibm.rational.common.test.editor.framework.extensions.ExtLabelProvider</code> class.	<ul style="list-style-type: none"> • <code>treeLabel</code> – The text to be displayed in the tree in the Main section of the editor. This text can be static or contain formatting. It is up to the implementing class to format and return proper textual representation of an element. • <code>statusLine</code> – The text to be displayed on the status line when the object is selected in the Main section tree of the editor. If omitted, the value of the <code>treeLabel</code> attribute will be used for this purpose. • <code>tooltip</code> – The text to be displayed in the tooltip (when appropriate). If omitted, the value of the <code>treeLabel</code> will be used for this purpose. • <code>menuText</code> – The text to be displayed in a pop-up menu, such as Add or Insert. If omitted, the value of the <code>modelObjectDescriptor</code> label will be used for this purpose. • <code>description</code> – The longer description of the model element. If omitted, the value of the <code>modelObjectDescriptor</code> label will be used for this purpose. • <code>icon</code> – The image that represents the model element. May be same or different as the one specified in the <code>modelObjectDescriptor</code> icon. • <code>class</code> – The instance of this class will be created when an object is set up to be referenced in the editor. There is a default base class to be used for this purpose, called <code>ExtLabelProvider</code>. It provides methods that return information, specified in the extension. All of its methods can be overridden by the extending class to provide appropriate formatting.
<code>layoutProvider</code> . See <code>com.ibm.rational.common.test.editor</code>	<ul style="list-style-type: none"> • <code>class</code> – The instance of this class will be created when the object details are displayed in the editor's Details section. A model element must have a <code>layoutProvider</code> class if this object is displayed in the Main section tree. There is a default base class to be used for this purpose, called <code>ExtLayoutProvider</code>. It provides methods for constructing, formatting, and refreshing Details section for the model element. The methods <code>layoutControls</code> and <code>refreshControls</code>

Child	Attributes description
<p><code>.framework-</code> <code>.extensions-</code> <code>.ExtLayout-</code> <code>Provider</code> class.</p> <p><code>content-</code> <code>Provider.</code> See <code>ExtCon-</code> <code>tentProvider</code> class.</p>	<p>must be overridden by the extending class. The <code>com.ibm.rational.common.test.editor.frame-</code> <code>work.extensions.ExtLayoutProvider</code> class implements the <code>SelectionListener</code> and <code>ModifyLis-</code> <code>tener</code> interfaces, so it can be used to listen to such events generated by your controls.</p> <ul style="list-style-type: none"> • <i>class</i> – The instance of this class will be created when the object hierarchy information needs to be discovered. A model element must have a <code>contentProvider</code> class if the object is displayed in the Main section tree. There is a default base class to be used for this purpose, called <code>ExtContentProvider</code>. It provides methods for discovering information about object children and parents. See the Eclipse <code>IStructuredContentProvider</code> interface for more details. In many cases there is no need to override any of the <code>ExtContentProvider</code> methods, but if some non-standard processing must be done, the first method to override is <code>getChildrenAsList()</code>, as it is called by other methods in this class.
<p><code>actionHan-</code> <code>dlr.</code> See the <code>ExtAc-</code> <code>tionHandler</code> class.</p>	<ul style="list-style-type: none"> • <i>class</i> – The instance of this class will be created when a new object of this type needs to be created or when the existing object needs to be removed from model or moved up or down. Your class must extend <code>ExtActionHandler</code>.

The testOptions extension point

The testOptions extension point is used to contribute user interface elements for displaying options for a protocol in the **Details** area of the root element in the tree, **Performance Test** or **A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule.** Each contribution is displayed in a separate tab in a **Tab** folder.

The testOptions extension point has the following attributes:

Attribute	Description
<i>label</i>	The text to be displayed in the tab title.
<i>toolTip</i>	The text to be displayed in the tab tooltip.
<i>image</i>	The image to be displayed in the tab title.
<i>feature_id</i>	The feature to which these options pertain, currently not filtered.
<i>order</i>	The numeric order of the tab, used to sort tabs. If no number is given or there are duplicate numbers, the label will be used for sorting.

Attribute	Description
<code>class</code>	The class that will be instantiated to display and handle options. The default implementation base class, <code>DefaultOptionsHandler</code> , provides convenience methods. This class is abstract, which means that it needs to be extended to provide concrete user interface elements. The default implementation of the <code>ExtLayoutProvider</code> class for Test objects, <code>DefaultTestLayoutProvider</code> , loads and displays relevant options declared through this extension point.

Contributing actions to the menu

You can use extensions to extend the menu, change navigation, and perform data correlation.

Test editor menus

The test editor has three menus. You can extend the menus by adding more actions such as remove, move up, or move down

The first menu is displayed when the user right-clicks inside the left side of the tree. The other two menus are displayed when the user clicks the **Add** and **Insert** buttons. The **Add** or **Insert** menus are also displayed as submenus in the tree menu.

The IDs of the menus and named group separators are listed in the following table:

ID	Group separator defined in <code>ITestEditorActionIDs</code>
<code>org.eclipse.hyades.test.ui.editor.TestSuiteEditorPart.tree.menu</code>	<ul style="list-style-type: none"> • <code>additions.new-start</code> • <code>additions.new-end</code> • <code>additions.edit-start</code> • <code>additions.edit-end</code> • <code>additions.find-start</code> • <code>additions.find-end</code> • <code>IWorkbenchActionConstants.MB_ADDITIONS</code>
<code>org.eclipse.hyades.test.ui.editor.TestSuiteEditorPart.action_add</code>	<ul style="list-style-type: none"> • <code>IWorkbenchActionConstants.MB_ADDITIONS</code>
<code>org.eclipse.hyades.test.ui.editor.TestSuiteEditorPart.action_insert</code>	<ul style="list-style-type: none"> • <code>IWorkbenchActionConstants.MB_ADDITIONS</code>

Creating actions

You can use the `com.ibm.rational.common.test.editor.framework.extensions.ExtActionHandler` class to remove, move up, and move down menu buttons

About this task

The editor or protocol extensions that are needed to manipulate the test, and the test add, remove, insert specific model elements, must use the `NewModelElementAction` class as described in the following conditions:

1. For every model element displayed in the tree, the `com.ibm.rational.common.test.editor.framework.TestEditor` class expects to find an instance of the `com.ibm.rational.common.test.editor.framework.extensions.ExtActionHandler` class and some `com.ibm.rational.common.test.editor.framework.kernel.actions.NewModelElementActionS` registered in `com.ibm.rational.common.test.editor.framework.TestEditorplug-in's` `com.ibm.rational.common.test.editor.framework.RptMenuManager`.
2. The `ActionHandlerS` are declared in the `plugin.xml` file, while the `com.ibm.rational.common.test.editor.framework.kernel.actions.NewModelElementAction` class must be created and registered programmatically through the protocol plug-in class. Only one set of actions is needed for multiple editors, because this class is context sensitive.
3. The `ActionHandlerS` of the selected elements are asked whether to enable the **Remove**, **Move up** and **Move down** buttons. The `com.ibm.rational.common.test.editor.framework.extensions.ExtActionHandler` examines the selection, looks for the objects of the recognized types, and then returns a value of true or false.
4. To enable or disable **Add** and **Insert** buttons and menus, the `TestEditor` class passes the selection to each of the registered `com.ibm.rational.common.test.editor.framework.kernel.actions.NewModelElementAction` instances. The action is expected to examine the selection, and enable or disable its state, based on whether this action can add its model object to the selected item.
5. The ID of each `com.ibm.rational.common.test.editor.framework.kernel.actions.NewModelElementAction` or the derived type must match the type of the model object that this action represents. The ID must have the same value as the one used in the `modelObjectDescriptor` extension point. The `com.ibm.rational.common.test.editor.framework.TestEditor` class uses this value to locate various providers for the specific model element.
6. There are two separate sets of actions maintained by the `com.ibm.rational.common.test.editor.framework.RptMenuManager` class, the **Add** and the **Insert** actions. While objects of the same type can be registered for both, the same instances of the same class cannot. This means that two instances of the `AddObject` action must be created and registered. If the object cannot be inserted, the insert action is not required. The same is true for the add action.
7. If an action is selected, the action calls its correspondent `ActionHandler` to create a new model object. At this point, the `ActionHandler` class must create and initialize a new model object, including any required children. The action will add the new model object to the selected parent, but only if the selected parent `ContentProvider` returns a list of the children that have `EList` type, the native model list. For a composite list of

children, for example the `ArrayList` assembled by provider, the `ActionHandler` is expected to add a new child to the parent.

- The protocol writers must use the `LoadTestNewModelElementAction` class as the base class for creating actions. The class supports filtering by feature.

Editor layer extension points

The editor layer enables you to write protocol extensions. During the initialization process, a number of extension points are checked for extensions that define different aspects of the test editor.

The editor layer extends classes from the common framework layer to provide specific implementations for their models. The extensions must be defined in the protocol-specific plug-ins extending the test editor.

Extension point	Description
<code>dataCorrelationHandler</code>	<ul style="list-style-type: none"> Defines an entry in the Data Correlation Handler Selection window. The handlers are queried and displayed in a dialog for the user to choose when there is more than one protocol capable of creating data correlation objects out of the selected text. The user interface descriptor of a handler is matched to its data correlation package through the <code>typeID</code> attribute in the extension point.
<code>dataCorrelationUICategory</code>	<ul style="list-style-type: none"> Defines a category for grouping built-in data sources that are represented in the user interface by the <code>dataCorrelationUIDescriptor</code> extension point.
<code>dataCorrelatorUIDescriptor</code>	<ul style="list-style-type: none"> Provides visual representation for built-in data source types. These are grouped under data correlation user interface categories in the Built-in Datasource Selection Wizard. The wizard is displayed when the user wants to create data correlation with a built-in data source.
<code>wizardPageContributor</code>	<ul style="list-style-type: none"> Used for contributing a protocol specific page or pages to the New Test Wizard.

The protocol handler can make a contribution to the details page of the test to protocol-specific test-wide options. If your protocol needs to support data correlation, you can use the class `DataCorrelationLabelProvider`. The class contains several methods to use with data correlation. Many of the methods deal with display aspects of the data correlation objects.

In data correlation there must be an attribute name assigned to every field or property of a model object that supports data correlation. These names are used for many purposes, one of which is to provide formatted labels for data correlation objects. Because the `LoadTest` editor creates labels early in a process cycle, these labels may not contain detailed formatting instructions. To overcome this issue, higher level protocol extensions or plug-ins can register `LabelFormatters` that are called when needed.

API classes

The following classes and methods can be used to extend the common editor framework:

Class	Methods
com.ibm.rational.common.test.editor.framework.extensions.ExtLabelProvider	<ul style="list-style-type: none"> • <code>getTestEditor()</code> - Returns the TestEditor object, the Load-TestEditor in the Performance Test Editor
com.ibm.rational.common.test.editor.framework.extensions.ExtLayoutProvider	<ul style="list-style-type: none"> • <code>getTestEditor()</code> - Returns the TestEditor object. • <code>getFactory()</code> - Returns the WidgetFactory for creating widgets and controls. • <code>getDetails()</code> - Returns Composite which is a parent for creating controls.

Contributing error handlers

An error handler is associated with an error condition. Error handlers specify the action to take when a specified error condition occurs. Error handlers are provided for conditions such as verification-point failures, connection failures, server timeouts, and data-correlation problems. You can add new types of errors and error handlers.

Defining the user interface for an error handler

After you create a new type of error, you must define the user interface for the error handler. If you do not define a user interface, the error handler is not available in the test editor or schedule editor.

To define the user interface for a new error handler, use the `com.ibm.rational.test.common.editor.framework.exceptionDefinition` extension point.

The creator class can implement the `com.ibm.rational.common.test.editor.framework.extensions.IExceptionCreator` class or extend the `com.ibm.rational.common.test.editor.framework.extensions.DefaultExceptionCreator` class.

The `com.ibm.rational.common.test.editor.framework.extensions.DefaultExceptionCreator` class provides a default implementation of the `IExceptionCreator` element, which uses the `isMyType(CBErrorType exceptionType)` method.

The following is an example definition using HTTP:

```
<extension point="com.ibm.rational.test.common.editor.framework.exceptionDefinition">
  <exceptionTypeDefinition
    creatorClass="com.ibm.rational.test.lt.http.editor.PageTitleErrorExceptionCreator"
    defaultBehavior="0"
    defaultMessage="%PageTitle.Vp.Failed"
    forFeature="com.ibm.rational.test.lt.feature.http"

    labelProvider="com.ibm.rational.test.lt.http.editor.ui.exceptions.PageTitleVpErrorLabelProvider">
  </exceptionTypeDefinition>
  <exceptionTypeDefinition
    creatorClass="com.ibm.rational.test.lt.http.editor.ResponseCodeVpErrorCreator"
    defaultBehavior="0"
    defaultMessage="%Resp.Code.Vp.Failed"
    forFeature="com.ibm.rational.test.lt.feature.http"

    labelProvider="com.ibm.rational.test.lt.http.editor.ui.exceptions.ResponseCodeVpErrorLabelProvider">
  </exceptionTypeDefinition>
  <exceptionTypeDefinition
    creatorClass="com.ibm.rational.test.lt.http.editor.ResponseSizeVpErrorCreator"
    defaultBehavior="0"
    defaultMessage="%Resp.Size.Vp.Failed"
    forFeature="com.ibm.rational.test.lt.feature.http"

    labelProvider="com.ibm.rational.test.lt.http.editor.ui.exceptions.RespSizeVpErrorLabelProvider">
  </exceptionTypeDefinition>
</extension>
```

You must specify the feature ID when you define the user interface for an error handler. By specifying the feature ID, the test editor can find the new types of errors and to filter out error types that are not applicable in multiprotocol tests.

Controlling how available error handlers are displayed

You can control how custom error handlers are displayed in the test and schedule editors.

For any test element, all core errors and all applicable protocol-specific errors are displayed automatically in the **Details** section of the **Advanced** page in the test editor, if the following conditions are satisfied:

- The element implements the `CBErrorHost` interface.
- The `canHostErrors` method returns true.
- The `isErrorGenerator` method returns false.

These conditions are tested in the `createExceptionsUi` method in the `ExtLayoutProvider` class. Typically, you do not modify the default behavior for a test element, but it is possible to override the `createExceptionsUi` method. If the `createExceptionsUi` method returns null, no user interface for error handling is displayed on the **Advanced** page in the test editor.

The ExceptionsUI class controls the user interface for error-handling on the **Advanced** page. The test-specific implementation is the TestExceptionsUI class. The schedule-specific implementation is the ScheduleExceptionsUI class. A shortcut way to create these classes is to call the createExceptionsUi method in the TestEditor class.



Note: Container objects in schedules do not display any user interface for error handling if there are no tests inside the containers.

Controlling how error handlers for specific elements are displayed

You can control how error handlers for specified test elements are displayed.

In the test editor, users can configure error handling for a specific model element that produces an error. For example, if a specific instance of a content verification point fails, the user can change the error-handling behavior for only that instance of the content verification point. To change how the error handlers for protocol-specific elements are displayed, you must know what types of errors can be generated for the element when tests run.

The test editor SDK provides classes for the display of the error-handling user interface.

The com.ibm.rational.test.lt.testeditor.main.exceptions.TestExceptionProducerUI class controls the error-handling user interface in the **Details** section of the **Advanced** page. The com.ibm.rational.test.lt.testeditor.main.exceptions.DialogExceptionProducerUI class controls the error-handling user interface in dialog boxes and properties pages.

Example

The following code is an example for the **Details** section of the **Advanced** page in the test editor. In the layoutControls method, add code similar to these lines:

```
m_exceptionUi = new TestExceptionProducerUI(
    getTestEditor(),
    new AuthenticationExceptionCreator());
m_exceptionUi.createErrorProducerContents(
    getDetails(),
    getNtlm(),
    (LoadTestWidgetFactory) getFactory());
```

The AuthenticationExceptionCreator class is the same class that is specified in the plugin.xml file in the exceptionDefinition extension point. In the refreshControls method, add code similar to this line:

```
m_exceptionUi.refresh( getNtlm() );
```

The following code is an example for dialog boxes and properties pages. In the createContents method or the createDialogArea method, add code similar to these lines:

```
m_errorHandling = new DialogExceptionProducerUI(
    m_page.getEditor(),
    new PageTitleErrorExceptionCreator());
m_errorHandling.createErrorProducerContents( parent, vp );
```

In the onCancelPressed method, which is called when the user clicks **Cancel** in a dialog box, add code similar to this line:

```
m_errorHandling.cleanup();
```

When the user clicks **OK** in a dialog box or property page, the changes are applied to the `CBError` object immediately, so no extra code is required. When the user clicks **Cancel** in a dialog box or property page, the changes must be undone using the `cleanup` method.

To receive notifications when changes are made to a `CBError` object, override the `updateEditor` method by using this code:

```
m_errorHandling = new DialogExceptionProducerUI(
    TestEditorPlugin.getEditorFor(m_datapool),
    new EndOfDatapoolExceptionCreator()){
    @Override
    protected void updateEditor() {
        getButton(OK).setEnabled(true);
        super.updateEditor();
    }
};
m_errorHandling.createErrorProducerContents(composite, m_datapool, null );
```

Creating an error type

The core performance test model includes the generic `CBError` object. The `CBError` object is a shell that contains the actual error, the `CBErrorType` object. Extend the `CBErrorType` object to add new types of errors, including protocol-specific errors. Associate error-handling behavior with an error by using the `CBErrorBehaviorEnum` object.

The following code is an example of creating a new protocol error type from the generic error object:

```
public abstract class ProtocolErrorTypeImpl extends CBErrorTypeImpl implements ProtocolErrorType {
    /**
     * <!-- begin-user-doc -->
     * <!-- end-user-doc -->
     * @generated
     */
    protected ProtocolErrorTypeImpl() {
        super();
    }

    /**
     * <!-- begin-user-doc -->
     * <!-- end-user-doc -->
     * @generated
     */
    protected EClass eStaticClass() {
        return ErrorsPackage.Literals.PROTOCOL_ERROR_TYPE;
    }

    public boolean isErrorGenerator(){
        return true;
    }

    /**
     * Imports needed at code generation time
     * so that the test runs correctly.
     */
}
```

```

*/
public List<String> getExecImport() {
    ArrayList<String> imports = new ArrayList<String>();
    imports.add("import com.ibm.rational.test.lt.execution.protocol.tes.*;");
    imports.add("import com.ibm.rational.test.lt.kernel.action.impl.KThrow;");
    imports.add("import com.ibm.rational.test.lt.kernel.services.*;");
    return imports;
}
} //ProtocolErrorTypeImpl

```

The following code is an example of creating a protocol error type from another protocol error type:

```

public class ProtocolNewErrorTypeImpl extends ProtocolErrorTypeImpl implements ProtocolNewErrorType {
    /**
     * <!-- begin-user-doc -->
     * <!-- end-user-doc -->
     * @generated
     */
    protected ProtocolNewErrorTypeImpl() {
        super();
    }
    /**
     * <!-- begin-user-doc -->
     * <!-- end-user-doc -->
     * @generated
     */
    protected EClass eStaticClass() {
        return ErrorsPackage.Literals.PROTOCOL_NEW_ERROR_TYPE;
    }

    public String getExecType(){
        return "ProtocolNewEvent" ;
    }
} //ProtocolNewErrorTypeImpl

```

For model elements that generate errors, declare these elements to be error generators by using this code:

```

public boolean isErrorGenerator(){
    return true;
}

```

Implementing code generation for error handling

During code generation, test elements are translated into code language elements. You must implement code generation for new error-handling methods.

To learn more about code generation, see [Extending code generation on page 750](#).

Complete these steps to implement code generation for objects that do not generate errors:

- For `Script` objects, add the **<PARAM name="codegen.core.attributes">** parameter to the creation template for the method. For `KAction` objects, add the **<PARAM name="codegen.core.attributes">** parameter to the `execute()` method.
- Add a call to the `translateCoreAttributes()` method. For example, the `LTTestScriptDefinition` class now includes this call:

```
LTTestTranslator.translateCoreAttributes(scriptTemplate, null, (CBActionElement)test);
```

Complete the following steps to implement code generation for methods that generate errors, such as connection failures and authentication failures:

- Add the **<PARAM name="errorBehavior">** parameter to the creation template for the method.
- Add a call to the `translateErrorBehavior()` method. For example, code generation for the `BasicAuthentication` object in HTTP includes these calls:

```
CBError err = (bAuth.getCBErrors().size()==0)?null:(CBError)bAuth.getCBErrors().get(0);
translateErrorBehavior(err, bAuthElem, ILanguageElement.TEMPLATE_CREATION);
```

Implementing execution for error handling

Events that require error handling at run time must have an associated `errorBehavior` action. Extend the `RPTEventGenerator` class to specify event behaviors.

`KAction` objects handle events by implementing the `IRPTEventHandler` method. When an event such as a verification-point failure or connection failure occurs, make the following call to the `KAction` object that is the parent of the event:

```
KAction.registerEvent(eventType, eventBehavior);
```

In the previous example, the `eventType` parameter is the type of failure. The `eventBehavior` parameter is the action to take when the failure occurs.

During the `KAction.finish()` procedure, all registered event behaviors are processed. Behaviors that are registered for a specific event are processed. Applicable behaviors that are specified at a higher level in the event hierarchy are processed.

The following code implements the `RPTEventGenerator` class:

```
public abstract class RPTEventGenerator implements IRPTEventGenerator{
    RPTEvent behavior = null;
    boolean behaviorSet = false;
    IKAAction act = null;
    RPTEvent eventType;

    public void setEventBehavior(IKAAction act, RPTEvent eventType, RPTEvent behavior){
        behaviorSet = true;
        this.behavior = behavior;
        this.act = act;
        this.eventType = eventType;
    }

    public RPTEvent getEventBehavior(){
```

```

    return behavior;
}

public RPTEvent getEventType(){
    return eventType;
}

public KAction getAction(){
    return act;
}
}

```

The following code examples demonstrate how to implement error handling at run time for the `ServerConnection` class.

```

public class ServerConnection extends RPTEventGenerator implements IServerConnection {

    public ServerConnection(String name, int port, ISSLInfo sslInfo,
        INtlmAuthenticationContext ntlmContext,
        IProxyServerInfo proxyServerInfo,
        boolean closeWhenTestCompletes, RPTEvent behav) {
        this.serverAddr = new InetAddressInfo(name, port);
        this.sslInfo = sslInfo;
        this.ntlmCxt = ntlmContext;
        this.proxyInfo = proxyServerInfo;
        this.inUse = true;
        this.closeWhenTestCompletes = closeWhenTestCompletes;
        setEventBehavior(null, new RPTConnectEvent(), behav);
    }
}

```

The behavior for a server connection failure now includes the following code:

```

registerEvent(((IRPTEventGenerator)m_Request.getServerConnection()).getEventType(),
    ((IRPTEventGenerator)m_Request.getServerConnection()).getEventBehavior());

```

Extending the schedule component

When you extend the schedule component in performance testing, another plug-in can contribute options to a schedule. Those options can be set up to persist in the schedule model, and a corresponding user interface can be displayed for editing the new options.

About this task

Another plug-in can contribute options to a schedule, have them persisted in the schedule model, and have a corresponding user interface displayed to edit those options. A schedule object represents a performance testing schedule. It is the main object in the schedule model. Schedules can have only one type of element added to them, a `UserGroups` object. You can use the `com.ibm.rational.test.common.schedule.UserGroup` interface to add a user group to a schedule.

The mechanism to extend scheduling is similar to the mechanism used to extend the test editor. See the "Extending the test editor" topic for more information.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

Extension points for extending schedules

During the process of extending the schedule object, a number of extension points are checked to determine whether they extend different aspects of the schedule object.

The following table lists the extension points:

Extension point	Description
<code>com.ibm.rational.test.common.schedule.editor.optionProvider</code>	This extension point enables a contributor to add options when a new schedule object is created through the File > New wizard command. The <code>class</code> attribute must be set to a class that implements the <code>IOptionProvider</code> interface from the <code>com.ibm.rational.test.common.schedule.extensions</code> package. A child node, <code>forTypes</code> , must be added also, with the <code>type</code> attribute set to <code>com.ibm.rational.test.common.schedule.Schedule</code> .
<code>com.ibm.rational.test.common.editor.frame.work.testOptions</code>	This extension point enables a contributor to specify that the user interface code display or edit the options. <ul style="list-style-type: none"> • Set the <code>label</code> attribute to the desired label string. Each set of contributed options will appear under its own tab in the editor. • Set the <code>order</code> attribute, if desired. A value of zero (0) must not be used, because it is used by the <i>User Load</i> tab. • Set the <code>forTestType</code> attribute to <code>com.ibm.rational.test.common.schedule.Schedule</code>.

If you are contributing options for both the Test and Schedule objects, you must create two instances of the extension definition in your `plugin.xml` file.

Public APIs for extending schedules

The public APIs contain the public interfaces and classes that you can use to extend the Schedule object.

The following table lists the public interfaces and classes:

Classes and interfaces

Description

<pre>com.ibm- .rational- .test.com- mon.sched- ule.editor- .extension- s.IOption- Provider</pre>	<ul style="list-style-type: none"> • The <code>IOptionProvider</code> interface is used in conjunction with the <code>optionProvider</code> extension point to allow contributors to initialize and add their options to a <code>Schedule</code> object when it is created by using the File > New wizard command. • The class has the method <code>public void setDefaultOptionValues(Schedule theSchedule)</code>, which is called after the schedule object is created. In this method, create your option object. This object must be a subclass of the <code>com.ibm.rational.test.common.models.behavior.CBOPtion</code> class. • Once you have created the schedule object, you can add the option object to the schedule object by calling the <code>addOptions()</code> method on the schedule object. See the Javadoc information for more details about the schedule interface.
<pre>com.ibm.ra- tional.test- .common.mod- els.behav- ior.CBOPtion</pre>	<ul style="list-style-type: none"> • This class has no functionality. It only serves as a common type for options that are added to test and schedule objects.
<pre>com.ibm.ra- tional.test- .common- .schedule- .Schedule</pre>	<ul style="list-style-type: none"> • For option contributors, the important method is the boolean <code>addOptions(CBOPtion options)</code> method. Use this method to add your option object to the <code>Schedule</code> object. This should be done through the <code>optionProvider</code> extension point. The method returns <code>true</code> if the option object is successfully added. • If you need to remove your options from the schedule, use the boolean <code>removeOptions(CBOPtion options)</code> method. The parameter is the option object to remove. The method returns a value of <code>true</code> if the option object is successfully removed. • Use the <code>com.ibm.rational.test.common.models.behavior.CBOPtion.getOptions(String str- Type)</code> method to retrieve your options from the schedule object. The parameter is the fully-qualified classname of the object type you want to retrieve. Note that the return type is <code>CBOPtion</code>, so it must correspond to the appropriate type. For example, to retrieve the general options for the schedule, the following code can be used: <code>ScheduleOptions2 theOptions = theSchedule.getOptions(ScheduleOptions2.class.getName());</code>

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help ContentsHCL OneTest Performance API Reference**.

Extending code generation

The code generation subsystem maps Load Test Behavior Model (LTBM) elements to objects of the code generation element model (IModelElement), which in turn are mapped to objects of its ILanguageElement class. As a result, a language element tree is created that contains all equivalent elements of the behavior model test element tree and at the same time determines the structure of code to be generated.

Before you begin

The language elements are typed, named, and have appropriate templates defined for them. Performance testing code generation takes place entirely within the Eclipse workbench. The Eclipse extension point mechanism is used for code generation extensibility to accommodate new protocols.

About this task

The `com.ibm.rational.test.lt.codegen.core` plug-in supports code generation for the generic `Load Test` script extending the `com.ibm.rational.test.lt.execution.core.impl.LTTestScript` class and generic test elements such as loops, containers, transactions, data pools, generic content verification points, data sources and substituters, and generic custom code. Specific protocols such as the `com.ibm.rational.test.lt.codegen.http` plug-in are implemented as extensions of the `com.ibm.rational.test.lt.codegen.core` plug-in. The `com.ibm.rational.test.lt.codegen.schedule` plug-in generates schedules implemented on top of `com.ibm.rational.test.lt.codegen.core` functionality.

For the code generation extension sample, see the plug-in `com.ibm.rational.test.lt.sdk.samples.codegen.socket`.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

Code generation

The `com.ibm.rational.test.lt.codegen.core.extLibraryDependency` extension point has been modified.

The `com.ibm.rational.test.lt.codegen.core.extLibraryDependency` extension point has been modified in the following ways:

- `supportedFeature` has been added with attribute `featureName`, which is the feature ID of this protocol extension as defined by extending the `com.ibm.rational.test.lt.licensing.feature` extension point. If this element is specified, and the test being generated does not contain this feature, the existence of external libraries will not be enforced. This will prevent the developer from requiring that libraries are invalid for various OS platforms. If the element is not defined, external libraries are enforced as before, so no existing code is broken. Essentially these external libraries will not be enforced for tests that do not contain this feature.
- `optionalExtLibraryLocation` has been added with attribute `pathname` (the same as the one for the existing `extLibraryLocation` element). Libraries defined by this element are deployed when present. If they are not present, however, no warning will be raised. An optional external library might have the supported feature defined, so if the test does not contain the feature, the optional library, even if it exists, will not be deployed.
- `optionalExtLibraryLocation` and `extLibraryLocation` have a new attribute, `RelativetoExternalFiles`. Set this attribute to true if you are providing a library that is in the `external_files` directory and to false if you are providing a library that is relative to your plug-in.

Extension points for code generation

During the initialization process a number of extension points are checked for existence of extensions that define different aspects of the code generation behavior.

These extensions must be defined in the protocol specific plug-ins extending core code generation classes.



Note: Java™ has a 64KB limit on the size of a class method code. When there is a possibility of creating large methods that approach this limit, a check has to be programmed into the code generation extension to ensure that the limit is not being exceeded. The guideline is to generate multiple smaller methods instead.

Extension point	Description
<code>com.ibm.rational.test.lt.codegen.core.typeDefDescription</code>	Defines language element types that specify their own translators.
<code>com.ibm.rational.test.lt.codegen.core.elementTranslatorMapping</code>	Defines what <code>AbstractTranslator</code> class extensions are responsible for generating code for code generation model elements.
<code>com.ibm.rational.test.lt.codegen.core.structureDefinition</code>	Defines the class that perform translation at the script level, the <code>ScriptDefinition</code> class.
<code>com.ibm.rational.test.lt.codegen.core.modelElementAdapter</code>	Defines the relationship between low-level behavior model elements and code generation model elements.
<code>com.ibm.rational.test.lt.codegen.core.modelReader</code>	Defines the class that reads the behavior model elements.
<code>com.ibm.rational.test.lt.codegen.core.TemplateLocation</code>	Defines the location of the directory that contains the templates.
<code>com.ibm.rational.test.lt.codegen.core.elementTypeTemplateBinding</code>	Associates element types with the text templates.
<code>com.ibm.rational.test.lt.codegen.core.elementTypeImportMapping</code>	Defines the packages to be imported into the script when a particular element type is used in the test.
<code>com.ibm.rational.test.lt.codegen.core.testProjectDependency</code>	Defines the projects and the plug-ins to be added to the test project's class path.

Generating test code

During the initialization phase of a code generation request, an object extending the `LTTestExtensionPreferences` class is created by calling all plug-ins implementing the `com.ibm.rational.test.lt.codegen.core.codegenProtocolExtension` extension point through their `supportFeatures()` method.

About this task

The `supportFeatures()` method gets the feature list of the test model object, and if it determines that it supports these features, it returns the appropriate `LTTestExtensionPreferences` object. The returned object implements the `com.ibm.rational.test.lt.codegen.core.config.IExtensionPreferences` interface, which is the public part of the `LTTestExtensionPreferences` class.

The code generation is controlled by the `com.ibm.rational.test.lt.codegen.core.CodeGenerator` class. This class uses an `EclipseCodegenConfiguration` object that stores the hash maps for translators, the model element adapters and the templates that are determined by language element types. The container test elements, data pools, and some other independent protocol-specific elements have their own translators defined. Elements that cannot exist on their own are translated as children of their containing elements by calling the `translateChildren()` method of the translator.

The code generator determines the proper translator and calls its `getTranslationFor()` method for the given model element.

The `ElementAdapter` class with its `getAdapterForType()` method determines the proper code generation model element for the given Behavior Model element.

The translation of test elements involves the following steps:

1. The test elements are read from the behavior model, and the appropriate translator is determined for each of them.
2. The language element object is created for the element and its template is determined.
3. The element attributes are read from the model and their corresponding parameters are substituted in the template.

Results

The generation of the code language elements is recursive and generates the language element tree containing all levels of the execution model test elements, each with the template object instantiated and containing all the requested test substitutions. The tree is implemented by using the language element containers as its nodes. The template substitutions for the language element containers for parameters defining multiple elements are processed by the `LangElemCollectionValue` class.

Creating the script class

After all model language elements are processed and the code language elements created, the code for the script class is generated.

About this task

The global script variables need to be declared at the top of the script, and their number and types are not known until all test elements are processed. The names of these variables and other associated information are gathered during the processing of the test element tree and are stored as temporary attributes of the test elements in the test model. The script class creation is accomplished by the `ScriptDefinition` class and is based on a separate script template.

The script template declares the imports and the script globals. It contains the parameters for script type-specific declarations and for methods creating and returning the test elements.

The creation of the top-level script class and test project involves the following steps:

1. The language element tree and the top level script element are created.
2. The generation of script text is performed and the text is stored in Eclipse storage units.
3. The test project is configured, the class path is determined and updated, and the project is built.

Example

The following code represents an example of the script code structure:

```
package customcode;

import com.ibm.rational.test.lt.execution.http.IHTTPRequest;
...

public class Google_Test_C240F3CB2D546DE2A9BDE160BDA411D9 extends
    com.ibm.rational.test.lt.execution.protocol.impl.HTTPTestScript {

    //GLOBAL DECLARATIONS
    private IBuiltInDataSource bds1 = new
        com.ibm.rational.test.lt.kernel.custom.impl.timestampdatasource();

    { builtInDCVars[50] = new BuiltInCorrelationVar(bds1);

        builtInDCVars[50].setProperty(1, "16");
        builtInDCVars[50].setProperty(2, "16"); }

    //TEST CLASS CONSTRUCTOR
    public Google_Test_C240F3CB2D546DE2A9BDE160BDA411D9(IContainer container, String
        invocationId) {

    super(container, "google", invocationId);

    setTimeoutScheme(IKTimeoutControl.CONTINUE);
    setArmEnabled(false);

    public void execute() {
        this.add(page_1(this));
        ...
        super.execute();
    }
    .....

    //page_1 CREATION METHOD
    private HTTPPage page_1(IContainer parent) {

        HTTPPage page = new HTTPPage(parent, "Google", .....) {

            public void execute() {
                this.add(request_1(this);
```

```

        ...
        super.execute();
    }
    .....
};
return page;
}

//request_1 CREATION METHOD
private HTTPAction request_1(IContainer parent) {
HTTPAction reqAction = new HTTPAction(parent, .....);
    .....
harvestContainer_16.addHarvestInstruction ("resp_content", dcVars[50], ..);
    .....
return reqAction;

}
}
}

```

Code generation templates

The code generation subsystem uses declaration templates and creation templates.

The declaration templates contain code for declarations of the methods and classes creating the test element. The creation template contains code that calls these methods to instantiate the test element objects.

The type of the template is reflected in the template name and is defined by the

`com.ibm.rational.test.lt.codegen.core.elementTypeTemplateBinding` extensions.



Note: It is possible to have name conflicts between code generation templates with the same names that come from different protocol extensions. Use template names containing a protocol name (for example: HTTPScript.template) to avoid this problem.

Example

The following example shows a creation template for a DCSubstituter object:

```

ISubRule sub_<PARAM name="subRuleIdx"> = newSubRule(<PARAM name="targetAttr">,
        <PARAM name="offset">,
            <PARAM name="length">,
            <PARAM name="isEncoded">,
            <PARAM name="dataSourceVarName">,
<PARAM name="parentInstancename">, addSubInstruction (sub_<PARAM name="subRuleIdx">);

```

New protocol extensions

This section describes the implementation of the code generation subsystem extension for the socket protocol as an example of using code generation extensibility features.

A dedicated code generation plug-in, called `com.ibm.rational.test.lt.sdk.samples.codegen.socket` was created with subpackages of config, lang, and model. The socket code generation subsystem extends the capabilities of the `codegen.core` plug-in.

The config package contains the `SocketExtensionPreferences` class which contains the `supportsFeatures()` method that expresses interest in tests with feature lists containing the socket feature. Also, its `getSupportedModelElements()` method declares which translator-associated model element types this codegen extension will support.

The lang package contains two main classes, a script definition (`SocketScriptDefinition`) class and a translator (`SocketTranslator`) class. The classes extend the appropriate super classes from the codegen.core plug-in, and defining methods for translating script and test elements. The script definition class overrides the `doScriptLevelTranslation()` method which, after calling its superclass, translates all socket script template parameters specific to the socket protocol. The translator class overrides the `getTranslationFor()` method, handling all non-socket functionality by calling its superclass, and all socket-specific test elements and parameters locally.

The model package contains the `SocketElementAdapter` class, which implements a `getAdapterFor()` method that states what Common Behavior element type this protocol extension handles, and returns the corresponding codegen model element.

The template directory was also added to the protocol extension plug-in, and socket-specific templates were created and placed there. All necessary extensions were defined in the protocol extension plugin.xml file.

For the code generation extension sample, see the plug-in `com.ibm.rational.test.lt.sdk.samples.codegen.socket`.

Public APIs of codegen.core

The public APIs contain the public interfaces and classes used to extend the code generation core subsystem.

The following table lists the public interfaces and classes:

Package	Public classes and interfaces
<code>com.ibm.rational.test.lt.codegen.core.config</code>	<ul style="list-style-type: none"> <code>IExtensionPreferences</code> <code>InitializationException</code> <code>ConfigurationException</code>
<code>com.ibm.rational.test.lt.codegen.core.template</code>	<ul style="list-style-type: none"> <code>ITemplate</code> <code>Template</code> <code>ITemplateParameter</code> <code>LangElementCollectionValue</code> <code>LangElementParameterValue</code>
<code>com.ibm.rational.test.lt.codegen.core.lang</code>	<ul style="list-style-type: none"> <code>IStructureDefinition</code> <code>ILanguageElement</code> <code>LanguageElement</code> <code>ITranslator</code>

Package	Public classes and interfaces
<code>com.ibm.rational.test.lt.codegen.core.model</code>	<ul style="list-style-type: none"> • <code>ICoreTranslationConstants</code> • <code>TranslationException</code>
<code>com.ibm.rational.test.lt.codegen.ltttest.config</code>	<ul style="list-style-type: none"> • <code>IModelElement</code> • <code>ModelElement</code> • <code>IModelElementAdapter</code>
<code>com.ibm.rational.test.lt.codegen.ltttest.lang</code>	<ul style="list-style-type: none"> • <code>LTTestExtensionPreferences</code> • <code>ILTTestTranslationConstants</code> • <code>LTTestScriptDefinition</code> • <code>LTTestTranslator</code>
<code>com.ibm.rational.test.lt.codegen.ltttest.model</code>	<ul style="list-style-type: none"> • <code>LTTestElementAdapter</code>

Extending the run-time environment

The run-time environment defines the plug-ins on which all other plug-ins depend. The run-time environment is responsible for defining a structure for plug-ins and the implementation detail behind them.

Blocked Action detection

The performance testing execution engine provides the capability of detecting that an Action is blocked. The definition of blocked is provided by the author of the Action. The definition is provided as an amount of time an Engine Worker Thread is allowed to be unresponsive in the course of executing the Action before it is considered blocked. The performance testing execution engine logs the detection of blocked actions at level FINEST in the Problem Determination Log. In the simplest case, the run would be hung, and discovery of the blocked action would come from examining the Problem Determination log after clicking **Stop** to end the run.

In addition to detecting blocked actions, a protocol can be notified that the action is blocked. A reference to the blocked Engine thread is provided with the action, making it possible for a protocol to send a message to the blocked thread. Upon notification of an action being blocked, the protocol can retry the action, finish the action and move to the next action, or finish the virtual user execution.

The `IKAction` interface will provide the following method as an entry point for protocols to react to a blocked action condition:

`public void blocked()`. The `KAction` object state, upon entering `blocked()`, is potentially corrupted. The protocol writer must assume any or all data associated with the Action is not safe. Also, any locks which may have been held during `execute()` have been released.

```
public long getBlockedTimeout()
```

Returns the action blocked timeout value. Default is 0, which means block indefinitely

```
public WorkerThread getWorkerThread()
```

Returns the Engine WorkerThread executing the action when the blocked state was detected.

`IEngine`

The performance testing engine interface has added `public boolean createWorker()`. The purpose of `createWorker()` is to allow the sentinel thread monitoring workers to add additional workers in response to having removed workers found executing blocked actions.

Extending subsystem management during a test run

A subsystem is a collection of classes in a discrete component within the performance testing engine that provides a service to many actions. For example, in test execution, the `KernelWait` subsystem manages think and sleep time for virtual users while a test is running.

About this task

Here are additional examples of the services that subsystems provide:

- Tracking server responses. For example, consider tests in which an HTTP request is sent to a server, and you need to know when the response returns. Instead of tying up a thread to wait for the response, a subsystem can do this while the rest of the threads perform other actions. The subsystem can provide notification or reissue an action when the server response occurs.
- Handling asynchronous communication.
- Managing the sleeping action for virtual users.
- Managing logging. For example, creating a custom execution history can take a long time. You can assign a subsystem to do this without tying up a thread with this process. You can set up a subsystem to take care of the special logging actions while the rest of the actions perform other things.

Creating the performance test engine subsystem sample

A performance test engine subsystem provides services to one or more actions. An action that uses a subsystem during a test run is known as a recurrent action. Most actions contain `finish()` at the end of their `execute()`. A recurrent action requests service from a subsystem before the end of `execute()`. After the service is provided, the subsystem that provides the service updates state information in the action and reissues the action for execution. Based on the state information, a performance test engine worker thread takes a different course of execution during the recurring call to `execute()` and eventually calls `finish()` to end the action.

The following list gives an overview of the steps that are required to use this sample:

- Creating the performance test engine subsystem sample
- Informing the performance test engine of the existence of the subsystem
- Creating a simple test and schedule
- Adding SampleAction and SampleSubsystem to the project
- Running the schedule with SampleAction using SampleSubsystem

Informing the test engine that the subsystem exists

Subsystems are identified to the performance test engine with extensions. Typically, a new protocol indicates in its plugin.xml file the class name of the subsystem that it is providing. This example modifies an existing performance test plug-in and specifies the class name of the sample plug-in in the `plugin.xml` file.

About this task

To identify the subsystem to the performance testing engine, complete the following steps.

1. Open a command prompt, and change directory to the `__PT_ACRONYM__` plug-ins directory.
2. Save a copy of the existing HTTP execution JAR. For example: `C:__BRAND_NAME__\common`

```
\plugins>copy com.ibm.rational.test.lt.execution.http_7.0.0.v200609010404.jar
com.ibm.rational.test.lt.execution.http_7.0.0.v200609010404.jar.orig
```
3. Make a new copy of the HTTP execution JAR for modification. For example: `C:__BRAND_NAME__\common`

```
\plugins>copy com.ibm.rational.test.lt.execution.http_7.0.0.v200609010404.jar foo.jar
```
4. Extract the plugin.xml file. For example: `C:__BRAND_NAME__\common\plugins>jar xvf foo.jar plugin.xml` **Note:**
 You must have a Java™ SDK in your PATH statement so that you have access to the JAR utility.
5. Add these lines to the plugin.xml file just above the `</plugin>` line at the bottom:

Example

```
<extension
  point="com.ibm.rational.test.lt.execution.Subsystem">
  <Subsystem
    feature="com.ibm.rational.test.lt.feature.lt"
    class="test.SampleSubsystem"/>
  </extension>
```

6. Start HCL OneTest™ Performance with the `-clean` option so that the plug-ins are reloaded, for example: `C:\IBM`

```
\RPT>eclipse -clean.
```

Creating a simple test and VU Schedule

You must create a simple performance test and VU Schedule. By creating and playing back a test and a VU Schedule, code is generated for both. Later, you will modify the generated test code so that the test contains a special action that uses the sample subsystem.

1. Create a performance test project (**File > New > Performance Test Project**).
2. Type a project name, for example, `testproj`, and then click **Finish**.
3. When the Create New Test From Recording prompt opens, click **Cancel**.

4. Right-click `testproj` and select **New > Test**.
5. Expand **Test Assets**, select **New Test**, and then click **Next**.
6. Name the test, for example, `subtest`, and then click **Next**.
7. In the Test Attributes window, click **Next**.
8. In the **Protocols and Features** window, select the **HTTP Protocol** check box.
9. In the HTTP Extension window, in the **Number of HTTP pages to generate** field, type **0** (zero).
10. Specify the connection details and click **Finish**.
11. Right-click `testproj` and select **New > Performance Schedule**.
12. Type a name for the schedule, for example, `Schtest`, and then click **Finish**.
13. Select **User Group 1**, and then click **Add > Test**.
14. Select `subtest`, click **OK**, and then click **File > Save**.
15. Right-click `Schtest`, and then select **Run as > Performance Schedule**.

Adding SampleAction and SampleSubsystem to the project

Import the performance test and schedule source files into the `testproj` project.

1. Open the Java™ perspective, open `src`, and then right-click `test`.
2. Select **Import**.
3. Import `SampleAction.java` and `SampleSubsystem.java`.
4. Edit the generated `subtest` file. The name will begin with `Subtest_Test_` and end in `.java`.
5. At the bottom of the constructor for the test, add the following line: `add(new SampleAction(this, "SampleAction"));`

Running the schedule with SampleAction using SampleSubsystem

Modify the generated test code to use `SampleAction`, so that the subsystem is active during the next playback.

1. Right-click the schedule and select **Run As > Performance Schedule**.
2. When the run completes, right-click the results in the Performance Test Runs view and select **Display Test Log**.
3. Click **Events**.
4. Open the test log hierarchy and navigate through the events of the schedule playback. Look for message events where `SampleAction` indicates it is requesting service from the subsystem, and where `SampleAction` recognizes that it has received services from `SampleSubsystem`.

SampleAction.java code sample

This is an example of `SampleAction.java`.

Example

SampleAction.java

```
package customcode;
```

```

import com.ibm.rational.test.lt.kernel.IKSubsystem;
import com.ibm.rational.test.lt.kernel.action.IContainer;
import com.ibm.rational.test.lt.kernel.action.impl.KAction;

public class SampleAction extends KAction {
    final String subsystemName = "test.SampleSubsystem";
    private boolean serviced = false;

    public SampleAction(IContainer arg0, String arg1) {
        super(arg0, arg1);
        // TODO Auto-generated constructor stub
    }

    public void execute() {
        if (serviced) {
            reportMessage("SampleAction execute(): service completed");
            finish();
        } else {
            IKSubsystem subsystem = getSubsystem(subsystemName);
            if (subsystem != null) {
                reportMessage("SampleAction execute(): requesting service");
                subsystem.enqueue(this);
            } else {
                reportMessage("SampleAction execute(): Cannot find subsystem '" + subsystemName + "'");
                finish();
            }
        }
    }

    public void setServiced() {
        serviced = true;
    }
}

```

SampleSubsystem.java code sample

This is an example of SampleSubsystem.java.

Exemple

SampleSubsystem.java

```

package customcode;

import com.ibm.rational.test.lt.kernel.action.IKAction;
import com.ibm.rational.test.lt.kernel.engine.impl.Queue;
import com.ibm.rational.test.lt.kernel.impl.KSubsystem;

/**
 * Sample __PT_ACRONYM__ Engine Subsystem
 */

public class SampleSubsystem extends KSubsystem {
    private Queue sampleSubsystemQueue;
    private boolean stopRequested = false;
    private SampleAction client;
}

```

```

public SampleSubsystem(String name) {
    super(name);
    sampleSubsystemQueue = new Queue();
    sampleSubsystemQueue.setBlocking(true); // Allows for waiting for something to appear on the queue
}

/*
 * Actions enter the subsystem for service via a call to enqueue().
 * An action can get a reference to the subsystem using the IKAAction
 * getSubsystem() method.
 *
 * @see
com.ibm.rational.test.lt.kernel.IKSubsystem#enqueue(com.ibm.rational.test.lt.kernel.action.IKAAction)
 */
public void enqueue(IKAAction action) {
    sampleSubsystemQueue.enqueue(action);
}

/*
 * Message to the subsystem to stop.
 *
 * @see com.ibm.rational.test.lt.kernel.IKSubsystem#shutdown()
 */
public void shutdown() {
    stopRequested = true;
}

/*
 * @see java.lang.Thread#run()
 */
public void run() {
    while(!stopRequested) {
        ringIn(); // Informs engine subsystem is healthy
        client = null;

        // If nothing to do wait for work
        updateJob("Idle");
        client = (SampleAction)sampleSubsystemQueue.dequeue(pingTime);

        // This subsystem's work will be to touch an attribute of the action
        if (client != null) {
            updateJob("Servicing " + client.getName()); // Good for debugging

            client.setServiced();
            dispatch(client); // Serviced action leaves subsystem
        }
    }
}
}
}
}

```

Extending initialization and finalization during a test run

You can specify code for your protocol to be executed by the HCL OneTest™ Performance engine threads at strategic points during startup and shutdown. For example, you can specify code to load libraries, unload libraries, or perform other initialization or cleanup as required by the protocol.

Before you begin

You create a class that implements `IKInitializeFinalize`. The interface requires the following methods:

- `public void initializeEngine()`
- `public void finalizeEngine()`
- `public void initializeWorker()`
- `public void finalizeWorker()`
- `public interface IKInitializeFinalize`

The `IKInitializeFinalize` interface provides a way for protocols to specify code that must be executed to the HCL OneTest™ Performance engine:

- once by the engine at startup
- once by the engine at shutdown
- once by each engine worker thread at startup
- once by each engine worker thread at shutdown

Use this startup and shutdown code when it is necessary for the engine to execute initialization or shutdown code. Also use this code for each worker thread before test execution occurs or after test execution finishes.

1. Specify a dependency on `com.ibm.rational.test.lt.execution` in the `plugin.xml` file of a protocol.
2. Use **Add under plugin Extensions** to specify an extension for `com.ibm.rational.test.lt.execution.InitializeFinalize`.
3. Create a new extension element called `InitializeFinalize`. This element must have the following properties:

Choose from:

- `class`: The class name that implements `IKInitializeFinalize`
- `id`: The protocol feature ID
- `dependsOn`: Leave blank

Example

For example:

- `class="com.ibm.rational.test.lt.execution.http.impl.HTTPInitializeFinalize"`
- `id="com.ibm.rational.test.lt.feature.http"`
- `dependsOn=`

Public APIs for run time

The public APIs contain the public interfaces and classes that you can use to extend the run-time environment functionality.

The following table lists the public packages:

Package	Description
<code>com.ibm.rational.test.lt-.kernel</code>	Contains the factory, counter, monitoring, and constants classes.
<code>com.ibm.rational.test.lt-.kernel.action</code>	Contains the classes and interfaces necessary to define conditions and the basic functionality that all kernel actions should implement
<code>com.ibm.rational.test.lt-.kernel.arbitrary</code>	Contains the <code>IArbitrary</code> interface.
<code>com.ibm.rational.test.lt-.kernel.custom</code>	Contains the interfaces that enable additions of custom code to a performance test.
<code>com.ibm.rational.test.lt-.kernel.engine</code>	Contains the interfaces used to manage the process of arranging actions onto queues.
<code>com.ibm.rational.test.lt-.kernel.io</code>	Contains the interfaces that get the buffer factory.
<code>com.ibm.rational.test.lt-.kernel.library</code>	Contains the class that loads the library.
<code>com.ibm.rational.test.lt-.kernel.logging</code>	Contains the interfaces to manage the cache.
<code>com.ibm.rational.test.lt-.kernel.runner</code>	Contains the <code>IRatlRunner</code> interface.
<code>com.ibm.rational.test.lt-.kernel.services</code>	Contains the interfaces for dataset, loop control, test information.
<code>com.ibm.rational.test.lt-.kernel.statistics</code>	Contains the interfaces for statistical counters.
<code>com.ibm.rational.test.lt-.kernel.util</code>	Contains the interfaces and classes for kernel utilities.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

Extending the test log viewer

After running a test, the test results are saved in a test log. Use the Test Log viewer to check the results for specific events, such as the script start and end, loop, invocation, message, or verdict.

About this task

You can extend the test log viewer for your protocol. For information about how to do this, see the Testing Performance Tools Platform (TPTP) documentation.

Extending evaluation results

Reports are a specification of how performance test data should be extracted from the statistical model and presented to the user.

About this task

The persisting reports contain no statistical data. After a report is created, it can be focused on any statistical model.

The user interface components for evaluation results are the Performance Test Runs view, Results Viewer, Report Wizard, Execution History Viewer (TPTP), Protocol Data view and Properties view in Eclipse. For details about how to use the user interface components, see the "Evaluating results" topic in the product information center.

Besides the user interface components, generic counters and aggregators also enable you to extend the evaluation of results. The generic counters represent the mechanism for specifying the model path of statistical data to be displayed in the Results Viewer. The generic counters are specified with the `com.ibm.rational.test.It.execution.results.DynamicCounter` extension point. The aggregators calculate the majority of the data contained in the statistical model and reduce the amount of data that must be transmitted. The aggregators are deployed through the `com.ibm.rational.test.It.execution.results.data.aggregation.Aggregator` extension point. For details about generic counters, see the "Generic counters" topic in the product information center. For details about extending evaluation results with aggregators and other classes and interfaces, see the Javadoc information.

For an example of extending evaluation results, see the plug-in `com.ibm.rational.test.It.sdk.samples.results.socket`.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

Aggregation of statistical data

To minimize network load, the guideline is to send only a minimal subset of the statistical data that can be used to aggregate the remainder of the statistical data. This aggregation is handled by the aggregation subsystem. There are two primary types of aggregators: transfer aggregators and standard aggregators. Transfer aggregators transfer data of like origin from individual nodes to the All Hosts node. Standard aggregators calculate additional data based on the data sent across the network from the execution engine. An example of a transfer aggregator task is creating the counter "Pages/Response Time/Google/Average interval" on the All Hosts node by considering the same counter from three drivers. An example of a standard aggregator task is calculating counter "Pages/Response Time/Google/Average cumulative" on the All Hosts node by tracking and weighting each "average interval" value in real time.

For more information on aggregation, refer to the Javadoc information for the following transfer aggregators:

- `MaxTransferAggregator`
- `MinTransferAggregator`
- `ScalarTransferLastValueAggregator`
- `AverageTransferAggregator`

as well as the following standard aggregators:

- AverageAggregator
- MaxAggregator
- MinAggregator
- PercentAggregator
- PercentAggregator_NonInclusive
- RateAggregator
- TotalScalarAggregator

Extending report counters

In performance test reports, generic counters are collectable queries in the user interface that dynamically gather specific information from the statistical model, such as the number of page hits, the response time, the response success, and information about verification points. Counters are dynamic. The counter wizard is used to add counters to reports. You can extend the counter wizard for specific protocols.

About this task

You extend the counter wizard in the ReportAction extension point, which is contained in the `com.ibm.rational.test.lt.execution.results` plug-in.



Note: Any report that provides mean average data now also provides standard deviation data. If a protocol-specific report contains a mean counter, the extension is able to add the corresponding standard deviation counter onto that report.

The ReportAction extension point

`ReportAction` enables interactions directly on the report screen as well as in the Performance Test Runs view. Use this extension point to enable report menus and menu items in the tree and to extend the counter wizard for your protocol. Implementors of generic `ReportActions` must extend the abstract class `com.ibm.rational.test.lt.execution.results.actions.ReportAction`, while implementors of `AddCounterAction` must specify relevant data in `plugin.xml` only.

The following table describes the elements and attributes of the `ReportAction` extension point.

Elements and attributes	Description
<code>ReportActionGroup</code>	Provides menu groupings for <code>ReportActions</code> that define the location of report actions within menus in reports.
<code>menuPos</code>	Position of a menu group in a report menu. Valid values: "start" (top) "mid1," "mid2," and "end" (bottom).
<code>ReportAction</code>	Provides interactivity on performance test reports and in the Performance Test Runs view. Defines the behavior of actions in reports.

Elements and attributes	Description
<code>groupID</code>	ID of the <code>ReportActionGroup</code> (menu grouping) that contains this <code>ReportAction</code> menu item.
<code>actionprovider</code>	Behavior provider that extends <code>com.ibm.rational.test.lt.execution.results.ReportAction</code> .
<code>Menutext</code>	Text for this menu item.
<code>Icon</code>	The project-relative path to the icon for this menu item.
<code>Tooltip</code>	Tooltip for menu item.
AddCounterAction	Provides a wizard for adding and removing specified statistical descriptors (counters).
<code>groupID</code>	ID of <code>ReportActionGroup</code> that contains this wizard action.
<code>Menutext</code>	Menu text for an agent.
<code>Icon</code>	The project-relative path to the icon in the wizard for this action.
<code>Tooltip</code>	Tooltip for this action.
<code>allowAllAvailable</code>	When true, a check box is displayed at the bottom of the wizard that reads "Automatically add localizedCounter-CategoryName counters to graphic as they appear in result." If the user selects this check box, any counter that appears in the stat model that meets the specifications of the AddCounter action is automatically added to the graphic of focus. For an example of this feature in use, see the Add/Remove Resource Counters wizard in the performance testing product.
<code>agentID</code>	Returns the ID of the <code>TRCAgent</code> to which this add wizard action applies. Agent IDs can be declared as "not" to specify systems under test, for example <code>!com.ibm.rational.test.lt.execution.results.XMLStatisticalDataProcessor</code> . Declaring a "!" ID causes all other agents to be included. If this attribute is blank, it defaults to <code>com.ibm.rational.test.lt.execution.dataprocessor.XMLStatisticalDataProcessor</code> which is the performance testing Statistical agent.
<code>wizardIcon</code>	The project-relative path to the icon that is displayed on the wizard page.

Elements and attributes	Description
<code>modelBasePath</code>	Specifies the root path from the statistical model from which <code>AddCounterAction</code> should pull counters for the add/remove action.
<code>cshelpID</code>	The context-sensitive help ID for this wizard.
<code>localizedCounterCategoryName</code>	The localized description of the type of counters this wizard processes. This string is substituted into the wizard to describe what the user is adding to the report. For example, in the string "Add/Remove Resource Counters Wizard", "Resource Counters" is the <code>localizedCounterCategoryName</code> . This name should be plural as indicated above.
<code>showScale</code>	Makes data readable within the space provided for it in the user interface. If <code>showScale</code> is true, the wizard enables the user to adjust the scale of counters relative to each other. It also enables a recommended scale to be calculated when data is added to a report from the wizard. For an example of this feature in use, see the Add/Remove Resource Counters wizard in the performance testing product.
<code>showScope</code>	<p>"Scope" refers to the nodes from which data is pulled from in the statistical model. If true, the user is provided with a control to specify the scope of a counter. Valid scopes are:</p> <ul style="list-style-type: none"> • All Locations Data is pulled from any node in the model that has data meeting the specifications of an <code>AddCounterAction</code>. • Systems Under Test Data is pulled from any node that does not contain an "Statistical Agent." This signifies that the node is a "driver node." • Selected location Data is pulled only from the node on which the report is focused. <p>For an example of this feature in use, see the Add/Remove Resource Counters wizard in the performance testing product.</p>

Elements and attributes	Description
<code>showAgents</code>	If true, the wizard shows agents in the tree hierarchy. If not included or if false, descriptors are shown as the root objects.
<code>defaultScope</code>	Specifies the default scope to be used by the wizard. "Scope" refers to the nodes in the wizard from which data is pulled from in the wizard. It can be used with or without the <code>showScope</code> attribute. Valid entries are: "CURRENT", "SUT", and "ALL."
Enablement	Controls when actions are visible and selectable.
<code>type</code>	Specifies selected object types for which this action should be enabled. Any object that can be identified in a report or in the Performance Test Runs view (PTR) can be specified as a valid type, for example, a chart or table as shown in the PTR or in a report.
<code>path</code>	Controls visibility. If no path is specified, the item is visible on any selection where "type" is valid. Path also specifies the model path to the data to be referenced within the selected object for menu-item visibility. Paths can contain wildcards, and more than one path can be specified. An example path specification is as follows: <pre data-bbox="831 1150 1421 1213"><path value="Pages,Response Time,*,Average Cumulative"/></pre>
<code>runstate</code>	Controls enablement (not visibility) by run state. If true, the menu item is enabled only if a run is in progress.
<code>Filterstate</code>	Controls enablement (not visibility) by filter state. If true, the menu item is enabled only if the selected object is filtered.

The following is an example of a `ReportAction` extension point.

```
<extension
  point="com.ibm.rational.test.lt.execution.results.ReportAction">

  <ReportAction
    groupId="com.ibm.rational.test.lt.execution.results.ImportGroup"
    menutext="%IMPORT_RESMON_COUNTERS"
    tooltip="%IMPORT_RESMON_COUNTERS_TOOLTIP"
    icon="icons/elcl16/import_resmon_data.gif"
    actionprovider="com.ibm.rational.test.lt.execution.rm.actions.ImportResourceCounters"
  <Enablement>
    <type objecttype="com.ibm.rational.test.lt.execution.results.view.countertree.MonitorTreeObject"/>
    <type objecttype="com.ibm.rational.test.lt.execution.results.view.countertree.GraphicTreeObject"/>
```

```

<type objecttype="com.ibm.rational.test.lt.execution.results.view.graphics.Graphic"/>
<runstate active="false"/>
</Enablement>
</ReportAction>
<AddCounterAction
  agentID="!com.ibm.rational.test.lt.execution.dataprocessor.XMLStatisticalDataProcessor"
  allowAllAvailable="true"
  cshelpID="add_resource_wiz"
  defaultScope="ALL"
  groupId="com.ibm.rational.test.lt.execution.results.addcountersgroup"
  icon="icons/elcl16/add_res_ctr.gif"
  localizedCounterCategoryName="%RESOURCE_COUNTERS_DESCRIPTION"
  menutext="%ADD_RESOURCE_COUNTER"
  showAgents="true"
  showScale="true"
  showScope="true"
  tooltip="%ADD_RESOURCE_COUNTER_TOOLTIP"
  wizardIcon="icons/wizban/add_res_wiz.gif"
  <Enablement>
    <type objecttype="com.ibm.rational.test.lt.execution.results.view.graphics.Graphic"/>
    <type objecttype="com.ibm.rational.test.lt.execution.results.view.countertree.GraphicTreeObject"/>
  </Enablement>
</AddCounterAction>
</extension>

```

Extending default reports

You can specify the default performance report for your protocol. In the Preferences page in performance testing, the **Determine the default report based on protocols in test** check box is selected by default. You can specify, for your protocol, the report that automatically displays during a test run. You define the default performance report in the `RPTReport` extension point, which is contained in the `com.ibm.rational.test.lt.execution.results` plug-in.

The RPTReport extension point

Use this extension point to define performance testing reports that are installed with the product and are available for "Restore to default."

The following table describes the elements and attributes of the RPTReport extension point.

Elements and attributes	Description
<code>Report</code>	A performance testing report definition. Can be a part of a <code>ReportGroup</code> , however, any report that applies only to a specific protocol should be assigned to that protocol report group.
<code>path</code>	The provider plug-in relative path to the <code>.view</code> file that defines the performance report, for example: <code>./reports/my_report.view</code> .

Elements and attributes	Description
<code>menuText</code>	A localized string for the Report menu item (for example, "Display Performance Report"). Localize this string using the "%" prefaced key. When this string is retrieved from the extension, the localized value from the plug-in resource bundle is provided. Refer to the example that follows in this topic.
<code>icon</code>	Plug-in root-relative path to an icon used with the menu of this report, for example: <code>/icons/my_icon.gif</code>
<code>restrictToPostRun</code>	To have this report available only after the run has completed, set this Boolean variable to "true."
<code>id</code>	A unique identifier for this report.
<code>PostRunGenerator</code>	Implies <code>restrictToPostRun</code> . If a report requires post-run processing (for example, a Page Percentile report that calculates its data based on the entire test run), this attribute can specify an implementor of <code>PostRunReportGenerator</code> . Refer to the example that follows in this topic.
<code>isTemplate</code>	Note: this attribute is not currently available.
ReportGroup	Grouping for protocol-specific reports. Results in a drop-down menu labeled with text provided in <code>menuText</code> . All protocol-specific reports must have a protocol group.
<code>menuText</code>	Text that labels the protocol group drop-down menu. Should be localized using the "%" prefaced key.
<code>capability_id</code>	Note: This attribute is not currently available.
<code>defaultReportID</code>	The ID of the default report for the protocol associated with this group.
<code>protocol_id</code>	The ID of the protocol to which this <code>ReportGroup</code> applies.

The following example shows an `RPTReport` extension point.

```
<Report
  menuText="%DISPLAY_TRANS_REPORT_LABEL"
  icon="icons/elcl16/trans_report.gif"
  path="reports/Transaction Report.view"
  id="com.ibm.rational.test.lt.execution.results.transactions"/>
<ReportGroup menuText="%HTTP_REPORTS">
  <Report
    menuText="%DISPLAY_PERF_REPORT_LABEL"
    icon="icons/report_default.gif"
    path="reports/default.view"
    id="com.ibm.rational.test.lt.execution.results.performance"/>
```

```

<Report
  menuText="%PAGEEL_REPORT"
  icon="icons/elcl16/pageelem_report.gif"
  path="reports/Page Element Report.view"
  id="com.ibm.rational.test.lt.execution.results.pageelement"/>
<Report
  restrictToPostRun="true"
  menuText="%PERCENTILE_REPORT"
  icon="icons/report_percent.gif"
  path="reports/Percentile Report.view"

  PostRunGenerator="com.ibm.rational.test.lt.execution.results.view.controller.PercentileReportControll
er"
  id="com.ibm.rational.test.lt.execution.result.percentile"/>
<Report
  menuText="%VP_REPORT"
  icon="icons/report_vp.gif"
  path="reports/Verification Point Report.view"
  id="com.ibm.rational.test.lt.execution.results.vp"/>
</ReportGroup>

```

Public APIs for evaluate results

The public APIs contain the public interfaces and classes that you can use to extend the evaluate results functionality.

The following table lists the public packages:

Package	Description
<code>com.ibm.rational.test.lt-.execution.results.data</code>	Contains the <code>IStatModelFacade</code> interface used to access and modify the performance testing statistical model.
<code>com.ibm.rational.test.lt-.execution.results.data-.aggregation</code>	Contains the aggregation classes used to aggregate statistical data in real time.
<code>com.ibm.rational.test.lt-.execution.results.data-.aggregation.aggregators</code>	Contains the aggregator classes that aggregate statistical data in real time and place the newly calculated data on the host which contains the aggregators dependency data.
<code>com.ibm.rational.test.lt-.execution.results.data-.aggregation.transferaggregators</code>	Contains the aggregator classes that aggregate statistical data based on data from all hosts in the resultset. Data calculated by these aggregators is placed in the "All Hosts" host.
<code>com.ibm.rational.test.lt-.execution.results.inter-nal.actions</code>	Contains the <code>PostRunReportGenerator</code> interface that is specified in the extension point <code>RPTRReport</code> . Post Report generators are used to calculate statistical data after a run has completed.

Package	Description
<code>com.ibm.rational.test.lt- .execution.results.view- .data.stringtranslator</code>	Contains the <code>IRPTStatStringTranslator</code> interface extended by the implementors of the <code>com.ibm.rational.test.lt.execution.results.StatisticalStringTranslator</code> class. String translators are used to localize strings contained in the statistical model.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

Chapter 9. Test Manager Guide

This guide describes how to keep track of the performance of the application by evaluating the test results. This guide is intended for test managers.

Evaluating results in web analytic reports

After the test run, evaluate the results in the web analytic reports. Web analytics collect data using new technologies thereby providing better user experience.

Comparing results among runs

To analyze the difference between two or more reports, you can compare them. For example, to analyze the performance of the application at different time slots or different milestone builds, you can compare two runs.

About this task

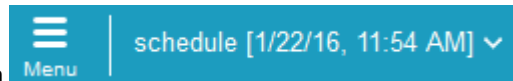
You can compare the test runs from the Test Navigator view or from the web analytics report itself. To compare test runs from the Test Navigator view, select the runs, right-click, and click **Compare Results**.


You can compare the test runs that are in the same project or in the different projects. When comparing multiple runs, you cannot compare multiple time-ranges or stages.

To compare runs from the web analytics report:

1. Open the run or report to serve as the basis for comparison.

2. Click the name of the run next to the **Menu** option



3. Click **Add**  and navigate to the run to compare with.

Multiple runs are displayed in the report.

4. **Optional:** To add, remove, or move the position of the runs, click **Manage**



Related information



[Comparing schedule stages on page 773](#)


Comparing schedule stages

When you are running a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule. that contains stages, time ranges are automatically created for each stage. You can view a report that compares these stages, and you also can set preferences to display the report automatically at the end of a staged run.

About this task

In addition to comparing stages, you can add time ranges and compare them. To view the compare report automatically at the completion of a run, click **Window > Preferences > Test > Performance Test Reports**, and select **Launch Compare report when staged run completes**.

1. Open the run that consists of stages.
By default, reports are displayed for the entire run.
2. Click the **Entire Run** menu  and select the stages to compare.
Both the running and completed stages show up in the list.
3. To add a new time range, click **Add**  in the **Entire run** menu.
4. In the **Time Range** dialog box, specify a name, start time, and end time of the run and click **Apply**.

 **Note:** When you compare stages in a run, you cannot compare data from various geographies at the same time.

Comparing results from various regions or agent locations

When you run a schedule that includes agents from different regions, use the Web Analytic reports to compare the performance data from these regions.

Before you begin

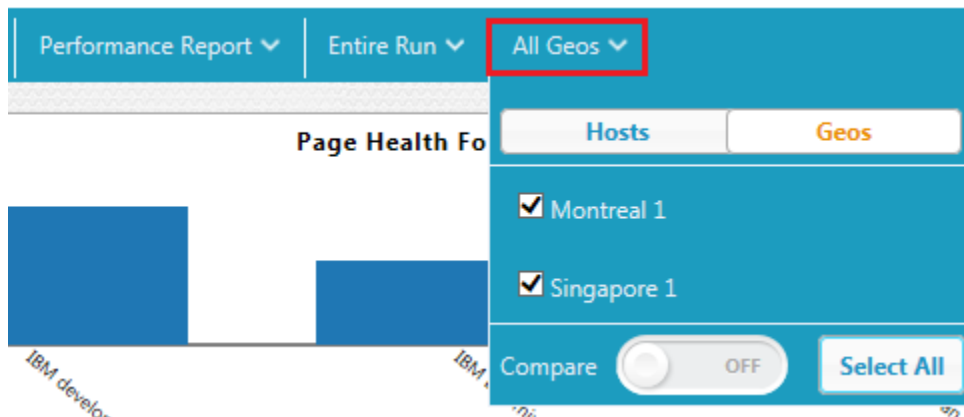
Run a schedule with on premise agents on different geographic regions. See [Running schedules on page 608](#).

About this task

When comparing agents, open the Location asset and in the **General Properties** tab, add an `RPT_GEO` property with any value. This value is then displayed in the **All Geos** menu of the report.

To compare performance data:

1. From the Test Navigator view, open the schedule run that includes remote agents. The name of the run corresponds with the name of the schedule and has a timestamp.
2. On the report toolbar, click **All Geos**.

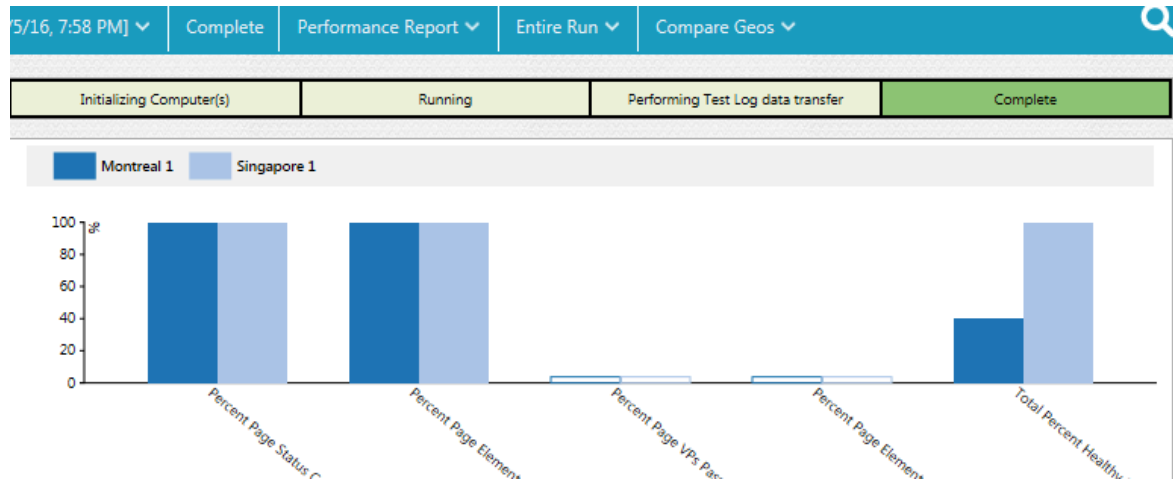


3. Select the regions that you want to compare and click **Compare**.



Note: When you compare data from various regions, you cannot compare stages in a run at the same time.

The report shows the data in the compare mode.



Generating functional test reports

You can generate functional test reports of your tests, which summarize the pass or fail verdicts of elements in the test log. Functional reports are generated from the test run as HTML files that use predefined report designs.

Before you begin

Before you can generate a functional report, you must successfully run a test or schedule and produce a test run.

The following report types are available:

- Extensible Stylesheet Language Transformation (XSLT) reports: These reports are faster to generate, but do not contain graphs.
- Business Intelligence and Reporting Tools (BIRT) report: These reports contain graphs but are slower to generate. You can customize and create your own BIRT report designs in the Report Design perspective of the workbench. BIRT report generation is not supported when the workbench is running in a VMWare Windows™ image.



Note: If you use your own XSLT style sheets, verify that the style sheets contain this line: `<xsl:param name="languagePack" select="'default'"/>`

1. In the **Test Navigator**, select a test run or runs.

You can use the Ctrl key to select multiple test runs or schedule runs. You cannot generate a functional report that contains more than 5000 calls or objects.

- In the **Test Runs** view, right-click the test runs and select **Generate Functional Test Reports**.

Result

This opens the **Generate HTML Functional Test Report** wizard.

- Select the location in the workspace where you want to generate the functional report, and type the **Functional report base name**. A time stamp and the type of report is appended to this base name when the report is generated.

If you want to keep the temporary XML file that is created to generate the report for debugging purposes, select **Keep intermediate XML data**.

- Click **Next**.
- Select a predefined report designs or click **Add** to add a custom BIRT report design or an XSLT style sheet.

Choose from:

- **Common Functional Test Report:** This produces a generic functional test report for all test protocols.
- **SAP Functional Report:** This produces a functional test report for SAP tests.
- **Services - Failed events:** This produces a functional test report for web service tests. The report contains only failed events. Events with other verdicts are not shown in the report.
- **Services - Failed tests:** This produces a functional test report for web service tests. The report contains only failed tests. Tests with other verdicts or other event types are not shown in the report.
- **Services - Full:** This produces a functional test report for web service tests. The report contains detailed information on all events.
- **Services - Summary:** This produces a brief summary functional test report for web service tests.
- **Services - Truncated:** This produces a functional test report for web service tests. The report contains detailed information on all events, but truncates XML contents after 500 characters.

One functional report is generated for each selected report design. Report designs marked with **(xslt)** use XSLT style sheets and are more suitable for larger reports.

- Click **Finish**.

Results

The functional reports are generated as HTML files in the specified location in the workspace.

Publishing test results to the server

The test results indicates the quality of application under test. Different stakeholders to the application might want to check the quality of application but do not have the workbench or desktop client installed. As a desktop client user, you can publish the test result to HCL OneTest™ Server so that others can view it from the web browser.

Before you begin

- Accessed HCL OneTest™ Server.
- Created an offline user token to connect to HCL OneTest™ Server from HCL OneTest™ Performance. For more information, refer to [Managing access to server](#).
- Created or joined a project in HCL OneTest™ Server.

- Configured the firewall so that HCL OneTest™ Server enables connection on the port 443.
- Upgraded legacy reports to the Web Analytics report format.



Note: You can right-click the report and select **Upgrade** to upgrade the legacy report to the Web Analytics report.

About this task

You can publish both performance and functional reports. You can set the publish parameters once in the **Preference** page so that you do not have to do it after every run or you can set the parameters every time for the specific result that you want to publish. Based on the parameters, the test result is published after the test run.

If you select **Prompt** from the drop-down list for the **Publish result after execution** option, after each test run, the **Publish Result** dialog box is displayed to publish reports instantly to HCL OneTest™ Server. You can modify the following options before publishing the results:

- If you want to publish reports to other than the default server added in the **Preferences** window, you can change the URL of HCL OneTest™ Server.



Note: If you change the server URL, you must enter an offline token to enable the publishing of test results and reports.

- By default, the **Result Name** field populates the name of the selected test result. You can provide a different name that you want to use.
- To identify specific test results, you can enter a tag or comment in the **Labels** field to associate it with the test result.
- You can change the project name if you want to publish reports to a different project.



Note: If there are no projects or if you are not a member of any project, you must create a project or become a member of a project on the server.

1. Click **Window > Preferences > Test > HCL OneTest™ Server**.
2. Specify the URL of the server and click **Test Connection**.
3. Enter the offline user token that you created on the server and click **OK**.
4. **Optional:** Click **Manage Offline Tokens** to view and remove the tokens that are associated with the desktop client, and click **Apply and Close**.

For example, if there is one instance of the desktop client for multiple testers to publish reports, each tester must remove the token created by other testers and add a new token.

5. Click the **Results** page from the navigation to apply settings for publishing reports.

6. Clear the **Use default** HCL OneTest Server **URL** check box if the URL of the server is different than that is specified at **Window > Preferences > Test > HCL OneTest Server**.

The format of the URL is `https://fully-qualified-domain-name:443`.

7. In **Publish result after execution** field, select when to publish test result.
In the initial stage when you are debugging a test, you might not want to publish the test result. Select one of the following options based on the requirement:
 - Select **Never** to never publish the test results to the server.
 - Select **Prompt** to prompt you to publish the test results after every test run.



Notes:

- A command line run will always publish the test results even if the workbench is set to **Prompt**.
 - After each test run, the **Publish Result** dialog box is displayed to publish reports instantly to HCL OneTest™ Server.
- Select **Always** to publish the test results after every test execution.
 8. In **Publish to project** field, select a project that you are a member of on the server. The test result is published to that project.

You cannot create a new project from the desktop client. If there are no projects or if you are not a member of any project, you must create a project or become a member of a project on the server.
 9. In **Reports**, select the reports that you want to publish to the server.
 10. Click **Apply and Close**.

Results

Test results are automatically published to the HCL OneTest™ Server, depending on the parameters that you have set.

What to do next

You can log in to HCL OneTest™ Server and analyze the test results. For more information, refer to [Test results and reports overview](#).

Related information

Publishing specific results to the server


Customizing reports

You can customize reports to specifically investigate a performance problem in more detail than what is provided in the default reports.

Creating custom reports

If the default reports do not address your needs, you can create your own reports.

About this task



Before you create a custom report, determine the ways in which the custom report will be different from or similar to the system-supplied reports. You can use a default report as a template, modify the counters, and save it with a different name. You can create a copy of pages or charts in a report that are based out of existing pages or charts. To copy the pages or charts, go to the Edit view and click the **Duplicate** icon. 

You can also create a report from scratch and add the required counters. Counters for test protocol are explained in the topics in the [Reports and counters on page 805](#) section.



Note: If a counter has a lot of data, the graph is not represented properly. To ensure that the graph is displayed properly, you must filter out some of the data.



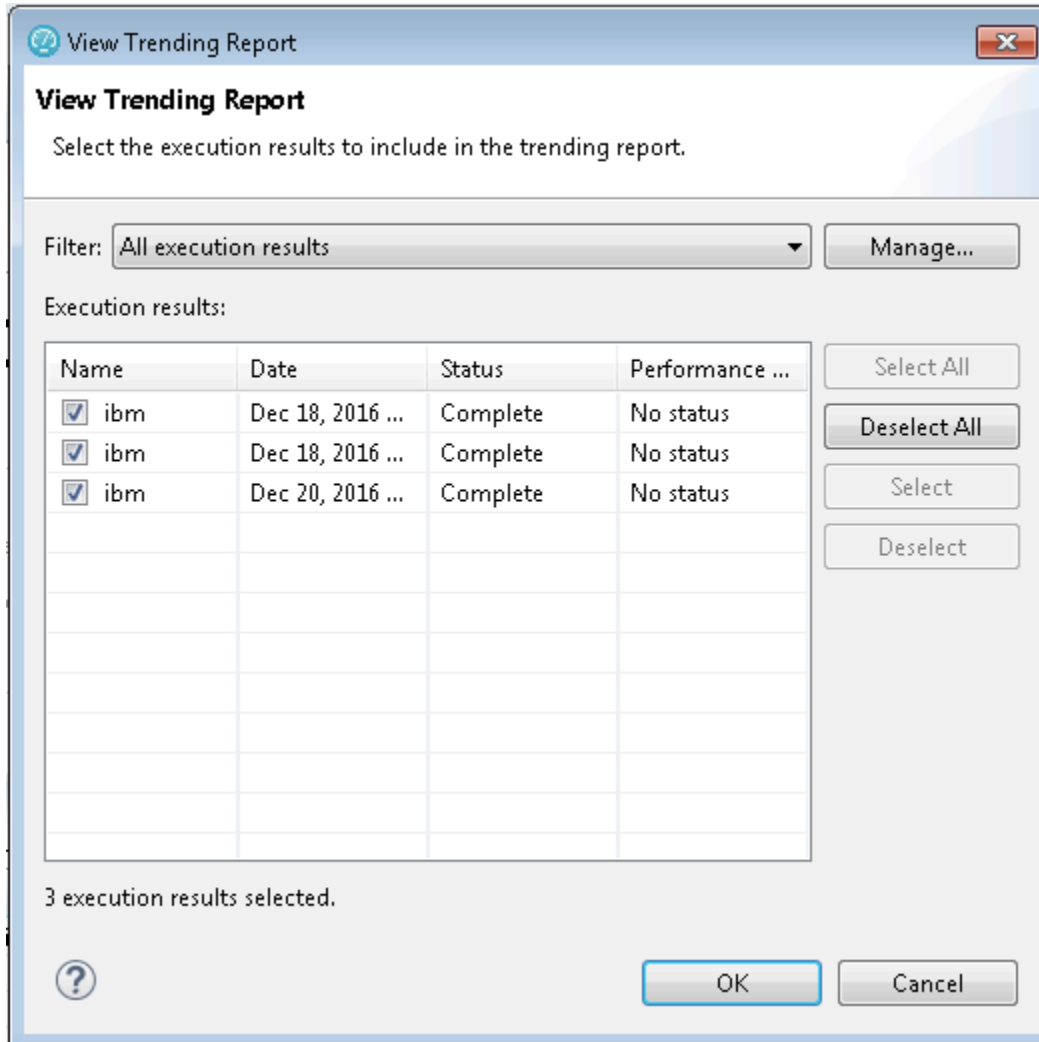
1. From the report, click **Menu**  and click **New**.
2. In **Create a new report** dialog box, specify a name and description about the new report and click **Create**.
3. To change the page title, click the default page title and specify a different name.
4. Click **Click to insert a row** and specify the number of columns to add the views.
Each view represents a bar chart, line chart, or pie chart.
5. Select a view. To add counters to the view, click **Settings** .
6. On the View Settings page, select a counter and add its details.
7. Click **Apply** and from the **Menu**, click **Save** to save the report.
8. To add more views to the report, repeat steps 4 through 7 again.

Viewing trending reports

To view the trend of response time for an application over a period of time, open the trend report for a run. In addition to the response time, you can view the trend for the loops, transactions, and performance requirements for the application.

About this task

The trend report can help you determine the response times of the application at different milestones. For instance, you can run the performance test for sprint or milestone builds and tag them. When generating the trend report, you can specify conditions such as results that are less than 60 days old and include 'milestone' tag.



You cannot save a trend report. So, if you close the report, you have to generate it again.

To view the trend report:

1. In the Test Navigator view, select the run for which to open the trend report.
2. Right-click the run and click **View Trend Report**.
3. To view the trend that is based on certain criteria, in **Filter**, select a filter criteria.
If there is no customized filtering criteria, create one by clicking **Manage** and then **Add**.
4. To save the criteria, click **Save**, specify a name to the filter, and click **OK**.
The results in the execution results table are filtered out according to the specified criteria.
5. Click **OK**.

Result

The trend report is generated.



Filtering results

By filtering the results that are displayed in a report, you can remove unnecessary data and focus on the data that is significant to you. If you save the changes, the report will contain these updates the next time that you open it.

About this task

If there is a lot of data in a bar or line chart that causes some of the data to not display by default, you can use the horizontal scroll bar on the X-axis to view the rest of the data. There is also a filter icon to filter the data based on the text on labels.

The table at the bottom of the chart also includes a lot of information. So, now, the tabular data is presented in pages. There is also the filter icon to filter the data in the table.

1. To filter the data in a report, open that report.
2. Click the **Menu** icon  and click **Edit**.
3. On the graph or table, click the **Settings** icon .
4. On the View Settings page, in the Filter section, select one of the following options and specify the values:

Option	Description
Filter by count	Display the specified number of items. For example, if you select this option and then type 15, the report will show the 15 items with the highest values (Show highest values) or the 15 items with the lowest values (Show lowest values).
Filter by value	Display items based on a comparison with the specified value. For example, if you select this option and then type 15, the report will show all of the items that are higher than 15 (Show counters above filter values) or lower than 15 (Show counters below filter value).
Filter by name	Display items that match the specified name. If you are filtering a table, the label is usually a page, and is listed in the left column. If you are filtering a graph, the label is a legend in the graph.

See the changes in the Preview section.

5. Click the **Apply** button.

If you are filtering a table by count or value, the **Primary counter for table filtering** lists the counters in the heading of the table that you are filtering. Select a primary counter. If a value is removed by the primary filter, all other data occupying the same row as the removed value is also removed, thus maintaining the integrity of the results.

What to do next

To remove a filter, follow the same steps but at step 4 modify or delete the value.

Adding additional counters on a separate page

You can add additional counters on a separate page without editing an existing report to investigate performance problems in detail.


Before you begin

You must have created and run the *test* or *schedule*.

About this task

Each report has its pre-defined counters that gather statistical information from the recorded test. If the counter information provided in the default reports do not address your needs, you can add additional counters on a separate page to diagnose the performance problems in detail.

You can click the **Menu** icon  and select **Hide All Instances** check box to hide the instance counters from


the counter tree. Similarly, you can click the **Menu** icon  and select the **Hide Percentile** check box to hide the percentile counters from the counter tree.

For example, when you want to add counters for the test report, you might want to hide percentile counters, which are specifically useful for the schedule runs.

1. Double-click the report in the **Test Navigator**.

2. Click the **Plus** icon .

A new page is displayed along with the execution report with a counter tree on the left pane.

3. Click the **Expand** icon  from the counter tree to view the available counters.

Alternatively, you can use the **Search** field to search the available counters by name.

4. Select the check box preceded with the counter name that you want to analyze.

For example, to analyze the standard deviation for the response time of all the pages:


a. Click **Pages > Response Time > All pages**.

b. Select the check box preceded with the counter name **StdDev**. The **StdDev** counter statistic is displayed as a graph.

Result

The selected counters are displayed in a graph in the right pane.

5. Rename the new tab for the new page by completing the following steps:

- a. Click **Up** arrow  of the new tab.
- b. Select **Rename**.
- c. Enter a new name for the new tab, and then click **OK**.

Result

The new page that you created is saved with the name you entered.

Results

You have added additional counters to be displayed on a separate page. The default chart as a line chart is displayed.

What to do next

You can change the graph view for the selected counters. See [Displaying counter data in tables or as graphs on page 783](#).

Related information

[Creating tests on page 180](#)

[Creating a VU Schedule on page 524](#)

[Running a local schedule or test on page 608](#)

Displaying counter data in tables or as graphs

You can display the statistics of counters in a table view or a graphical view so that you can analyze the counters information efficiently.

Before you begin

You must have completed the following tasks:

- Created and ran the *test* or *schedule*.
- Added counters on a separate page and renamed the page. See [Adding additional counters on a separate page on page 782](#).

1. Double-click the *test* or *schedule* report in the Test Navigator.


Result

The Performance Report is displayed.

2. Click the page you created and renamed.

Result



The counters that you added are displayed in a graph in the right pane. The default chart is the line chart.



3. Click the **Settings** icon  to modify the view for the counter.



Result

The available controls are displayed in the **Options** section.

4. Select one of the following options depending on the chart type that you want to view in the graph:
 - Lines chart
 - Bars chart
 - Pie chart
 - Table
5. Modify the following parameters for the view you select:

Options	Description	Lines	Bars	Pie	Table
Title	Displays a title you enter for the graph	✓	✓	✓	✓
Show legend	Enables you to select the position where you want the legend to be displayed in the graph.	✓	✓	✓	✗
Color items	Displays the selected option as a color item in the legend. The available options are as follows: <ul style="list-style-type: none"> ◦ Pages ◦ Transactions ◦ Resources ◦ Counters  Note: Depending on the test element that you add in your test or schedule, the preceding options are available.	✓	✗	✗	✗
Dash items	Displays the selected option as a dashed item in the legend. The available options are as follows: <ul style="list-style-type: none"> ◦ Pages ◦ Transactions ◦ Resources ◦ Counters  Note: Depending on the test element that you add in your test or schedule, the preceding options are available.	✓	✗	✗	✗
Source	Enables you to select the information of counters in the graph by its source type. The available options are as follows: <ul style="list-style-type: none"> ◦ All sources ◦ Load agents ◦ Resource monitoring 	✓	✓	✓	✗




Options	Description	Lines	Bars	Pie	Table
Adapt Y scale on selection	Select the check box to compute the minimum and maximum limits on the Y-axis.	✓	✓	✓	X
Palette	Displays the predefined color combination for the legend in the graph.	✓	✓	✓	X
	 Note: The requirements report always uses verdict colors.				
Adapt Y scale on zoomed data	Select the check box to adjust the Y scale according to zoomed data.	✓	X	X	X
Show time ranges	Select the check box to display the time range in the background of the graph.	✓	X	X	X
X axis unit	Enables you to select the options that you want to view as the X axis unit in the graph.	✓	X	X	X
Line smoothing	Clear the check box to apply corners in the graph.	✓	X	X	X
Time line visibility	Enables you to select the options that you want to view as the time line of the graph. The available options are as follows: <ul style="list-style-type: none"> ◦ None: The time line is not visible and you cannot create a new time range in the graph. ◦ Small (faster): The time line of the graph is in partial view. ◦ Full (draw curves slower): The time line of the graph is in full view.  Note: When you select Small (faster) or Full (draw curves slower) option, you can drag the time line to create a new time range on the graph.	✓	X	X	X
Individual scale for each	Sets the scale for each counter when multiple counters are displayed simultaneously. The available options are as follows: <ul style="list-style-type: none"> ◦ Unit ◦ Counter ◦ Result/Host/Geo 	✓	✓	X	X


Options	Description	Lines	Bars	Pie	Table
X Axis Main Items	<p>Displays the selected option in the X axis. The available options are as follows:</p> <ul style="list-style-type: none"> ◦ Not assigned ◦ Counters ◦ Time ranges/Results/Hosts/Geos 	X	✓	X	X
X Axis Secondary Items	Displays the selected option in the X axis as a secondary item.	X	✓	X	X
Stacked Items	<p>Displays the selected option as stack instead of separate bars in the graph.</p> <p>For example, when you select Counters for X-axis Main Items and Time ranges/Results/Hosts/Geos for X axis Secondary Items, the other option Not assigned is selected for Stacked Items.</p>	X	✓	X	X
Show values	Displays the value of counters in the graph.	X	✓	X	X
Orientation	Enables you to select orientation to view bar charts horizontally or vertically.	X	✓	X	X
Labels display policy	<p>Displays the labels in the graph. The available options are as follows:</p> <ul style="list-style-type: none"> ◦ Hidden: To hide the labels in the bar chart ◦ Adaptative: To be able to accommodate labels within the frame of the bar chart ◦ Fixed: To display the fixed labels. <p> Note: If you select Fixed, long labels might not be visible.</p>	X	✓	X	X
Show values axis	Clear the check box to remove the values from the Y-axis in the graph.	X	✓	X	X
Horizontal bar thickness	<p>Use an up-down control button to increment or decrement the thickness of the horizontal bar in the graph.</p> <p> Note: The maximum value you can set is 50.</p>	X	✓	X	X
Donut items	<p>Displays the selected option as a Pie chart in the graph:</p> <ul style="list-style-type: none"> ◦ Counters ◦ Time ranges/Results/Hosts/Geos 	X	X	✓	X

Options	Description	Lines	Bars	Pie	Table
Arc items	Displays the selected option as an Arc in the graph. For example, when you select Counters for Arc items, then the other option Time ranges/Results/Hosts/Geos is selected for Donut items and vice-versa.	X	X	✓	X
Pie style	Displays the graph either in the format of Pie or Donut.	X	X	✓	X
Row items	Displays the selected information in the row of a table. The available options are as follows: <ul style="list-style-type: none"> ◦ Not assigned ◦ Counters ◦ Time ranges/Results/Hosts/Geos 	X	X	X	✓
Column items	Displays the selected information in the column of the table.	X	X	X	✓
Group items	Displays the selected option information in the group. For example, when you select Counters for Row items and Time ranges/Results/Hosts/Geos for Column items, the other option Not assigned is selected for Group items.	X	X	X	✓
Draw mini bars	Clear the check box to remove the mini bar from the values.	X	X	X	✓

6. Select the **Counters** tab and perform the following steps to remove, move, or hide the counters in the graph and set the **Cumulated** value:

a. To remove, move, or hide the counters in the graph:

Action	Description
 Up-Down control icon	To move the counter on the graph.
 View icon	To hide the counter on the graph. Click the View icon again to display the hidden counter on the graph.
 Delete icon	To remove the counter from the graph.

b. Click the **Expand** icon  to change the **Cumulated** value for the selected counter.

- c. Select one of the following options based on the requirement when you want to display the cumulation values on the graph:

Option	Description
No	To display the value of the last interval on the current time range.
From beginning of time range	To display the cumulation of all values of the current time range.
From beginning of run	To display the cumulation of all values from the beginning of the run to the end of the current time range.

**Note:**

- For line charts, the default value is **No**.
- For the bar chart, pie chart, and tables, the default value is **From beginning of time range**.

7. Click the **Time Ranges** drop-down list and select the following options to display the data in the graph either for the entire run or number of users specified in the *schedule*:

Option	Description
Entire run	Select the check box to display the graph for the <i>test</i> for the entire run.
X Users	Select the check box to display the graph for the number of users specified in the <i>schedule</i> . Where <i>x</i> is the number of users specified while creating in the <i>schedule</i> .

Customizing the appearance of graphs in a report


To display the data in a table, bar chart, or line chart in a manner that caters to your test requirements, use the controls that are available in the View Options of a report.

1. In the Test Navigator, expand the project until you locate the run.
Each run begins with the name of the schedule or test, and ends with the date of the run in brackets.
2. Double-click the run.

Result

The default report opens.



3. Click the **Menu** icon and click the **Edit** icon.
4. Click the **Settings** icon  for the graph or table to modify.

5. The controls that are available in the View Options section depend on the graph type: bar chart, line chart, or table. For each graph type, only the applicable controls are displayed. You can adjust the following controls:

Option	Description
Adapt Y Scale	To compute minimum and maximum limit on the Y axis, select the check box. (all charts)
Title	Specify a title to the graph.
Show title	To hide the title, clear the check box.
X Axis Main items	Select the item to view on the X Axis.
Stacked Items	Select the item such as Pages or Time Ranges to view them in stack instead of separate bars.
Adapt Y scale on zoomed data	To adjust the Y scale according to zoomed data, select the check box. (line charts)
Show time ranges	To display the time range in the background of the chart, select the check box.
Line smoothing	To apply corners, clear the check box.
Orientation	To view bar charts horizontally or vertically, select an orientation.
Labels display policy	To hide the labels in a bar chart, select Hidden . To be able to accommodate labels within the frame of a bar chart, select Adaptative . If you select Fixed , long labels might not be visible.
Time line visibility	To view the time line of the chart in partial or full view, select Small or Full options. Drag the time line to create a new time range. If those options are specified, you can drag and create a new time range on the chart itself. If you select None , the time line is not visible and you cannot create a new time range on the chart.

6. After making the changes, click **Apply** and from the Menu click **Save**.

To apply the changes to other reports, you can export the report definition and import it back. See [Exporting report metadata on page 795](#).

Changing the report displayed during a run

Use this page to select the default report that opens during a run. Typically, you select **Determine default report based on protocols in test**, which determines the protocols that you are testing and automatically opens the appropriate protocol-specific reports.







1. Open the Default Report Preferences page. Click **Window > Preferences > Test > Performance Test Reports > Default Report**.
2. In the Default Report window, select **Determine default report based on protocols in test** or a specific default report to display a customized report or if the default reports do not meet your needs. Note, however, that you will have to change this setting when you record other protocols.
3. Click **Apply**, and then click **OK**.

Modifying counters in a graph

To gather additional information for diagnosing performance problems, you can modify the counters that are displayed in a graph.

About this task

Counters are specific in-built queries that gather statistical information from the recorded test. The information can be the number of page hits, response time, and user load. By default, each report has pre-defined counters. You can add or remove the counters from the graphs in the report.

1. From the Test Navigator, double-click the report to modify the counters.
2. To add or remove a counter from a report, click the **Menu** icon .
3. Click the **Edit** icon .
4. To modify counters on a specific graph, click **Settings** icon .
5. On the **View Settings** page, select the **Counters** tab and perform the following sub-steps to add, remove, or move the counters in a graph:
 - a. To add a counter, click the **Plus** button  and select the counters from the drop-down list.
 - b. To remove the selected counter, click the **Remove** button .
 - c. To move a counter, use the **up-down** control buttons .

The Preview section displays the result of the actions.

6. **Optional:** For a selected counter, you can change the component of the counter. Based on the counter selection, the **Component** field shows the options available for that counter.
7. **Optional:** You can change the **Cumulated** value for the selected counter if you want to show the cumulation values on a graph. Select one of the following options based on the requirement:
Choose from:

- Select **No** to display the value of the last interval on the current time range.
- Select **From beginning of time range** to display the cumulation of all values of the current time range.
- Select **From beginning of run** to display the cumulation of all values from the beginning of the run to the end of the current time range.


**Notes:**

- For line charts, the default value is **No**.
- For bar chart, pie chart, and tables, the default value is **From beginning of time range**.
- The fields **Label**, **Path**, and **Unit** are non-editable.

8. Click **Apply**.

9. Click **Save** from the menu.

10. **Optional:** To create another report with these changes, click **Save As**.

11. To exit the edit mode, click the **Edit** icon .

Results

You have updated the counter information for the specific report.

Correcting time offset

Response time breakdown and resource monitoring data is time stamped using the system clock of the host computer. If there are differences between the system clocks of the host computers that you include in a test, then response time breakdown and resource monitoring data are skewed in reports. The best practice is to synchronize the system clocks on all computers that you include in a test. When this is not possible, you can correct the time offset of each host computer after a test run. Typically, correct the time offset on all computers to match the system clock of the workbench computer.

After you run tests with resource monitoring or response time breakdown enabled, follow these steps to correct the time offset:

1. In the **Test Runs** view, right-click the host where you want to correct the time offset; then click **Correct Time Offset**.
2. Select a **Shift Direction** of positive or negative. A positive shift moves the response time breakdown and resource monitoring data on the selected host to the right. A negative shift moves the response time breakdown and resource monitoring data on the selected host to the left.
3. Type the hours, minutes, or seconds of the time offset you want to use, and click **OK**.

Results

The response time breakdown and resource monitoring data on the selected host displays with a corrected time offset.

Export test results

You can export the test result in different formats to share it with different stakeholders.

Creating an executive summary from the workbench

To create a printable report that summarizes the findings of the performance test run on a single view, create an executive summary. You can export the data of the test run as an executive summary from a single report or from multiple reports such as Performance Report, Mobile and Web UI Statistical Report, Transaction Report, and Loop Report. You can then open the summary in a word-processing program to further format and annotate the data.

About this task

You export the executive summary to a local or a shared directory. You can export a test run from the Web Analytics report, from the test workbench, and from the command line.

When you use the workbench approach to create an executive summary, you can choose to create the summary for multiple runs and multiple report types at the same time. When you use the Web Analytics reports or the command line, you create executive summary for a particular run and a report at a time.

To create an executive summary from the workbench:

1. Click **File > Export > Test > Executive Summary**. You can also right-click the runs to create executive summaries for from the Test Navigator view and click **Export > Test > Executive Summary**. Each run would have one executive summary.
2. In **Export Directory**, specify the folder path to save the executive summary and click **Next**.
3. Select the runs to create the executive summary for. To create an executive summary for comparing two runs, select the **Generate a compare report** check box and select the main run to compare the report with and click **Next**.
4. Select a report to export and click **Finish**.

What to do next

A folder with the name of the run is created on the specified folder. To view the executive summary, open the `index.html` file.

Creating an executive summary from the Web Analytics report



To create a printable report that summarizes the findings of the performance test run on a single view, create an executive summary. You can choose to view the executive summary on a web browser or save it on a computer.

About this task

To generate an executive summary for a particular report such as Transaction report or Performance report, open that report and then follow the steps in this topic. To generate an executive summary for multiple reports or test runs at the same time, see [Creating Executive Summary from Workbench on page 792](#).

To create an executive summary from the Web Analytics report:

1. Open the test run to create executive summary for. The test run opens in a web browser.
2. From the dropdown, open the report for which to create executive summary.

3. Click the **Menu** icon , click the **Share** icon , and click **Executive Summary**.
4. To view the executive summary of the report in another browser tab, click **View on another tab or page of the browser**. To save the executive summary, click **Save as an HTML file on the local computer**.
5. Click **Generate**.

Exporting reports to HTML format



When you export a test run and share it, people can analyze test data without using the test workbench. You can also email the test run or post it on a web server. The exported run can be displayed and printed from any browser. A test run contains multiple reports. You can choose to export any or all of the reports.

About this task

You can export a single run to a local directory or multiple runs in the compare mode to a directory. In addition to exporting a test run from Web Analytics, you can export it from the test workbench itself and from command line.

To export from the workbench, select a single run or multiple runs and click **Export > Test > Performance Test Run Statistics as HTML application**. To generate a single report comparing multiple runs, in the Export wizard, select the **Generate a compare report** check box and select a base run from the dropdown. To generate one report for each run, do not select the check box.

To export from Web Analytics:

1. Open the test run to export.
The test run opens in an external or internal web browser.
2. Click the **Menu** icon , click the **Share** icon , and click **Export Session to HTML**.
3. Select the type of report to export and click **Export**.
4. When you export from the workbench, specify a path to the folder to save the exported report.
Your current project is the default save location. You can create a folder outside of the project to store exported reports.

When you export from an external browser, the report is compressed and saved to the default download location of the browser.

What to do next

You can now share the test run with others. You can also export the test run from command line.

 Related information

[Running a test or schedule from a command line on page 627](#)

Exporting results to a CSV file

To further analyze test results, you can export all statistics or specific statistics captured during a run to a CSV file.

About this task

You can export a single run to a local directory or multiple runs in the compare mode to a directory. You can export the runs from Web Analytics report, workbench, and command line. To export from the workbench, select a single run or multiple runs and click **Export > Test > Performance Test Run Statistics as CSV File**. To export data of specific time ranges, on a subsequent page select a time range.

To export the run from command line, see the parameters in the [Running a test or schedule from a command line on page 627](#) topic.

1. Open the test run to export.

2. Click the **Menu** icon  , click the **Share** icon  , and click **Export Session to CSV**.

3. Select the encoding system for the export.
4. Complete either one of the following steps:

Choose from:

- To export only the last value of each counter from the results or to export data of specific time ranges, select **Simple**.



Note: When you export data of specific time ranges, for example, 5 Users or 15 Users, a separate column is created in the CSV file for each time range.

- To create multiple CSV files if the number of columns exceed the specified value, select the **Split output if column exceeds** check box and specify a value.
- To export all of the data for the run, select **Full**.

To include description about the name of the run, node name, and time range for each counter, select the **Include per instance counters**.

- To export data of each location (agent) in a separate section in the CSV file, select the **Export each agent separately**

To export data of each location (agent) to separate CSV files, select the **One file per agent** check box.

5. Click **Export**. If you export from the workbench, the report is saved in the specified folder. If you export from an external browser, the report is downloaded in a compressed format to the default download location of the browser.

What to do next

You can now analyze and share the report with people who are not using the workbench.

Related information

[Exporting reports to HTML format on page 793](#)

Sharing URL of test run

When you share the URL of the test run with other people, they can view and analyze the test results on a browser on their computer if the test workbench is running on your computer at that time.

To share the URL of the test run:

1. Open the test run to share.

2. Click the **Menu** icon  and click the **Share** icon  and select **Share Execution Result URL**.

A unique URL is created for the test run.

3. Copy the URL and click **Close**.

What to do next

You can now share the URL of the test run with anybody.

Exporting report metadata

To share report metadata with another test workbench user, export the report definition. Use this option to share customized report formats with other users. The recipient imports the metadata with Eclipse's **Import** option and views the report from the Test Navigator or in the list of reports in the web report.

To export report metadata:

1. Click **File > Export**.
2. In the **Export** window, expand the **Test** folder, select **Report Definitions**, and click **Next**.
3. In **Save to File**, select the file that will contain the report. This file is created if it does not exist.
4. In **Select Report**, select the report to export, and then click **Finish**.

The file is saved in the `.report` format.

What to do next

To apply another report definition to your reports, import that report metadata by clicking **File > Import > Report Definition**, and browse to the `.report` file.

Viewing response time breakdown

You can do detailed analysis of the response time to find bottlenecks in the HTTP traffic of the application.

Viewing page element responses

You can view the response times for individual page elements in reports, to determine which elements are the slowest.

About this task

Page element response times do not include client delay or connection time. Because page elements can be returned in parallel from the server under test, the page response time is not necessarily the sum of the page element response times.

1. Open the web analytics reports.
2. On the Page Performance report, click a page (represented by a bar) and click **Page Element Responses**.
The Page Element Responses report displays response time for all of the elements of the page.
3. To return to the original report, click the Page Performance link in the breadcrumb.

Viewing page response time contributions

You can view the response time contributions for individual pages to determine how much time was actually taken by the page to load and the time taken for the connection to go through and the delay on the client side of each page.

Before you begin

Because page elements can be returned in parallel from the server under test, the page response time is not necessarily the sum of the page element response times. Client delay and connection time also contribute to page response time. The page response time can be greater than the sum of the page element response times if, for example, a lengthy connection time adds a delay. Connection time includes the time required for Domain Name Services (DNS) lookups. Conversely, the page response time can be less than the sum of the page element response times if multiple page elements are returned in parallel.

1. Open the web analytics report.
2. On the Page Performance report, click a page (represented by a bar) and click **Page Response Time Contributions**.
The Page Response Time Contributions report shows the average response time taken for Connection Time, Client Delay Time, and Page Element Response Time.
3. To return to the original report, click the Page Performance link in the breadcrumb.

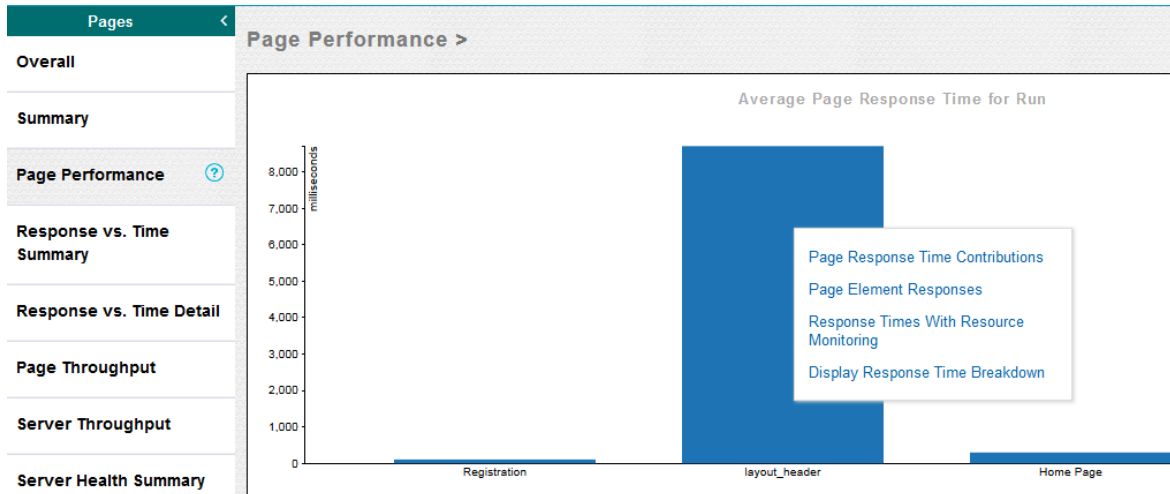
Viewing response time breakdown data

In addition to viewing page element and page response time, you can do further analysis to view the response time data for each method.

Before you begin

You must have instrumented the servers. See [Instrumenting local servers on page 108](#).

1. Open the web analytics reports.
2. On the Page Performance report, click a page (represented by a bar) and click **Display Response Time Breakdown**.



You can also directly access the response time breakdown data from the Page Element Responses report.

3. In the workbench, the **Page Element Selection** dialog is open.
4. Select a page element to view response time breakdown for and click **Finish**.

Logs overview

HCL OneTest™ Performance uses logs to store different types of information, which you can use to determine the reason for a test failure.

HCL OneTest™ Performance has the following logs:

Test logs

The test log contains a historical record of events that occurred during a test run or a schedule run, as well as the status of each verification point. The test log sets a verdict for each run as follows:

- **Pass** indicates that all verification points matched or received the expected response and all the test steps successfully completed. For example, a response code verification point is set to `PASS` when the recorded response code is received during playback. If your test does not contain verification points, `PASS` means that all primary requests in the test were successful.
- **Fail** indicates that at least one verification point did not match the expected response or that the expected response was not received or a Web UI step did not run successfully.
- **Error** indicates one of the following results: a primary request was not successfully sent to the server, no response was received from the server for a primary request, or the primary request response was incomplete or could not be parsed.
- The verdict is set to **Inconclusive** only if you provide custom code that defines a verdict of **Inconclusive**.

The verdict is rolled up from the child elements to the test level. For example, if a user group contains 25 virtual users, and five virtual users have failed verdicts, that user group has only one failed verdict, not five.

The test log file is stored in binary format with a `.executiond1r` file name extension in the project directory of your workspace. You can also view the test log in the user interface.

For more information about viewing test logs, see [Viewing test logs on page 605](#).

Problem determination logs

You can set the level of information that is saved in the problem determination log during a run. By default, only warnings and severe errors are logged. Typically, you change this log level only when requested to do so by the Support person.

The problem determination logs contain internal information about the playback engine. These logs are particularly useful for debugging problems such as Kerberos authentication, SSL negotiation, and resource constraints on an agent. The log files are named `CommonBaseEvents00.log` and are located in the deployment directory. For example, if you play back a schedule on an agent and set `C:\Agent` as the deployment directory, the problem determination log files are in a directory similar to `C:\Agent\deployment_root\\A1E14699848784C00D2DEB73763646462\CommonBaseEvents00.log`. If a large amount of log information is generated, multiple `CommonBaseEvents` files are created.

For more information about setting problem determination level, see [Setting the problem determination level on page 606](#).

Agent logs

Look in `%TEMP%` directory for the `majordomo.log` file. This file contains information about the attempts to contact the workbench including information about any failures and the reason for the failures.

On the Microsoft™ Windows operating system, the `%TEMP%` directory is typically at `%USERPROFILE%\AppData\Local\Temp`.

If the majordomo service is configured to log in as Local System Account, then the `%TEMP%` directory is at `%SystemRoot%\TEMP`, typically `C:\Windows\TEMP`.

Error logs




If an error message is displayed when you run tests, try looking up the error message in the *Performance testing error messages* section of the online help. Only the most common error messages are listed. If no error message is displayed when you encounter a problem, open the error log by clicking **Window > Show View > Error Log**. If the workbench shuts down while running tests, restart the workbench and examine the error log. By default, warning and error messages are logged. You can increase the default logging level by clicking **Window > Preferences > Logging**. The log file is stored in the `.metadata` directory of your workspace. To avoid excessive logging, the Logging Level should be adjusted for individual Logger Names in the Loggers tab. For example, to get more information about a problem connecting with IBM® Rational® Quality Manager, increase the Logging Level for `com.ibm.rational.test.it.rqm.adapter` Logger Name. For the licensing issue, adjust the level for `com.ibm.rational.test.it.licensing` Logger Name. When you no longer need the extra logging, use the **Restore Default** button in the Logging Preferences to reset all the levels to their recommended defaults.

Viewing test logs

To see a record of all the events that occurred during a test run or a schedule run, as well as the status of each verification point, open the test log for that run. You can also compare an event from the test log with the request or response in the test to view the differences between the recording and the playback of the test.

About this task

The test log file is stored in binary format with a `.executiond1r` file name extension in the project directory of your workspace. You can also view the test log in the user interface.

1. In the Test Navigator view, right-click the executed test; then click **Display Test Log**.
2. On the **Overview** tab, view the verdict summary for the executed test. To see the potential data correlation errors in a separate view, click **Display Potential Data Correlation Errors**.
3. On the **Events** tab, view the errors, failures, and passes for each event in the test.
 - To navigate to the verdict type, click the **Select the verdict type**  icon.
 - To compare an event or request in the test log with the response or request of the test, right-click an event and click **Compare With Test > Request**, **Compare With Test > Response Data**, or **Compare With Test > Response Headers**. The compare editor highlights the differences.
4. On the **Data Correlation** tab, see all the references and substitutions that occurred during a test execution, as well as the data correlation errors. By default, you view both references and substituters. To view only substituters, click the **Show References**  icon. To view the correlation data for each virtual user that was executed, click the **Merge Users**  icon. This icon is enabled only for a schedule. In the **Data Correlation** section, when you click an event, you can see the correlation data in either the Content View or the Table View.

What to do next

From the test log, you can submit, search, and open defects in a defect tracking system. For details on configuring the test log preferences and working with defects, see [Associating defects with a test log](#).

Viewing errors while running tests


To view errors and other events while a test is running, use the Execution Event Console view. If problems occur in a test run, you can examine the Execution Event Console view to determine whether to stop or continue the test.

1. Open the **Execution Event Console** view by clicking **Window > Show View > Execution Event Console**.
2. In the **Execution Event Console** view, click the **Filters** toolbar button in the upper, right corner.

Result

The **Event Console Configuration** window opens.

3. Select the types of messages and verdicts to display in the event console, and then click **OK**.
You can also limit the number of events that are displayed per user and per run, and you can limit events to specific user groups or agent computers (locations). To configure other settings for the event console, click **Settings**.
4. Run performance tests as you normally do.

5. While a test is running, double-click an event in the **Execution Event Console** view to open the **Event Details** window.
 - a. To change the order in which events are listed, click the **View Menu** toolbar button, and then select **Group By**.
6. To load events from the test log, ensure that the Test Log view is open and in the Console view, click the **Load Test Log Events** icon .

Viewing reports after a run

Reports are generated and displayed automatically after a run. Each test result begins with the name of the schedule or test, and ends with the timestamp of the run in brackets.

About this task

In version 8.5.1 or later, for a service test report, you can choose not to generate a report automatically after a run by clicking **Window > Preferences > Test > Performance Test Reports > Service Test Reports** and clearing the **Functional Test Report generation after Test execution**.

1. In the Test Navigator, expand the project until you locate the run.
2. Do either of the following:
 - To view the default report, double-click the run. To change the default report, Open the Default Report Preferences page. Click **Window > Preferences > Test > Performance Test Reports > Default Report**
 - To view another report, right-click the test run, click **Display Report**, and then select the report to display.



Note: You can also view reports remotely from a web browser. For information about viewing reports remotely, see [Accessing reports remotely on page 800](#).

Accessing reports remotely

Before executing a schedule or test, you can enable an option so that you can access reports remotely from a web browser. When you make changes to a report, the changes are saved to the workspace where the workbench is running.

1. On the HCL OneTest™ Performance workbench, click **Windows > Preferences > Test > Performance Test Reports > Web Reports**.
2. To enable remote access to reports, select the **Allow remote access from a web browser** check box.
3. To enable the remote control of schedule execution tasks, select the **Allow control of schedule execution from the web browser** check box.
4. **Optional:** By default, the non-secure port number for web reports is 8080. If this port number is used by another service, you can type another port number.
5. **Optional:** To provide security for web reports, select the **Security is required to access reports** check box.

a. By default, the secure port number for web reports is 8443. If this port number is used by another service, you can type another port number.

b. Select the **User authentication is required to access reports** check box and specify the login credentials.

You must use the same login credentials to access reports remotely.

6. Click **OK**.

7. To access web analytics reports remotely, open a web browser and type

`http://host_name:8080/analytics/web/index.html`. To access a secured report, type

`https://host_name:8080/analytics/web/index.html` and specify the login credentials if you have set it.

To access old web reports remotely, on another computer, open a web browser and type

`http://host_name:8080/RPTWeb/WebAnalytics/`. To access a secured report, type

`https://host_name:8080/RPTWeb/WebAnalytics/` and specify the login credentials if you have set it.

The host name is the HCL OneTest™ Performance workbench computer name and the port number is as specified in **Windows > Preferences > Test > Performance Test Reports > Web Reports**.

Exporting test logs

To process data from a performance test in another application or to use search tools to locate text in a test log, export the test log to a text file.

1. In the Test Navigator, right-click the run, and select **Export Test Log**.

a. **Optional:** To export only a portion of the test log, open the test log by right-clicking the test run and then selecting **Display Test Log**. Right-click the elements to export, and then select **Export Log Element**.

Result

The **Export Test Log** window opens.

2. In the Export Test Log window, specify a location for saving the file, and then select options as follows:

Option	Description
Export format	Select default encoding or Unicode encoding.
Include event time stamps	Select to include event time stamps.
Include detailed protocol data	Select to include detailed protocol data. This option is available only for HTTP test runs.
Include response content	Select to include response content. This option is available only for HTTP test runs.
Include known binary data	Select to export binary data. This option is available only for HTTP test runs.

3. Click **Finish**.

Result


The test log is exported to a text file.

Exporting event log

To view all the events that occurred during the run of a test from another file, you can export this data from the Event Log panel, to an XML, CSV, or text file.

Before you begin

You must run a test to view data in the Event Log panel.


1. On the Event Log panel toolbar click the **View Menu** arrow icon  and select **Export Event Log**.
2. In the Save dialog box, specify the location and format in which you want to save the events.

Exporting event console output

To view errors and other events of a test run from another file, you can export this data from the Execution Event Console view to an XML, CSV, or text file.

Before you begin

- Ensure that the Execution Event Console view is open by clicking **Window > Show View > Execution Event Console**.
- Ensure that the test is run and the Execution Event Console view contains data.

1. From the Execution Event Console view toolbar, click the **View Menu** arrow icon  and select **Export**.
2. In the Save dialog box, specify the location and format in which you want to save the events.

Viewing adjustments to page response times

To see the adjustments to page response times that are measured during a test run or a schedule run, open the test log for that run.

About this task

HTTP page response times are adjusted for increased accuracy by using advanced techniques to exclude processing time that is not related to the server under test. Other measurements, such as page element response times, are not affected.

1. In the Test Navigator, right-click the run, and then click **Display Test Log**.

Result

The test log opens, displaying the **Overview** page.

2. Click the **Events** tab.
3. Expand the elements in the **Events** hierarchy, and navigate to the page that contains the adjustment.
4. The last element for an HTTP page is the page stop event. Select the page stop event.

Result

Under **Extended Properties**, the **rtaa** property is the adjustment to the page response time, in milliseconds, made to exclude processing time that is unrelated to the server under test.

Disabling adjustments to page response times

Starting with HCL OneTest™ Performance Version 8.1.1, HTTP page response times are adjusted for increased accuracy by using advanced techniques to exclude processing overhead not related to the server under test. To configure the product to use the previous method for calculating page response times, set the RPT_VMARGS property rptPre811PageResponseTimes.

About this task

Typically, you do not disable adjustments to page response times unless you want to compare results gathered from versions of the product prior to 8.1.1 with results gathered from versions of the product starting with 8.1.1.

1. In the Test Navigator, right-click the location on which to disable page response time adjustment.
2. Click **Open**.
3. Under **Property Groups**, click the **General Properties** link, and then click **Add**.
4. In the **New Property** window:
 - a. In the **Property Name** field, type `RPT_VMARGS`.
 - b. In the **Operator** field, confirm that the operator is `=`.
 - c. In the **Property Value** field, type `-DrptPre811PageResponseTimes`, and then click **OK**.

Viewing resource monitoring data

You can analyze the performance of the computer resources, application server, or database servers by viewing the resource monitoring data in web analytics reports.

Adding resource counters to reports

To view performance data of resource counters that are not shown in the report by default, you can add the resource counters.

Before you begin

You must have enabled capturing of resource monitoring data in the schedule. See [Enable resource monitoring on page 576](#).

1. Open the Performance Report, and from the list of Pages, click **Resources**.

2. Click the **Menu** icon  and then click the **Edit** icon .
3. To update the graph, click the **Settings** icon  on the graph. There would be multiple Settings icons for a report.
4. On the View Settings page, click **Counters**, and click the **Add** icon .

- From the dropdown, select **Resource Monitoring** and then from the **Component** dropdown, select a unit of measurement such as Min, Max, or Average.
- Click **Apply**, click **Save**, and then click the **Edit** icon.

Results

The changes are reflected in the resource monitoring graph.




Filtering resource counters

To view the performance data that is more important for you, filter the resource monitoring counters. You can filter resource counters the same way you filter other results displayed in reports.

Before you begin

To view the Resources page in the Performance Report, you must have enabled capturing of resource monitoring data in the schedule. See [Enable resource monitoring on page 576](#).

- Open the Performance Report, and from the list of Pages, click **Resources**.

- Click the **Menu** icon  and then click the **Edit** icon .
- To update the graph, click the **Settings** icon  on the graph. There would be multiple Settings icons for a report.
- On the View Settings page, click **Filters**.
- From the dropdown, click one of the following filter types:

Option	Description
Filter by count	Display the specified number of items. For example, if you select this option and then type 15, the report will show the 15 items with the highest values (Show highest values) or the 15 items with the lowest values (Show lowest values).
Filter by value	Display items based on a comparison with the specified value. For example, if you select this option and then type 15, the report will show all of the items that are higher than 15 (Show counters above filter value) or lower than 15 (Show counters below filter value).
Filter by label	Display items that match the specified label. If you are filtering a table, the label is typically a page, and is listed in the left column. If you are filtering a graph, the label is a legend in the graph.

- From the **Primary counter for table filtering** dropdown, click a primary resource counter.

Typically, this is the **Resource Monitoring** counter.

7. Click **Apply**, click **Save**, and then click the **Edit** icon.

Results

The changes are reflected in the resource monitoring graph.

Reports and counters

See the description of the counters for all of the performance reports.

Requirements report

From version 9.2.0.1, the Performance Requirements report is renamed to Requirements report. The Requirements report validates the performance and functional requirements that you set in a test or in a schedule.

Validation is accomplished by comparing the data in the run to the requirements that you set on the data.

Status Summary page

With the Status Summary page, you can quickly analyze the requirements that are defined in a test result. The page contains two tables.

- The first table provides the overall status of the run and the percentage of performance requirements that passed.
- The second table lists all of the standard requirements that you defined.

Each requirement has a row in the table that explains the target of the performance requirement (for example, an HTTP page), specification, and status.

Overall Summary page

The Overall Summary page provides a high-level, graphical analysis of requirements, both standard and supplemental. The Overall Summary page contains two tables and two pie charts.

The first table presents the pass/fail status and percentage pass data for requirements and supplemental requirements.

Both requirements and supplemental requirements are represented in the center of the report by a pie chart. In both cases, failures are shown in red and passes are shown in green.

The final graphic on this page presents numeric details for each requirement type and for the union of both types.

Details page

The Details page provides a detailed analysis of standard requirements. It contains two tables and a bar chart.

As with the Status Summary page, the overall status of the run and percent passed value for the run are presented in a table at the top of the page. The bar chart at the center of the report displays the margin of each performance requirement. The *margin* is a percentage value that allows improvement or regression analysis with regards to

a requirement. When a performance requirement is in the passed state, it will have a margin ≥ 0 . The margin is calculated as follows:

$$\% \text{ Margin} = \text{abs}(\text{specification} - \text{observed}) / \text{specification}$$

When a performance requirement is in the "failed" state, it will have a value ≤ 0 . The margin is calculated as follows:

$$\% \text{ Margin} = -\text{abs}(\text{specification} - \text{observed}) / \text{specification}$$

Margins are useful in comparison mode because you can detect improvement or regressions before they are significant enough to change the requirement from pass to fail. The table at the bottom of the Details page contains a representative row for each performance requirement. Each requirement is presented with an explanation of its target, observed result, specification, margin, and status. Passed statuses are shown in green and failed statuses are shown in red.

Supplemental Details page

The Supplemental Details page shows a detailed analysis of supplemental requirements. It contains the same data as the Details page, except that it pertains to supplemental requirements.

Synchronization Point report

This report provides information about the synchronization points in test runs and lets you manually release a user from a synchronization point. To release a user, right-click in the report and select **Manage Synchronization Points**.

The Synchronization Point report contains the following information for each synchronization point in the run:

- The name of the synchronization point.
- Time-out Value. The time after which the synchronization point is automatically released. A value of 0 means that the synchronization point will be released after the arrival of the first user.
- Users Late. The number of virtual users that have arrived "late"; that is, after the synchronization point was released.
- Users Arrived. The number of virtual users that have arrived at the synchronization point.
- Current State. The state of the synchronization point. The state can be:
 - Inactive. No users have arrived or all locations are inactive.
 - Active. At least one user has arrived.
 - Released. All users have arrived, been release manually, or have timed out.
- Users Expected. The number of virtual users still expected to arrive.
- Run Duration. The time between the first user's arrival and the first user's release. The count begins as soon as any location reports an Active state, and stops as soon as any location reports a released state. The accuracy of the time might be adversely affected by a large statistics interval.

Loops report

This report summarizes the functionality of loops in a test.

Loop Invocation Details

This tab displays the number of times the loop was invoked in a test and loops that were invoked but did not complete.

Loop Iteration Details

This tab displays how many iterations does each loop run and how many iterations in a loop were successful.

Loop Iteration Health

This tab displays the health of each iteration of a loop. You can also view the error that cause the failure of an iteration of a loop.

Agents Health Report

With the Agents Health Report, you can view the usage data of CPU, Memory, Threads, and JVM Heap for the agent machines involved in the run. The report shows usage data for the Agent Host and Agent Process on the agent machines. By default, the **Enable Agent Health Report** check box at **Window > Preferences > Test > Test Execution** is selected. Also, by default, the report opens in the Compare mode. You cannot disable the compare mode.

Report

CPU

The CPU usage graph shows the usage data for the Agent Host and Agent processes. Note that the SAP, SOA, and HCL OneTest™ API test extensions runs as external processes on the agent locations and therefore are not counted in the Agent process but are counted in the Agent Host with all others system processes (including the agent).

Memory

The Memory usage graph shows the memory consumed by the Agent processes in terms of average and percentage and by the Agent Host. The average memory consumption of the agent is usually very little. Therefore, the graph also displays the memory consumption in terms of percentage. If the CPU and Memory consumption for the agents is high, you can decide to add more locations to the execution of the performance tests.

JVM Heap

The JVM Heap graph shows the percentage of memory used compared to the total memory allocated and the total JVM memory allocated. If the memory used is high, you can decide to increase the JVM Heap size by adding the `-Xmx` property to the agent location.

Threads

Threads Usage graph shows the percentage of threads used for the currently running tasks compared to the maximum of threads that the tool can create. It also shows the number of tasks waiting for the threads to be made available to execute them.

Rate Runner report

You can use the Rate Runner report to view how all the Rate Runner groups in the Rate Schedule have performed during the run.

Rate Runners

The **Rate Runners** page displays the graph with X-axis showing the time and Y-axis showing the rate or workload generated per second. The top section of the report displays the following fields:

State: Status of the run for rate generators. One of the following statuses is displayed:

- Not started
- Arriving: At least one user has arrived at the synchronization point.
- Active: At least one iteration of workload.
- Inactive: No users or clients are running.

Users or **Clients:** Number of users or clients in the Arriving or Active state.

Target Rate: Number of iterations specified in Rate Generator or Rate Schedule. The report shows the value in seconds. For example, if the iteration rate is 2 every minute, the Target Rate would show 0.032 per second.

Workloads Started - First time: Time taken for the first workload to start after the test run. The time taken includes the time to create the users or clients, begin the workload, meet at the rate synchronization point, run any iteration delay, and then start the workload.

Workloads Started - Elapsed time: Time taken for the first workload to start and the last workload to end.

Workloads Started: Number of workloads already started at any given point of time during the run. For example, for a target rate of 4 every minute, at the end of one minute, the number of workloads started should be 4. However, it should increment throughout the run based on the target rate. So, at a target rate of 4 every minute and a total duration of 10 minutes, the number of workloads at the end of the run should show `Workloads Started = 40` and `Workloads Completed = 40`.

Workloads Started - Rate: Actual rate achieved for the run. Compare it with the Target Rate to determine the performance of the application.

Percent Target: Comparison of the Target Rate with the Actual Rate (Workloads Started - Rate) to indicate by percentage how close the workload is to reach the specified target rate.

Percent Late - Percent: The percent of workloads in a user group or Rate Runner group that did not begin to run at the specified time.

Health

Use this page to view the number of workloads that did not meet the target rate.

Throughput

Use this page to view the number of workloads that passed successfully.

Transaction report

This report summarizes the success of transactions in the run, plots on a graph the response trend of each transaction during a specific sample interval, and shows the transaction throughput.

This report plots the sample intervals within a run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

Overall page

The *Overall Transaction Rate* graph shows the average elapsed time for all transactions during a specific sample interval. *Elapsed time* is the actual time spent within the transaction container. If you have staged loads in the schedule, this graph also delineates the stages with time range markers, in various colors, at the top.

The table under the graph lists the following information:

- The average elapsed time for all transactions in the entire run.
- The standard deviation of the elapsed time. The standard deviation tells you how tightly the data is grouped about the mean. For example, assume that System A and System B both have an average elapsed time of 12 ms. However, this does not mean that the elapsed times are similar. System A might have elapsed times of 11, 12, 13, and 12 ms. System B might have elapsed times of 1, 20, 25, and 2 ms. Although the mean time is the same, the standard deviation of System B is greater and the elapsed time is more varied.
- The longest transaction in the entire run.
- The shortest transaction in the entire run.
- The average net server time for all transactions in the entire run.
- The standard deviation of the net server time.
- The longest net server time in the entire run.
- The shortest net server time in the entire run.

Elapsed Time vs. Time

The *Elapsed Time vs. Time* graph shows the average response of each transaction during a specific sample interval. Each transaction is represented by a separate line. If you have staged loads in the schedule, this graph also delineates the stages with time range markers, in various colors, at the top.

The table under the graph lists the following information for each transaction:

- The minimum elapsed time for the entire run.
- The average elapsed time for the entire run. This is similar to the graph above, but the information in the table is the average for the entire run rather than the average per sample interval.
- The standard deviation of the average response time. The standard deviation tells you how tightly the data is grouped about the mean. For example, System A and System B both have an average response time of 12 ms. However, this does not mean that the response times are similar. System A might have response times of

11, 12, 13, and 12 ms. System B might have response times of 1, 20, 25, and 2. Although the mean time is the same, the standard deviation of System B is greater and the response time is more varied.

- The maximum elapsed time for the entire run.
- The rate, per second, at which the transaction was completed.
- The number of attempts for the transaction.

Net End-to-End time vs. Time

Net end to end time for a transaction is a measured time of interactions with the server and a client such as a browser or a device. Typically, this does not include think times or processing time by the workbench.

Net Server Time vs. Time

Net server time for a transaction is a measured time of interactions with the server. Typically, this does not include think times or processing time by the product. The server interaction calculation is protocol specific. For example, in HTTP protocol, the net server time is exactly the sum of all page response times. The elapsed time (wall clock time) includes think time and other product processing overhead. For a Web UI test, the net server time includes the sum of time spent on the server and network.

Transaction Throughput

These line graphs provide an overview of the transaction frequency and the number of users that are adding load, both over the course of a run.

- The *Transaction Hit Rate* graph shows the overall rates for starting and completing transactions during a specified sample interval. If you have staged loads in the schedule, this graph also delineates the stages with time range markers, in various colors, at the top. The summary table under the graph lists the transaction rate per second and the number of transactions that were completed for the entire run.
- The *User Load* graph shows the number of active users and the number of users that completed testing over the course of the run. The summary table under the graph lists the number of active users, the number of users that completed testing, and the total number of users for the latest sample interval.

Transaction Health

The *Transaction Health* tab displays whether the transaction is healthy or unhealthy. When you define error conditions for a transaction, you can set whether the behavior of the errors affect the health of the run. If any one condition is met in a transaction, the transaction is marked unhealthy. The Transaction Health tab displays the percentage of healthy and unhealthy transactions. The graphic bar is displayed only if there are no errors.



Note:

During the test run, the elapsed time is recorded for unhealthy transactions. You can select the **Discard time measurements for unhealthy transactions** check box from the **Test Execution Preferences (Windows**



> **Preferences > Test > Test Execution**), if you do not want to record the elapsed time for the unhealthy transactions.

Transaction Details

The *Transaction Completion Percentage* graph in the Transaction Details tab displays the overall percentage of the successful transactions. The Transaction Details section shows the transactions that were attempted, completed, exited, and the percentage of completed transactions.

Transaction Percentile report

This report shows the 85th, 90th, and 95th percentile elapsed times for all users, the union of all transactions in a run, and for the 10 slowest transactions in a run.

The default percentiles in this report, 85, 90, and 95, are sufficient for most purposes. If you need a report on a different percentile set, click **Window > Preferences > Test > Percentile Analysis Targets** to change the percentiles in this report and in the Page Percentile report.

The Summary page of this report has a graph with three bars, which represent the 85th percentile, 90th percentile, and 95th percentile elapsed times for all users and for all transactions in the run. For the 85th percentile bar, 85% of all users achieved the indicated elapsed time or better. For the 90th percentile bar, 90% of all users achieved the indicated elapsed time or better. And for the 95% percentile bar, 95% of all users achieved the indicated elapsed time or better.

The 85%, 90%, and 95% pages show the elapsed time percentiles of the 10 slowest transactions in the run. For example, if you click the tab for the 85th percentile, and the total for a transaction is 110 (the total is beneath each bar), you know that 85 percent of the elapsed times for that transaction are less than or equal to 110 milliseconds (ms).

This graph provides an overall idea of the elapsed times for each transaction. For example, the Transaction report might indicate that a login transaction is one of the 10 slowest transactions. However, it is possible that only one instance of the login transaction was extremely slow while the other instances of the login transaction were within acceptable range. The Transaction Percentile report shows which transactions have slow elapsed time averages because they were slow in general, not because a few elapsed times (out of many) were extremely slow.

The table beneath the graph provides the following information for each transaction:

- The minimum elapsed time for the run.
- The average elapsed time for the run.
- The standard deviation of the elapsed time. The standard deviation tells you how tightly the data is grouped about the mean. For example, assume that System A and System B both have an average elapsed time of 12 ms. However, this does not mean that the elapsed times are similar. System A might have elapsed times of 11, 12, 13, and 12 ms. System B might have elapsed times of 1, 20, 25, and 2 ms. Although the mean time is the same, the standard deviation of System B is greater and the elapsed time is more varied.
- The maximum elapsed time for the run.

- The 85th percentile for the run. That is, for this particular transaction, 85% of the elapsed times were equal to or faster than this time.
- The 90th percentile for the run. That is, for this particular transaction, 90% of the elapsed times were equal to or faster than this time.
- The 95th percentile for the run. That is, for this particular transaction, 95% of the elapsed times were equal to or faster than this time.
- The number of attempts in the run.

Transaction Net Server Time Percentile report

This report shows the 85th, 90th, and 95th percentile net server times for all users and the union of all transactions in a run. The report also shows the 10 slowest transactions in a run. The net server time corresponds to the cumulative server response times within a transaction. Net server time does not include think times and delays, which are included in the elapsed time.

The default percentiles in this report, 85th, 90th, and 95th, are sufficient for most purposes. However, if you must report on a different percentile set, click **Window > Preferences > Test > Percentile Analysis Targets** to change the percentiles in this report and in the Page Percentile report.

The Summary page of this report has a graph with three bars, which represent the 85th percentile, 90th percentile, and 95th percentile net server times for all users and for all transactions in the run. For the 85th percentile bar, 85% of all users achieved the indicated net server time or better. For the 90th percentile bar, 90% of all users achieved the indicated net server time or better. And for the 95% percentile bar, 95% of all users achieved the indicated net server time or better.

The 85%, 90%, and 95% pages show the net server time percentiles of the 10 slowest transactions in the run. For example, if you click the tab for the 85th percentile, and the total for a transaction is 110 (the total is beneath each bar), you know that 85 percent of the net server times for that transaction are less than or equal to 110 milliseconds (ms).

This graph provides an overall idea of the net server times for each transaction. For example, the Transaction report might indicate that a login transaction is one of the 10 slowest transactions. However, it is possible that only one instance of the login transaction was extremely slow while the other instances of the login transaction were within acceptable range. The Transaction Net Server Time Percentile report shows which transactions have slow net server time averages because they were slow in general, not because a few net server times (out of many) were extremely slow.

The table beneath the graph provides more detailed information for each transaction:

- The minimum net server time for the run.
- The average net server time for the run.
- The standard deviation of the net server time. The standard deviation tells you how tightly the data is grouped about the mean.
- The maximum net server time for the run.

- The 85th percentile for the run. That is, for this particular transaction, 85% of the net server times were equal to or faster than this time.
- The 90th percentile for the run. That is, for this particular transaction, 90% of the net server times were equal to or faster than this time.
- The 95th percentile for the run. That is, for this particular transaction, 95% of the net server times were equal to or faster than this time.
- The number of attempts in the run.

Rate Generator report

Use this report to determine if the transactions run at the rate that you specify.

The report contains three pages. See the description of each of the page.

Rate Generators

State: Status of the run for rate generators. One of the following statuses is displayed:

- Not started
- Arriving: At least one user has arrived at the synchronization point.
- Active: At least one iteration of workload.
- Inactive: No users or clients are running.

Users or Clients: Number of users or clients in the Arriving or Active state.

Target Rate: Number of iterations specified in Rate Generator or Rate Schedule. The report shows the value in seconds. For example, if the iteration rate is 2 every minute, the Target Rate would show 0.032 per second.

Workloads Started - First time: Time taken for the first workload to start after the test run. The time taken includes the time to create the users or clients, begin the workload, meet at the rate synchronization point, run any iteration delay, and then start the workload.

Workloads Started - Elapsed time: Time taken for the first workload to start and the last workload to end.

Workloads Started: Number of workloads already started at any given point of time during the run. For example, for a target rate of 4 every minute, at the end of one minute, the number of workloads started should be 4. However, it should increment throughout the run based on the target rate. So, at a target rate of 4 every minute and a total duration of 10 minutes, the number of workloads at the end of the run should show `Workloads Started = 40` and `Workloads Completed = 40`.

Workloads Started - Rate: Actual rate achieved for the run. Compare it with the Target Rate to determine the performance of the application.

Percent Target: Comparison of the Target Rate with the Actual Rate (Workloads Started - Rate) to indicate by percentage how close the workload is to reach the specified target rate.

Percent Late - Percent: The percent of workloads in a user group or Rate Runner group that did not begin to run at the specified time.

Health

Use this page to view the number of workloads that did not meet the target rate.

Throughput

Use this page to view the number of workloads that passed successfully.

Related information

[Creating rate generators in user groups on page 562](#)

HTTP performance test reports

When you test an HTTP system, reports are produced during a run and saved after a run. You can then analyze the reports to know the performance of the system under test.

In a performance report, you can sort the order of HTTP pages that are captured in a test or schedule either by alphabetical order or order of execution of the HTTP pages. The default sorting of the HTTP pages is by order of execution.

Performance report

The performance report summarizes the validity of the run and the data that is most significant to the run. The report also shows the response trend of the slowest 10 pages in the test and the graph of the response trend of each page for a specified interval.

Contents

- [Overall page on page 815](#)
- [Summary page on page 816](#)
- [Page Performance page on page 818](#)
- [Response vs. Time Summary page on page 818](#)
- [Response vs. Time Detail page on page 819](#)
- [Page Throughput page on page 819](#)
- [Server Throughput page on page 820](#)
- [Server Health Summary page on page 820](#)
- [Server Health Detail page on page 821](#)
- [Caching Details page on page 821](#)
- [Resources page on page 821](#)
- [Page Element Responses on page 824](#)
- [Page Response Time Contributions on page 824](#)

- [Page Size on page 824](#)
- [Errors on page 824](#)
- [Page Health on page 824](#)

Overall page

The Overall page provides the following information:

- A progress indicator that shows the state of the run.
- A pie chart that shows the information about the overall verification point passed and failed for the test run if they were set. For the schedule run, it displays the overall requirements that passed and failed.

Item	Description	Verdict
Page VPs	Displays the verdict of the page title verification points if they were set.	<ul style="list-style-type: none"> • Passed • Failed • Inconclusive • Error
Page Element VPs	Displays the verdict of the response code or response size verification points if they were set.	<ul style="list-style-type: none"> • Passed • Failed • Inconclusive • Error
Page Status Codes	<p>Displays the success and failure rate for the entire run.</p> <p>If a primary request includes verification points, the Page Status Code Successes value indicates that the verification point for the response code is passed.</p> <p>If a primary request has no verification points, the Page Status Code Successes value indicates that the server received the primary request and returned a response with one of the following status codes:</p> <ul style="list-style-type: none"> • 200 or 300 category • 400 or 500 category, which is an expected response code 	<ul style="list-style-type: none"> • Passed • Failed

Item	Description	Verdict
Page Element Status Codes	<p>Displays the success and failure rate for the entire run.</p> <p>If a primary request includes verification points, the Page Element Successes value indicates that the response code verification point passed for that request.</p> <p>If a request has no verification points, the Page Element Successes indicates that the server received the request and returned a response with one of the following status codes:</p> <ul style="list-style-type: none"> • 200 or 300 category • 400 or 500 category, which is an expected response code 	<ul style="list-style-type: none"> • Passed • Failed
Page Health	Displays the total health of the pages, transactions, and loops for the test or schedule run.	<ul style="list-style-type: none"> • Healthy • Unhealthy

- If you click any individual chart, you can go to that specific report to analyze the status in detail.
- If you click any legend (for example, Passed), the chart is updated to show only the other verdicts of a test or a schedule run. For example, the Page Status Codes has a legend as Passed and Failed. If you click Passed, the chart is updated to show only errors during a test or a schedule run.
- Similarly, if you double-click any legend, the chart is updated to show only the selected verdict by removing all other verdicts from the chart. Thus you can focus on only one counter which you want to investigate in detail.

Summary page

The **Summary** page summarizes the most important data about the test run, so that you can analyze the final or intermediate results of a test at a glance.

The **Summary** page displays the following `Run Summary` information:

- The name of the test.
- The number of users that are active and the number of users that have completed testing. This number is updated during the run.
- The elapsed time. This is the run duration, which is displayed in hours, minutes, and seconds.
- The status of the run. This can be `Initializing Computers`, `Adding Users`, `Running`, `Transferring data to test log`, `Stopped`, **OR** `Complete`.
- `Displaying results for computer: All Hosts`. To see summary results for individual computers, click the computer name in the Performance Test Runs view.

The **Summary** page displays the following `Page Summary` information:

- The total number of page attempts and hits. A *page attempt* means that a primary request was sent; it does not include requests within the page. A *hit* means that the server received the primary request and returned any complete response.
- The average response time for all pages. *Response time* is the sum of response times for all page elements (including the connect time and inter-request delays). Response time counters omit page response times for pages that contain requests with status codes in the range of 4XX (client errors) to 5XX (server errors). The only exception is when the failure (for example, a 404) is recorded and returned, and the request is not the primary request for the page. Page response times that contain requests that time out are always discarded.
- The standard deviation of the average response time for all pages.
- The maximum response time for all pages.
- The minimum response time for all pages.
- A summary of the results for page verification points, if these verification points were set.

The **Summary** page displays the following `Page Element Summary` information:

- The total number of page element attempts and hits. A *page element attempt* means that a request was sent. A *hit* means that the server received the request and returned any complete response.
- The total number of page elements where no request was sent to the server because the client determined that the page elements were fresh in the local cache.
- The average response time for all page elements. *Response time* is the time between the first request character sent and the last response character received. Response times for HTTP requests that time out or that return an unexpected status code (the recorded and played back codes do not match) in the range of 4XX (client errors) to 5XX (server errors) are discarded from the reported values.
- The standard deviation of the average response time. The standard deviation tells you how tightly the data is grouped about the mean. For example, System A and System B both have an average response time of 12 ms. However, this does not mean that the response times are similar. System A might have response times of 11, 12, 13, and 12 ms. System B might have response times of 1, 20, 25, and 2. Although the mean time is the same, the standard deviation of System B is greater and the response time is more varied.
- The percentage of verification points that passed.
- A summary of the results for page element verification points, if these verification points were set.

If you have set transactions in your test, the **Summary** page displays the following `Transaction` information:

- The minimum, maximum, and average response time for all transactions. *Response time* is the actual time spent within the transaction container.
- The standard deviation of the average response time. The standard deviation tells you how tightly the data is grouped about the mean. For example, System A and System B both have an average response time of 12 ms. However, this does not mean that the response times are similar. System A might have response times of 11, 12, 13, and 12 ms. System B might have response times of 1, 20, 25, and 2. Although the mean time is the same, the standard deviation of System B is greater and the response time is more varied.
- The total number of transactions that were started and completed.

Page Performance page

The **Page Performance** page shows the average response of the slowest 10 pages in the test as the test progresses. With this information, you can evaluate system response during and after the test.

The bar chart shows the average response time of the 10 slowest pages. Each bar represents a page that you visited during recording. As you run the test, the bar chart changes, because the 10 slowest pages are updated dynamically during the run. For example, the Logon page might be one of the 10 slowest pages at the start of the run, but then, as the test progresses, the Shopping Cart page might replace it as one of the 10 slowest. After the run, the page shows the 10 slowest pages for the entire run.

The table under the bar chart provides the following additional information:

- The minimum response time for each page in the run. *Response time* is the time between the first request character sent and the last response character received. Response time counters omit page response times for pages that contain requests with status codes in the range of 4XX (client errors) to 5XX (server errors). The only exception is when the failure (for example, a 404) is recorded and returned, and the request is not the primary request for the page. Page response times that contain requests that time out are always discarded.
- The average response time for each page in the run. This matches the information in the bar chart.
- The standard deviation of the average response time. The standard deviation tells you how tightly the data is grouped about the mean. For example, System A and System B both have an average response time of 12 ms. However, this does not mean that the response times are similar. System A might have response times of 11, 12, 13, and 12 ms. System B might have response times of 1, 20, 25, and 2. Although the mean time is the same, the standard deviation of System B is greater and the response time is more varied.
- The maximum response time for each page in the run.
- The number of attempts per second to access each page. An *attempt* means that a primary request was sent; it does not include requests within the page.
- The total number of attempts to access the page.

To display the 10 slowest page element response times, right-click a page and click **Display Page Element Responses**.

Response vs. Time Summary page

The **Response vs. Time Summary** page shows the average response trend as graphed for a specified interval. It contains two line graphs with corresponding summary tables. When a schedule includes staged loads, colored time-range markers at the top of the graph delineate the stages.

- The Page Response vs. Time graph shows the average response time for all pages during the run. Each point on the graph is an average of what has occurred during that interval. The table after the graph lists the total average response time for all pages in the run and the standard deviation of the average response time.
- The Page Element response vs. Time graph shows the average response time for all page elements during the run. Each point on the graph is an average of what has occurred during that interval. The table under the graph lists the total average response time for all page elements in the run and the standard deviation of the average

response time. The table also lists the total number of page elements where no request was sent to the server because the client determined that the page elements were fresh in the local cache. You set the **Statistics sample interval** value in the schedule, as a schedule property.

Response vs. Time Detail page

The **Response vs. Time Detail** page shows the response trend as graphed for the sample intervals. Each page is represented by a separate line.

The Average Page Response Time graph shows the average response of each page for each sample interval. When a schedule includes staged loads, colored time-range markers at the top of the graph delineate the stages. The table after the graph provides the following additional information:

- The minimum page response time for the run. *Response time* is the time between the first request character sent of the primary request and the last response character received. Response time counters omit page response times for pages that contain requests with status codes in the range of 4XX (client errors) to 5XX (server errors). The only exception is when the failure (for example, a 404) is recorded and returned, and the request is not the primary request for the page. Page response times that contain requests that time out are always discarded.
- The average page response time for the run. This is similar to the graph, but the information in the table includes the entire run.
- The maximum page response time for the run.
- The standard deviation of the average response time. The standard deviation tells you how tightly the data is grouped about the mean. For example, System A and System B both have an average response time of 12 ms. However, this does not mean that the response times are similar. System A might have response times of 11, 12, 13, and 12 ms. System B might have response times of 1, 20, 25, and 2. Although the mean time is the same, the standard deviation of System B is greater and the response time is more varied.
- The rate of page attempts per interval for the most recent statistics sample interval. A *page attempt* means that the primary request was sent; it does not include requests within the page. You set the **Statistics sample interval** value in the schedule, as a schedule property.
- The number of page attempts per interval.

Page Throughput page

The **Page Throughput** page provides an overview of the frequency of requests being transferred per sample interval.

- The Page Hit Rate graph shows the page attempt rate and page hit rate per sample interval for all pages.

A *page attempt* means that the primary request was sent; it does not include requests within the page.

A *hit* means that the server received the primary request and returned any complete response.

When a schedule includes staged loads, colored time-range markers at the top of the graph delineate the stages. The summary table after the graph lists the total hit rates and counts for each page in the run.

- The User Load graph shows active users and users that have completed testing, over the course of a run. The summary table after the graph lists the results for the most recent sample interval. You set the **Statistics**

sample interval value in the schedule, as a schedule property. As the run nears completion, the number of active users decreases and the number of completed users increases. The summary table after the graph lists the active and completed users for the entire run.



Note: To set the sample interval value, open the schedule, click the **Statistics** tab, and then view or modify **Statistics sample interval**.

If the number of requests and hits are not close, the server might be having trouble keeping up with the workload.

If you add virtual users during a run and watch these two graphs in tandem, you can monitor the ability of your system to keep up with the workload. As the page hit rate stabilizes, even though the active user count continues to climb and the system is well-tuned, the average response time will naturally slow down. This response time reduction happens because the system is running at its maximum effective throughput level and is effectively throttling the rate of page hits by slowing down how quickly it responds to requests.

Server Throughput page

The **Server Throughput** page lists the rate and number of bytes that are transferred per interval and for the entire run. The page also lists the status of the virtual users for each interval and for the entire run.

- The Byte Transfer Rates graph shows the rate of bytes sent and received per interval for all intervals in the run. When a schedule includes staged loads, colored time-range markers at the top of the graph delineate the stages. The summary table after the graph lists the total number of bytes sent and received for the entire run.
- The User Load graph shows active users and users that have completed testing, per sample interval, over the course of a run. You set the **Statistics sample interval** value in the schedule, as a schedule property. As the run nears completion, the number of active users decreases and the number of completed users increases. The summary table after the graph lists the active and completed users for the entire run.

The bytes sent and bytes received throughput rate, which is computed from the client perspective, shows how much data HCL OneTest™ Performance is pushing through your server. Typically, you analyze this data with other metrics, such as the page throughput and resource monitoring data, to understand how network throughput demand affects server performance.

Server Health Summary page

The **Server Health Summary** page gives an overall indication of how well the server is responding to the load.

- The Page Health chart shows the total number of page attempts, page hits, and status code successes for the run. The table under the bar chart lists the same information.

A *page attempt* means that a primary request was sent; it does not include requests within the page.

A *hit* means that the server received the primary and returned any complete response.

A *success* means that the response code verification point passed for that request. If a primary request has no verification points, the Success value indicates that the server received the primary request and returned a response with one of the following status codes:

- 200 or 300 category
- 400 or 500 category, which is an expected response code
- The Page Element Health chart shows the total number of page element attempts, page element hits, status code successes, and page element redirections for the run. The table under the bar chart lists the same information and the total number of page elements where no request was sent to the server because the client determined that the page elements were fresh in the local cache.

Server Health Detail page

The **Server Health Detail** page provides specific details for the 10 pages with the lowest success rate.

- The bar chart shows 10 pages with the lowest success rate.
- The summary table under the chart lists, for all pages, the number of attempts, hits, and successes in the run and the attempts per second during the run.

An *attempt* means that a primary request was sent; it does not include requests within the page.

A *hit* means that the server received the primary and returned any complete response.

A *success* means that the response code verification point passed for that request. If a primary request has no verification points, the Success value indicates that the server received the primary request and returned a response with one of the following status codes:

- 200 or 300 category
- 400 or 500 category, which is an expected response code

Caching Details page


The **Caching Details** page provides specific details on caching behavior during a test run.

- The Caching Activity graph shows the total number of page element cache attempts, page element cache hits, and page element cache misses for the run. These values correspond to responses from the server, indicating whether the content has been modified. Additionally, the bar chart shows the total number of page elements in the cache that were skipped for the run. That value indicates the cache hits that were still fresh in the local cache, where communication with the server was not necessary.
- The Page Element Cache Hit Ratios graph shows the percentage of cache attempts that indicate server-confirmed success and client-confirmed success for the run. Server-confirmed cache hits occur when the server returns a 304 response code. Client-confirmed cache hits occur when the content is still fresh in the local cache and no communication with the server is required.

Resources page

The **Resources** page shows information about all the resource counters that were monitored during the *schedule* run.

The **Resources** page displays the following information as mentioned in the table:

If...	Then the Resources page displays...
<p>If you did not add any Resource Monitoring source to a performance schedule</p>	<p>A message that states that you must set up the Resource Monitoring sources to view the resource details.</p>
<p>If you added Resource Monitoring sources to a performance schedule</p>	<ul style="list-style-type: none"> • The Resource Monitoring sources that were monitored during the schedule run. • All resource counters for those Resource Monitoring sources that were monitored during the schedule run. • The Unavailable sources section that lists the Resource Monitoring sources that were unavailable or unreachable during the schedule run. <p> Note: The Unavailable sources section is displayed only if any of the Resource Monitoring sources were unreachable or unavailable during the schedule run.</p>
<p>If you added Resource Monitoring sources by using labels to a performance schedule</p>	<ul style="list-style-type: none"> • The following information in the Server sources matching the labels set in the schedule section: <ul style="list-style-type: none"> ◦ Labels and the Resource Monitoring sources associated with those labels that were monitored during the schedule run. ◦ Resource monitoring sources that were unavailable or unreachable during the schedule run. ◦ An empty array (<code>[]</code>) when you used labels that were not tagged to any Resource Monitoring source in HCL OneTest™ Server. • All resource counters for the Resource Monitoring sources that were monitored during the schedule run.

If...	Then the Resources page displays...
<p>If you ran a performance schedule by using the <code>overridermlabels</code> command from the HCL OneTest™ Performance command line</p>	<ul style="list-style-type: none"> • The following information in the Server sources matching the labels set with the command-line flag <code>-overridermlabels</code> section: <ul style="list-style-type: none"> ◦ Labels that you used to add the Resource Monitoring sources to the schedule for the schedule run. ◦ Resource monitoring sources associated with those labels that were monitored during the schedule run. ◦ Resource monitoring sources that were unavailable or unreachable during the schedule run. ◦ An empty array (<code>[]</code>) when you used labels that were not tagged to any Resource Monitoring source in HCL OneTest™ Server. • All resource counters for the Resource Monitoring sources that were monitored during the schedule run.

The Legend shows the Resource Monitoring type and its resource counters. When you have multiple Resource Monitoring sources, the resource counters for the respective sources are displayed in front of their Resource Monitoring source name. You can customize the resource counter information displayed in a graph by clicking any individual resource counter or type of source. You can click or double-click any individual resource counter for the following results:

- A single click on the resource counter hides the data displayed on the graph. Click the resource counter again to display the data in the graph.
- A double-click on the resource counter removes information about all other resource counters from the graph and displays only the information about the selected resource counter.



Tip: You can click **Select All** option to restore all the resource counter information on the graph.

When you click on any of the sources, the graph removes all the resource counters of other sources and displays only the resource counters of the selected source.

For example, you have an Apache httpd server and a Windows Performance Host as a Resource Monitoring source. When the *schedule* completes, the **Resources** page displays the resource counter information of both the sources. If you want to analyze the resource counters for any one of the sources, you can click the Apache httpd server or the

Windows Performance Host. Based on your selection, the graph is updated to show the selected source resource counters information.

The Performance Summary table under the graph lists the most recent values of the resource counters that were monitored during the schedule run. The first two columns show the **Type** of the source and **Name** of the resource counter. This table also lists the minimum, maximum, and average values of the resource counters that were monitored during the schedule run.

Page Element Responses

The **Page Element** page shows the 10 slowest page element responses for the selected page.

Page Response Time Contributions

The **Page Response Time Contributions** page shows how much time each page element contributes to the overall page response time and the client delay time and connection time.

Page Size

This page lists the size of each page of your application under test. The size of the page contributes to the response time. If part of a page or an entire page is cached, then those requests coming from the cache will not contribute to the total page size.

The size of a page is mostly determined by the size of its elements. Each bar in this report represents a page. To view the Page Elements Size report, click a bar and select **Page Element Sizes**. All the elements that are on the page show up with sizes.

Errors

This page lists the number of errors and the corresponding actions that occurred in the test or schedule. The Error Conditions section displays the number of error conditions met. The Error Behavior section displays how each error condition was handled. You should have already defined how to handle errors in the **Advanced** tab of the test editor, schedule editor, or compound test editor.

Page Health

Use this page(report) to determine if the pages of your application have errors. If a page contains any error, the report displays that the page is not 100% healthy. If there are pages that are not 100% healthy, the report displays another section listing such pages and the errors reported.

Page Element report

This report summarizes the most important page element data for the run.

The graphs in this report show time intervals, attempts, hits, and successes.

- The *interval* depends on the **Statistics sample interval** value that you set for the schedule.
- An *attempt* means that a request was sent.
- A *hit* means that the server received the request and returned any response.

- A *success* means that the response code verification point passed for that request. If the request has no verification point, a success means that the server received a request and returned a response where the status code was in the 200 or 300 category, or returned an expected response in the 400 or 500 category.
- The *response time* is the time between the first request character sent and the last response character received. The response time does not include HTTP requests that time out or requests that return a status code in the range of 4XX (client errors) - 5XX (server errors) or requests that timed out. Response times for HTTP requests that time out or that return an unexpected status code (the recorded and played back codes do not match) in the range of 4XX (client errors) to 5XX (server errors) are discarded from the reported values.

Overall page

The Overall line graph shows the average response time for all page elements during a specified interval. When you have set staged loads in the schedule, this graph delineates the stages with time range markers, in various colors, at the top. The table after the graph provides the following information:

- The average response time for all page elements in the entire run
- The standard deviation of the average response time. The standard deviation tells you how tightly the data is grouped about the mean. For example, System A and System B both have an average response time of 12 ms. However, this does not mean that the response times are similar. System A might have response times of 11, 12, 13, and 12 ms. System B might have response times of 1, 20, 25, and 2. Although the mean time is the same, the standard deviation of System B is greater and the response time is more varied.
- The average number of page elements attempted for the entire run
- The total page element attempts for the entire run

Response vs. Time Summary page

The Response vs. Time Summary line graph shows the response time for the 10 slowest page elements in the run. When you have set staged loads in the schedule, this graph delineates the stages with time range markers, in various colors, at the top. The table after the graph lists the parent page and the page element, and provides the following information:

- The average response time for that page element during the entire run
- The standard deviation of the average response time. The standard deviation tells you how tightly the data is grouped about the mean. For example, System A and System B both have an average response time of 12 ms. However, this does not mean that the response times are similar. System A might have response times of 11, 12, 13, and 12 ms. System B might have response times of 1, 20, 25, and 2. Although the mean time is the same, the standard deviation of System B is greater and the response time is more varied.
- The attempts per second during the most recent sample interval
- The number of attempts during the most recent sample interval

Response vs. Time Detail page

The Response vs. Time Detail page shows the response time for each page element in the run. The table lists the parent page and page element, and provides the following information for each page element in the entire run:

- The average response time
- The standard deviation of the average response time. The standard deviation tells you how tightly the data is grouped about the mean. For example, System A and System B both have an average response time of 12 ms. However, this does not mean that the response times are similar. System A might have response times of 11, 12, 13, and 12 ms. System B might have response times of 1, 20, 25, and 2. Although the mean time is the same, the standard deviation of System B is greater and the response time is more varied.
- The number of attempts
- The number of attempts per second

Page Element Throughput page

The Page Element Throughput page shows the average response trend during a specified interval. It contains two line graphs with corresponding summary tables:

- The Page Element Hit Rate graph shows the combined attempt rate and hit rate for all page elements during the last recorded interval. The table after the graph lists one number: the average hit rate for all pages in the run. When a schedule includes staged loads, colored time-range markers at the top of the graph delineate the stages.
- The User Load graph shows active users compared to users that have completed testing. The table after the graph lists the number of active users, the number of users that have completed testing, and the total user count for the entire run.

Server Health Detail page

The Server Health Detail bar chart shows the percentage of successes for the 10 slowest page elements in the run. The table under the chart lists the parent page and page element, and provides the following information for the entire run:

- The number of attempts
- The number of hits
- The number of successes
- The percent of successes (matches the information in the bar chart)
- The number of attempts per second

Caching Details page

The Caching Details page provides specific details on caching behavior during a test run.

- The Caching Activity graph shows the total number of page element cache attempts, page element cache hits, page element cache misses for the run. These values correspond to responses from the server indicating whether the content has been modified. Additionally, the bar chart shows the total number of page elements in cache skipped for the run. That value indicates the cache hits that were still fresh in the local cache, where communication with the server was not necessary.
- The Page Element Cache Hit Ratios graph shows the percentage of cache attempts that indicate server-confirmed success and client-confirmed success for the run. Server-confirmed cache hits occur when the

server returns a 304 response code. Client-confirmed cache hits occur when the content is still fresh in the local cache and no communication with the server is required.

- The summary table under the charts lists the the total number of page elements found fresh in the cache for the run. This value indicates the cache hits that were still fresh in the local cache, where communication with the server was not necessary. The table also lists the number of attempts, the total number of page element cache attempts sent to the server, and the total number of page element cache hits confirmed by the server for the run. Additionally, the table lists the percentage of cache attempts that indicate server-confirmed success and client-confirmed success for the run.

Why response time of a page does not equal the sum of its requests

The response time for a page typically differs from the sum of its requests. This does not mean that your data is incorrect. The difference can be caused by concurrent requests, page connection times, inter-request delays, and custom code within a page.

The most common reason for the sum of the individual request times within a page to exceed the total page response time is that requests are often sent concurrently (in parallel) to a server. Thus some of the individual request response times overlap so the sum of the request response times would exceed the page response time.

Additionally, the page response time can exceed the sum of the individual request response times within the page for the following reasons:

- The individual request response times do not include time to establish connections but the page response time does include the connection request time.
- Inter-request delays are not reflected in the individual request response time but are reflected in the page response time.
- Custom code placed within a page is executed serially (after waiting for all previous individual requests to complete) and thus contributes to the page response time. It does not affect individual request response times. However, we recommend that you place custom code outside of a page, where it will not affect page response time. For more information, see [Reducing the performance impact of custom code on page](#).

Page Percentile report

This report shows the 85th, 90th, and 95th percentile response times for all users and all pages in a run, as well as for the 10 slowest pages in a run.

The default percentiles in this report, 85, 90, and 95, are sufficient for most purposes. However, if you are required to report on a different percentile set, click **Window > Preferences > Test > Percentile Analysis Targets** to change the percentiles in this report and in the Transaction Percentile report.

The Summary page shows a graph with three bars, which represent the 85th percentile, 90th percentile, and 95th percentile response times for all users and for all pages in the run. For the 85th percentile bar, 85% of all users achieved the indicated response time or better. For the 90th percentile bar, 90% of all users achieved the indicated response time or better. And for the 95% percentile bar, 95% of all users achieved the indicated response time or better.

The 85%, 90%, and 95% pages show the response-time percentiles of the 10 slowest pages in the run. For example, if you click the tab for the 85th percentile, and the total for a page is 110 (the total is beneath each bar), you know that 85 percent of the response times for that page are less than or equal to 110 milliseconds (ms).

This graph provides an overall idea of the response times for each page. For example, the Page Performance report might indicate that a Login page is one of the 10 slowest pages. However, it is possible that one page attempt was extremely slow, but the other attempts were within range. The Page Percentile report shows which pages have slow responses because they were slow in general, not because a few responses (out of many) were extremely slow.

The table beneath the graph provides more detailed information for each page:

- The minimum response time for the run.
- The average response time for the run.
- The standard deviation of the average response time. The standard deviation tells you how tightly the data is grouped about the mean. For example, System A and System B both have an average response time of 12 ms. However, this does not mean that the response times are similar. System A might have response times of 11, 12, 13, and 12 ms. System B might have response times of 1, 20, 25, and 2. Although the mean time is the same, the standard deviation of System B is greater and the response time is more varied.
- The maximum response time for the run.
- The 85th percentile for the run. That is, for this particular page, 85% of the response times were equal to or faster than this time.
- The 90th percentile for the run. That is, for this particular page, 90% of the response times were equal to or faster than this time.
- The 95th percentile for the run. That is, for this particular page, 95% of the response times were equal to or faster than this time.
- The number of attempts in the run.

Verification Points report

This report shows the status of the verification points in your tests.

This report is displayed if your tests verify page titles, the return code for a page element, or the response size of a page element. To have your tests verify these items, complete the following steps:

1. Before recording, set the verification points. Click **Window > Preferences > Test > Test Generation > HTTP Test Generation** and under **Automatically include verification point of** select one of the verification point behavior that you want to test. Or, during test editing, enable page verification points individually by right-clicking the test item.
2. In the schedule, set the level for logging statistics to **Pages** or **All**.

The VP Pass Rate vs. Time graph on the Summary page lists the percentage of verification points that passed per sample interval during the last recorded interval. You set the **Statistics sample interval** value in the schedule, as a schedule property. When a schedule includes staged loads, colored time-range markers at the top of the graph delineate the stages.

The Page Verification Points page lists the following information for each page that has verification points:

- The number of verification points that passed in the run
- The number of verification points that failed in the run
- The percentage of verification points that passed

The Page Element Verification Points page lists the following information for each page element that has verification points:

- The name of the page and the element
- The number of verification points that passed in the run
- The number of verification points that failed in the run
- The percentage of verification points that passed

SAP performance test reports

When you test an SAP application, these performance test reports are produced during a run and saved after a run.

SAP Performance report

The SAP Performance report summarizes the health of the run, displays the data most significant to the run, shows the response trend of the slowest 10 transactions in the test, and graphs the response trend of each transaction for a specified interval.

Summary page

The *Summary* page summarizes the most important data about the test run, so that you can analyze the final or intermediate results of a test at a glance.

The *SAP Summary* section displays the following information:

- A progress indicator that shows the state of the run.
- The number of virtual users that are active and the number of virtual users that have completed testing. This number is updated during the run.
- The elapsed time. This figure is the run duration, which is displayed in hours, minutes, and seconds.
- The location and name of the test suite or schedule.
- Results for computer, All Hosts. To see summary results for individual computers, click the computer name in the Performance Test Runs view.
- The status of the run. This can be Initializing Computers, Adding Users, Running, Performing Execution History Data Transfer, Stopped, or Complete.
- The total number of virtual users that are simulated during the test.

The *Transaction Summary* section displays the following information:

- The minimum, maximum, average, and standard deviation execution time for all transactions. *Execution time* is the sum of response times for all screens of a transaction (including the connect time and inter-request delays).
- The total number of transactions that were completed and started.

The *Screen Summary* section displays the following information:

- The minimum, maximum, average, and standard deviation screen request response times for all SAP screens. *Response time* is the time between the first request character that is sent and the last response character that is received.
- The total number of SAP screens that were completed and started.
- Percent of verification points that passed.
- Total verification points that failed.
- Total verification point errors.
- Total verification points that passed.
- Total verification points that are inconclusive.

The *Element Summary* section displays the following information:

- The total number of SAP set, SAP get, and SAP call elements that were attempted.
- The total number of SAP set, SAP get, and SAP call elements that were completed.
- The percentage of completion of all SAP set, SAP get, and SAP call elements.

Screen Performance page

The *Screen Performance* page shows the average response time of the longest SAP screen requests in the test as the test progresses. With this information, you can evaluate system response during and after the test.

If you select **Do not measure performance on this screen** on a SAP screen in the test editor, then the report does not include the response time results for that screen.

The bar chart shows the average response time of the 10 slowest transactions. Each bar represents a page that you visited during recording. As you run the test, the bar chart changes, because the 10 slowest screens are updated dynamically during the run. For example, the SAP Easy Access screen might be one of the 10 slowest pages at the start of the run, but then, as the test progresses, Display Material (Initial Screen) might replace it as one of the 10 slowest. After the run, the page shows the 10 slowest transactions for the entire run.

The *Performance Summary* table that follows the bar chart provides the following additional information for each SAP screen:

- The minimum response time for each SAP screen in the run. *Response time* is the time between the first request character that is sent and the last response character that is received.
- The average response time for each SAP screen in the run. This matches the information in the chart.
- The maximum response time for each SAP screen in the run.
- The standard deviation response time for each SAP screen in the run.

- The rate per second at which each SAP screens is started.
- The total number of times each SAP screen is started.

Response vs. Time Summary page

The *Response vs. Time Summary* page shows the average response trend as graphed for a specified interval. The line graph shows the average response time for all SAP screens during the run. Each point on the graph is an average of what has occurred during that interval. You set the **Statistics sample interval** in the schedule, as a schedule property.

The table that follows the graph displays the average and standard deviation response time for all SAP screens during the run.

Response vs. Time Detail page

The *Response vs. Time Detail* page shows the response trend by screen as graphed for a specified interval. Each separate line represents a SAP screen. You set the **Statistics sample interval** in the schedule, as a schedule property.

The line graph shows the average response of each page during a specified interval. The table that follows the graph provides the following additional information for each SAP screen:

- The minimum SAP screen request response time for the run. This is the time between the moment the input is validated in the SAP GUI and the time the resulting SAP screen is displayed.
- The average SAP screen request response time for the run. This is similar to the graph, but the information in the table is for the duration of the entire run.
- The maximum SAP screen request response time for the run.
- The standard deviation for SAP screen request response time for the run.
- The rate per second at which each SAP screen is started for the most recent interval.
- The total number of times each SAP screen is started for the most recent interval.

Screen Throughput page

The *Screen Throughput* page provides an overview of the frequency of requests that are being transferred per interval.

- The line graph on the left shows two lines that represent the rate for intervals at which SAP screens are started and the rate at which SAP screens are completed. The summary table under the graph lists the start rates and the completed rates and counts for each SAP screen in the run.
- The line graph on the right shows active users and users that have completed testing, over the course of a run. The summary table under the graph lists the results for the most recent sample interval. You set the **Statistics sample interval** in the schedule, as a schedule property.

Server Health page

The *Server Health Summary* page shows an overall indication of how well the server is responding to the load.

The bar chart on the left represents the total number of SAP screens started. The bar chart on the right represents the percentage of SAP screens completed compared to the SAP screens started on each interval. The percentage can be over 100% if more screens were completed than started on a given interval.

The table that follows the charts lists the same information for each SAP screen.

Batch Input Transaction Rate page

The *Batch Input Transaction Rate* page displays the number of batch input tests that were run during the test run. The **Batch Input Transaction** graph represents the volume of batch input transactions that were processed during the test run.

SAP Verification Points report

The SAP Verification Points report shows the status of the verification points in your tests.

Summary page

The Summary page displays a line graph representing the percentage of verification points that passed per interval. You set the **Statistics sample interval** value in the schedule, as a schedule property.

Below the graph, the Verification Point Summary table lists the following information:

- The percentage of verification points that passed during the run.
- The number of verification points that were tested.
- The number of verification points that passed.
- The number of verification points that failed.
- The number of verification points that produced an error.
- The number of verification points that were inconclusive.

Screen Verification Points page

The Screen Verification Points page lists the following information for each page element that has verification points:

- The number of verification points that passed during the run
- The number of verification points that failed during the run
- The percentage of verification points that passed during the run

Verification points set on the screen title are counted with verification points set on get events in the screen.

Citrix performance test reports

When you test a Citrix XenApp application, these reports are produced during a run and saved after a run.

Citrix Performance report

The Citrix performance report summarizes the validity of the run, graphs show response times, average response time, and the server health depending on requests.

Citrix Overall page

The Overall page provides the following information:

- A progress indicator that shows the state of the run.
- The bar chart indicates the overall success of the run with the percentage of window and image synchronization successes and the percentage of verification point successes. *Synchronization success* indicates that the expected window and image events in the test match the actual window and image events in the test run.

Performance Summary page

The Summary page summarizes general data about the test run, so that you can analyze the final or intermediate results of a test at a glance.

The Run Summary table displays the following information:

- The number of virtual users that are active and the number of virtual users that have completed testing. This number is updated during the run.
- The elapsed time. This is the total duration of the run, which is displayed in hours, minutes, and seconds.
- The location and name of the test.
- The status of the run. This can be Initializing Computer(s), Adding Users, Running, Performing Execution History Data Transfer, Stopped, or Complete.
- The total number of virtual users emulated during the test.

The Citrix Summary section displays the following information:

- The statistics values (average, standard deviation, maximum, minimum) of the average response time for all response time measurements. *Response times* are determined by measurements that are located in the tests. Response time measurements can be automatically generated between the last input action before a window create event and the window create event. The table does not display values that equal zero.
- Total user actions for run. This indicates the total number of user input actions that were emulated during the run.
- The total number of window synchronization attempts.
- The total number of window synchronization successes.
- The total number of window synchronization timeouts. A timeout occurs when the synchronization fails.
- The total number of image synchronization attempts.
- The total number of image synchronization successes.
- The total number of image synchronization timeouts. A timeout occurs when the synchronization fails.

Server Performance Summary page

The server Performance Summary page shows the average time virtual users are connected, active, and disconnected.

The bar chart shows the average response time of each test phase. Each bar represents a particular phase in the test: connecting to the Web interface, connecting to the server, identifying, activity phase and disconnecting. As you run the test, the bar chart changes, because the time measurements are updated dynamically during the run.

The table under the bar chart provides the following additional information for each window:

- The minimum response time during the run.
- The average response time during the run. This matches the information in the chart.
- The maximum response time during the run.
- The standard deviation response time during the run.

Response vs. Time Summary page

The Response vs. Time Summary page shows the average response trend as graphed for a specified interval. You set the **Statistics sample interval** value in the schedule, as a schedule property. *Response times* are determined by measurements that are located in the tests. Response time measurements can be automatically generated between the last input action before a window create event and the window create event.

The line graph shows the average response time for all measurements during the run. Each point on the graph is an average of what has occurred during that interval. The table under the graph lists the statistics values of the total average response time for all measurements in the run.

Response vs. Time Details page

The Response vs. Time Details page shows the response trend as graphed for a specified interval. You set the **Statistics sample interval** value in the schedule, as a schedule property. *Response times* are determined by measurements that are located in the tests. Response time measurements can be automatically generated between the last input action before a window create event and the window create event.

The line graph shows the average response time of each measurement for a specified interval. Each measurement is represented by a separate line.

The table under the graph provides the following additional information for each response time measurement:

- The minimum response time during the run.
- The average window response time during the run. This is similar to the graph, but the information in the table includes the entire run.
- The maximum window response time during the run.
- The standard deviation window response time during the run.

User Action Throughput page

The User Action Throughput page provides an overview of the frequency of requests being transferred per interval. You set the **Statistics sample interval** value in the schedule, as a schedule property.

- The line graph on the left shows the user action rate per interval for all windows. This represents the activity of virtual user input actions per second for each interval. The table under the graph lists the user action rate per second for the entire run, and the total number of user actions for the run.
- The line graph on the right shows active users and users that have completed testing, over the course of a run. The summary table under the graph lists the results for the most recent sample interval.

Server Health Summary page

The Server Health Summary page provides an overall indication of how well the server has performed. The graph does not display values that equal zero.

The bar chart shows the following information:

- The total number of window synchronization attempts.
- The total number of window synchronization successes.
- The total number of window synchronization timeouts.
- The total number of image synchronization attempts.
- The total number of image synchronization successes.
- The total number of image synchronization timeouts.
- The total number of Citrix server errors or errors encountered during test execution.

Server Timeout page

The Server Timeout page shows when the synchronization timeouts and server errors occurred during the run. The graph does not display values that equal zero.

The line graph shows the following information:

- Citrix window synchronization timeouts.
- Citrix image synchronization timeouts.
- Citrix server errors or errors encountered during test execution.

Resources page

The Resources page shows all resource counters monitored during the schedule run.

- The line chart shows the values of the resources counters monitored during the schedule run.

The chart scales automatically to accommodate the highest resource counter value.

- The summary table under the chart lists the average values of the resource counters monitored during the schedule run. This table is organized by resource monitoring hosts.

Citrix Verification Points report

The Citrix Verification Points report shows the status of the verification points in your tests.

Summary page

The Summary page displays a line graph representing the percentage of verification points that passed per interval. You set the **Statistics sample interval** value in the schedule, as a schedule property.

Below the graph, the Window Verification Point Summary table lists the following information:

- The percentage of window verification points that passed during the run
- The number of window verification points that were attempted
- The number of window verification points that passed
- The number of window verification points that failed

The Image Synchronization Verification Point Summary table lists the following information:

- The percentage of image synchronization verification points that passed during the run
- The number of image synchronization verification points that were tested
- The number of image synchronization verification points that passed
- The number of image synchronization verification points that failed

Citrix Verification Points page

The Citrix Verification Points page contains tables with verification point details.

The Window Verification Points table lists the following information:

- The number of window verification points that passed during the run
- The number of window verification points that failed during the run
- The number of window verification points that caused an error during the run
- The number of window verification points that were inconclusive during the run
- The percentage of window verification points that passed during the run

The Image Synchronization Verification Points table lists the following information:

- The number of image synchronization verification points that passed during the run
- The number of image synchronization verification points that failed during the run
- The number of image synchronization verification points that caused an error during the run
- The number of image synchronization verification points that were inconclusive during the run
- The percentage of image synchronization verification points that passed during the run

Citrix response time percentile report

This report shows the 85th, 90th, and 95th percentile response times for all users in a run.

The default percentiles in this report, 85, 90, and 95, are sufficient for most purposes. However, if you are required to report on a different percentile set, click **Window > Preferences > Test > Percentile Analysis Targets** to change the percentiles in this report.

The Summary page shows a graph with three bars, which represent the 85th percentile, 90th percentile, and 95th percentile response times for all users in the run. For the 85th percentile bar, 85% of all users achieved the indicated response time or better. For the 90th percentile bar, 90% of all users achieved the indicated response time or better. And for the 95% percentile bar, 95% of all users achieved the indicated response time or better.

The 85%, 90%, and 95% pages show the response-time percentiles of the 10 slowest window events in the run. For example, if you click the tab for the 85th percentile, and the total for a page is 110 (the total is beneath each bar), you know that 85 percent of the response times for that window event are less than or equal to 110 milliseconds (ms).

This graph provides an overall idea of the response times for each window event. For example, the Response Time Measurements page of the Citrix performance report might indicate that a Login screen is one of the 10 slowest measurements. However, it is possible that one attempt was extremely slow, but the other attempts were within range. The Citrix response time percentile report shows which measurements have slow responses because they were slow in general, not because a few responses (out of many) were extremely slow.

The table beneath the graph provides more detailed information for each window event:

- The minimum response time for the run.
- The average response time for the run.
- The standard deviation of the average response time. The standard deviation tells you how tightly the data is grouped about the mean. For example, System A and System B both have an average response time of 12 ms. However, this does not mean that the response times are similar. System A might have response times of 11, 12, 13, and 12 ms. System B might have response times of 1, 20, 25, and 2. Although the mean time is the same, the standard deviation of System B is greater and the response time is more varied.
- The maximum response time for the run.
- The 85th percentile for the run. That is, for this particular measurement, 85% of the response times were equal to or faster than this time.
- The 90th percentile for the run. That is, for this particular measurement, 90% of the response times were equal to or faster than this time.
- The 95th percentile for the run. That is, for this particular measurement, 95% of the response times were equal to or faster than this time.
- The number of attempts in the run.

Related reference

[Percentile analysis preferences on page 1194](#)

Web service reports

When you test a web service, these reports are produced during a run and saved after a run.

Service Performance report

The Service Performance report summarizes the validity of the run, summarizes the data most significant to the run, shows the response trend of the slowest 10 service calls in the test, the server health depending on requests, and graphs the response trend of each service calls for a specified interval.

Overall page

The Overall page provides the following information:

- A progress indicator that shows the state of the run.
- The bar graph on the left indicates the percentage of successful service calls during the run.
- The bar graph on the right indicates the percentage of verification points with a Pass status for the run.

Summary page

The Summary page summarizes the most important data about the test run, so that you can analyze the final or intermediate results of a test at a glance.

The Run Summary table displays the following information:

- The number of virtual users that are active and the number of virtual users that have completed testing. These numbers are updated during the run.
- The elapsed time. This is the total duration of the run, which is displayed in hours, minutes, and seconds.
- The location and name of the test.
- The results for the computer and for all computers. To see summary results for individual computers, click the computer name in the **Performance Test Runs** view.
- The status of the run. This can be Initializing Computers, Adding Users, Running, Performing Execution History Data Transfer, Stopped, or Complete.
- The total number of virtual users emulated during the test.

The Call Summary section displays the following information:

- The percentage of verification points with a Pass status.
- The total number of verification points with a Fail status.
- The total number of verification points with an Error status.
- The total number of attempted service calls.
- The total number of successful service calls.
- The total number of service calls that produced a timeout.

The Bytes Summary section displays the following information:

- The minimum, maximum, and average number of bytes sent and received for each call in the run.
- The byte rate per second for the run.
- The total number of bytes sent and received for the run.

Response Time Results page

The Response Time Results page shows the average response of the service calls in the test as the test progresses. With this information, you can evaluate system response during and after the test. The delay between the moment a service call is invoked and the moment the corresponding message return is received, determines the *Response times*.

The bar chart shows the average response time of each service call. Each bar represents a service call that was invoked during the test. As you run the test, the bar chart changes, because the response times are updated dynamically during the run.

The table that follows the bar chart provides the following additional information for each service call:

- The minimum response time during the run.
- The average response time during the run. This matches the information in the chart.
- The maximum response time during the run.
- The standard deviation response time during the run.

Response Time vs. Time Summary page

The Response vs. Time Summary page shows the average response trend as graphed for a specified interval. You set the **Statistics sample interval** value in the schedule, as a schedule property. Measurements that are located in the tests determine the *Response times*. Response time measurements can be automatically generated between the last input action before a service call and the corresponding message return event.

The line graph shows the average response time for all measurements during the run. Each point on the graph is an average of what has occurred during that interval. The table that follows the graph lists one number: the total average response time for all measurements in the run.

Response Time vs. Time Details page

The Response vs. Time Details page shows the response trend as graphed for a specified interval. You set the **Statistics sample interval** value in the schedule, as a schedule property. The delay between the moment a service call is invoked and the moment the corresponding message return is received determines the *Response times*.

The line graph shows the average response time of each measurement for a specified interval. A separate line represents each measurement.

The table under the graph provides the following additional information for each response time measurement:

- The minimum response time during the run.
- The average service call response time during the run. This is similar to the graph, but the information in the table includes the entire run.
- The maximum service call response time during the run.
- The standard deviation service call response time during the run.

Data Volume page

The Data Volume page provides details about the volume of data that is sent to and received from the service. You set the **Statistics sample interval** value in the schedule, as a schedule property.

- The **Sent and Received** line graph shows the total bytes sent and received per interval.
- The **Received Summary** table lists, for each call, the received volume rate (bytes per second) for the entire run, the minimum and maximum received bytes per interval, and the average number of bytes received for each call.
- The **Sent Summary** table lists, for each call, the sent volume rate (bytes per second) for the entire run, the minimum and maximum sent bytes per interval, and the average number of bytes sent for each call.

Call Throughput page

The Call Throughput page provides an overview of the frequency of service calls that are being transferred per interval. You set the **Statistics sample interval** value in the schedule, as a schedule property.


- The line graph shows the calls that are started and ended per interval. Ended calls can be: *success*, *fail*, or *timeout*.
- The **Performance Summary** table lists the details of the number of call starts, successes, failures or timeouts for each call and for the run.

Resources page

The **Resources** page shows information about all the resource counters that were monitored during the *schedule* run.

The **Resources** page displays the following information as mentioned in the table:

If...	Then the Resources page displays...
If you did not add any Resource Monitoring source to a performance schedule	A message that states that you must set up the Resource Monitoring sources to view the resource details.
If you added Resource Monitoring sources to a performance schedule	<ul style="list-style-type: none"> • The Resource Monitoring sources that were monitored during the schedule run. • All resource counters for those Resource Monitoring sources that were monitored during the schedule run. • The Unavailable sources section that lists the Resource Monitoring sources that were unavailable or unreachable during the schedule run.

If...	Then the Resources page displays...
	 Note: The Unavailable sources section is displayed only if any of the Resource Monitoring sources were unreachable or unavailable during the schedule run.
<p>If you added Resource Monitoring sources by using labels to a performance schedule</p>	<ul style="list-style-type: none"> • The following information in the Server sources matching the labels set in the schedule section: <ul style="list-style-type: none"> ◦ Labels and the Resource Monitoring sources associated with those labels that were monitored during the schedule run. ◦ Resource monitoring sources that were unavailable or unreachable during the schedule run. ◦ An empty array (<code>[]</code>) when you used labels that were not tagged to any Resource Monitoring source in HCL OneTest™ Server. • All resource counters for the Resource Monitoring sources that were monitored during the schedule run.
<p>If you ran a performance schedule by using the <code>overrideLabels</code> command from the HCL OneTest™ Performance command line</p>	<ul style="list-style-type: none"> • The following information in the Server sources matching the labels set with the command-line flag <code>-overrideLabels</code> section: <ul style="list-style-type: none"> ◦ Labels that you used to add the Resource Monitoring sources to the schedule for the schedule run. ◦ Resource monitoring sources associated with those labels that were monitored during the schedule run. ◦ Resource monitoring sources that were unavailable or unreachable during the schedule run.

If...	Then the Resources page displays...
	<ul style="list-style-type: none"> ◦ An empty array ([]) when you used labels that were not tagged to any Resource Monitoring source in HCL OneTest™ Server. • All resource counters for the Resource Monitoring sources that were monitored during the schedule run.

The Legend shows the Resource Monitoring type and its resource counters. When you have multiple Resource Monitoring sources, the resource counters for the respective sources are displayed in front of their Resource Monitoring source name. You can customize the resource counter information displayed in a graph by clicking any individual resource counter or type of source. You can click or double-click any individual resource counter for the following results:

- A single click on the resource counter hides the data displayed on the graph. Click the resource counter again to display the data in the graph.
- A double-click on the resource counter removes information about all other resource counters from the graph and displays only the information about the selected resource counter.



Tip: You can click **Select All** option to restore all the resource counter information on the graph.

When you click on any of the sources, the graph removes all the resource counters of other sources and displays only the resource counters of the selected source.

For example, you have an Apache httpd server and a Windows Performance Host as a Resource Monitoring source. When the *schedule* completes, the **Resources** page displays the resource counter information of both the sources. If you want to analyze the resource counters for any one of the sources, you can click the Apache httpd server or the Windows Performance Host. Based on your selection, the graph is updated to show the selected source resource counters information.

The Performance Summary table under the graph lists the most recent values of the resource counters that were monitored during the schedule run. The first two columns show the **Type** of the source and **Name** of the resource counter. This table also lists the minimum, maximum, and average values of the resource counters that were monitored during the schedule run.

Web Service Verification Points report

The web service verification points report shows the status of the verification points in your tests.

Summary page

The **Summary** page displays a bar graph representing the percentage of successful web service calls for the test run. You can set the **Statistics sample interval** value in the schedule, as a schedule property.

The *Verification Point Summary Tab* table lists the following information:

- The number of verification points that were attempted, passed, and failed during the test run.
- The percentage of verification points that passed per interval during the test run.

Verification Points Detail page

The **Verification Points Detail** page displays the details for all types of verification points that were checked during the test run.

The *Verification Points Detail* table lists the following information:

- The number of verification points that passed, failed, caused an error, were inconclusive during the test run.
- The percentage of verification points that passed during the test run.

Response Contain Verification Points page

The **Response Contain Verification Points** page displays the details of contain verification points that were checked during the test run.

The *Response Contain Verification Points* table lists the following information:

- The number of verification points that passed, failed, caused an error, were inconclusive during the test run.
- The percentage of verification points that passed during the test run.

Response Equal Verification Points page

The **Response Equal Verification Points** page displays the details of equal verification points that were checked during the test run.

The *Response Equal Verification Points* table lists the following information:

- The number of verification points that passed, failed, caused an error, were inconclusive during the test run.
- The percentage of verification points that passed during the test run.

Response Properties Verification Points page

The **Response Properties Verification Points** page displays the details of the verification point for the properties that were checked during the test run.

The *Response Properties Verification Points* table lists the following information:

- The number of verification points that passed, failed, caused an error, and were inconclusive during the test run.
- The percentage of verification points that passed during the test run.

Response Query Verification Points page

The **Response Query Verification Points** page displays the details of query verification points that were checked during the test run.

The *Response Query Verification Points* table lists the following information:

- The number of verification points that passed, failed, caused an error, were inconclusive during the test run.
- The percentage of verification points that passed during the test run.

Response XSD Verification Points

The **Response XSD Verification Points** page displays the details of verification points that were checked during the test run.

The *Response XSD Verification Points* table lists the following information:

- The number of verification points passed, failed, caused an error, were inconclusive during the test run.
- The percentage of verification points that passed during the test run.

Response Attachment Verification Points page

The **Response Attachment Verification Points** page displays the details of attachment verification points that were checked during the test run.

The *Response Attachment Verification Points* table lists the following information:

- The number of verification points that passed, failed, caused an error, were inconclusive during the test run.
- The percentage of verification points that passed during the test run.

Response Text Verification Points

The **Response Text Verification Points** page displays the details of verification points that were checked during the test run.

The *Response Text Verification Points* table lists the following information:

- The number of verification points that passed, failed, caused an error during the test run.
- The percentage of verification points that passed during the test run.

Callback Verification Points

The **Callback Verification Points** page displays the details of verification points that were checked during the test run.

The *Callback Verification Points* table lists the following information:

- The number of verification points that passed, failed, caused an error, were inconclusive during the test run.
- The percentage of verification points that passed during the test run.

WSDL Coverage report

The Web Services Description Language (WSDL) Coverage report displays all the bindings, methods, and ports of a web service that were used in the test, compound test, or schedule. In this report, you can see the methods or ports that are not called by the test so that they require more tests for better coverage.

To generate a WSDL Coverage report, in the Test Navigator view, right-click a service test result, and click **WSDL Coverage Report**.

Global Summary

This section displays the number of tests, verification points, and defects available in the report.













WSDL Coverage Summary

This section includes sub-sections for the name of all the compound tests, schedules, and test runs for each WSDL service. For example, if you run a compound test that contains four tests and each test runs a different WSDL service, this section displays a table with four rows for the same compound test but different WSDL services used. Another table displays each test run with the corresponding WSDL service. The tables would display a column for the overall coverage percentage of each WSDL service.

WSDL Coverage Details

This section also includes sub-sections for the names of all the compound tests, schedules, and test runs for each WSDL service. However, this section provides detail coverage information at the bindings, methods, and ports level for the WSDL service. It reports the verdict and the number of requests sent and responses received. The Coverage column of the table indicates whether the methods or ports are called by the test. If the method is called, it shows as 100%. If the method is not called, it shows as 0%. The percentages are then rolled up to the bindings and WSDL service level.

For example, in the following screenshot, there is one binding `BasicHttpBinding_IStockQuoteService` with two methods. The method `GetStockQuote` is called by the test. But, the `GetWorldMajorIndices` method was not called. Therefore, at the binding level, you see 50% coverage. As a user of the web service, if the `GetWorldMajorIndices` method is required, you can check why it was not used by the test.

WSDL	Coverage
▲  StockQuoteService	50 % 
▲  BasicHttpBinding_IStockQuoteService	50 % 
▲  GetStockQuote	100 % 
 BasicHttpBinding_IStockQuoteService	100 % 
▲  GetWorldMajorIndices	0 % 
 BasicHttpBinding_IStockQuoteService	0 % 

Socket performance test reports

When you test a socket application, these performance test reports are produced during a run and saved after a run.

Socket Performance report

The Socket Performance report summarizes the health of the run, displays the data most significant to the run, shows the response trend of the slowest 10 transactions in the test, and graphs the response trend of each transaction for a specified interval, for socket API performance tests.

Overall page

The Overall page summarizes the most important data about the test run, so that you can analyze the final or intermediate results of a test at a glance.

The top of the page displays a progress indicator bar that shows the state of the run.

The Virtual Users Activity bar chart displays the number of virtual users that are active and the number of virtual users that have completed testing. This number is updated during the run.

The Run Summary section displays the following information:

- The name of the test or schedule.
- The number of virtual users that are active and the number of virtual users that have completed testing. This number is updated during the run.
- The total number of virtual users that are simulated during the test.
- The elapsed time. This figure is the run duration, which is displayed in hours, minutes, and seconds.
- The status of the run. This can be Initializing Computers, Adding Users, Running, Performing Execution History Data Transfer, Stopped, or Complete.
- Results for the computer, All Hosts. To see summary results for individual computers, click the computer name in the Performance Test Runs view.

Connect Performance page

The Connect Performance page shows the performance of socket connection actions. With this information, you can evaluate system response during and after the test.

The Connect Actions vs. Time bar chart shows the attempted and successful connection actions during the test run.

The Connect Times vs. Time bar chart shows average, maximum, minimum, and standard deviation of connection times during the test run.

The Connect Actions Summary displays this information:

- The number of connections that were attempted
- The number of connections that were successful

The Connect Times Summary displays the average, maximum, minimum, and standard deviation of connection times.

Send/Receive Performance page

The Send/Receive Performance page shows the data throughput trend as graphed for a specified interval. The line graph shows the average response time for all socket send and receive actions during the run. Each point on the graph is an average of what has occurred during that interval. You set the **Statistics sample interval** in the schedule as a schedule property.

The Send/Receive Actions vs. Time bar chart shows the attempted and successful send and receive actions during the test run.

The Response Times vs. Time bar chart shows average, maximum, minimum, and standard deviation of response times during the test run.

The Connect Actions Summary displays this information:

- The number of send and receive actions that were attempted
- The number of send and receive actions that were successful

The Connect Times Summary displays the average, maximum, minimum, and standard deviation of response times.

Byte Performance page

The Byte Performance page provides an overview of the frequency of requests that are being transferred per interval:


- The Exchanged Bytes vs. Time line graph shows the quantity of bytes sent and received during the test run.
- The Exchanged Bytes summary table after the graph lists the total number of bytes sent and received.

HTTP counters

HTTP counters, which are displayed in the Performance Test Runs view, enable you to customize your reports with dynamic information that is updated for each run.

Byte counters

These counters provide throughput information regarding the rate and the number of bytes sent and received during a sample interval and during a run.

The counters in the following tables provide an aggregate value for all tests. Folders that contain aggregate counters have a clock superimposed on them: .

Counter name	Description
Bytes Received [for Run]	The total number of bytes received for all tests for the entire run
Bytes Sent [for Run]	The total number of bytes sent for all tests for the entire run


Counter name	Description
Count [for Interval]	The total number of bytes received for all tests within the most recent sample interval
Rate [per second] [for Interval]	The bytes per second received for all tests within the most recent sample interval
Rate [per second] [for Run]	The bytes per second received for all tests for the entire run

Counter name	Description
Count [for Interval]	The total number of bytes sent for all tests within the most recent sample interval
Rate [per second] [for Interval]	The bytes per second sent for all tests within the most recent sample interval
Rate [per second] [for Run]	The bytes per second sent for all tests for the entire run


Page counters

These counters provide information about page attempts, page hits, response time, response success, and verification points. Some counters produce an aggregate value, where the values for several pages are rolled up into one value; others produce values for each page.

Aggregate counters use the values for all the pages in a test to produce a single value for a report. This value is rolled up from all values that satisfy the counter. When you drag an aggregate counter onto a report, one value is displayed.

Folders that contain aggregate counters have a clock superimposed on the folder icon: . The tables below that list aggregate counters have this icon in their title.

Individual counters produce values for each item that satisfies the counter, rather than a single rolled-up value.

Folders that contain individual counters have an asterisk superimposed on the folder icon: . The tables below that list individual counters have this icon in their title.

Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

Adjustment counters

The counters in this section provide information about adjustments made to page response times.

Counter name	Description
Average [for Interval]	The average of all adjustments applied during the most recent sample interval
Average [for Run]	The average of all adjustments applied for the entire run

Counter name	Description
Maximum [for Interval]	The maximum adjustment applied during the most recent sample interval
Maximum [for Run]	The maximum adjustment applied for the entire run
Minimum [for Interval]	The minimum adjustment applied during the most recent sample interval
Minimum [for Run]	The minimum adjustment applied for the entire run
Standard Deviation [for Interval]	The standard deviation for adjustments applied within the most recent sample interval
Standard Deviation [for Run]	The standard deviation for adjustments applied for the entire run

The counters in the following table are available only after a run, and only after you have displayed the Page Percentile report.

Counter name	Description
85	85% of the users had a maximum adjustment of this amount applied
90	90% of the users had a maximum adjustment of this amount applied
95	95% of the users had a maximum adjustment of this amount applied

Attempt counters

The counters in this section provide information about *attempts*. When an attempt refers to a page, it means that the primary request was sent; it does not include requests within the page. When an attempt refers to a page element, it means that a request was sent.

Counter name	Description
Count [for Interval]	The number of attempts for a specific page within the most recent sample interval
Count [for Run]	The number of attempts for a specific page for the entire run
Rate [per second] [for Interval]	The rate at which attempts occurred for a specific page during the most recent sample interval
Rate [per second] [for Run]	The rate at which attempts occurred for a specific page for the entire run

Counter name	Description
Count [for Interval]	The total number of page attempts completed for a specific page during the most recent sample interval

Counter name	Description
Count [for Interval]	The total number of page attempts completed for a specific page element for the most recent sample interval

Counter name	Description
Count [for Interval]	The number of attempts for a specific page element within the most recent sample interval
Count [for Run]	The number of attempts for a specific page element for the entire run
Rate [per second] [for Interval]	The rate at which attempts occurred for a specific page element during the most recent sample interval
Rate [per second] [for Run]	The rate at which attempts occurred for a specific page element for the entire run

Counter name	Description
Page Attempt Rate [per second] [for Interval]	The rate at which page attempts occurred within the most recent sample interval
Page Attempt Rate [per second] [for Run]	The rate at which page attempts occurred for the entire run
Page Element Attempt Rate [per second] [for Interval]	The rate at which page element attempts occurred within the most recent sample interval
Page Element Attempt Rate [per second] [for Run]	The rate at which page element attempts occurred for the entire run
Total Page Attempts [for Interval]	The number of page attempts during the most recent sample interval
Total Page Attempts [for Run]	The number of page attempts for the entire run
Total Page Element Attempts [for Interval]	The number of page element attempts within the most recent sample interval
Total Page Element Attempts [for Run]	The number of page element attempts for the entire run

Example

Each counter in the following table provides a single value to the report. This aggregated value has been calculated from all pages.

Hit counters

Counter name	Description
Count [for Interval]	The number of hits for a specific page within the most recent sample interval
Count [for Run]	The number of hits for a specific page for the entire run
Rate [per second] [for Interval]	The rate at which hits occurred for a specific page during the most recent sample interval
Rate [per second] [for Run]	The rate at which hits occurred for a specific page for the entire run

Counter name	Description
Count [for Interval]	The number of hits for a specific page element within the most recent sample interval
Count [for Run]	The number of hits for a specific page element for the entire run
Rate [per second] [for Interval]	The rate at which hits occurred for a specific page element during the most recent sample interval
Rate [per second] [for Run]	The rate at which hits occurred for a specific page element for the entire run

Counter name	Description
Page Element Hit Rate [per second] [for Interval]	The rate of page element hits during the most recent sample interval
Page Element Hit Rate [per second] [for Run]	The rate of page element hits for the entire run
Page Hit Rate [per second] [for Interval]	The rate of page hits during the most recent sample interval
Page Hit Rate [per second] [for Run]	The rate of page hits for the entire run
Total Page Element Hits [for Interval]	The total number of page element hits within the most recent sample interval
Total Page Element Hits [for Run]	The total number of page element hits for the entire run
Total Page Hits [for Interval]	The total number of page hits during the most recent sample interval
Total Page Hits [for Run]	The total number of page hits for the entire run

Response Time counters

Response time counters omit page response times for pages that contain requests with status codes in the range of 4XX (client errors) to 5XX (server errors). The only exception is when the failure (for example, a 404) is recorded and

returned, and the request is not the primary request for the page. Page response times that contain requests that time out are always discarded.

Counter name	Description
Average [for Interval]	The average response time for a specific page within the most recent sample interval
Average [for Run]	The average response time for a specific page for the entire run
Maximum [for Interval]	The maximum response time for a specific page within the most recent sample interval
Maximum [for Run]	The maximum response time for a specific page for the entire run
Minimum [for Interval]	The minimum response time for a specific page within the most recent sample interval
Minimum [for Run]	The minimum response time for a specific page for the entire run
Standard Deviation [for Interval]	The standard deviation for a specific page within the most recent sample interval
Standard Deviation [for Run]	The standard deviation for a specific page for the entire run

Counter name	Description
Average [for Interval]	The average response time for a specific page within the most recent sample interval
Average [for Run]	The average response time for a specific page for the entire run

The counters in the following table are available only after a run, and only after you have displayed the Page Percentile report.

Counter name	Description
85	85% of the users experienced this response time or better on this specific page
90	90% of the users experienced this response time or better on this specific page
95	95% of the users experienced this response time or better on this specific page

The counters in the following table are available only after a run, and only after you have displayed the Page Percentile report.

Counter name	Description
85	85% of the users experienced this response time or better
90	90% of the users experienced this response time or better
95	95% of the users experienced this response time or better

Response time counters omit page response times for pages that contain requests with status codes in the range of 4XX (client errors) to 5XX (server errors). The only exception is when the failure (for example, a 404) is recorded and returned, and the request is not the primary request for the page. Page response times that contain requests that time out are always discarded.

Counter name	Description
Average Response Time for All Page Elements [ms] [for Interval]	The average response time for all page elements within the most recent sample interval
Average Response Time for All Page Elements [ms] [for Run]	The average response time for all page elements for the entire run
Average Response Time for All Pages [ms] [for Interval]	The average response time for all pages within the most recent sample interval
Average Response Time for All Pages [ms] [for Run]	The average response time for all pages for the entire run
Maximum Response Time for All Pages [ms] [for Interval]	The maximum response time for all pages within the most recent sample interval
Maximum Response Time for All Pages [ms] [for Run]	The maximum response time for all pages for the entire run
Minimum Response Time for All Pages [ms] [for Interval]	The minimum response time for all pages within the most recent sample interval
Minimum Response Time for All Pages [ms] [for Run]	The minimum response time for all pages for the entire run
Response Time Standard Deviation for All Page Elements [for Interval]	The standard deviation for all page elements within the most recent sample interval

Counter name	Description
Response Time Standard Deviation for All Page Elements [for Run]	The standard deviation for all page elements for the entire run
Response Time Standard Deviation for All Pages [for Interval]	The standard deviation for all pages within the most recent sample interval
Response Time Standard Deviation for All Pages [for Run]	The standard deviation for all pages for the entire run

Status Code Success counters

A *status code success* means that the response code verification point passed for that request. If the request has no verification points, a success means that the server received a request and returned a response where the status code was in the 200 or 300 category, or returned an expected response in the 400 or 500 category.

Counter name	Description
Count [for Interval]	The number of status code successes for a specific page during the most recent sample interval
Count [for Run]	The number of status code successes for a specific page for the entire run
Percent Status Code Success [for Interval]	The percentage of status code successes for a specific page during the most recent sample interval
Percent Status Code Success [for Run]	The percentage of status code successes for a specific page for the entire run

Counter name	Description
Count [for Interval]	The number of status code successes for a specific page element during the most recent sample interval
Count [for Run]	The number of status code successes for a specific page element for the entire run
Percent Status Code Success [for Interval]	The percentage of status code successes for a specific page element during the most recent sample interval
Percent Status Code Success [for Run]	The percentage of status code successes for a specific page element for the entire run

Counter name	Description
Percent Page Element Status Code Success [for Interval]	The percentage of status code successes for all page elements during the most recent sample interval
Percent Page Element Status Code Success [for Run]	The percentage of status code successes for all page elements for the entire run
Percent Page Status Code Success [for Interval]	The percentage of status code successes for all pages during the most recent sample interval
Percent Page Status Code Success [for Run]	The percentage of status code successes for all pages for the entire run
Total Page Element Status Code Successes [for Interval]	The number of status code successes for all page elements during the most recent sample interval
Total Page Element Status Code Successes [for Run]	The number of status code successes for all page elements for the entire run
Total Page Status Code Successes [for Interval]	The number of status code successes for all pages within the most recent sample interval
Total Page Status Code Successes [for Run]	The number of status code successes for all pages for the entire run

Verification Point counters

The tables in this section provide information about verification points. The verdict for a verification point can be `PASS`, `Fail`, `Error`, `OR Inconclusive`.

- **Pass** indicates that all verification points matched or received the expected response. For example, a response code verification point is set to `PASS` when the recorded response code is received during playback. If your test does not contain verification points, `PASS` means that all primary requests in the test were successful.
- **Fail** indicates that at least one verification point did not match the expected response or that the expected response was not received.
- **Error** indicates one of the following results: a primary request was not successfully sent to the server, no response was received from the server for a primary request, or the primary request response was incomplete or could not be parsed.
- The verdict is set to **Inconclusive** only if you provide custom code that defines a verdict of **Inconclusive**.

Counter name	Description
Percent Pass	The percentage of page title verification points that passed for a specific page for the entire run

Counter name	Description
Percent Pass	The percentage of response code or response size verification points that passed for a specific page element for the entire run

Counter name	Description
Count [for Interval]	The number of response code or response size verification points classified as <code>Error</code> for a specific page element during the most recent sample interval
Count [for Run]	The number of response code or response size verification points classified as <code>Error</code> for a specific page element for the entire run

Counter name	Description
Count [for Interval]	The number of response code or response size verification points that failed for a specific page element during the most recent sample interval
Count [for Run]	The number of response code or response size verification points that failed for a specific page element for the entire run

Counter name	Description
Count [for Interval]	The number of response code or response size verification points classified as <code>Inconclusive</code> for a specific page element during the most recent sample interval
Count [for Run]	The number of response code or response size verification points classified as <code>Inconclusive</code> for a specific page element for the entire run

Counter name	Description
Count [for Interval]	The number of response code or response size verification points that passed for a specific page element during the most recent sample interval
Count [for Run]	The number of response code or response size verification points that passed for a specific page element for the entire run

Counter name	Description
Count [for Interval]	The number of response code or response size verification points classified as <code>ERROR</code> for a specific page during the most recent sample interval
Count [for Run]	The number of response code or response size verification points classified as <code>ERROR</code> for a specific page for the entire run

Counter name	Description
Count [for Interval]	The number of response code or response size verification points that failed for a specific page during the most recent sample interval
Count [for Run]	The number of response code or response size verification points that failed for a specific page for the entire run

Counter name	Description
Count [for Interval]	The number of response code or response size verification points classified as <code>Inconclusive</code> for a specific page during the most recent sample interval
Count [for Run]	The number of response code or response size verification points classified as <code>Inconclusive</code> for a specific page for the entire run

Counter name	Description
Count [for Interval]	The number of response code or response size verification points that passed for a specific page during the most recent sample interval
Count [for Run]	The number of response code or response size verification points that passed for a specific page for the entire run

Counter name	Description
Percent Page Element VPs Passed [for Interval]	The percentage of response code or response size verification points that passed during the most recent sample interval

Counter name	Description
Percent Page Element VPs Passed [for Run]	The percentage of response code or response size verification points that passed for the entire run
Percent Page VPs Passed [for Interval]	The percentage of page title verification points that passed during the most recent sample interval
Percent Page VPs Passed [for Run]	The percentage of page title verification points that passed for the entire run
Total Page Element VPs Attempted [for Interval]	The number of response code or response size verification points executed during the most recent sample interval
Total Page Element VPs Attempted [for Run]	The number of response code or response size verification points executed for the entire run
Total Page Element VPs Error [for Interval]	The number of response code or response size verification points with a verdict of <code>ERROR</code> within the most recent sample interval
Total Page Element VPs Error [for Run]	The number of response code or response size verification points with a verdict of <code>ERROR</code> for the entire run
Total Page Element VPs Failed [for Interval]	The number of response code or response size verification points that failed during the most recent sample interval
Total Page Element VPs Failed [for Run]	The number of response code or response size verification points that failed for the entire run
Total Page Element VPs Inconclusive [for Interval]	The number of response code or response size verification points that were marked as <code>Inconclusive</code> within the most recent sample interval
Total Page Element VPs Inconclusive [for Run]	The number of response code or response size verification points that were marked as <code>Inconclusive</code> for the entire run
Total Page Element VPs Passed [for Interval]	The number of response code or response size verification points that passed during the most recent sample interval
Total Page Element VPs Passed [for Run]	The number of response code or response size verification points that passed for the entire run
Total Page VPs Attempted [for Interval]	The percentage of page title verification points that were executed during the most recent sample interval
Total Page VPs Attempted [for Run]	The percentage of page title verification points that were executed for the entire run
Total Page VPs Error [for Interval]	The percentage of page title verification points with a verdict of <code>ERROR</code> during the most recent sample interval
Total Page VPs Error [for Run]	For each page that contains verification points, the total with a verdict of <code>ERROR</code> for the entire run

Counter name	Description
Total Page VPs Failed [for Interval]	The number of page title verification points that failed during the most recent sample interval
Total Page VPs Failed [for Run]	The number of page title verification points that failed for the entire run
Total Page VPs Inconclusive [for Interval]	The number of page title verification points that were marked as <i>Inconclusive</i> during the most recent sample interval
Total Page VPs Inconclusive [for Run]	The number of page title verification points that were marked as <i>Inconclusive</i> for the entire run
Total Page VPs Passed [for Interval]	The number of page title verification points that passed during the most recent sample interval
Total Page VPs Passed [for Run]	The number of page title verification points that passed for the entire run

Run counters

These counters provide information about the active users in the run, the users that have completed the run, and the HTTP status codes that were received.

Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule as a schedule property.

Run

Counter name	Description
Displaying Results for computer	The name of the host.
Run Status	The run status or errors.

> Active Users

Counter name	Description
Count [for Run]	The number of users that are currently active.

Counter Name	Description
Count [for Interval]	The length of time it took to collect all the statistics to send for the most recent sample interval. Typically a brief period. This value has limited use in a custom report.

Counter name	Description
Count [for Run]	The number of users that have completed the run.

Counter Name	Description
Count [for Inter- val]	The number of HTTP status codes between 100 and 199 that were received during the most recent sample interval.
Count [for Run]	The number of HTTP status codes between 100 and 199 that were received in the entire run.

Counter name	Description
Count [for Inter- val]	The number of HTTP status codes between 200 and 299 that were received during the most recent sample interval.
Count [for Run]	The number of HTTP status codes between 200 and 299 that were received in the entire run.

Counter name	Description
Count [for Inter- val]	The number of HTTP status codes between 300 and 399 that were received during the most recent sample interval.
Count [for Run]	The number of HTTP status codes between 300 and 399 that were received in the entire run.

Counter name	Description
Count [for Inter- val]	The number of HTTP status codes between 400 and 499 that were received during the most recent sample interval.
Count [for Run]	The number of HTTP status codes between 400 and 499 that were received in the entire run.

Counter name	Description
Count [for Inter- val]	The number of HTTP status codes between 500 and 599 that were received during the most recent sample interval.
Count [for Run]	The number of HTTP status codes between 500 and 599 that were received in the entire run.

Counter name	Description
Count [for Run]	The duration of the run, in milliseconds (ms).

Counter name	Description
Elapsed Time [H:M:S]	The duration of the run, in hours, minutes, and seconds.

Run > Sample Interval Length

Counter name	Description
Count [for Run]	The length of the sample interval.

Counter name	Description
Executed Test	The web address of the schedule or test that was run.


Counter name	Description
Count [for Run]	The clock value at the start of the run, in milliseconds (ms).

Counter Name	Description
Count [for Run]	The total number of users that were involved in the run.

Test counters


These counters provide information about the execution time of the tests in a run. Some counters produce an aggregate value, where the values for all tests are rolled up into one value; others produce individual values for each test.

Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

The counters in the following table provide an aggregate value for all tests. Folders that contain aggregate counters have a clock superimposed on them: .

Counter name	Description
Average [for Interval]	The average execution time for all tests within the most recent sample interval
Average [for Run]	The average execution time for all tests in the entire run

Counter name	Description
Maximum [for Interval]	The maximum execution time for all tests within the most recent sample interval
Maximum [for Run]	The maximum execution time for all tests in the entire run
Minimum [for Interval]	The minimum execution time for all tests within the most recent sample interval
Minimum [for Run]	The minimum execution time for all tests in the entire run
Standard Deviation [for Interval]	The standard deviation for all tests within the most recent sample interval
Standard Deviation [for Run]	The standard deviation for all tests in the entire run


The counters in the following table provide individual values for each test. Folders that contain individual counters have an asterisk superimposed on them: .

Counter name	Description
Average Execution Time for All Tests [ms] [for Interval]	The average execution time for each test within the most recent sample interval
Average Execution Time for All Tests [ms] [for Run]	The average execution time for each test in the entire run
Execution Time Standard Deviation for All Tests [for Interval]	The standard deviation for each test within the most recent sample interval
Execution Time Standard Deviation for All Tests [for Run]	The standard deviation for each test in the entire run
Maximum Execution Time for All Tests [ms] [for Interval]	The maximum execution time for each test within the most recent sample interval
Maximum Execution Time for All Tests [ms] [for Run]	The maximum execution time for each test in the entire run
Minimum Execution Time for All Tests [ms] [for Interval]	The minimum execution time for each test within the most recent sample interval
Minimum Execution Time for All Tests [ms] [for Run]	The minimum execution time for each test in the entire run


Transaction counters

These counters provide information about transactions that were attempted, transactions that were completed, and the elapsed time for the transactions. Some counters produce an aggregate value, where the values for all transactions are rolled up into one value; others produce individual values for each transaction.


Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

The counters in the following table provide an aggregate value for all transactions. Folders that contain aggregate counters have a clock superimposed on them: .

Counter name	Description
Count [for Interval]	The total number of transactions that were attempted within the last recorded interval
Count [for Run]	The total number of transactions that were attempted in the entire run
Rate [per second] [for Interval]	The number of transactions that were attempted per second in the last recorded interval
Rate [per second] [for Run]	The number of transactions that were attempted per second in the entire run

The counters in the following table provide individual values for each transaction. Folders that contain individual counters have an asterisk superimposed on them: .

Counter name	Description
Total Transactions Started [for Interval]	The number of transactions that were started within the last recorded interval
Transaction Start Rate [per second] [for Interval]	The rate that transactions that were started within the last recorded interval
Transaction Start Rate [per second] [for Run]	The rate that transactions that were started in the entire run

The counters in the following table provide an aggregate value for all transactions. Folders that contain aggregate counters have a clock superimposed on them: .

Counter name	Description
Count [for Interval]	The total number of transactions that were completed within the last recorded interval
Count [for Run]	The total number of transactions that were completed in the entire run
Rate [per second] [for Interval]	The number of transactions that were completed per second in the last recorded interval
Rate [per second] [for Run]	The number of transactions that were completed per second in the entire run

The counters in the following table provide individual values for each transaction. Folders that contain individual counters have an asterisk superimposed on them: 📁*.

Counter name	Description
Total Transactions Completed [for Run]	The total number of transactions that were completed
Transaction Completion Rate [per second] [for Interval]	The rate that transactions were completed in the last recorded interval
Transaction Completion Rate [per second] [for Run]	The rate that transactions were completed in the entire run

The counters in the following table provide an aggregate value for all transactions. Folders that contain aggregate counters have a clock superimposed on them: 🕒.

Counter name	Description
Average [for Interval]	The average elapsed time for all transactions within the most recent sample interval
Average [for Run]	The average elapsed time for all transactions in the entire run
Maximum [for Interval]	The maximum elapsed time for all transactions within the most recent sample interval
Maximum [for Run]	The maximum elapsed time for all transactions in the entire run
Minimum [for Interval]	The minimum elapsed time for all transactions within the most recent sample interval
Minimum [for Run]	The minimum elapsed time for all transactions in the entire run
Standard deviation [for Interval]	The standard deviation for all transactions within the most recent sample interval
Standard deviation [for Run]	The standard deviation for all transactions for the entire run

The counters in the following table provide individual values for each transaction. Folders that contain individual counters have an asterisk superimposed on them: 📁*.

Counter name	Description
Average Elapsed Time for All Transactions [ms] [for Interval]	The average elapsed time for each transaction within the most recent sample interval
Average Elapsed Time for All Transactions [ms] [for Run]	The average elapsed time for each transaction in the entire run

Counter name	Description
Elapsed Time Standard Deviation for All Transactions [for Interval]	The standard deviation for each transaction within the most recent sample interval
Elapsed Time Standard Deviation for All Transactions [for Run]	The standard deviation for each transaction in the entire run
Maximum Elapsed Time for All Transactions [ms] [for Interval]	The maximum elapsed time for each transaction within the most recent sample interval
Maximum Elapsed Time for All Transactions [ms] [for Run]	The maximum elapsed time for each transaction in the entire run
Minimum Elapsed Time for All Transactions [ms] [for Interval]	The minimum elapsed time for each transaction within the most recent sample interval
Minimum Elapsed Time for All Transactions [ms] [for Run]	The minimum elapsed time for each transaction in the entire run

Recording with Internet Protocol v6

To record Internet Protocol version 6 traffic, set the HTTP recording preferences to **Firefox** and select **Record using IPv6**.

1. Click **Windows > Preferences > Test > RPT HTTP recording**.
2. In **Application to record**, select **Firefox**.
You cannot use Internet Explorer to record IPv6.
3. Select **Record using IPv6**.

Result

You are now ready to record IPv6 traffic.

Changing HTTP recording preferences

You can change the behavior of the recorder by changing the preference settings. The default settings, however, are appropriate for recording under Windows® or Linux® systems.

1. Open the HTTP Recording page. Click **Window > Preferences > Test > Recording > Browsers Recording**.
2. Select the setting to change.

Enable the HCL OneTest™ Performance toolbar in browsers

Click to install the annotation toolbar. This enables you to add comments and transactions, and to change page names during recording.

Verify annotation toolbar is installed before recording

Click to verify that the annotation toolbar is installed in the web browser before recording.


3. After changing a setting, click **Apply**.


SAP counters

SAP generic counters, which are displayed in the Performance Test Runs view, enable you to customize your SAP reports with dynamic information that is updated for each run.

SAP screen counters

These counters provide information about screens started, screens completed, response times, and verification points. Some counters produce an aggregate value, where the values for several screens are rolled up into one value; others produce values for each screen.

Aggregate counters use the values for all the screens in a test to produce a single value for a report. This value is rolled up from all values that satisfy the counter. When you drag an aggregate counter onto a report, one value is displayed. Folders that contain aggregate counters have a clock superimposed on the folder icon: . The tables below that list aggregate counters have this icon in their title.

Individual counters produce values for each item that satisfies the counter, rather than a single rolled-up value. Folders that contain individual counters have an asterisk superimposed on the folder icon: . The tables below that list individual counters have this icon in their title.

Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

SAP Screen Request Response Time counters

The counters in this section provide information about the SAP application response time. This is the time that elapses from the point that the SAP GUI client sends a request and the moment the response is received and displayed by the SAP GUI client.

Counter name	Description
Average [for Interval]	The average response time for a specific SAP screen during the most recent sample interval
Average [for Run]	The average response time for a specific SAP screen for the entire run
Maximum [for Interval]	The maximum response time for a specific SAP screen during the most recent sample interval
Maximum [for Run]	The maximum response time for a specific SAP screen for the entire run
Minimum [for Interval]	The minimum response time for a specific SAP screen during the most recent sample interval
Minimum [for Run]	The minimum response time for a specific SAP screen for the entire run
Standard Deviation [for Interval]	The standard deviation of response times for a specific SAP screen during the most recent sample interval

Counter name	Description
Standard Deviation [for Run]	The standard deviation of response times for a specific SAP screen for the entire run

Counter name	Description
Average Screen Request Response Time for All SAP Screens [ms] [for Interval]	The average response time for all SAP screens within the most recent sample interval
Average Screen Request Response Time for All SAP Screens [ms] [for Run]	The average response time for all SAP screens for the entire run
Maximum Screen Request Response Time for All SAP Screens [ms] [for Interval]	The maximum response time for all SAP screens within the most recent sample interval
Maximum Screen Request Response Time for All SAP Screens [ms] [for Run]	The maximum response time for all SAP screens for the entire run
Minimum Screen Request Response Time for All SAP Screens [ms] [for Interval]	The minimum response time for all SAP screens within the most recent sample interval
Minimum Screen Request Response Time for All SAP Screens [ms] [for Run]	The minimum response time for all SAP screens for the entire run
Standard Deviation Screen Request Response Time for All SAP Screens [ms] [for Interval]	The standard deviation of response times for all SAP screens during the most recent sample interval
Standard Deviation Screen Request Response Time for All SAP Screens [ms] [for Run]	The standard deviation of response times for all SAP screens for the entire run

SAP Screen Request Interpretation Time counters

The counters in this section provide information about the SAP GUI interpretation time. This is the duration from the time that the SAP GUI client receives data from the server and the moment when this data is displayed. This measures the health of the SAP GUI client used for the test rather than the performance of the SAP server.

Counter name	Description
Average [for Interval]	The average interpretation time for a specific SAP screen during the most recent sample interval
Average [for Run]	The average interpretation time for a specific SAP screen for the entire run
Maximum [for Interval]	The maximum interpretation time for a specific SAP screen during the most recent sample interval
Maximum [for Run]	The maximum interpretation time for a specific SAP screen for the entire run

Counter name	Description
Minimum [for Interval]	The minimum interpretation time for a specific SAP screen during the most recent sample interval
Minimum [for Run]	The minimum interpretation time for a specific SAP screen for the entire run
Standard Deviation [for Interval]	The standard deviation of interpretation times for a specific SAP screen during the most recent sample interval
Standard Deviation [for Run]	The standard deviation of interpretation times for a specific SAP screen for the entire run

Counter name	Description
Average Screen Request Interpretation Time for All SAP Screens [ms] [for Interval]	The average interpretation time for all SAP screens during the most recent sample interval
Average Screen Request Interpretation Time for All SAP Screens [ms] [for Run]	The average interpretation time for all SAP screens for the entire run
Maximum Screen Request Interpretation Time for All SAP Screens [ms] [for Interval]	The maximum interpretation time for all SAP screens during the most recent sample interval
Maximum Screen Request Interpretation Time for All SAP Screens [ms] [for Run]	The maximum interpretation time for all SAP screens for the entire run
Minimum Screen Request Interpretation Time for All SAP Screens [ms] [for Interval]	The minimum interpretation time for all SAP screens during the most recent sample interval
Minimum Screen Request Interpretation Time for All SAP Screens [ms] [for Run]	The minimum interpretation time for all SAP screens for the entire run
Standard Deviation Screen Request Interpretation Time for All SAP Screens [ms] [for Interval]	The standard deviation of interpretation times for all SAP screens during the most recent sample interval
Standard Deviation Screen Request Interpretation Time for All SAP Screens [ms] [for Run]	The standard deviation of interpretation times for all SAP screens for the entire run

SAP Screens Started counters

The counters in this section provide information about screens that are started.

Counter name	Description
Count [for Interval]	The number of started screens for a specific SAP screen during the most recent sample interval

Counter name	Description
Count [for Run]	The number of started screens for a specific SAP screen for the entire run
Rate [per second] [for Interval]	The rate at which screens were started for a specific SAP screen during the most recent sample interval
Rate [per second] [for Run]	The rate at which screens were started for a specific SAP screen for the entire run

Counter name	Description
SAP Screens Started Rate [for Interval]	The rate at which SAP screens were started within the most recent sample interval
SAP Screens Started Rate [for Run]	The rate at which SAP screens were started for the entire run
Total SAP Screens Started [for Interval]	The number of started SAP screens for the entire run
Total SAP Screens Started [for Run]	The number of started SAP screens during the most recent sample interval

SAP Screens Completed counters

The counters in this section provide information about SAP screens that were completed.

Counter name	Description
Count [for Interval]	The number of completed screens for a specific SAP screen during the most recent sample interval
Count [for Run]	The number of completed screens for a specific SAP screen for the entire run
Rate [per second] [for Interval]	The rate at which screens were completed for a specific SAP screen during the most recent sample interval
Rate [per second] [for Run]	The rate at which screens were completed for a specific SAP screen for the entire run

Counter name	Description
SAP Screens Completed Percent [for Interval]	The percentage of completed SAP screens within the most recent sample interval
SAP Screens Completed Percent [for Run]	The percentage of completed SAP screens for the entire run

Counter name	Description
SAP Screens Completed Rate [for Interval]	The rate at which SAP screens were completed within the most recent sample interval
SAP Screens Completed Rate [for Run]	The rate at which SAP screens were completed for the entire run
Total SAP Screens Completed [for Interval]	The number of completed SAP screens within the most recent sample interval
Total SAP Screens Completed [for Run]	The number of completed SAP screens for the entire run

Verification Point counters

The tables in this section provide information about SAP screen verification points. The verdict for a verification point can be `Pass`, `Fail`, `Error`, or `Inconclusive`.

- `Pass` indicates that the verification point matched or received the expected response. For example, a response code verification point is set to `Pass` when the recorded response code is received during playback. If your test does not contain verification points, it means that the connection succeeded.
- `Fail` indicates that the verification point did not match the expected response or that the expected response was not received.
- `Error` indicates that the primary request was not successfully sent to the server, no response was received from the server, or the response was incomplete or could not be parsed.
- The verdict is set to `Inconclusive` only if you provide custom code that defines a verdict of `Inconclusive`.

Counter name	Description
Percent Pass	The percentage of verification points that passed for a specific SAP screen for the entire run

Counter name	Description
Count [for Interval]	The number of verification points classified as <code>Error</code> for a specific SAP screen during the most recent sample interval
Count [for Run]	The number of verification points classified as <code>Error</code> for a specific SAP screen for the entire run

Counter name	Description
Count [for Interval]	The number of verification points that failed for a specific SAP screen during the most recent sample interval
Count [for Run]	The number of verification points that failed for a specific SAP screen for the entire run

Counter name	Description
Count [for Interval]	The number of verification points classified as <i>Inconclusive</i> for a specific SAP screen during the most recent sample interval
Count [for Run]	The number of verification points classified as <i>Inconclusive</i> for a specific SAP screen for the entire run

Counter name	Description
Count [for Interval]	The number of verification points that passed for a specific SAP screen during the most recent sample interval
Count [for Run]	The number of verification points that passed for a specific SAP screen for the entire run

Counter name	Description
Percent SAP Screens VPs Passed [for Interval]	The percentage of verification points that passed within the most recent sample interval
Percent SAP Screens VPs Passed [for Run]	The percentage of verification points that passed for the entire run
Total SAP Screens VPs Attempted [for Interval]	The number of verification points executed within the most recent sample interval
Total SAP Screens VPs Attempted [for Run]	The number of verification points executed for the entire run
Total SAP Screens VPs Error [for Interval]	The number of verification points with a verdict of <i>Error</i> within the most recent sample interval
Total SAP Screens VPs Error [for Run]	The number of verification points with a verdict of <i>Error</i> for the entire run
Total SAP Screens VPs Failed [for Interval]	The number of verification points that failed within the most recent sample interval
Total SAP Screens VPs Failed [for Run]	The number of verification points that failed for the entire run
Total SAP Screens VPs Inconclusive [for Interval]	The number of verification points that were marked as <i>Inconclusive</i> within the most recent sample interval
Total SAP Screens VPs Inconclusive [for Run]	The number of verification points that were marked as <i>Inconclusive</i> for the entire run

Counter name	Description
Total SAP Screens VPs Passed [for Interval]	The number of verification points that passed within the most recent sample interval
Total SAP Screens VPs Passed [for Run]	The number of verification points that passed for the entire run

SAP Elements Attempted counters

The counters in this section provide information about SAP set elements, SAP get elements or SAP calls that were attempted.

Counter name	Description
Count [for Interval]	The number of attempted elements for a specific SAP screen during the most recent sample interval
Count [for Run]	The number of attempted elements for a specific SAP screen for the entire run
Rate [per second] [for Interval]	The rate at which elements were attempted for a specific SAP screen within the most recent sample interval
Rate [per second] [for Run]	The rate at which elements were attempted for a specific SAP screen for the entire run

Counter name	Description
SAP Elements Attempted Rate [for Interval]	The rate at which SAP elements were attempted within the most recent sample interval
SAP Elements Attempted Rate [for Run]	The rate at which SAP elements were attempted for the entire run
Total SAP Elements Attempted [for Interval]	The number of attempted SAP elements for the entire run
Total SAP Elements Attempted [for Run]	The number of attempted SAP elements within the most recent sample interval

SAP Elements Completed counters

The counters in this section provide information about SAP set elements, SAP get elements or SAP calls that terminated normally. If an element does not complete, it is counted in the SAP Elements Attempted, but not in SAP Elements Complete.

Counter name	Description
Count [for Interval]	The number of completed elements for a specific SAP screen during the most recent sample interval
Count [for Run]	The number of completed elements for a specific SAP screen for the entire run
Rate [per second] [for Interval]	The rate at which elements were completed for a specific SAP screen during the most recent sample interval
Rate [per second] [for Run]	The rate at which elements were completed for a specific SAP screen for the entire run

Counter name	Description
SAP Elements Completed Percent [for Interval]	The percentage of completed SAP elements within the most recent sample interval
SAP Elements Completed Percent [for Run]	The percentage of completed SAP elements for the entire run
SAP Elements Completed Rate [for Interval]	The rate at which SAP elements were completed within the most recent sample interval
SAP Elements Completed Rate [for Run]	The rate at which SAP elements were completed for the entire run
Total SAP Elements Completed [for Interval]	The number of completed SAP elements within the most recent sample interval
Total SAP Elements Completed [for Run]	The number of completed SAP elements for the entire run



Citrix counters

Citrix generic counters, which are displayed in the Performance Test Runs view, enable you to customize your Citrix reports with dynamic information that is updated for each run.


Citrix window synchronization counters

Citrix window synchronization counters provide information about window synchronizations that were attempted, completed, or produced a timeout within the specified timeout limit. Window synchronization is the mechanism used by the test to compare windows from the test with actual windows in the run.


Some counters produce an aggregate value, where the values for several web service message returns are combined into a single value; others produce values for each web service message return.

- *Aggregate* counters use the values for all the elements in a test to produce a single value for a report. This value is rolled up from all values that satisfy the counter. When you drag an aggregate counter onto a report, one value is displayed. Folders that contain aggregate counters have a clock superimposed on the folder icon: . In the tables that follow, the aggregate counters are listed after the titles that contain this icon.
- *Individual* counters produce values for each item that satisfies the counter, rather than an aggregate value. Folders that contain individual counters have an asterisk superimposed on the folder icon: . In the tables that follow, the individual counters are listed after the titles that contain this icon.


Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

The counters in the following table provide an aggregate value for all window synchronizations. Folders that contain aggregate counters have a clock superimposed on them: .

Counter name	Description
Count [for Interval]	The total number of window synchronizations that were attempted within the last recorded interval
Count [for Run]	The total number of window synchronizations that were attempted in the entire run

The counters in the following table provide an aggregate value for all window synchronizations. Folders that contain aggregate counters have a clock superimposed on them: .

Counter name	Description
Count [for Interval]	The total number of window synchronizations that succeeded within the last recorded interval
Count [for Run]	The total number of window synchronizations that succeeded in the entire run
Percent Citrix Synchronization Success for Interval	The percentage of window synchronizations that succeeded within the last recorded interval
Percent Citrix Synchronization Success for Run	The percentage of window synchronizations that succeeded in the entire run



The counters in the following table provide an aggregate value for all transactions. Folders that contain aggregate counters have a clock superimposed on them: .

Counter name	Description
Count [for Interval]	The total number of window synchronizations that produced a timeout within the last recorded interval
Count [for Run]	The total number of window synchronizations that produced a timeout in the entire run


Citrix image synchronization counters

Citrix image synchronization counters provide information about image synchronizations that were attempted, completed, or produced a timeout within the specified timeout limit. Image synchronization is the mechanism used by the test to compare image areas from the test with actual image area in the run.


Some counters produce an aggregate value, where the values for several web service message returns are combined into a single value; others produce values for each web service message return.

- *Aggregate* counters use the values for all the elements in a test to produce a single value for a report. This value is rolled up from all values that satisfy the counter. When you drag an aggregate counter onto a report, one value is displayed. Folders that contain aggregate counters have a clock superimposed on the folder icon: . In the tables that follow, the aggregate counters are listed after the titles that contain this icon.
- *Individual* counters produce values for each item that satisfies the counter, rather than an aggregate value. Folders that contain individual counters have an asterisk superimposed on the folder icon: . In the tables that follow, the individual counters are listed after the titles that contain this icon.


Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

The counters in the following table provide an aggregate value for all image synchronizations. Folders that contain aggregate counters have a clock superimposed on them: .

Counter name	Description
Count [for Interval]	The total number of image synchronizations that were attempted within the last recorded interval
Count [for Run]	The total number of image synchronizations that were attempted in the entire run

The counters in the following table provide an aggregate value for all image synchronizations. Folders that contain aggregate counters have a clock superimposed on them: .

Counter name	Description
Count [for Interval]	The total number of image synchronizations that succeeded within the last recorded interval
Count [for Run]	The total number of image synchronizations that succeeded in the entire run
Percent Citrix Synchronization Success for Interval	The percentage of image synchronizations that succeeded within the last recorded interval
Percent Citrix Synchronization Success for Run	The percentage of image synchronizations that succeeded in the entire run

The counters in the following table provide an aggregate value for all transactions. Folders that contain aggregate counters have a clock superimposed on them: .

Counter name	Description
Count [for Interval]	The total number of image synchronizations that produced a timeout within the last recorded interval
Count [for Run]	The total number of image synchronizations that produced a timeout in the entire run

Counter name	Description
Percent Pass	The percentage of image synchronization verification points that passed for a specific window event for the entire run

Counter name	Description
Count [for Interval]	The number of image synchronization verification points classified as <code>ERROR</code> for a specific window event during the most recent sample interval
Count [for Run]	The number of image synchronization verification points classified as <code>ERROR</code> for a specific window for the entire run

Counter name	Description
Count [for Interval]	The number of image synchronization verification points that failed for a specific window event during the most recent sample interval
Count [for Run]	The number of image synchronization verification points that failed for a specific window event for the entire run

Counter name	Description
Count [for Interval]	The number of image synchronization verification points classified as <i>Inconclusive</i> for a specific window event during the most recent sample interval
Count [for Run]	The number of image synchronization verification points classified as <i>Inconclusive</i> for a specific window event for the entire run

Counter name	Description
Count [for Interval]	The number of image synchronization verification points that passed for a specific window event during the most recent sample interval
Count [for Run]	The number of image synchronization verification points that passed for a specific window event for the entire run



Counter name	Description
Percent Image Synchronization VPs Passed For Interval	The percentage of image synchronization verification points that passed during the most recent sample interval
Percent Image Synchronization VPs Passed For Run	The percentage of image synchronization verification points that passed for the entire run
Total Image Synchronization VPs Attempted [for Interval]	The number of image synchronization verification points executed during the most recent sample interval
Total Image Synchronization VPs Attempted [for Run]	The number of image synchronization verification points executed for the entire run
Total Image Synchronization VPs Error [for Interval]	The number of image synchronization verification points with a verdict of <i>Error</i> during the most recent sample interval

Counter name	Description
Total Image Synchronization VPs Error [for Run]	The number of image synchronization verification points with a verdict of <code>ERROR</code> for the entire run
Total Image Synchronization VPs Failed [for Interval]	The number of image synchronization verification points that failed during the most recent sample interval
Total Image Synchronization VPs Failed [for Run]	The number of image synchronization verification points that failed for the entire run
Total Image Synchronization VPs Inconclusive [for Interval]	The number of image synchronization verification points that were marked as <code>Inconclusive</code> within the most recent sample interval
Total Image Synchronization VPs Inconclusive [for Run]	The number of image synchronization verification points that were marked as <code>Inconclusive</code> for the entire run
Total Image Synchronization VPs Passed [for Interval]	The number of image synchronization verification points that passed during the most recent sample interval
Total Image Synchronization VPs Passed [for Run]	The number of image synchronization verification points that passed for the entire run


Citrix timer counters

Citrix timer counters provide information about the response time of window events in a run. *Response times* are determined by measurements that are located in the tests.


Some counters produce an aggregate value, where the values for several web service message returns are combined into a single value; others produce values for each web service message return.

- *Aggregate* counters use the values for all the elements in a test to produce a single value for a report. This value is rolled up from all values that satisfy the counter. When you drag an aggregate counter onto a report, one value is displayed. Folders that contain aggregate counters have a clock superimposed on the folder icon: . In the tables that follow, the aggregate counters are listed after the titles that contain this icon.
- *Individual* counters produce values for each item that satisfies the counter, rather than an aggregate value. Folders that contain individual counters have an asterisk superimposed on the folder icon: . In the tables that follow, the individual counters are listed after the titles that contain this icon.

Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

The counters in the following table provide an aggregate value for all tests. Folders that contain aggregate counters have a clock superimposed on them: .

Counter name	Description
Average Response Time For All Timers For Interval [ms]	The average response time for all measurements within the most recent sample interval
Average Response Time For All Timers For Run [ms]	The average response time for all measurements in the entire run
Maximum Response Time For All Timers For Interval [ms]	The maximum response time for all measurements within the most recent sample interval
Maximum Response Time For All Timers For Run [ms]	The maximum response time for all measurements in the entire run
Minimum Response Time For All Timers For Interval [ms]	The minimum response time for all measurements within the most recent sample interval
Minimum Response Time For All Timers For Run [ms]	The minimum response time for all measurements in the entire run
Standard Deviation Response Time For All Timers For Interval [ms]	The standard deviation response time for all measurements during the most recent sample interval
Standard Deviation Response Time For All Timers For Run [ms]	The standard deviation response time for all measurements in the entire run

The counters in the following table provide individual values for each test. Folders that contain individual counters have an asterisk superimposed on them: .


Counter name	Description
Average [for Interval]	The average response time for all measurements within the most recent sample interval
Average [for Run]	The average response time for all measurements in the entire run
Maximum [for Interval]	The maximum response time for all measurements within the most recent sample interval
Maximum [for Run]	The maximum response time for all measurements in the entire run
Minimum [for Interval]	The minimum response time for all measurements within the most recent sample interval
Minimum [for Run]	The minimum response time for all measurements in the entire run
Standard Deviation [for Interval]	The standard deviation response time for all measurements during the most recent sample interval

Counter name	Description
Standard Deviation [for Run]	The standard deviation response time for all measurements in the entire run


Citrix user action counters

Citrix user action counters provide information about emulated user input actions in the run.

Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

The counters in the following table provide an aggregate value for all transactions. Folders that contain aggregate counters have a clock superimposed on them: .

Counter Name	Description
Count [for Interval]	The total number of user input actions within the most recent sample interval
Count [for Run]	The total number of user input actions in the entire run
Rate [per second] [for Interval]	The rate per second of user input actions within the most recent sample interval
Rate [per second] [for Run]	The rate per second of user input actions in the entire run

The counters in the following table provide individual values for each test. Folders that contain individual counters have an asterisk superimposed on them: .

Counter Name	Description
Total User Actions For Interval	The total number of user input actions within the most recent sample interval
Total User Actions For Run	The total number of user input actions in the entire run
User Action Rate For Interval	The rate per second of user input actions within the most recent sample interval
User Action Rate For Run	The rate per second of user input actions in the entire run



Citrix window counters

Citrix window counters provide information about verification points.

The verdict for a verification point can be `Pass`, `Fail`, `Error`, or `Inconclusive`.

- `Pass` indicates that the verification point matched or received the expected response. For example, a response code verification point is set to `Pass` when the recorded response code is received during the run. If your test does not contain verification points, it means that the connection succeeded.
- `Fail` indicates that the verification point did not match the expected response or that the expected response was not received.
- `Error` indicates that the primary request was not successfully sent to the server, no response was received from the server, or the response was incomplete or could not be parsed.
- The verdict is set to `Inconclusive` only if you provide custom code that defines a verdict of `Inconclusive`.

Some counters produce an aggregate value, where the values for several window events are rolled up into one value; others produce values for each window event.

- *Aggregate* counters use the values for all the elements in a test to produce a single value for a report. This value is rolled up from all values that satisfy the counter. When you drag an aggregate counter onto a report, one value is displayed. Folders that contain aggregate counters have a clock superimposed on the folder icon: . The tables below that list aggregate counters have this icon in their title.
- *Individual* counters produce values for each item that satisfies the counter, rather than a single rolled-up value. Folders that contain individual counters have an asterisk superimposed on the folder icon: . The tables below that list individual counters have this icon in their title.

Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

Counter name	Description
Percent Pass	The percentage of window title verification points that passed for a specific window event for the entire run

Counter name	Description
Count [for Interval]	The number of window title verification points classified as <code>Error</code> for a specific window event within the most recent sample interval
Count [for Run]	The number of window title verification points classified as <code>Error</code> for a specific window for the entire run

Counter name	Description
Count [for Interval]	The number of window title verification points that failed for a specific window event during the most recent sample interval
Count [for Run]	The number of window title verification points that failed for a specific window event for the entire run

Counter name	Description
Count [for Interval]	The number of window title verification points classified as <code>Inconclusive</code> for a specific window event within the most recent sample interval
Count [for Run]	The number of window title verification points classified as <code>Inconclusive</code> for a specific window event for the entire run

Counter name	Description
Count [for Interval]	The number of window title verification points that passed for a specific window event during the most recent sample interval
Count [for Run]	The number of window title verification points that passed for a specific window event for the entire run

Counter name	Description
Percent Window VPs Passed For Interval	The percentage of window title verification points that passed during the most recent sample interval
Percent Window VPs Passed For Run	The percentage of window title verification points that passed for the entire run
Total Window VPs Attempted [for Interval]	The number of window title verification points executed during the most recent sample interval
Total Window VPs Attempted [for Run]	The number of window title verification points executed for the entire run
Total Window VPs Error [for Interval]	The number of window title verification points with a verdict of <code>Error</code> during the most recent sample interval
Total Window VPs Error [for Run]	The number of window title verification points with a verdict of <code>Error</code> for the entire run

Counter name	Description
Total Window VPs Failed [for Interval]	The number of window title verification points that failed within the most recent sample interval
Total Window VPs Failed [for Run]	The number of window title verification points that failed for the entire run
Total Window VPs Inconclusive [for Interval]	The number of window title verification points that were marked as <i>Inconclusive</i> during the most recent sample interval
Total Window VPs Inconclusive [for Run]	The number of window title verification points that were marked as <i>Inconclusive</i> for the entire run
Total Window VPs Passed [for Interval]	The number of window title verification points that passed during the most recent sample interval
Total Window VPs Passed [for Run]	The number of window title verification points that passed for the entire run



Service counters

Service counters, which are displayed in the Test Runs view, enable you to customize your service test reports with dynamic information that is updated for each run.

Service call counters

Service call counters provide information about the calls that are invoked during the service test.

Some counters produce an aggregate value, where the values for several service calls are combined into a single value; others produce values for each service call.

- *Aggregate* counters use the values for all the elements in a test to produce a single value for a report. This value combines all values that satisfy the counter. When you drag an aggregate counter onto a report, one value is displayed. Folders that contain aggregate counters have a clock superimposed on the folder icon: . In the tables that follow, the aggregate counters are listed after the titles that contain this icon.
- *Individual* counters produce values for each item that satisfies the counter, rather than an aggregate value. Folders that contain individual counters have an asterisk superimposed on the folder icon: . In the tables that follow, the individual counters are listed after the titles that contain this icon.

Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

Counter name	Description
Average [for Interval]	The average time, in milliseconds, required to establish a connection to the web service for all web service calls within the most recent sample interval
Average [for Run]	The average time, in milliseconds, required to establish a connection to the web service for all web service calls for the entire run
Standard deviation [for Interval]	The standard deviation time, in milliseconds, required to establish a connection to the web service for all web service calls within the most recent sample interval
Standard deviation [for Run]	The standard deviation time, in milliseconds, required to establish a connection to the web service for all web service calls for the entire run
Maximum [for Interval]	The longest time, in milliseconds, required to establish a connection to the web service for all web service calls within the most recent sample interval
Maximum [for Run]	The longest time, in milliseconds, required to establish a connection to the web service for all web service calls for the entire run
Minimum [for Interval]	The shortest time, in milliseconds, required to establish a connection to the web service for all web service calls within the most recent sample interval
Minimum [for Run]	The shortest time, in milliseconds, required to establish a connection to the web service for all web service calls for the entire run

Counter name	Description
Average connection time [ms] [for Interval]	The average time, in milliseconds, required to establish a connection to the web service for a specific web service call within the most recent sample interval
Average connection time [ms] [for Run]	The average time, in milliseconds, required to establish a connection to the web service for a specific web service call for the entire run
Connection time standard deviation [ms] [for Interval]	The standard deviation time, in milliseconds, required to establish a connection to the web service for a specific web service call during the most recent sample interval
Connection time standard deviation [ms] [for Run]	The standard deviation time, in milliseconds, required to establish a connection to the web service for a specific web service call for the entire run
Maximum connection time [ms] [for Interval]	The longest time, in milliseconds, required to establish a connection to the web service for a specific web service call within the most recent sample interval
Maximum connection time [ms] [for Run]	The longest time, in milliseconds, required to establish a connection to the web service for a specific web service call for the entire run
Minimum connection time [ms] [for Interval]	The shortest time, in milliseconds, required to establish a connection to the web service for a specific web service call within the most recent sample interval

Counter name	Description
Minimum connection time [ms] [for Run]	The shortest time, in milliseconds, required to establish a connection to the web service for a specific web service call for the entire run

Counter name	Description
Average [for Interval]	The average data volume, in bytes, received as a response from the web service for all web service calls during the most recent sample interval
Average [for Run]	The average data volume, in bytes, received as a response from the web service for all web service calls for the entire run
Count [for Interval]	The total data volume, in bytes, received as a response from the web service for all web service calls during the most recent sample interval
Count [for Run]	The total data volume, in bytes, received as a response from the web service for all web service calls for the entire run
Maximum [for Interval]	The largest data volume, in bytes, received as a response from the web service for all web service calls during the most recent sample interval
Maximum [for Run]	The largest data volume, in bytes, received as a response from the web service for all web service calls for the entire run
Maximum Count [for Run]	The largest cumulated data volume, in bytes, received as a response from the web service for all web service calls for the entire run
Minimum [for Interval]	The smallest data volume, in bytes, received as a response from the web service for all web service calls during the most recent sample interval
Minimum [for Run]	The smallest data volume, in bytes, received as a response from the web service for all web service calls for the entire run
Minimum Count [for Run]	The smallest cumulated data volume, in bytes, received as a response from the web service for all web service calls for the entire run
Rate [per second] [for Interval]	The data volume throughput, in bytes per second, received as a response from the web service for all web service calls within the most recent sample interval
Rate [per second] [for Run]	The data volume throughput, in bytes per second, received as a response from the web service for all web service calls for the entire run
Standard deviation [for Interval]	The standard deviation data volume, in bytes, received as a response from the web service for all web service calls within the most recent sample interval
Standard deviation [for Run]	The standard deviation data volume, in bytes, received as a response from the web service for all web service calls for the entire run

Counter name	Description
Average received bytes for all calls [Bytes] [for Interval]	The average data volume, in bytes, received as a response from the web service for all web service calls during the most recent sample interval
Average received bytes for all calls [Bytes] [for Run]	The average data volume, in bytes, received as a response from the web service for all web service calls for the entire run
Maximum received bytes [Bytes] [for Interval]	The largest data volume, in bytes, received as a response from the web service for a specific web service call during the most recent sample interval
Maximum received bytes [per Interval] [for Run]	The largest data volume, in bytes, received as a response from the web service for a specific web service call for the entire run
Maximum received bytes for all calls [Bytes] [for Interval]	The largest data volume, in bytes, received as a response from the web service for all web service calls during the most recent sample interval
Maximum received bytes for all calls [Bytes] [for Run]	The largest data volume, in bytes, received as a response from the web service for all web service calls for the entire run
Minimum received bytes [Bytes] [for Interval]	The smallest data volume, in bytes, received as a response from the web service for a specific web service call during the most recent sample interval
Minimum received bytes [per Interval] [for Run]	The smallest data volume, in bytes, received as a response from the web service for a specific web service call for the entire run
Minimum received bytes for all calls [Bytes] [for Interval]	The smallest data volume, in bytes, received as a response from the web service for a specific web service call during the most recent sample interval
Minimum received bytes for all calls [Bytes] [for Run]	The smallest data volume, in bytes, received as a response from the web service for a specific web service call for the entire run
Rate received bytes [for Interval]	The data volume throughput, in bytes per second, received as a response from the web service for all web service calls within the most recent sample interval
Rate received bytes [for Run]	The data volume throughput, in bytes per second, received as a response from the web service for a specific web service call for the entire run
Received bytes standard deviation [ms] [for Interval]	The standard deviation data volume, in bytes, received as a response from the web service for a specific web service call within the most recent sample interval
Received bytes standard deviation [ms] [for Run]	The standard deviation data volume, in bytes, received as a response from the web service for a specific web service call for the entire run
Total received bytes [Bytes] [for Interval]	The total data volume, in bytes, received as a response from the web service for a specific web service call during the most recent sample interval
Total received bytes [Bytes] [for Run]	The total data volume, in bytes, received as a response from the web service for a specific web service call for the entire run

Counter name	Description
Average [for Interval]	The average data volume, in bytes, sent as a call from the web service for all web service calls during the most recent sample interval
Average [for Run]	The average data volume, in bytes, sent as a call from the web service for all web service calls for the entire run
Count [for Interval]	The total data volume, in bytes, sent as a call from the web service for all web service calls within the most recent sample interval
Count [for Run]	The total data volume, in bytes, sent as a call from the web service for all web service calls for the entire run
Maximum [for Interval]	The largest data volume, in bytes, sent as a call from the web service for all web service calls during the most recent sample interval
Maximum [for Run]	The largest data volume, in bytes, sent as a call from the web service for all web service calls for the entire run
Maximum Count [for Run]	The largest cumulated data volume, in bytes, sent as a call from the web service for all web service calls for the entire run
Minimum [for Interval]	The smallest data volume, in bytes, sent as a call from the web service for all web service calls during the most recent sample interval
Minimum [for Run]	The smallest data volume, in bytes, sent as a call from the web service for all web service calls for the entire run
Minimum Count [for Run]	The smallest cumulated data volume, in bytes, sent as a call from the web service for all web service calls for the entire run
Rate [per second] [for Interval]	The data volume throughput, in bytes per second, sent as a call from the web service for all web service calls during the most recent sample interval
Rate [per second] [for Run]	The data volume throughput, in bytes per second, sent as a call from the web service for all web service calls for the entire run
Standard deviation [for Interval]	The standard deviation data volume, in bytes, sent as a call from the web service for all web service calls during the most recent sample interval
Standard deviation [for Run]	The standard deviation data volume, in bytes, sent as a call from the web service for all web service calls for the entire run

Counter name	Description
Average sent bytes for all calls [Bytes] [for Interval]	The average data volume, in bytes, sent as a call from the web service for all web service calls during the most recent sample interval

Counter name	Description
Average sent bytes for all calls [Bytes] [for Run]	The average data volume, in bytes, sent as a call from the web service for all web service calls for the entire run
Maximum sent bytes [Bytes] [for Interval]	The largest data volume, in bytes, sent as a call from the web service for a specific web service call during the most recent sample interval
Maximum sent bytes [per Interval] [for Run]	The largest data volume, in bytes, sent as a call from the web service for a specific web service call for the entire run
Maximum sent bytes for all calls [Bytes] [for Interval]	The largest data volume, in bytes, sent as a call from the web service for all web service calls during the most recent sample interval
Maximum sent bytes for all calls [Bytes] [for Run]	The largest data volume, in bytes, sent as a call from the web service for all web service calls for the entire run
Minimum sent bytes [Bytes] [for Interval]	The smallest data volume, in bytes, sent as a call from the web service for a specific web service call during the most recent sample interval
Minimum sent bytes [per Interval] [for Run]	The smallest data volume, in bytes, sent as a call from the web service for a specific web service call for the entire run
Minimum sent bytes for all calls [Bytes] [for Interval]	The smallest data volume, in bytes, sent as a call from the web service for a specific web service call during the most recent sample interval
Minimum sent bytes for all calls [Bytes] [for Run]	The smallest data volume, in bytes, sent as a call from the web service for a specific web service call for the entire run
Rate sent bytes [for Interval]	The data volume throughput, in bytes per second, sent as a call from the web service for all web service calls during the most recent sample interval
Rate sent bytes [for Run]	The data volume throughput, in bytes per second, sent as a call from the web service for a specific web service call for the entire run
Sent bytes standard deviation [ms] [for Interval]	The standard deviation data volume, in bytes, sent as a call from the web service for a specific web service call within the most recent sample interval
Sent bytes standard deviation [ms] [for Run]	The standard deviation data volume, in bytes, sent as a call from the web service for a specific web service call for the entire run
Total sent bytes [Bytes] [for Interval]	The total data volume, in bytes, sent as a call from the web service for a specific web service call during the most recent sample interval
Total sent bytes [Bytes] [for Run]	The total data volume, in bytes, sent as a call from the web service for a specific web service call for the entire run

Counter name	Description
Average [for Interval]	The average time, in milliseconds, required to receive a response from the web service for all web service calls within the most recent sample interval
Average [for Run]	The average time, in milliseconds, required to receive a response from the web service for all web service calls for the entire run
Standard deviation [for Interval]	The standard deviation time, in milliseconds, required to receive a response from the web service for all web service calls within the most recent sample interval
Standard deviation [for Run]	The standard deviation time, in milliseconds, required to receive a response from the web service for all web service calls for the entire run
Maximum [for Interval]	The longest time, in milliseconds, required to receive a response from the web service for all web service calls within the most recent sample interval
Maximum [for Run]	The longest time, in milliseconds, required to receive a response from the web service for all web service calls for the entire run
Minimum [for Interval]	The shortest time, in milliseconds, required to receive a response from the web service for all web service calls within the most recent sample interval
Minimum [for Run]	The shortest time, in milliseconds, required to receive a response from the web service for all web service calls for the entire run

Counter name	Description
Average connection time [ms] [for Interval]	The average time, in milliseconds, required to receive a response from the web service for a specific web service call within the most recent sample interval
Average connection time [ms] [for Run]	The average time, in milliseconds, required to receive a response from the web service for a specific web service call for the entire run
Response time standard deviation [ms] [for Interval]	The standard deviation time, in milliseconds, required to receive a response from the web service for a specific web service call during the most recent sample interval
Response time standard deviation [ms] [for Run]	The standard deviation time, in milliseconds, required to receive a response from the web service for a specific web service call for the entire run
Maximum response time [ms] [for Interval]	The longest time, in milliseconds, required to receive a response from the web service for a specific web service call within the most recent sample interval
Maximum response time [ms] [for Run]	The longest time, in milliseconds, required to receive a response from the web service for a specific web service call for the entire run
Minimum response time [ms] [for Interval]	The shortest time, in milliseconds, required to receive a response from the web service for a specific web service call within the most recent sample interval

Counter name	Description
Minimum response time [ms] [for Run]	The shortest time, in milliseconds, required to receive a response from the web service for a specific web service call for the entire run

Counter name	Description
Count [for Interval]	The number of times web service calls have been invoked during the most recent sample interval
Count [for Run]	The number of times web service calls have been invoked during the entire run
Maximum [for Interval]	The maximum number of times web service calls have been invoked within an interval
Maximum [for Run]	The maximum number of times web service calls have been invoked during the entire run
Rate [per second] [for Interval]	The number of times per second that web service calls have been invoked during the most recent sample interval
Rate [per second] [for Run]	The number of times per second that web service calls have been invoked for the entire run

Counter name	Description
Maximum Rate Web Service Call Started [for Run]	The maximum number of times per second that a specific web service call has been invoked for the entire run
Maximum Web Service Call Started [for Interval]	The maximum number of times a specific web service call has been invoked during the most recent sample interval
Maximum Web Service Call Started [for Run]	The maximum number of times a specific web service call has been invoked for the entire run
Minimum Rate Web Service Call Started [for Run]	The minimum number of times per second that a specific web service call has been invoked for the entire run
Minimum Web Service Call Started [for Interval]	The minimum number of times a specific web service call has been invoked during the most recent sample interval
Minimum Web Service Call Started [for Run]	The minimum number of times a specific web service call has been invoked for the entire run
Rate Web Service Call Started [for Interval]	The number of times per second that a specific web service call has been invoked during the most recent sample interval
Rate Web Service Call Started [for Run]	The number of times per second that a specific web service call has been invoked for the entire run

Counter name	Description
Total Web Service Call Started [for Interval]	The number of times that a specific web service call has been invoked during the most recent sample interval
Total Web Service Call Started [for Run]	The number of times that a specific web service call has been invoked for the entire run

Counter name	Description
Count [for Interval]	The number of times web service calls have returned a valid response during the most recent sample interval
Count [for Run]	The number of times web service calls have returned a valid response during the entire run
Maximum [for Interval]	The maximum number of times web service calls have returned a valid response within an interval
Maximum [for Run]	The maximum number of times web service calls have returned a valid response during the entire run
Rate [per second] [for Interval]	The number of times per second that web service calls have returned a valid response during the most recent sample interval
Rate [per second] [for Run]	The number of times per second that web service calls have returned a valid response for the entire run

Counter name	Description
Maximum Rate Web Service Call Success [for Run]	The maximum number of times per second that a specific web service call has returned a valid response for the entire run
Maximum Web Service Call Success [for Interval]	The maximum number of times a specific web service call has returned a valid response during the most recent sample interval
Maximum Web Service Call Success [for Run]	The maximum number of times a specific web service call has returned a valid response for the entire run
Minimum Rate Web Service Call Success [for Run]	The minimum number of times per second that a specific web service call has returned a valid response for the entire run
Minimum Web Service Call Success [for Interval]	The minimum number of times a specific web service call has returned a valid response during the most recent sample interval
Minimum Web Service Call Success [for Run]	The minimum number of times a specific web service call has returned a valid response for the entire run
Rate Web Service Call Success [for Interval]	The number of times per second that a specific web service call has returned a valid response during the most recent sample interval

Counter name	Description
Rate Web Service Call Success [for Run]	The number of times per second that a specific web service call has returned a valid response for the entire run
Total Web Service Call Success [for Interval]	The number of times that a specific web service call has returned a valid response during the most recent sample interval
Total Web Service Call Success [for Run]	The number of times that a specific web service call has returned a valid response for the entire run

Counter name	Description
Count [for Interval]	The number of times web service calls have failed within the most recent sample interval
Count [for Run]	The number of times web service calls have failed during the entire run
Maximum [for Interval]	The maximum number of times web service calls have failed within an interval
Maximum [for Run]	The maximum number of times web service calls have failed during the entire run
Rate [per second] [for Interval]	The number of times per second that web service calls have failed during the most recent sample interval
Rate [per second] [for Run]	The number of times per second that web service calls have failed for the entire run

Counter name	Description
Maximum Rate Web Service Call Failures [for Run]	The maximum number of times per second that a specific web service call has failed for the entire run
Maximum Web Service Call Failures [for Interval]	The maximum number of times a specific web service call has failed during the most recent sample interval
Maximum Web Service Call Failures [for Run]	The maximum number of times a specific web service call has failed for the entire run
Minimum Rate Web Service Call Failures [for Run]	The minimum number of times per second that a specific web service call has failed for the entire run
Minimum Web Service Call Failures [for Interval]	The minimum number of times a specific web service call has failed during the most recent sample interval
Minimum Web Service Call Failures [for Run]	The minimum number of times a specific web service call has failed for the entire run
Rate Web Service Call Failures [for Interval]	The number of times per second that a specific web service call has failed during the most recent sample interval

Counter name	Description
Rate Web Service Call Failures [for Run]	The number of times per second that a specific web service call has failed for the entire run
Total Web Service Call Failures [for Interval]	The number of times that a specific web service call has failed during the most recent sample interval
Total Web Service Call Failures [for Run]	The number of times that a specific web service call has failed for the entire run

Counter name	Description
Count [for Interval]	The number of times web service calls produced a timeout during the most recent sample interval
Count [for Run]	The number of times web service calls produced a timeout during the entire run
Maximum [for Interval]	The maximum number of times web service calls produced a timeout within an interval
Maximum [for Run]	The maximum number of times web service calls produced a timeout during the entire run
Rate [per second] [for Interval]	The number of times per second that web service calls produced a timeout during the most recent sample interval
Rate [per second] [for Run]	The number of times per second that web service calls produced a timeout for the entire run

Counter name	Description
Maximum Rate Web Service Call Timeouts [for Run]	The maximum number of times per second that a specific web service call produces a timeout for the entire run
Maximum Web Service Call Timeouts [for Interval]	The maximum number of times a specific web service call produces a timeout during the most recent sample interval
Maximum Web Service Call Timeouts [for Run]	The maximum number of times a specific web service call produces a timeout for the entire run
Minimum Rate Web Service Call Timeouts [for Run]	The minimum number of times per second that a specific web service call produces a timeout for the entire run
Minimum Web Service Call Timeouts [for Interval]	The minimum number of times a specific web service call produces a timeout during the most recent sample interval
Minimum Web Service Call Timeouts [for Run]	The minimum number of times a specific web service call produces a timeout for the entire run

Counter name	Description
Rate Web Service Call Timeouts [for Interval]	The number of times per second that a specific web service call produces a timeout during the most recent sample interval
Rate Web Service Call Timeouts [for Run]	The number of times per second that a specific web service call produces a timeout for the entire run
Total Web Service Call Timeouts [for Interval]	The number of times that a specific web service call produces a timeout during the most recent sample interval
Total Web Service Call Timeouts [for Run]	The number of times that a specific web service call produces a timeout for the entire run



Service verification point counters

Service verification point counters provide information about verification points.

The verdict for a verification point can be `Pass`, `Fail`, `Error`, or `Inconclusive`.

- `Pass` indicates that the verification point matched or received the expected response. For example, a response code verification point is set to `Pass` when the recorded response code is received during the run. If your test does not contain verification points, it means that the connection succeeded.
- `Fail` indicates that the verification point did not match the expected response or that the expected response was not received.
- `Error` indicates that the primary request was not successfully sent to the server, no response was received from the server, or the response was incomplete or could not be parsed.
- `Inconclusive` is returned only if you provide custom code that defines an `Inconclusive` verdict.

Some counters produce an aggregate value, where the values for several web service message returns are combined into a single value; others produce values for each web service message return.

- *Aggregate* counters use the values for all the elements in a test to produce a single value for a report. This value is rolled up from all values that satisfy the counter. When you drag an aggregate counter onto a report, one value is displayed. Folders that contain aggregate counters have a clock superimposed on the folder icon: . In the tables that follow, the aggregate counters are listed after the titles that contain this icon.
- *Individual* counters produce values for each item that satisfies the counter, rather than an aggregate value. Folders that contain individual counters have an asterisk superimposed on the folder icon: . In the tables that follow, the individual counters are listed after the titles that contain this icon.

Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule, as a schedule property.

Counter name	Description
Percent Pass	The percentage of all verification points that passed for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of all verification points classified as <code>ERROR</code> for a specific web service message return during the most recent sample interval
Count [for Run]	The number of all verification points classified as <code>ERROR</code> for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of all verification points that failed for a specific web service message return during the most recent sample interval
Count [for Run]	The number of all verification points that failed for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of all verification points classified as <code>Inconclusive</code> for a specific web service message return during the most recent sample interval
Count [for Run]	The number of all verification points classified as <code>Inconclusive</code> for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of all verification points that passed for a specific web service message return during the most recent sample interval
Count [for Run]	The number of all verification points that passed for a specific web service message return for the entire run

Counter name	Description
Percent All VPs Passed For Interval	The percentage of all verification points that passed during the most recent sample interval
Percent All VPs Passed For Run	The percentage of all verification points that passed for the entire run
Total All VPs Attempted [for Interval]	The number of all verification points executed within the most recent sample interval
Total All VPs Attempted [for Run]	The number of all verification points executed for the entire run
Total All VPs Error [for Interval]	The number of all verification points with a verdict of <code>ERROR</code> during the most recent sample interval
Total All VPs Error [for Run]	The number of all verification points with a verdict of <code>ERROR</code> for the entire run
Total All VPs Failed [for Interval]	The number of all verification points that failed within the most recent sample interval
Total All VPs Failed [for Run]	The number of all verification points that failed for the entire run
Total All VPs Inconclusive [for Interval]	The number of all verification points that were marked as <code>Inconclusive</code> during the most recent sample interval
Total All VPs Inconclusive [for Run]	The number of all verification points that were marked as <code>Inconclusive</code> for the entire run
Total All VPs Passed [for Interval]	The number of all verification points that passed within the most recent sample interval
Total All VPs Passed [for Run]	The number of all verification points that passed for the entire run

Counter name	Description
Percent Pass	The percentage of attachment verification points that passed for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of attachment verification points classified as <code>ERROR</code> for a specific web service message return during the most recent sample interval

Counter name	Description
Count [for Run]	The number of attachment verification points classified as <code>Error</code> for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of attachment verification points that failed for a specific web service message return during the most recent sample interval
Count [for Run]	The number of attachment verification points that failed for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of attachment verification points classified as <code>Inconclusive</code> for a specific web service message return during the most recent sample interval
Count [for Run]	The number of attachment verification points classified as <code>Inconclusive</code> for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of attachment verification points that passed for a specific web service message return during the most recent sample interval
Count [for Run]	The number of attachment verification points that passed for a specific web service message return for the entire run

Counter name	Description
Percent Attachment VPs Passed For Interval	The percentage of attachment verification points that passed during the most recent sample interval
Percent Attachment VPs Passed For Run	The percentage of attachment verification points that passed for the entire run
Total Attachment VPs Attempted [for Interval]	The number of attachment verification points executed during the most recent sample interval

Counter name	Description
Total Attachment VPs Attempted [for Run]	The number of attachment verification points executed for the entire run
Total Attachment VPs Error [for Interval]	The number of attachment verification points with a verdict of <code>ERROR</code> during the most recent sample interval
Total Attachment VPs Error [for Run]	The number of attachment verification points with a verdict of <code>ERROR</code> for the entire run
Total Attachment VPs Failed [for Interval]	The number of attachment verification points that failed during the most recent sample interval
Total Attachment VPs Failed [for Run]	The number of attachment verification points that failed for the entire run
Total Attachment VPs Inconclusive [for Interval]	The number of attachment verification points that were marked as <code>Inconclusive</code> during the most recent sample interval
Total Attachment VPs Inconclusive [for Run]	The number of attachment verification points that were marked as <code>Inconclusive</code> for the entire run
Total Attachment VPs Passed [for Interval]	The number of attachment verification points that passed during the most recent sample interval
Total Attachment VPs Passed [for Run]	The number of attachment verification points that passed for the entire run

Counter name	Description
Percent Pass	The percentage of contain verification points that passed for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of contain verification points classified as <code>ERROR</code> for a specific web service message return during the most recent sample interval
Count [for Run]	The number of contain verification points classified as <code>ERROR</code> for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of contain verification points that failed for a specific web service message return during the most recent sample interval
Count [for Run]	The number of contain verification points that failed for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of contain verification points classified as <i>Inconclusive</i> for a specific web service message return during the most recent sample interval
Count [for Run]	The number of contain verification points classified as <i>Inconclusive</i> for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of contain verification points that passed for a specific web service message return during the most recent sample interval
Count [for Run]	The number of contain verification points that passed for a specific web service message return for the entire run

Counter name	Description
Percent Contain VPs Passed For Interval	The percentage of contain verification points that passed during the most recent sample interval
Percent Contain VPs Passed For Run	The percentage of contain verification points that passed for the entire run
Total Contain VPs Attempted [for Interval]	The number of contain verification points executed within the most recent sample interval
Total Contain VPs Attempted [for Run]	The number of contain verification points executed for the entire run
Total Contain VPs Error [for Interval]	The number of contain verification points with a verdict of <i>Error</i> during the most recent sample interval

Counter name	Description
Total Contain VPs Error [for Run]	The number of contain verification points with a verdict of <code>Error</code> for the entire run
Total Contain VPs Failed [for Interval]	The number of contain verification points that failed during the most recent sample interval
Total Contain VPs Failed [for Run]	The number of contain verification points that failed for the entire run
Total Contain VPs Inconclusive [for Interval]	The number of contain verification points that were marked as <code>Inconclusive</code> during the most recent sample interval
Total Contain VPs Inconclusive [for Run]	The number of contain verification points that were marked as <code>Inconclusive</code> for the entire run
Total Contain VPs Passed [for Interval]	The number of contain verification points that passed during the most recent sample interval
Total Contain VPs Passed [for Run]	The number of contain verification points that passed for the entire run

Counter name	Description
Percent Pass	The percentage of equal verification points that passed for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of equal verification points classified as <code>Error</code> for a specific web service message return during the most recent sample interval
Count [for Run]	The number of equal verification points classified as <code>Error</code> for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of equal verification points that failed for a specific web service message return during the most recent sample interval
Count [for Run]	The number of equal verification points that failed for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of equal verification points classified as <code>Inconclusive</code> for a specific web service message return during the most recent sample interval
Count [for Run]	The number of equal verification points classified as <code>Inconclusive</code> for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of equal verification points that passed for a specific web service message return during the most recent sample interval
Count [for Run]	The number of equal verification points that passed for a specific web service message return for the entire run

Counter name	Description
Percent Equal VPs Passed For Interval	The percentage of equal verification points that passed during the most recent sample interval
Percent Equal VPs Passed For Run	The percentage of equal verification points that passed for the entire run
Total Equal VPs Attempted [for Interval]	The number of equal verification points executed within the most recent sample interval
Total Equal VPs Attempted [for Run]	The number of equal verification points executed for the entire run
Total Equal VPs Error [for Interval]	The number of equal verification points with a verdict of <code>Error</code> during the most recent sample interval
Total Equal VPs Error [for Run]	The number of equal verification points with a verdict of <code>Error</code> for the entire run
Total Equal VPs Failed [for Interval]	The number of equal verification points that failed during the most recent sample interval
Total Equal VPs Failed [for Run]	The number of equal verification points that failed for the entire run
Total Equal VPs Inconclusive [for Interval]	The number of equal verification points that were marked as <code>Inconclusive</code> during the most recent sample interval

Counter name	Description
Total Equal VPs Inconclusive [for Run]	The number of equal verification points that were marked as <code>Inconclusive</code> for the entire run
Total Equal VPs Passed [for Interval]	The number of equal verification points that passed during the most recent sample interval
Total Equal VPs Passed [for Run]	The number of equal verification points that passed for the entire run

Counter name	Description
Percent Pass	The percentage of query verification points that passed for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of query verification points classified as <code>Error</code> for a specific web service message return during the most recent sample interval
Count [for Run]	The number of query verification points classified as <code>Error</code> for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of query verification points that failed for a specific web service message return during the most recent sample interval
Count [for Run]	The number of query verification points that failed for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of query verification points classified as <code>Inconclusive</code> for a specific web service message return during the most recent sample interval
Count [for Run]	The number of query verification points classified as <code>Inconclusive</code> for a specific web service message return for the entire run

Counter name	Description
Count [for Interval]	The number of query verification points that passed for a specific web service message return during the most recent sample interval
Count [for Run]	The number of query verification points that passed for a specific web service message return for the entire run


Counter name	Description
Percent Query VPs Passed For Interval	The percentage of query verification points that passed during the most recent sample interval
Percent Query VPs Passed For Run	The percentage of query verification points that passed for the entire run
Total Query VPs Attempted [for Interval]	The number of query verification points executed within the most recent sample interval
Total Query VPs Attempted [for Run]	The number of query verification points executed for the entire run
Total Query VPs Error [for Interval]	The number of query verification points with a verdict of <code>ERROR</code> during the most recent sample interval
Total Query VPs Error [for Run]	The number of query verification points with a verdict of <code>ERROR</code> for the entire run
Total Query VPs Failed [for Interval]	The number of query verification points that failed during the most recent sample interval
Total Query VPs Failed [for Run]	The number of query verification points that failed for the entire run
Total Query VPs Inconclusive [for Interval]	The number of query verification points that were marked as <code>Inconclusive</code> during the most recent sample interval
Total Query VPs Inconclusive [for Run]	The number of query verification points that were marked as <code>Inconclusive</code> for the entire run
Total Query VPs Passed [for Interval]	The number of query verification points that passed during the most recent sample interval
Total Query VPs Passed [for Run]	The number of query verification points that passed for the entire run


Socket counters

With socket generic counters, which are displayed in the Performance Test Runs view, you can customize your socket reports with dynamic information that is updated for each run.

Socket counters

These counters provide information about connections started, send and receive actions completed, and connection times. Some counters produce an aggregate value, where the values for several send and receive actions are represented in one value; others produce values for each action.

Aggregate counters use the values for all the connections in a test to produce a single value for a report. This value is calculated from all values that satisfy the counter. When you drag an aggregate counter onto a report, one value is displayed. Folders that contain aggregate counters have a clock superimposed on the folder icon: . The following tables that list aggregate counters have this icon in their title.

Individual counters produce values for each connection that satisfies the counter, rather than a single aggregate value. Folders that contain individual counters have an asterisk superimposed on the folder icon: . The following tables that list individual counters have this icon in their title.

Some counters pertain to *intervals* in the run. You set the **Statistics sample interval** value in the schedule as a schedule property.

Bytes Received counters

The counters in this section provide information about the number of bytes that were received.

Counter name	Description
Count [for Interval]	The number of bytes received for a specific connection during the most recent sample interval
Count [for Run]	The number of bytes received for a specific connection for the entire run

Bytes Sent counters

The counters in this section provide information about the number of bytes that were sent.

Counter name	Description
Count [for Interval]	The number of bytes sent for a specific connection within the most recent sample interval
Count [for Run]	The number of bytes sent for a specific connection for the entire run

Connect Attempts counters

The counters in this section provide information about the number of attempts to establish a connection with the server.

Counter name	Description
Count [for Interval]	The number of attempts to establish a connection within the most recent sample interval
Count [for Run]	The number of attempts to establish a connection for the entire run

Connect Times counters

The counters in this section provide information about the connection response time. This is the time, in milliseconds, that elapses from the point that the test client attempts to establish a connection with the server and the moment the connection is established.

Counter name	Description
Average [for Interval]	The average connection time for a specific connection during the most recent sample interval
Average [for Run]	The average connection time for a specific connection for the entire run
Maximum [for Interval]	The maximum connection time for a specific connection during the most recent sample interval
Maximum [for Run]	The maximum connection time for a specific connection for the entire run
Minimum [for Interval]	The minimum connection time for a specific connection during the most recent sample interval
Minimum [for Run]	The minimum connection time for a specific connection for the entire run
Standard Deviation [for Interval]	The standard deviation of connection times for a specific connection during the most recent sample interval
Standard Deviation [for Run]	The standard deviation of connection times for a specific connection for the entire run

Counter name	Description
Average [ms] [for Interval]	The average connection time for all connections within the most recent sample interval
Average [ms] [for Run]	The average connection time for all connections for the entire run

Counter name	Description
Maximum [ms] [for Interval]	The maximum connection time for all connections within the most recent sample interval
Maximum [ms] [for Run]	The maximum connection time for all connections for the entire run
Minimum [ms] [for Interval]	The minimum connection time for a all connections within the most recent sample interval
Minimum [ms] [for Run]	The minimum connection time for a all connections for the entire run
Standard Deviation [ms] [for Interval]	The standard deviation of connection times for all connections during the most recent sample interval
Standard Deviation [ms] [for Run]	The standard deviation of connection times for all connections for the entire run

Connects counters

The counters in this section provide information about the number of connections that were established.

Counter name	Description
Count [for Interval]	The number of established connections for a specific connection during the most recent sample interval
Count [for Run]	The number of established connections for a specific connection for the entire run

Receive Attempts counters

The counters in this section provide information about the number of attempts to receive data from the connection.

Counter name	Description
Count [for Interval]	The number of receive attempts for a specific connection during the most recent sample interval
Count [for Run]	The number of receive attempts for a specific connection for the entire run

Receives counters

The counters in this section provide information about the number of receive actions that were successfully completed that were received.

Counter name	Description
Count [for Interval]	The number of completed receives for a specific connection during the most recent sample interval
Count [for Run]	The number of completed receives for a specific connection for the entire run

Send Attempts counters

The counters in this section provide information about the number of attempts to send data from the connection.

Counter name	Description
Count [for Interval]	The number of send attempts for a specific connection during the most recent sample interval
Count [for Run]	The number of send attempts for a specific connection for the entire run

Sends counters

The counters in this section provide information about the number of send actions that were successfully completed.

Counter name	Description
Count [for Interval]	The number of completed sends for a specific connection during the most recent sample interval
Count [for Run]	The number of completed sends for a specific connection for the entire run

Chapter 10. Troubleshooting Guide

This guide describes how to analyze and resolve some of the common problems that you might encounter while you work with HCL OneTest™ Performance.

Troubleshooting performance testing

This topic provides information about how to troubleshoot several problems with HCL OneTest™ Performance.

If you run tests and encounter problems, make sure that you have followed all the [Performance testing tips on page 180](#).

If an error message is displayed when you run tests, try looking up the error message in the *Performance testing error messages* section of the online help. Only the most common error messages are listed. If no error message is displayed when you encounter a problem, open the error log by clicking **Window > Show View > Error Log**. If the workbench shuts down while running tests, restart the workbench and examine the error log. By default, warning and error messages are logged. You can increase the default logging level by clicking **Window > Preferences > Logging**. The log file is stored in the `.metadata` directory of your workspace. To avoid excessive logging, the Logging Level should be adjusted for individual Logger Names in the Loggers tab. For example, to get more information about a problem connecting with IBM® Rational® Quality Manager, increase the Logging Level for `com.ibm.rational.test.lt.rqm.adapter` Logger Name. For the licensing issue, adjust the level for `com.ibm.rational.test.lt.licening` Logger Name. When you no longer need the extra logging, use the **Restore Default** button in the Logging Preferences to reset all the levels to their recommended defaults.

You might encounter some of these problems while performance testing:

Connectivity problems between workbench and agent computers

If the workbench stops or locks up when you attempt to start running tests, it is important to confirm that all the agent computers are running. Perform the following steps to confirm your installation is properly configured:


- Confirm that there is sufficient disk space available on the workbench computer and the agent computers.
- Restart the workbench computer.
- Verify the network connectivity between the workbench computer and agent computers. To confirm the hostname in `majordomo.config` file can be DNS resolved on the agent machine, use a shell ping to the workbench hostname. If the ping results fail use the IP address of the workbench instead.
- Confirm the server port number on the test workbench computer. Click **Window > Preferences > Server**. This is the port number that should be specified in `majordomo.config` file on the agent machines.
- Restart the agent computers and verify the Majordomo process is running.

- On the agent machines, set the optional debug flag in the majordomo.config file. Set the value equal to true; the default value is false. You do not have to restart the agent. Within about ten seconds it should automatically pick up the changes to majordomo.config.

Look in %TEMP% directory for the majordomo.log file. This file contains information about the attempts to contact the workbench including information about any failures and the reason for the failures.

On the Windows operating system, the %TEMP% directory is typically at %USERPROFILE%\AppData\Local\Temp.

If the majordomo service is configured to log in as Local System Account, then the %TEMP% directory is at %SystemRoot%\TEMP, typically C:\Windows\TEMP.

- You can check the agent status on the workbench computer by clicking the  icon. For the Agent Controller, you can attempt to share files between the workbench computer and agent computers. Click **Window > Preferences > Agent Controller > Hosts**, and then add the agent computers as hosts, and click **Test Connection** to test connectivity to the instances of the Agent Controller that are running on the agent computers.

Recording configuration problems

No HTTP traffic is captured while recording

See [Recording reliable HTTP tests on page 182](#) for instructions on configuring your web browser. If you are attempting to use Internet Explorer to record tests from a secure website, see [Configuring Internet Explorer for recording from a secure web site on page 183](#). Disable firewalls on the workbench computer and the agent computers.

No traffic is captured while recording

Ensure that the recorder type that you select matches the protocol in use by the system under test. For example, do not attempt to use the HTTP recorder if the system under test uses the Citrix protocol.

No test is generated after recording

When the test generator cannot create a test from the recorded traffic, typically an error message is displayed or written to the error log. Try looking up the error message in the *Performance testing error messages* section of the online help.

Recorder controls are not available

If you use a workspace from a different version of the product, the recorder controls might not be available. Instead, the recorder controls from the other version of the product are displayed. Click **Window > Reset Perspective** to reset the **Performance Test** or **Service Test** perspective. Alternately, click **File > New > Other** to select the wizard to use.

Problems running large tests or long-run tests

If a test runs but ends with errors, check that the workbench computer and agent computers meet the hardware and software requirements that are detailed in the installation guide. Pay close attention to the memory and disk space

requirements. See [Increasing memory allocation on page 638](#) for more information on how to set the maximum heap size to avoid out-of-memory errors. Monitor processor and memory usage on the workbench and agent computers and watch for excessive processor use or excessive memory use by `javaw.exe` or `java.exe` processes.

Run tests with fewer virtual users that use the default schedule settings to determine whether the behavior is linked to the number of users. Examine the test log for error messages that the system under test generates. Run tests with a single virtual user and make sure that the system under test is not generating errors, before you attempt to run tests with a large number of users. If you encounter problems, restart the workbench and agent computers before attempting to run tests again.

If the workbench shuts down while running tests, search for file names that begin with `javacore`. The name of `javacore` files includes the date, time, and process ID. If you find a `javacore` file with a date, time, and process ID matching the workbench, open the file in a text editor. You can find the reason for failure at the beginning of the `javacore` file.

Data correlation errors

If you can record tests successfully, but the expected behavior is not triggered in your application when you run tests, you might need to perform manual data correlation. Typically when additional data correlation is needed, the test log includes messages similar to this message: `Unable to extract the value.` To troubleshoot data correlation problems, try running tests using only one virtual user running on the workbench computer, and compare the playback to the recorded test to determine which responses from the system under test are unexpected. See [Debugging HTTP tests on page 644](#) to learn how to use the test log and the **Protocol Data** view to troubleshoot HTTP tests. To learn more about data correlation, see [Correlating response and request data on page 443](#).

Common errors integrating with IBM® Rational® Quality Manager

All modes of the adapter use the Eclipse error log. You can view the log by opening the workbench and clicking **Window > Show View > Error Log**. By default, warning and error messages are logged. You can turn on more detailed logging for the adapter by clicking **Window > Preferences > Logging**. The log component for the adapter is named `com.ibm.rational.test.lt.rqm.adapter`.

If you are running the adapter as a Windows™ service or from the command line, you can view the `adapter.log` file without opening the test workbench.

Problem	Solution or cause
Where do you look for errors or warnings?	In the workbench, click Window > Show View > Error Log .
You do not see the adapter available for selection.	<ul style="list-style-type: none"> • Verify that the Engineering Test Management server address that is provided to the adapter is correct. Provide the correct address. • Check the provided login and password. Provide the correct password.

Problem	Solution or cause
The adapter continuously fails to connect to Engineering Test Management.	Make sure that the server is running. If necessary, restart the server or check network connectivity.
The adapter is displayed as red in the selection dialog box.	<ul style="list-style-type: none"> • The adapter is not communicating with the server. • The adapter might already be in use.
You attempt to import a script from the adapter but no scripts are found.	<ul style="list-style-type: none"> • Make sure the project path that is entered in Engineering Test Management is a project under the workspace that is associated with the running adapter. You have to enter only the project name. This is less error prone than typing the complete project path, but either forms are acceptable. • If running from the command line or as a service, be certain the <code>WORKSPACE_DIR</code> environment variable that is set in the <code>adapter.config</code> file is the same path as seen in the select workspace dialog box when running the test workbench. Be careful not to set the path to a project folder under the workspace directory. • Make sure that you are not using a workspace that contains a project that was copied from a shared location. A workspace that contains projects from shared locations cannot be used for projects that are not shared.
The adapter is running from the command line or as a service, and tests continue to fail.	Run the adapter in GUI mode so that you can see what happens when the test workbench runs the test script.
Adapter Windows™ services does not start. A error message states that the service failed to start in a timely fashion.	Ensure that the computer has .NET 2.0 or later. This platform can be installed from the Windows™ Update Site or manually. For more information on installing .NET, see http://support.microsoft.com/kb/923100 .
When testing shared assets, the execution fails with and an <code>IOException</code> message is displayed.	<p>The most likely cause is that the Engineering Test Management to UNC shared location is not set up correctly.</p> <ul style="list-style-type: none"> • From Engineering Test Management, ensure that you can access the UNC shared directory without being prompted for a password. You might have to map a drive on Windows™ for the Engineering Test Management system to log into the UNC share. • Ensure that you have defined the shared resource in Engineering Test Management under Admin > System Properties > Resources.

Problem	Solution or cause
When testing shared assets, the execution fails with a low level model error.	<ul style="list-style-type: none"> • Ensure that the test-script points to a shared location that still exists. If you have associated a Engineering Test Management test script with a shared location that has changed (for example if the IP address has been reassigned) you might need to reassociate every test script • Ensure that the UNC shared directory that is specified in Engineering Test Management points to a project.
Service tests that were created in a previous version of the product cannot be run.	Ensure that the adapter has the required protocol extensions installed. The test assets located on the shared location can only be run on an adapter workspace that supports those protocols.
The adapter cannot connect to the server, and one of the following error messages is displayed:	Upgrade every SOA asset to the latest version.
<ul style="list-style-type: none"> • <code>Communications error with server</code> • <code>Error occurred while registering the adapter</code> 	<ul style="list-style-type: none"> • When using Engineering Test Management 3.0 or later, the server URL that is configured for the adapter must exactly match the public URI of the Engineering Test Management server. The server public URI is available on the Engineering Test Management administration page. By default the administration page is at <code>https://server-name:9443/qm/admin</code>. • The adapter user must be a member of the Engineering Test Management project area. Open the project area administration page on the Engineering Test Management server to determine whether the adapter user is a member of the project area. For Engineering Test Management 3.0 and later, the adapter user must be a member in the test team member role, not the test team contributor role. This error can also occur if you have modified these roles from their defaults.

Performance testing error messages

Find more information about the error messages.

PRXE0101W

%1
terminating
due
to
exception:
%2

PRXE4943W

Transaction
[%1]
has
been
aborted.

PRXE4951I

User
group
[%1]
was
not
found.

RMSE0003W

RMSE0003W
There
are
currently
no
selected
counters
for
the
source
named
{0}.

Explanation: The source has no counters selected.

System action: Execution of the schedule will continue but the information related to this source won't be collected.

User response: Consider selecting at least one counter from the Resource Monitoring Service web console.

RMSE0004W

RMSE0004W

The
source
named
{0}
is
no
longer
available.

Explanation: This source has been removed from the Service web console after it was added to this schedule.

System action: Execution of the schedule will continue but the information related to this source won't be collected.

User response: Consider adding it back, then edit the schedule to update the sources to be monitored during its execution.

RMSE0005W

RMSE0005W

The
source
named
{0}
is
reporting
the
error
message
{1}.

Explanation: Look at the reported error.

System action: Execution of the schedule will continue but the information related to this source won't be collected.

User response: Consider fixing it from the Resource Monitoring Service web console.

RMSE0006W

The
server
does
not
support
resource
monitoring
labels.

Explanation: The server does not support resource monitoring labels.

System action: Execution of the schedule will continue but the resource monitoring counters won't be collected.

User response: Consider using a service that supports this feature.

RPAC0001W

The
JAR
%1
referenced
in
preferences
could
not
be
found.
Preferences
on
the
cloud
workbench
will
be
cleared.

Explanation: The Resource Monitoring preferences list a JAR file that is required for an instrumented application server type. This JAR file must be mapped to a new location and transferred to the cloud workbench. But this transaction failed, because the file could not be found locally.

System action: Execution in the cloud will continue but the instrumented application server types that require the listed JAR file might fail.

User response: Open the child preference page under Test -> Performance Resource Monitoring. Ensure that the listed files exist and can be found in a valid location.

RPHD1032E

Error
occurred
while
instructing
__PT_ACRONYM__
engine
to
enable
real-
time
protocol
data
for
user:
%1.
It's
possible
that
no
data
will
be
seen
for
this
user
in
the
Protocol
Data
view.

Explanation: There was a general error when starting real-time browsing in the Protocol Data View.

System action: The Protocol Data View will not be updated in real-time during this run. This does not affect test execution or post-run usage of the view.

User response: Ensure there is a stable connection with the Performance Test Agent and System Under Test. If problem persists, contact support.

RPHD1034E

Error
occurred
while
instructing
__PT_ACRONYM__
engine
to
disable
real-
time
protocol
data
for
user:
%1.
It's
possible
that
data
for
this
user
will
continue
to
be
displayed
in
the
Protocol
Data
view.

Explanation: = There was a general error when ending real-time browsing in the Protocol Data View.

System action: None.

User response: If the Protocol Data View no longer updates for additional runs or when the test editor selection is changed, closing the view and reopening it may help.

RPHE0001E

example
of
translatable
error
message
%1

RPHE0010W

Unknown
authentication
scheme
'%1'
discovered
in
HTTP
401
response,
ignoring.

RPHE0011W

Unrecognized
authentication
header
'%1'
discovered
in
HTTP
401
response,
ignoring.

RPHE0012W

No authentication headers found in HTTP 401 response, ignoring.

RPHE0013W

The server requested NTLM authentication but no NTLM authentication context was supplied with this request. Authentication is not possible.

RPHE0014W

NTLM
authentication
failed
for
this
request.
Verify
that
the
NTLM
authentication
context
values
for
this
request
are
correct.

RPHE0100W

Host
name
'%1'
can
not
be
resolved.

Explanation: A connection could not be established with the host. This can occur if the testing environment changes so that the host name is no longer correct. This can also occur when running a test on a different computer, such as an agent computer, from the workbench computer that was used for recording, if the new computer cannot resolve the host name.

User response: If the host name is incorrect due to a change in the testing environment, update the host name in the test. Otherwise, try to resolve the host name using the command `nslookup <hostname>`. Run `nslookup` on the agent computer if the error is happening on the agent computer. If `nslookup` is also unable to resolve the name, contact your network administrator. If `nslookup` resolves the host name, but the test continues to fail, try changing the host name to a fully-qualified host name. Alternatively, edit the hosts file.

RPHE0101W

Encountered
error
while
updating
dynamic
cookie
cache
while
interpreting
'Set-
Cookie'
header
with
value
'%1'
sent
from
web-
server
'%2'
retrieving
URI
'%3'.
Explanation
message:
'%4'.
Cache
not
updated
to
include
this
cookie
value.

RPHE0102W

Unexpected challenge(HTTP status code=401) received during HTTP playback to web-server '%1' retrieving URI '%2'. This behavior differs from the behavior recorded during test creation. For authentication to playback correctly a challenge must be recorded during test creation.

RPHE0103W

Authentication
failed
during
HTTP
playback
to
web-
server
'%1'
retrieving
URI
'%2'.
Probable
cause:
username
'%3'
and/
or
password
'%4'
incorrect.

RPHE0104W

Exception
occurred
during
attempt
to
write
request
to
web-
server
'%1'
getting
url
'%2'.
Explanation:
%3

RPHE0105W

General
un-
handled
exception
occurred
during
socket
I/O
read
from
web-
server
'%1'
retrieving
URI
'%2'.
Explanation
message:
'%3'.

Explanation: This error occurs when the server abruptly closes the connection to the virtual user. Servers might close connections if the virtual user is detected as a security risk due to an invalid cookie, failed SSL negotiation, or an improperly formatted request.

User response: Compare the request that was sent at run time (in the test log) to the one that is in the test. To determine if differences between the requests are valid, record the test again and compare the two requests.

RPHE0106W

A
read
time-
out
occurred
during
a
socket
I/O
read
from
web-
server
'%1'
retrieving
URI
'%2'.
Since
this
URI
is
the
primary
request
for
the
current
page
all
secondary
requests
will
be
skipped
and
the
next
page
will
be
attempted.
Current
time-
out
value
of

Explanation: The server did not return the response data before the timeout interval elapsed. If the server is under heavy load, the behavior can be caused by bottlenecks on the server or the agent computers. This error can also occur if an incorrect request is sent and the server is unable to respond.

User response: If the server is under heavy load, examine the server and agent computers to find and fix bottlenecks. Increase the timeout value. To stop tests or virtual users when this error occurs, enable error handling in the test and configure the server timeout error condition. If the server is not under heavy load, examine the request to ensure that it is valid and accurate.

RPHE0107W

A
read
time-
out
occurred
during
a
socket
I/O
read
from
web-
server
'%1'
retrieving
URI
'%2'.
This
secondary
request
will
be
skipped.
Current
time-
out
value
of
'%3'
milliseconds
should
be
increased
if
long
delays
are
expected
on
this
request.

Explanation: The server did not return the response data before the timeout interval elapsed. If the server is under heavy load, the behavior can be caused by bottlenecks on the server or the agent computers. This error can also occur if an incorrect request is sent and the server is unable to respond.

User response: If the server is under heavy load, examine the server and agent computers to find and fix bottlenecks. Increase the timeout value. To stop tests or virtual users when this error occurs, enable error handling in the test and configure the server timeout error condition. If the server is not under heavy load, examine the request to ensure that it is valid and accurate.

RPHE0108W

A
connect
time-
out
occurred
during
a
socket
I/O
connect
to
web-
server
'%1'
attempting
to
retrieve
URI
'%2'.
Since
this
URI
is
the
primary
request
for
the
current
page
all
secondary
requests
will
be
skipped
and
the
next
page
will
be
attempted.

Explanation: This error can occur if the server or agent computer is under heavy load. This error can also occur if the server or host computer is not configured with enough connections, or if the agent computer is not configured with enough sockets.

User response: Examine the server and agent computers to find and fix bottlenecks. To stop tests or virtual users when this error occurs, enable and configure error handling in the test.

RPHE0109W

A
connect
time-
out
occurred
during
a
socket
I/O
connect
to
web-
server
'%1'
attempting
to
retrieve
URI
'%2'.
This
secondary
request
will
be
skipped.

Explanation: This error can occur if the server or agent computer is under heavy load. This error can also occur if the server or host computer is not configured with enough connections, or if the agent computer is not configured with enough sockets.

User response: Examine the server and agent computers to find and fix bottlenecks. To stop tests or virtual users when this error occurs, enable and configure error handling in the test.

RPHE0110W

Unexpected challenge(HTTP status code=407) received while accessing HTTP proxy '%1' retrieving URI '%2'. This behavior differs from the behavior recorded during test creation. For authentication to playback correctly a challenge must be recorded during test creation.

Explanation: When the test was recorded, no basic authentication was required on the proxy server. When the test is run, the proxy server is requesting basic authentication information that is not in the test.

User response: Record the test again to capture basic authentication information. Play back the new test, or add the basic authentication information to the request in the original test.

RPHE0111W

Authentication
failed
accessing
proxy-
server
'%1'
retrieving
URI
'%2'.
Probable
cause:
username
'%3'
and/
or
password
'%4'
incorrect.

Explanation: Basic authentication failed when connecting to the proxy server. This can occur if an incorrect user name or password is supplied.

User response: Ensure that user name and password are correct.

RPHE0112W

An
error
occurred
during
decoding
of
content
received
from
web-
server
'%1'
attempting
to
retrieve
URI
'%2'.
Explanation
message:
'%3'.

RPHE0113E

Error encountered during the process of URI substitution for host=%1 and URI = %2 . Data correlation supplied a malformed URI= %3 . Explanation: %4. If you attempted to perform a custom data substitution on this URI ensure it has proper URI syntax.

If you did not

RPHE0113W

An
error
occurred
during
encoding
of
an
annotated
execution
history
event
property.
Explanation
message:
'%1'.

RPHE0114E

An
error
was
encountered
during
transform
of
response
data.
%1

Explanation: The response data was not in a format that the data transformer could interpret. This can occur when an error is returned from the server instead of valid response data.

User response: Examine the response data for errors.

RPHE0114W

Exception
occurred
during
attempt
to
write
request
to
proxy-
server
'%1'
getting
URL
'%2'
on
host
'%3'.
Explanation:
%4.

RPHE0115E

An
error
was
encountered
during
un-
transformation
of
request
data.
%1

Explanation: The transformed request data could not be converted into the format required by the server. This can occur because of a faulty data substitution. This can also occur if you manually edit the request data and invalidate the transformed data format.

User response: Correct the faulty substitution or the invalid data formatting.

RPHE0115W

Unable
to
successfully
establish
a
connection
to
web-
server
'%1'
retrieving
URI
'%2'.
Web-
server
closing
the
connection
after
connection
was
just
established.

RPHE0117W

Unexpected
exception
occurred
during
connection
close
to
web-
server
'%1'
retrieving
URI
'%2'.
Explanation:
%3.

RPHE0118W

HTTP
parsing
error
encountered
while
retrieving
URI
'%1'
from
web-
server
'%2'.
If
this
URI
is
the
primary
request
for
the
current
page
all
secondary
requests
will
be
skipped
and
the
next
page
will
be
attempted.

RPHE0119E

IP
aliasing
is
enabled
but
no
IP
address
was
found
for
virtual
user
%1.
Verify
correct
network
interface
name(s)
are
specified.

RPHE0120E

Exception
occurred
during
attempt
to
connect
to
proxy-
server
'%1'
getting
URL
'%2'
on
host
'%3'.
Explanation:
%4.

RPHE0121E

Unable
to
authenticate
with
the
proxy-
server.
Possible
solution:
re-
record
test
due
to
possible
proxy-
server
'%1'
authentication
changes.

RPHE0122W

Web-server
'%1'
unexpectedly
closed
the
connection
while
in
the
process
of
retrieving
URI
'%2'.
The
response
body
MAY
be
incomplete
due
to
a
missing
"chunk".
If
missing
chunk
was
last
(zero
length)
chunk,
data
is
complete.

RPHE0123W

Infinite
redirection
loop
detected
getting
URL
'%1'.
If
this
is
expected
and
understood
increase
RPT_VMARGS
rptMaxRedirection
parameter.
Redirected
history
%2

RPHE0124W

Unexpected
server
redirection
occurred
getting
URL
'%1'.
We
were
redirected
to
the
same
URI
which
issued
this
request.
Redirected
history
%2

RPIB0007E

%1

RPKG0090E

Exception
thrown
while
creating
connection
variables

Explanation: Exception thrown while creating connection variables

System action: Can not create the connection variable

User response: None required

RPKG0100E

Exception
thrown
by
the
launch
configuration
core

Explanation: Exception thrown by the launch configuration

System action: None required

User response: None required

RPKG0101E

Exception
thrown
during
an
update
to
a
launch
configuration

Explanation: Exception thrown during an update to a launch configuration

System action: None required

User response: None required

RPKG0110E

The
data
source
type
%1
is
not
expected

Explanation: The data source type %1 is not expected

System action: None required

User response: None required

RPSF0172E

__PT_ACRONYM__/
SAP:

Unable

to

start

SAP

GUI,

please

check

SAP

GUI

installation.

Explanation: SAP GUI can't be reached.

System action: The test is stopped.

User response: Install SAP GUI with scripting options as recommended by SAP.

RPTA0001I

Setting

the

log

verbosity

left

me

with

%1

users

RPTA0002E

A
Test
cannot
be
launched
on
the
specified
Driver

RPTA0003E

%1

RPTA0004E

A
Test
could
not
be
launched
on
Driver:
%1.
The
Test
Execution
Framework
was
not
able
to
deliver
an
Executor.
This
is
an
internal
error,
please
contact
support.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

Explanation: The Test and Performance Tools Platform (TPTP) infrastructure did not produce an executor for the test. This error message might display if firewalls are active on the local computer or the agent computer.

User response: Disable firewalls on both the local computer and the agent computer. If you do not want to disable firewalls, you can instead enable a firewall-aware connection. For more information on enabling a firewall-aware connection, see [Running with a workbench behind a firewall](#). On the local computer, check the properties of the location that represents the agent computer. This error can occur if the deployment root directory is not specified correctly in the location that represents the agent computer. Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Restart the Agent Controller. Restart the application.

RPTA0009E

A
Test
could
not
be
launched
on
Driver:
%1
due
to
an
internal
error.
Please
see
Problem
Determination
Log.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

Explanation: An exception was thrown during an attempt to obtain the operating system attribute of the location asset.

User response: Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Open the location asset representing the agent computer in the Test Navigator, and verify that all information and properties are correct. Delete the location asset representing the agent computer in the Test Navigator, and create a new location asset. You might need to delete the location and create a new one, if the location asset representing the agent computer asset is corrupted.

RPTA0010E

An
error
has
been
encountered
while
launching
a
Test
on
Driver:
%1.
Please
see
Problem
Determination
Log.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

Explanation: An exception was thrown while starting the test. The exception did not contain an error message.

User response: Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Restart the Agent Controller. Restart the application.

RPTA0011E

An
error
has
been
encountered
while
launching
a
Test
on
Driver:
%1.
An
Executor
was
not
returned
and
neither
was
an
error
message.
This
is
an
internal
error,
please
contact
support.

Explanation: The Test and Performance Tools Platform (TPTP) infrastructure produced neither an executor for this test nor error messages.

User response: Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Restart the Agent Controller. Restart the application.

RPTA0012E

An
error
has
been
encountered
while
launching
a
Test
on
Driver:
%1.
There
are
no
Data
Processors
present.
This
is
an
internal
error,
please
contact
support.

RPTA0013E

An error has been encountered while launching a Test on Driver: %1. Data Processors have not been configured correctly. This is an internal error, please contact support. For more information, see the Troubleshooting section of the online help.

Explanation: The test application was unable to configure the Data Processor for either the test log or the statistics portion of the test infrastructure.

User response: Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Restart the Agent Controller. Restart the application.

RPTA0014E

A
Test
could
not
be
launched
on
Driver:
%1.
The
Test
Execution
Framework
encountered
an
Exception.
This
is
an
internal
error,
please
contact
support.

RPTA0015E

An
error
was
encountered
while
launching
a
Test
on
%1.
\nPlease
examine
your
Deploy
Directory:
%2,
the
error
could
be
caused
by
one
of
the
following:
\n
\n1.
The
Deploy
Directory
path
must
be
absolute
(start
with
Drive
Letter
or
"/").
\n2.
The
Deploy
Directory
path

RPTA0016E

An
error
has
been
encountered
while
launching
the
test.
A
required
dataset
%1
is
missing
or
invalid
in
your
project.

RPTA0017E

An
error
has
been
encountered
while
launching
the
test.
A
required
dataset
%1
has
been
replaced.
One
or
more
test(s)
are
referencing
a
different
version
of
the
dataset.

RPTA0018E

ready

RPTA0019E

not
ready
on
port

RPTA0020E

Check
Agents
Failed

RPTA0021E

%1
deployment
directory
%2
format
not
compatible
for
operating
system
%3.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

Explanation: The deployment directory that is specified in the location asset representing the agent computer is incorrect for the operating system that is specified in the location asset.

User response: Open the location representing the agent computer in the Test Navigator, and edit the deployment directory or the operating system.

RPTA0023E

Virtual users have exited prior to stage completion.

At the end of stage %1 there were %2 users running when %3 were expected.

A common reason for this is a schedule which has assigned an insufficient amount of work (for one or more User Groups), to

Explanation: During schedule execution, at the end of the current stage, the actual number of users running did not match the expected number of users. For example, if the current stage specifies that 100 users should run for 1 hour and only 90 users are running at the end of the hour, this message is displayed.

User response: Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Typically, this message is displayed when virtual users did not have enough work to do for the duration of the stage. For schedules that contain more than one stage, verify that the workload under each user group is contained inside an infinite loop. Use infinite loops because the stage duration is controlled by the time when users stop. If virtual users have sufficient workload, look in the test log for more information about why virtual users stopped. The virtual users that stopped might have encountered errors. By default, this message is displayed when the number of expected users does not match the number of actual users running at the end of a stage. You can change this setting to specify the percentage of users that may stop during a stage without being considered an error. To change the error condition, create the -DrptStopTolerance property in the eclipse.ini file in the installation directory. For example, -DrptStopTolerance=80 specifies that 80% of the users may stop unexpectedly during stage execution without being considered an error.

RPTA0024E

Exception
encountered
adding
or
removing
users.

Explanation: This error message is displayed when a dataset reference between a test and a dataset is broken. Whenever a dataset is used, a reference is created in the test. The reference is a link that points to the physical dataset file in the test project. This link can break if the test is copied or imported into another project without copying or importing the associated dataset file. This link can also break if the dataset file is deleted.

System action: None.

User response: Do not copy or import individual test assets. Instead, copy or import entire projects. If you have already copied or imported individual test assets, copy the dataset from the previous project or create a new dataset that contains the same information. Open the test with the broken reference and link the dataset to the test. \nDo not delete dataset files.

RPTA0025E

The
schedule
has
no
user
group.

RPTA0026E

The RPT_VMARGS option rptPre811PageResponseTimes is specified on at least one location and is missing from at least one other location. Please ensure that either all locations include this option or none do. See "adjusted page response time for increased accuracy" in the help for more information.

RPTA0025I

Run
Completed
(%1)

RPTA0026I

Run
Terminated
(%1)

RPTA0027I

%1:
%2

RPTA0031E

Location
template
file
%1
is
not
found
(referenced
from
location
file
%2)

Explanation: A location template file referenced by a location file is missing or inaccessible.

User response: Create a location template file with the given name. If the location template file exists but is in a closed project, open the project.

RPTA0032I

Found
location
template
[%1]
for
[%2]
(instances
found:
%3)

RPTA0033I

%1
remote
location(s)
associated
with
location
template
[%2]

RPTA0034E

Cannot
change
stage
duration
if
Until
Finished
specified

RPTA0035E

Duration
time
specified
is
less
than
what
has
already
elapsed

RPTA0036E

Schedule
must
be
in
the
Running
state
to
change
stage
duration

RPTA0037E

Agent
%1
not
ready,
time
of
last
contact:
%2

Explanation: The specified agent is not in contact with the workbench. The schedule cannot run until all agents that are used in the schedule are actively connected.

System action: Install and configure the HCL OneTest™ Performance load generation agent on the agent computer.

User response: Ensure that the specified agent has a HCL OneTest™ Performance load generation agent installed and is properly configured to this workbench. Restart the schedule. See the online help for information about how to install and configure the load generation agent.

RPTA0038E

No
successful
contact

RPTA0039E

Unknown
host
'%1'

Explanation: The specified agent name is not resolving in the Domain Name System (DNS).

User response: Ensure that the agent name is spelled correctly in the location.

RPTA0040E

Unable
to
complete
deployment
to
agents
because
of
an
unexpected
error
in
the
publish
phase.
%1

Explanation: A deployment error occurred that is likely a low-level I/O error or an unrecoverable internal error.

User response: Check the exception messages for possible causes such as a lack of hard-disk space.

RPTA0041E %1

Explanation: The specified agent is not in contact with the workbench. The schedule cannot run until all agents that are used in the schedule are actively connected.

System action: Install and configure the __PT_RR_SHORTNAME__ load generation agent on the agent computer.

User response: Ensure that the specified agent has a __PT_RR_SHORTNAME__ load generation agent installed and is properly configured to this workbench. Restart the schedule. See the online help for information about how to install and configure the load generation agent.

RPTA0042E Agent
version
%1
incompatible
on
host
%2.
Minimum
agent
version
%3
required.

Explanation: The version of the __PT_AGENT_ACRONYM__ is not compatible with a feature in the schedule.

System action: The schedule cannot be launched so schedule execution ends.

User response: Upgrade the __PT_AGENT_ACRONYM__ on the machine specified to match the workbench version.

RPTA0043E Error
encountered

Explanation: An unexpected error occurred.

User response: Look for more details about the error in the message posted.

RPTA0100W

Failed
to
delete
file
%1

RPTA0518E

An
error
has
been
encountered
while
launching
the
test.
A
required
dataset
%1
is
missing
or
invalid
in
your
project.

Explanation: A test contains a link to a dataset that cannot be found or that is corrupted. This can happen when a project is not imported completely, or when a file is deleted.

System action: The test run does not start.

User response: Open the test. On the Common Options page, fix the broken link so that it points to a valid dataset file or delete the link.

RPTC0003E

Wrong
type
of
project
'%1'.

RPTC0004E

Unable
to
access
test
variable
initialization
file.
Make
sure
the
specified
file
path
is
accessible:
%1

RPTC0005E

Error
while
processing
XML
file
containing
variable
initializations.
Make
sure
the
file
contains
valid
XML
of
the
expected
format:
%1

RPTC0006E

Error
while
gather
test
variable
initializations.
No
variable
initializations
will
be
honored
for
this
run.

RPTC0008I

Setting
Variable
[name='%1',
value='%2',
source='%3',
user
group='%4',
location='%5']

RPTC00020E

Unexpected
I/O
error
while
communicating
with
workbench
%1

Explanation: During test-log transfer a network error occurred on the agent communicating to the workbench.

System action: The agent re-attempts to communicate with the workbench.

User response: If the problem persists, inspect error and take corrective action.

RPTC1001W

The
file
path
specified
for
the
Zip
Utility
is
invalid.

RPTC1002W

Could
not
get
the
classpath
for
project
'%1'.

RPTC1009I

Undefined

RPTC1011I

%1:
Request
delivered

RPTC1012I

%1:
successfully
added
%2
to
the
configuration
file

RPTC1013I

%1:
successfully
removed
%2
from
the
configuration
file

RPTC1014l

%1:
%2
is
already
in
the
configuration
file

RPTC1015l

%1:
Request
timed
out

RPTC1016l

%1:
Agent
not
ready

RPTC1017l

%1:
Agent
not
known

RPTC1018l

%1:
Unknown
host
exception

RPTC1019l

%1:
%2

RPTC1020I

License
type:
%1

Explanation: Lists the brand of licensing being used (either HCL or IBM).

System action: License checkouts will attempt to acquire a license of the corresponding type.

User response: No action required.

RPTC1021I

License
valid:
%1

Explanation: Indicates whether a valid license was successfully acquired (true/false).

System action: If true, the functionality associated with the acquired license will be enabled.

User response: If false, check your license configuration.

RPTC1030E

Unable
to
replace
dataset
'%1'
with
'%2':
%3.

Explanation: An error occurred attempting to replace datasets.

System action: Execution will complete with error.

User response: Refer to the error message for more details, change the command line options related to replacing datasets.

RPTC1031E

The
dataset
'%1'
doesn't
exist.

Explanation: Unable to locate the specified dataset referenced in the dataset command line option.

System action: Command line execution will be cancelled.

User response: Change the command line options related to replacing datasets.

RPTC1032E

The
dataset
'%1'
is
incompatible
with
existing
dataset
'%2'.

Explanation: The specified replacement dataset does not have compatible columns, type, etc.

System action: Execution will complete with error.

User response: Ensure the dataset has the same columns of the dataset it is replacing.

RPTe0005W

Unable
to
attach
requirements
report
into
RQM
result,
because
the
default
requirements
report
has
been
deleted.
You
can
recreate
the
default
reports
by
click
restore
defaults
button
on
the
Default
Reports
preference
page.

Explanation: When a test run started by __QM_NAME__ completes, the default report is attached to the __QM_NAME__ execution results. This error occurs when the report selected as the default report on the Default Report preferences page does not exist.

System action: No report is attached to the __QM_NAME__ execution results.

User response: Click Window > Preferences > Test > Performance Test Reports > Default Report to open the Default Report preferences page. Check that the selected report exists. Click Restore Defaults to reset the default reports.

RPTE0011W

Unexpected
error
while
releasing
system
resources
for
test
log
export.
This
may
cause
an
increased
memory
footprint,
until
HCL
OneTest™
Performance
is
restarted.

Explanation: Test log export has completed (possibly with errors described earlier in the workspace log), but when releasing assets used during the export operation, there was an unexpected error.

System action: Memory allocated to this operation may not have been freed. Previous errors are likely to be present explaining the root cause.

User response: It is advisable to restart the application to free memory allocated during this operation. The exported test log file may be available but there may be errors.

RPTE0147E

The
password
saved
for
an
encrypted
column
in
dataset
"%1"
was
invalid.
Set
a
new
password
in
the
Automation
Security
preference
page.

Explanation: The value saved in the Automation Security preference page for the specified dataset was not correct. It will be ignored.

System action: The password in the preference is ignored. If running from the workbench, it will prompt for a password before execution. Otherwise, execution will fail.

User response: Update the password in the Test - Test Execution - Automation Security preference page.

RPTH0130I

No
sample
time
closely
matches
request
at
time=
%1

RPTH049E

A
statistical
adapter
is
missing
reference
to
the
target
result.

Explanation: This is an internal error when loading results files. It could indicate that the result is corrupted, or it could only be a timing issue.

System action: The result cannot be opened.

User response: Close all reports and restart the workbench. If the result still does not open, kill any CPU-intensive processes running in the background.

RPT10069E

Local
on
premise
agent
%1
not
in
contact
with
this
workbench.

RPT10070E

See
Error
Log
for
more
details.

RPT10071I

There
was
an
error
while
updating
the
workspace
after
downloading
remote
files.

RPT10072E

Modify
majordomo.config
on
%1
and
configure
it
to
poll
this
workbench.

RPT10072I

Remote
Launch
Status:
%1

RPT10073E

Project
is
NULL

RPT10074E

Exception
occurred
while
creating
and
unzipping
project:
%1

RPTl0075E

Error
running
schedule.
Could
not
find
schedule
%1
in
project
%2.

RPTl0110I

Provision
time
(MM:SS):
%1

RPTl0111I

Launch
time
(MM:SS):
%1

RPTl0112I

Execution
time
(MM:SS):
%1

RPTl0113I

Results
transfer
time
(MM:SS):
%1

RPTI0141E

```
-----  
\nError  
Dialog  
\n%1:  
%2\nConsult  
workspace  
error  
log  
(workspace)/.metadata/  
.log)  
for  
further  
information.  
\n-----  
\n
```

Explanation: This message is displayed to the command-line output when an error occurs during execution. It displays details about the error and directs the user where to find additional information.

System action: None.

User response: This message occurs as a generic way to display errors during command-line execution. Consult the workspace log for further details including additional error messages.

RPTI0142E

The
Usage
Metrics
version
%1
required
by
the
licensed
component
%2
is
not
available.

Explanation: The license that you are using requires Usage Metrics reporting for a later version of the product.

System action: The execution will not start.

User response: Update the product to a newer version, or obtain a license that is applicable to the current version of the product.

RPTI0143E

The
licensing
system
failed
to
return
Usage
Metrics
enablement
for
component
%1.

Explanation: An error occurred while determining if the license requires Usage Metrics reporting.

System action: The execution will not start.

User response: Verify that the license is not meant for a newer version of the product. Otherwise, contact support.

RPTI0144W

No
RTCP
instance
is
available
to
report
Usage
Metrics.
No
Usage
Metrics
will
be
reported
for
this
execution.

Explanation: The license enables Usage Metrics reporting, but either the preference for the Usage Metrics server is not set, or it is set but the server is not active or reachable.

System action: The execution will be done normally, but the Usage Metrics will not be logged. This is allowed by the license you are using.

User response: If you have set up __QUALITY_SERVER__, go to Preferences > Test > __QUALITY_SERVER__, and fill in the server details for Usage Metrics reporting. Verify that the server can be reached from this machine by going to <http://servername:7828> in a browser on the local machine.

RPTl0145E

No
RTCP
instance
is
available
to
report
Usage
Metrics.
Per
license
policy,
execution
cannot
happen
unless
a
RTCP
is
defined
and
running.

Explanation: The license requires Usage Metrics reporting, but either the preference for the Usage Metrics server is not set, or it is set but the server is not active or reachable.

System action: The execution will not start.

User response: Install `__QUALITY_SERVER__` (if not done already), then go to Preferences > Test > `__QUALITY_SERVER__`, and fill in the server details for Usage Metrics reporting. Verify that the server can be reached from this machine by going to `http://servername:7828` in a browser on the local machine.

RPTI0146E

TPTP
Datapools
and
Datasets
cannot
coexist
in
the
same
test.
Test
run
aborted.

Explanation: A legacy datapool and a new dataset were both detected in the same test.

System action: Test execution will be aborted and will not be successful until the test contains only one of the two asset types (dataset or datapool).

User response: With the latest version of this product, convert the datapool to a dataset, then open the test containing the legacy datapool in the test editor and save it. Then, restart test execution.

RPTJ0063E

An
IOException
was
encountered
while
creating
the
Annotation
File
on
Driver:
%1

RPTJ0075E

An
IOException
was
encountered
while
creating
the
Execution
Log
File
on
Driver:
%1 ::
%2

RPTJ1002E

Driver
%1
returned
an
unrecognized
response:
%2.
The
last
command
sent
was:
%3

RPTJ1003E

While
waiting
for
an
acknowledgement
from
the
Driver,
an
unrecognized
response
was
received.

RPTJ1004E

The
workbench
was
waiting
for
an
Acknowledgement
from
the
__VENDOR_NAME__
Agent
Controller
on
Driver
%1
and
none
was
received.

Explanation: A required response from an agent was not received.

System action: Execution ends because the required acknowledgement from the agent was not received.

User response: Monitor resource usage on the agent. Add additional agents if memory or CPU usage is high on a any agent.

RPTJ1005E

Error
while
processing
a
message
from
the
__VENDOR_NAME__
Agent
Controller.

Explanation: An unexpected error occurred while handling a command from a load generating agent.

System action: Execution ends because of an unexpected error while communicating with an agent.

User response: Check the workbench Error Log for more information.

RPTJ1006E

Execution
failure.
No
status
received
from
location
%1
in
%2
seconds.
Workbench
memory
usage
at
%3
percent
of
the
configured
JVM
heap.
Possible
location
or
workbench
overload.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

Explanation: The workbench cannot communicate with the agent computer.

User response: Try running the schedule again, using default values for all parameters and running at reduced user load levels. It is possible one agent computer is overloaded. If you can run successfully with the default values, make changes to the schedule settings or user load incrementally to determine the cause of failure. Increase the statistics interval to 60 seconds and try running the schedule again. Check the error log for messages that might indicate the cause of the failure. Click Window > Show View > Error Log to open the error log.

RPTJ1007E

The
 Driver:
 %1
 has
 encountered
 a
 communication
 error.
 Please
 refer
 to
 Problem
 Determination
 Log
 for
 more
 details.
 For
 more
 information,
 see
 the
 Troubleshooting
 section
 of
 the
 online
 help.

Explanation: The agent computer that the message specifies encountered a problem when trying to run a command sent from the workbench.

User response: Check the error log for messages from the agent computer that the error message specifies. Click Window > Show View > Error Log. Check the test log for any failures from virtual users. This message might be displayed when you add or remove users manually or by means of schedule stages.

RPTJ1008E

The
Driver:
%1
has
become
unresponsive,
possibly
due
to
an
out-
of-
memory
condition.
At
last
notification
this
Driver
was
using
%2
percent
of
its
allocated
memory.
Please
refer
to
the
"Increasing
memory
allocation"
Help
topic
for
information
on
how
to
increase
memory
allocation.

For
more

Explanation: The workbench cannot communicate with the agent computer. The agent computer might have a memory allocation problem.

User response: Try running the schedule again, using the default values for Test Log and Problem Determination log levels. Follow the instructions in Increasing memory allocation. Set the memory allocation to the size of physical memory minus 256 megabytes, up to a limit of 1500 megabytes. For example, on an agent computer with one gigabyte of physical memory, set the memory allocation to 756 megabytes.

RPTJ1009E

The
Driver:
%1
is
running
%2,
however
the
user
selected
%3
as
the
Drivers
operating
system.

RPTJ1010E

Error
while
transferring
file
on
Driver:
%1.
Transfer
FROM:
%3
TO:
%2

RPTJ1011E

The
'%1'
Protocol/
Feature
is
not
supported
on
the
%2
platform,
so
the
Test
%3
can't
be
executed
on
location
%4.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

Explanation: The test includes a feature or protocol that is not supported on one of the agent computers where it is scheduled to run.

User response: Edit the schedule and associate the user groups that include the problem test with agent computers that support the feature or protocol.

RPTJ1012E

The
operating
system
(%1)
for
location
%2
is
not
recognized.
Please
use
an
operating
system
that
matches
or
begins
with
the
name
of
one
of
the
recognized
platforms:
%3

RPTJ1013E

No
valid
license
key
for
%1
Protocol/
Feature
found.
The
Test
%3
cannot
be
executed.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

Explanation: The test includes a feature or protocol that requires a license for the number of virtual users that are included in the run.

System action: The test run stops.

User response: Run the __BRAND_NAME__ License Key Administrator and check for available license keys for the feature or protocol and number of users that you want. To learn more about license keys, see the installation guide.
\nAdd the required license key or point to a server that has the required license key.

RPTJ1015E

The specified operating system (%1) for location %2 is inconsistent with the actual platform (%3) running at that location. Please update the operating system to match and then try again. For more information, see the Troubleshooting section of the online help.

Explanation: The operating system that is specified in the agent computer asset does not match the operating system that is running on the computer at the specified address.

User response: 1. Open the schedule in the test editor. 2. Select the user group that runs on the location mentioned in the error message. 3. In the Schedule Element Details, click the Locations tab, and then select the location mentioned in the error message. 4. Click Edit. 5. Select the appropriate value from the Operating system list. 6. Click OK.

RPTJ1016E

After
deploying
File:
%2
to
Driver:
%1,
%3
Byte(s)
where
found
on
the
socket.
Please
refer
to
the
Problem
Determination
Log
for
more
details.

RPTJ1017E

An
IOException
occurred
while
deploying
File:
%2
to
Driver:
%1.
Please
refer
to
the
Problem
Determination
Log
for
more
details.

RPTJ1018E

A
SocketException
occurred
while
deploying
File:
%2
to
Driver:
%1.
Please
refer
to
the
Problem
Determination
Log
for
more
details.

RPTJ1019E

An
UnsupportedEncodingException
occurred
while
deploying
File:
%2
to
Driver:
%1
Please
refer
to
the
Problem
Determination
Log
for
more
details.

RPTJ1020E

An
IOException
occurred
while
deploying
File:
%2
to
Driver:
%1.
\nA
possible
cause
is
that
the
__VENDOR_NAME__
Agent
Controller
was
started
by
a
non-
root
user.
\nThe
Agent
Controller
needs
to
be
started
by
the
root
user.

Explanation: Deployment of test assets to an agent failed.

System action: Execution ends because required test assets could not be copied to an agent.

User response: Ensure that the Majordomo process is started by the root user.

RPTJ1021E

An
InactiveAgentException
has
occurred
while
deploying
to
Driver:
%1.
Please
refer
to
the
Problem
Determination
Log
for
more
details.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

Explanation: The Test and Performance Tools Platform (TPTP) infrastructure threw an InactiveAgentException when the TPTP infrastructure attempted to communicate with the Agent Controller.

User response: Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Restart the Agent Controller on the agent computer.

RPTJ0121I

Send
RATEGENERATORS
to:
%1,
string
'%2'

RPTJ1022E

The
workbench
received
notification
that
the
execution
process
on
Driver
%1
has
terminated.

Explanation: The process running on the agent computer ended unexpectedly.

User response: Ensure that there is at least one successful test run, possibly with fewer virtual users, so that the maximum memory value for the agent is set correctly. Check the javacore* file on the agent computer or the logs in the deployment directory for further information on the process failure.

RPTJ1023E

Communication

with

Driver

%1

has

been

lost,

possibly

due

to

an

out-

of-

memory

condition.

At

last

notification

this

Driver

was

using

%2

percent

of

its

allocated

memory.

Please

refer

to

the

"Increasing

memory

allocation"

Help

topic

for

information

on

how

to

increase

memory

allocation.

For

RPTJ1024E

Error
during
initialization
of
annotation
transfer
progress
listener.

RPTJ1025I

Run
Completed
(%1)

RPTJ1026I

Run
Terminated
(%1)

RPTJ1030E

Non-
fatal
internal
exception
occurred
during
code
generation
optimization.
Code
generation
will
not
use
meta-
cache.

RPTJ1041E

The
'%1'
Protocol/
Feature
is
disabled
due
to
a
licensing
configuration
error.

RPTJ1042E

%1
Failure
checking
out
license
for
'%2'
Protocol/
Feature
and
%3
virtual
users.
The
Test
%4
cannot
be
executed.

RPTJ1043E

%1
The
'%2'
Protocol/
Feature
is
not
supported
on
the
%3
platform,
so
the
Test
%4
can't
be
executed
on
location
%5.

RPTJ1044E

Timed
out
after
%1
seconds
while
waiting
for
the
license
server.
Ensure
that
network
connectivity
to
the
license
server
exists
and
that
the
license
server
is
running.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

RPTJ1100I

A
hang
has
been
avoided
during
execution
history
receipt
with
%1
by
a
forceful
load
test
executor
state
change

RPTJ1101E

A
session
on
driver
%1
did
not
release
promptly.
Please
check
the
agent
controller.

RPTJ1102W

The
testLog
event
loader
thread
in
the
workbench
has
ended
before
processing
all
testLog
events
from
%1.
The
testLog
may
be
incomplete.

RPTJ1103W

The
test
executor
for
%1
has
been
artificially
set
to
HISTORY_COMPLETE
because
the
testLog
event
loader
thread
is
not
longer
running.

Explanation: A monitoring process indicates that the test log loader stopped prematurely. This is not a definite indication of a problem.

User response: Check that the expected events exist at the end of the test log. If so, no further action is necessary.

RPTJ1104E

Remote
debug
never
received
event
%1,
process
exit
value
%2

Explanation: Expected remote debug event was not received

User response: Check the Error Log for remote process failure reason

RPTJ1141E

Temporary
dataset
file
%1
not
created.

Explanation: Temporary dataset file can't be created on the system.

System action: Original dataset is used.

User response: Check corresponding file properties on the system.

RPTJ1142E

Temporary
dataset
data
are
not
generated:
%1

Explanation: Error reached during data generation.

System action: Original dataset is used.

User response: Check corresponding connection information.

RPTJ1200W

Failed
to
delete
file
%1

RPTJ1220E

An
InactiveAgentException
has
occurred
attempting
to
send
[%1]
to
driver
%2

RPTJ1221E

The
CommandHandler
for
%1
has
encountered
an
exception
while
processing
%2

RPTJ1240E

Driver
%1
has
reported
a
NOK.
The
last
command
sent
to
that
driver
was:
%2

Explanation: A schedule command sent from the workbench to the agent computer could not be run by the agent computer.

User response: Run the schedule using the default settings. Look for unusual assignments of numbers of virtual users to agent computers at stage transitions.

RPTJ1241E

Driver
%1
has
reported
a
NOK
with
the
message:
%2.
The
last
command
sent
to
that
driver
was:
%3

RPTJ1242E

Driver
%1
has
reported
a
%2
status

RPTJ1244E

The
AgentCommandListener
for
%1
has
encountered
an
exception
while
processing
%2

RPTJ1245E

Driver
%1
has
reported
that
it
is
no
longer
receiving
messages
from
the
workbench.
The
previous
message
received
from
this
driver,
%2
milliseconds
ago,
was
%3.
At
present
no
commands
have
been
sent
to
this
driver.

Explanation: Schedule commands sent from the workbench to the agent computer were not received by the agent computer.

User response: Ensure that there is at least one successful test run, possibly with fewer virtual users, so that the maximum memory value for the agent is set correctly. Use more agent computers to run the schedule.

RPTJ1261E

The
ResponseHandler
for
%1
has
encountered
an
exception
while
processing
%2

RPTJ1270E

Failure
attempting
to
launch
test
execution.

RPTJ1271E

The
process
executing
the
test
has
ended
unexpectedly.

Explanation: The process that runs tests could not start, or it stopped before the test run ended.

System action: The test run stops.

User response: Check the core files or the logs for further information on the process failure. If you are using Java Virtual Machine (JVM) arguments, check the argument syntax and try running tests without the arguments. Run the test inside a schedule.

RPTJ1280E

The communication path for returning test results from %1 has not been established. Check network connectivity between that machine and the workbench including any firewalls.

RPTJ1400I

%1%
%2/%3
files
%4/%5
bytes
deployed

RPTK0000I

%1

RPTK1001E

__PT_ACRONYM__

has
detected
the
presence
of
an
invalid
Virtual
Tester
license
key.
If
you
have
recently
upgraded
__PT_ACRONYM__,
note
that
this
is
a
new
check
performed
by
release
7.0.1
or
later,
and
instructions
for
replacing
invalid
Virtual
Tester
license
keys
should
have
already
been
sent
to

Explanation: Invalid Virtual Tester license key(s).

System action: System will not execute schedule run(s) that require a Virtual Tester license if one is not available.

User response: You must replace all invalid Virtual Tester license keys. If you need further assistance, please contact your sales representative or Technical Support.

RPTK1016E

The
specified
license
server's
version
level
is
not
compatible
with
this
version
of
__PT_ACRONYM__.

Explanation: The specified license server's version level is not compatible with this version.

System action: Incompatible version.

User response: Check the license server's version.

RPTK1019E

Unable
to
verify
system
time.

Explanation: The system time has been tampered with since the last successful license check.

System action: Future license checks will automatically fail.

User response: Contact Technical Support.

RPTK1020E

Unable
to
locate
license
directory.

Explanation: Unable to locate license directory.

System action: Stop execution.

User response: Please verify that the license directory exists.

RPTK1021E

License
has
expired.

Explanation: An expired license was found.

System action: Request a license key from user.

User response: Enter a new license key.

RPTK1022E

Invalid
license
file.

Explanation: A valid license was not found.

System action: Request a license key from user.

User response: Enter a valid license key.

RPTK1023E

Unable
to
find
a
license
supporting
%1
virtual
users.

Explanation: The currently installed license key(s) do not support enough VUs for this operation.

System action: Request a license key from user.

User response: Enter another license key to enable more VUs.

RPTL0001W

Unable
to
retrieve
data
from
the
test.

RPTL0002W

Failed
to
store
test
data
into
annotations.

RPTL0003W

Failed
to
attach
the
annotation
to
the
test.

RPTL0004W

Unable
to
open
test
annotation
to
read
data.

Explanation: The test appears to be corrupted.

System action: Attempts to open the test fail.

User response: Make sure your disk has enough space. If it does, try recreating the test from the recording.

RPTL0005W

Failed
to
create
a
temporary
file
to
save
test
data.

RPTL0006W

Failed
to
load
test.
Path
%1
is
invalid.

RPTL0007W

Failed
adding
element
from
an
un-
registered
feature
%1.

RPTL0008E

Cannot
load
a
test
created
by
a
future
version
%1.
Please
upgrade
your
install.

RPTL0009I

Test
%1
is
of
an
older
version
%2.

RPTL0010E

Error
creating
metadata
cache.

RPTL0011E

Error
reading
metadata
cache
for
%1.

RPTR0000W

%1

RPTR0001W

Failed
to
add
annotation
to
execution
history
for
file
%1

RPTR0002W

Unexpected
error
in
data
validity
check
of
LoadTimeEObjectConsumer

RPTR0003W

Failed
to
add
properties
to
parent
id
%1

RPTR0004W

Failed
to
delete
temp
file
%1

RPTR2001E

Unexpected
exception
in
container
complete
loader.
Heap
growth
likely.

RPTR2003W

Execution
Variables
-
Output

RPTS1000E

Unable
to
start
the
agent
communication
service
because
of
an
error:
%1.
RPT
will
not
be
able
to
execute
schedules.

Explanation: The agent communication service could not start. This service is a lightweight web server that agents use to communicate with the workbench and to serve web reports. Typically, this error occurs when a server process on the workbench computer is listening on the same port that HCL OneTest™ Performance requires. This error can also occur when two instances of HCL OneTest™ Performance run on the same workbench.

User response: If multiple instances of the HCL OneTest™ Performance workbench are running on the same computer, close all but one instance. These instances include HCL OneTest™ Performance workbenches that are running on multiple user desktop systems. If the error message RPTS1002E_PORTS_CONSUMED is also displayed in the error log, see the message for that error. After the error is resolved, restart HCL OneTest™ Performance.

RPTS1002E

RPT
is
unable
to
execute
a
schedule
because
one
of
the
ports(%1)
it
uses
to
communicate
with
the
agents
has
been
taken
by
another
RPT(or
other
server)
process.
Ensure
only
one
RPT
instance
is
running.

Explanation: Typically, this error occurs when a server process on the workbench computer is listening on the same port that HCL OneTest™ Performance requires. This error can also occur when two instances of HCL OneTest™ Performance run on the same workbench.

User response: Identify and stop the other process or service on the workbench that is using the ports that HCL OneTest™ Performance requires. Restart HCL OneTest™ Performance. You can also change the ports that HCL OneTest™ Performance uses by configuring the workbench and all agent computers. To change the ports, click Window > Preferences > Test > Server, and click Preferences > Test > Performance Test Reports > Web Reports.

RPTS1510E

Unable
to
stop
the
agent
communication
service
because
of
an
error:
%1

RPTS1001I

Agent
communication
service
listening
on
ports(%1)

Explanation: The agent communication service requires these local server ports to communicate with agents.

User response: This message is for informational purposes only.

RPTX0001E

The
combination
of
transformer
and
feature
you
have
selected
is
invalid.
Transformer
(%1)
was
not
expecting
data
type
(%2).

RPTX0002E

The combination of feature and transformer you have selected is invalid. Feature (%1) was not expecting data type (%2) to be returned by transformer (%3).

RPTX0003E

Transformer (%1) has experienced a fatal error. Additional information (%2).

RPTX0004E

Feature
(%1)
has
experienced
a
fatal
error.
Additional
information
(%2).

RPTX0005E

No
class
can
be
found
for
the
specified
transformer
id
(%1).
Please
check
to
make
sure
you
have
installed
this
transformer.

RPTX0006E

Class
definition
missing.
Please
add
jar
that
contains
definition
of
(%1)
to
the
classpath
of
the
test
project.

Explanation: Some requests or responses contain data that is encoded for Google Web Toolkit (GWT). To decode the data, HCL OneTest™ Performance requires access to the class definition.

User response: Add the JAR file that contains the class definitions to the classpath of the test project.

RPTX0007E

The
transformation
raised
a
GWT
serialization
exception:
%1

Explanation: The Google Web Toolkit (GWT) transformation could not be applied because of the indicated reason.

User response: Verify that the test elements containing the GWT encoded or decoded data are correct

RPTX0008E

The
Silverlight
decoder
raised
an
exception:
%1

Explanation: The Microsoft Silverlight decoder did not work because of the indicated reason.

User response: Verify that the test elements containing the Silverlight encoded data are correct

RPTX0009E

The
Silverlight
encoder
raised
an
exception:
%1

Explanation: The Microsoft Silverlight encoder did not work because of the indicated reason.

User response: Verify that the elements containing the Silverlight decoded data are correct

RPTX0010E

The
GraniteDS
transformer
made
an
error
when
encoding
or
decoding:
%1

Explanation: The GraniteDS encoder did not work because of the indicated reason.

User response: Verify that the elements containing the GraniteDS encoded or decoded data are correct.

RPXD002W

The
time
to
extract
references
seems
excessive.
It
was
%1
milliseconds.

Explanation: It is taking a long time to extract data from your response for your references.

System action: None.

User response: Examine each of the regular expressions for your references. Make sure they don't have .* with no qualifiers or other poorly formed constructs. When you write the regular expression in the test you can click verify to get an idea of how long it is taking to execute.

RPXE0061

Loop
iteration
started
late
by
%1
milliseconds

Explanation: A scheduled loop iteration started execution later than expected given the specified rate.

System action: Execution continues along with attempt to catch up in order to maintain desired rate.

User response: Add additional users or agents to increase capacity in order to maintain desired rate.

RPTX1010l

Start
of
OT-
Performance
project
resolve.
Repository=<
%1>,
Bootstrap=<
%2>

RPTX1011l

Attempting
to
resolve
asset=<
%1>

RPTX1012l

End
of
OT-
Performance
project
resolve.
No
detected
errors

RPTX1017I

Downloaded
asset
%1
from
remote
repository,
local
asset
created.

RPTX1018I

Using
local
cached
version
of
asset
%1.

RPTX1019I

OT-
Performance
testsuite=<
%1>
found
the
following
dependencies=<
%2>

RPTX1081E

Exception
occurred
while
uploading
Mobile
report.

Explanation: A low-level exception occurred uploading the mobile report. It is unexpected.

System action: The RQM report will fail to upload.

User response: If possible take corrective action, otherwise contact support.

RPTX1082E

An
error
occurred
when
generating
the
HTML/
zip
report.

Explanation: The HTML generator for the Execution Report has failed.

System action: No execution report uploaded into RQM results

User response: Ensure that the temporary directory is accessible on your file system.

RPTX2001E

Adapter
unable
to
start
test
because
__PT_ACRONYM__
is
already
executing
a
test.

Explanation: The adapter received a request to start a test while another test on the adapter is in-progress.

System action: The adapter ignores the request to launch another test.

User response: Wait for the test which is currently executing on the adapter to complete, then re-initiate the launch.

RPTX2002E

Error
encountered
parsing
RQM
adapter
preferences:
%1.
Please
enter
proper
credentials
in
the
Eclipse
Quality
Adapter
preference
page
(Windows-
>Preferences).

RPTX2003E

Project
<
%1>
could
not
be
found
during
RQM
import.

RPTX2004E

Test
log
is
unavailable,
no
test
results
returned
to
RQM.

RPTX2005E

Statistics
log
is
unavailable,
no
statistic
results
returned
to
RQM:
%1

RPTX2006W

Display
unavailable,
no
__PT_ACRONYM__
HTML
reports
will
be
attached
to
RQM
execution
results.

Explanation: The adapter requires access to a virtual display to generate HTML reports. The adapter was unable to successfully create a display so HTML reports may be unavailable.

System action: HTML reports are not generated at the end of execution.

User response: If HTML reports are required, start the adapter with display access. Refer to documentation on how to start the adapter with a display.

RPTX2007I

Start
RQM
Execution
Request
Project=
%1
Name=
%2

RPTX2008I

Start
RQM
Import
Request
Project=
%1

RPTX2009I

End
RQM
Execution
Request

RPTX2010I

End
RQM
Import
Request

RPTX2011E

Unable
to
interpret
RQM
configuration
file
%1.
If
file
was
hand
edited
make
sure
parameters
are
the
correct
format.
If
you
are
unable
to
get
this
file
into
the
correct
format,
please
erase
and
re-
configure.

RPTX2012E

Invalid
RQM
connection
parameter:
%1.
Adapter
was
not
launched.

RPTX2013E

Adapter
was
stopped
while
a
test
was
executing.
The
results
of
this
test
may
be
unreliable.

RPTX2014E

Adapter
was
stopped
while
preparing
to
run
an
RQM
script.
There
are
no
results
for
the
attempted
test
script
run.

RPTX2015E

Testsuite
'%1'
or
project
'%2'
does
not
exist.
Ensure
workspace
started
by
adapter
contains
project
and
testsuite.

RPTX2016l %1

RPTX2017E %1
Reason:
 %2

RPTX2018W %1

RPTX2019l The
 RQM
 Adapter
 has
 been
 disconnected.

RPTX2020l The
 RQM
 Adapter
 has
 stopped.

RPTX2021E Unexpected
 error
 occurred
 while
 executing
 RQM
 test
 script.

RPTX2022E

Unexpected
error
occurred
while
processing
an
import
request
from
RQM.

RPTX2023W

Error
occurred
while
update
the
run
status
back
to
the
RQM
server.
This
may
cause
the
RQM
test
progress
page
to
contain
inaccurate
data.

RPTX2024E

Unable
to
attach
the
following
file
to
the
RQM
results.
This
may
cause
the
attached
HTML
report
not
to
render
correctly.
File
name:
%1

RPTX2025E

Error
occurred
while
registering
the
adapter:
%1.

RPTX2026E

Error
occurred
setting
the
default
adapter
name.
Please
set
the
name
in
the
Eclipse
Quality
Adapter
preference
page
(Windows-
>Preferences).

RPTX2027W

Multiple
test
runs
were
detected
when
the
stop
request
was
received
from
RQM.

RPTX2029W

Was
unable
to
perform
stop
request
from
RQM.
Likely
the
run
was
already
shutting
down
when
the
request
came
in.

RPTX2030I

Request
to
stop
the
test
is
being
delayed
until
the
appropriate
run
state
is
reached.

RPTX2031

A
request
to
stop
the
currently
running
test
has
been
received
by
RQM.

RPTX2032

Successfully
issue
a
stop
command
to
the
running
test.
Please
wait
for
the
test
to
end.

RPTX2033E

Error
attempting
to
stop
a
test.

RPTX2034E

Unable
to
create
directory
%1
no
further
information.
Ensure
user
has
permission
to
create
directory
in
that
location.

RPTX2035E

Error
occurred
while
attempting
to
automatically
update
pre-8.0
asset
%1
for
RQM
execution.

RPTX2036E

RQM
remote
resource
access
is
not
supported
for
pre-8.0
SOA
assets.
Please
update
your
entire
SOA
project
to
8.0
or
greater
before
sharing.

RPTX2037E

Launch
was
aborted:
%1

RPTX2050E

Unable
to
download
remote
asset
%1
into
local
workspace.
Remote
repository
%2.
Ensure
RQM
system
has
connectivity
to
the
remote
repository
and
the
file
exists.

RPTX2051E

Unable
to
browse
%1
in
remote
repository
%2.
Ensure
RQM
system
has
connectivity
to
the
remote
repository
and
the
directory
exists.

RPTX2055E

Error
occurred
reading
the
adapter
connection
file.

RPTX2056E

Error
occurred
saving
the
adapter
connection
file.

RPTX2057E

Unable
to
complete
import
operation
because
the
specified
path
is
not
in
the
adapters
workspace.
Try
specifying
the
only
the
project
name.

RPTX2058E

The specified script < %1> is not in the workspace currently being used by the adapter. You can only execute scripts which are in the adapters workspace.

RPTX2060E

The script path specified by RQM does not seem to be valid. Please ensure the RQM test script has a script path which contains the project and script name.

RPTX2061W

Run
verdict
is
inconclusive
because
no
performance
requirements
exist
in
the
last
user
stage
for
the
associated
VU
Schedule.

RPTX2062W

Run
verdict
is
inconclusive
because
there
are
zero
performance
requirements
in
the
last
user
stage
of
the
associated
VU
Schedule.

RPTX2063W

No
time
range
was
generated
for
the
user
stage
of
the
associated
VU
Schedule.
Performance
requirements
reported
to
RQM
will
be
based
on
the
default
time
range.

RPTX2070E

Error
occurred
while
setting
the
RQM
project
area.
Make
sure
a
valid
project
area
is
specified
on
the
Quality
Manager
Adapter
preference
page.
The
adapter
is
attempting
to
connect
to
RQM
using
the
default
project
area.

RPTX2071E

Error occurred while retrieving list of project areas. Please verify Quality Manager connection information. See error log for more details.

RPTX2072E

Error occurred calling for the web analytics dashboard link.

Explanation: HCL OneTest™ Performance could not open the external URL for the dashboard that references Rational Quality Manager records. This error occurs when a problem exists with the classpath for the result analysis.

User response: Ensure that Rational Quality Manager is version 4.0 or later and HCL OneTest™ Performance is version 8.3 or later. If the error log contains startup errors, resolve the errors and check whether the problem is resolved.

RPTX2073E

Error
occurred
while
translating
RQM
server
execution
variables
to
__PT_ACRONYM__.

Explanation: An unexpected error occurred while setting up execution variables.

System action: Execution variables are unavailable during execution.

User response: Contact support if this error persists.

RPTX2074E

Error
connecting
OT-
Studio
adapter
and
successful
connecting
OT-
Performance
adapter.
This
suggest
RQM
does
not
support
OT-
Studio
script
type
introduced
in
4.0.3.
If
OT-
Performance
adapter
is
not
required
it
can
get
disabled
by
adding
-DrtwStartAdapter=false
in
eclipse.ini.

Explanation: Error connecting OT-Studio adapter and successful connecting OT-Performance adapter. This suggest RQM does not support OT-Studio script type introduced in 4.0.3. If OT-Studio adapter is not required it can get disabled by adding -DrtwStartAdapter=false in eclipse.ini.

User response: Use a RQM system supporting OT-Studio script type. Add -DrtwStartAdapter=false in eclipse.ini to disable OT-Studio script type. To import and execute OT-Studio assets under the OT-Performance adapter script type also add -DrptAvoidRQMImportFiltering in eclipse.ini

RPTX2075E

Unable
to
interpret
expression
<
%1>
from
RQM
control
file
<
%2>.
Ignoring.
Reason
<
%3>

Explanation: The RQM control file is of an invalid format.

System action: The control file instruction will be ignored.

User response: Change the file so it follows the specified format supplied by support.

RPTX2077E

Unable
to
browse
to
<
%1>.
Make
sure
it
exists
on
the
shared
location.

Explanation: The project referenced does not exist on the shared location.

System action: The RQM execution will stop.

User response: Ensure all required projects exist on the shared location.

RPTX2079W

Errors
attempting
to
load
available
SmartCard
aliases.
See
documentation
on
how
to
configure
your
system
to
use
SmartCard
to
authenticate
to
Quality
Manager.

Explanation: An error occurred while attempting to load SmartCard aliases.

System action: User is unable to configure SmartCard using the preference UI.

User response: Ensure com.ibm.security.capi.IBMCAC is listed as provider 1 in file *{install}\SDP\jdk\jre\lib\security\java.security*. See documentation on manual steps required to configure adapter for SmartCard usage. Contact support if issues persist.

RPWF0011E

Error
occurred
while
completing
test
generation

RPWF0012E

Error
occurred
while
processing
a
packet
at
test
generation

RPWF0021E

WSDL
Exception
raised
while
processing
WSDL
source

RPWF0032E

Error
while
generating
test
from
Axis
recording

RPWF0051E

Error
occurred
while
setting
classpath
entry
for
recorder

RPWF0052E

I/O
exception
occurred
while
resolving
keystore
or
truststore
path

RPWF0056E

Error
occurred
while
launching
web
services
HTTP
proxy

RPWF0066E

Error
occurred
while
launching
axis
client
recorder
agent

RPWF0071E

Exception
thrown
while
creating
a
wizard
page
control

RPWF0072E

Exception
thrown
while
parsing
URL:
%1

RPWF0074E

Exception
thrown
while
finishing
the
axis
recording
wizard

RPWF0075E

Exception
thrown
while
looking
for
an
available
port

RPWF0076W

Exception
thrown
while
adding
SOA
Tester
certificate
to
the
trustore
%1

RPWF0081W

A
proxy
authorization
%1
is
used
without
any
proxy

RPWF0082W

No
free
name
can
be
found;
reusing
%1

RPWF0083E

Resource
file
%1
not
found
in
workspace
%2

RPWF0084E

Workspace
location
cannot
be
determined

RPWF0085E

Cannot
retrieve
the
operation
name
from
the
envelope
%1

RPWF0101E

Core
exception
thrown
using
org.eclipse.debug.core
plugin

RPWF0102E

Exception
thrown
during
launch
configuration
update

RPWF0103E

Exception
thrown
while
resolving
a
bundle
entry
path

RPWF0104E

Exception
thrown
while
identifying
localhost
IP
address

RPWF0111E

Exception
thrown
while
creating
a
substitution:
%1

RPWF0112E

Exception
thrown
while
creating
a
reference:
%1

RPWF0121W

Unknown
format.
Skipping
the
test
generation
for:
%1

RPWF0122W

Skipped
call:
%1

RPWF0123W

Skipped
request:
It
could
be
that
provided
password
was
not
ok

RPWF0124W

Attachments
not
generated.

RPWF0130W

Could
not
find
project
for
URI:
%1

RPWF0131W

Loading
XSD
Schema
failed:
%1

RPWF0132E

Error
while
generating
test
from
Generic
Service
Client:
Can't
show
wizard

RPWF0140E

An
error
has
occurred:
%1

RPWH0007W

Unhandled
Security
Algorithm
'%1'

RPWH0009W

Unable
to
serialize
data

RPWH0010W

Unable
to
deserialize
data

RPWH0012E

Unable
to
open
editor
for
'%1'

RPWH0014E

Parse
Error
in
'%1'

RPWH0015E
Unable
to
create
resource
'%1'

RPWH0016E
Failed
to
export
source
text
'%1'

RPWH0017E
A
connection
error
occurred
on
'%1',
please
check
the
URL
or
the
network
configuration

Explanation: A connection error occurred.

System action: URL can not be reached, action is aborted.

User response: Check the URL or the network configuration.

RPWS0001E

Exception
raised
during
data
harvest
execution

Explanation: Reference can't be performed.

System action: Reference is not performed: get empty data.

User response: Check the corresponding reference.

RPWS0002E

Exception
raised
during
data
substitution
execution

Explanation: Substitution can't be performed.

System action: Substitution is not performed: write recorded data.

User response: Check the corresponding substitution.

RPWS0003E

Exception
raised
on
harvest
data
management

Explanation: Reference can't be performed.

System action: Reference is not performed: get empty data.

User response: Check the corresponding reference.

RPWS0004E

Exception
raised
on
substitution
data
management

Explanation: Substitution can't be performed.

System action: Substitution is not performed: write recorded data.

User response: Check the corresponding substitution.

RPWS0005E

Exception
raised
during
WebSocket
read
action

Explanation: Read action can't be performed.

System action: No data are receive.

User response: Check the application side, may be the server closes the connection.

RPWS0006E

Exception
raised
during
WebSocket
write
action

Explanation: Write action can't be performed.

System action: No data are sent.

User response: Check the application side, may be the server closes the connection.

RPWS0007E

Unable
to
get
WebSocket
connection

Explanation: WebSocket connection information is wrong.

System action: No data will be receive or sent on this connection.

User response: Check the WebSocket connection, may be the test is corrupted.

RPWS0008E

Unable
to
read
from
a
closed
connection

Explanation: WebSocket connection is closed.

System action: No data will be sent on this connection.

User response: Check why the WebSocket server closed connection.

RPWY0002E

An
exception
occurred
in
%1

Explanation: An exception was detected.

System action: Current action is aborted.

User response: Check the cause of the exception.

RPWY0003I

Information:
%1
(%2)

RPWY0004W

Warning:
%1
(%2)

RPWY0005E

An
error
occurred
while
importing
external
schema
%1

RPWY0006E

Unable
to
correlate
automatically

RPWY0007E

An
exception
%1
occurred
in
%2

Explanation: An exception was detected.

System action: Current action is aborted.

User response: Check the cause of the exception.

RPWZl0002E

Exception
raised
during
WebSocket
connection
creation.

Explanation: The workbench could not create a connection for WebSocket elements in split test.

System action: No connection are created by the workbench.

User response: The user need to create manually the connection, or to get the upgraded HTTP request in the split selection.

RPXD0001E

Unknown
Segment
Offset/
Length
for
Segmented
Dataset:
%1

RPXD0002E

Bad
Dataset
Mode:
%1

RPXD0003E

Dataset
not
initialized:
%1

RPXD0004E

End
of
non-
wrapped
dataset
reached:
%1

RPXD0005E

Dataset
with
multiple
Equivalence
Classes
cannot
be
segmented

RPXD0006E

segmented
DatapoolMap
null:
%1

RPXD0007F

No
registered
data
correlation
handler
for
this
IKAction

RPXD0017W

Pattern
matching
failed
for:
regex
(%1)
str
(%2)

RPXD0018E

Skipping
substitution,
reference
value
was
null.
original
string:
(%1)
offset:
(%2)

RPXD0019E

Data
Correlation:
Failed
Substitution
\nReference[%1]\nSubstitution[%2]\n\n\nDetails:
\n\n\nA
failed
reference
occurred
in
a
prior
request.
Since
the
reference
named
[%3]
was
null,
we
were
unable
to
substitute
a
new
value
for
the
substituter
named
[%4],
original
string
[%5]
at
offset
[%6]
and
this
request
may
have

Explanation: A reference for an expected data substitution is null.

System action: None.

User response: To find the failed reference, open the test and go to the substitution site. Right-click the substitution site and select Go To > Reference. When troubleshooting failed references, start with the first error message. The first failed reference can cause subsequent failed references. Search the test log "for unable to extract" to find the first error message. \n\nExamine the request that generated the response. The request contains a value that might need to be correlated. For example, the request might contain a username that must be unique to play back the test successfully. In that case, use a dataset to provide a list of unique username values. You might need to manually correlate a value by using the Test Data Sources view. Values that typically are correlated include timestamps, dates, ids, and other alphanumeric strings. \nIf you no longer need the data correlation mentioned in the message, remove that data correlation from the test.

RPXD0020E

Data
Correlation:
Failed
Extraction
\nReference[%1]\n
\nDetails:
\n
\nWe
were
unable
to
extract
the
value
for
the
reference
named
[%2],
with
the
regular
expression
[%3].
This
could
mean
a
later
request
will
fail.
Please
compare
the
response
in
the
test
log
to
the
corresponding
response
in
the

Explanation: The response received during playback is different from the response received when the test was recorded. The data correlation code was unable to use the regular expression expected value.

System action: None.

User response: Examine the request that generated the response. The request contains a value that might need to be correlated. For example, the request might contain a username that must be unique to play back the test successfully. In that case, use a dataset to provide a list of unique username values. You might need to manually correlate a value by using the Test Data Sources view. Values that typically are correlated include timestamps, dates, ids, and other alphanumeric strings. \nIf you no longer need the data correlation mentioned in the message, remove that data correlation from the test.

RPXD0021E

Dataset
%1
is
accessed
using
different
dataset
modes
by
different
tests.

RPXD0021W

Setting
variable
%1
to
value
%2.

RPXE0001W

%1

RPXE0010W

Engine
shutdown
problem
joining
workers

RPXE0011W

Failed
to
report
exception

RPXE0012W

Schedule
failed
to
load

RPXE0013W

Unable
to
create
test

RPXE0014W

Setting
log
level
to
%1

RPXE0015W

Attempt
to
add
object
to
Schedule
which
is
not
a
UserGroup

RPXE0016W

Virtual
User
%1
experienced
error
%2

RPXE0017W

Connect
timeout
for
action
%1
(%2)
user
%3

RPXE0018W

Read
timeout
for
action
%1
(%2)
user
%3

RPXE0019W

Connect
exception
for
action
%1
(%2)
user
%3

RPXE0021W

Read
exception
for
action
%1
(%2)
user
%3

RPXE0023W

Iterating
over
keys
exception

RPXE0024W

CancelledKeyException

RPXE0025W NullPointerException

RPXE0027W UserGroup
exception

RPXE0028W User
Group
%1
does
not
implement
createTesterWorkload()

RPXE0029W Worker
caught
throwable

RPXE0030W Connection
leak,
I/O
state
%1

RPXE0031W Exception
finishing
connection
for
action
%1
(%2)
user
%3

RPXE0033W

Finish
read
get
buffer
interrupted
for
action
%1
(%2)
user
%3

RPXE0035W

Finish
read
exception
for
action
%1
(%2)
user
%3

RPXE0036W

Engine
thread
startup
exception

RPXE0037W

Engine
request
to
report
exception

RPXE0038W

Exception
creating
cache
file,
cacheFileName:
%1,
extension:
%2,
dir:
%3

RPXE0039W

User
%1
experienced
exception
%2

RPXE0040W

User
%1
caught
exception
trying
to
report
severe
error.

RPXE0041W

Engine
hard
stop
after
%1
second
timeout

RPXE0042I

%1
received
request
to
stop

RPXE0043I

Forced
stop
of
action
%1

RPXE0044W

No
IP
address
was
found
for
the
local
host

RPXE0045W

Ignoring
invalid
network
interface
%1

RPXE0046W

Could
not
find
any
usable
network
interfaces

RPXE0047E

SyncPointSubsystem
Unknown
sync
point:
%1

RPXE0048W

%1
STOPUSERS
users=
%2
stagger=
%3
timelimit=
%4
active
users=
%5

RPXE0049W

%1
had
%2
non-
sampled
users
asked
to
stop
active
users=
%3

RPXE0050W

%1
had
%2
sampled
users
asked
to
stop

RPXE0051W

%1
after
wait
for
compliance
active
users=
%2
target=
%3

RPXE0052W

%1
abandon
user
%2

RPXE0053W

%1
abandoned
%2
users

RPXE0054W

%1
end
stop
%2
users
SUCCESS
active
users=
%3

RPXE0055W

%1
end
stop
%2
users
FAIL
active
users=
%3

RPXE0056W

%1
occurred
in
%2.
Message:
%3

RPXE0057E

Exception
while
reading
test
variable
initialization
file:
%1

RPXE0058E

Exception
while
initializing
virtual
users
test
variables.

RPXE0059E

Unable
to
get
Kerberos
ticket
from
KDC
for
server
%1.

RPXE0060E

Failed
to
load
test
from
'%1'
due
to
exception:
%2

Explanation: While trying to find and load class files required to execute the test a problem was encountered.

User response: See exception description for failure reason.

RPXE0100W

%1
terminated
due
to
exception:
%2

RPXE0102W

IKAction:
%1
(%2)
caught
Exception
in
preFinish()
for
%3
(%4)

RPXE0103W

IKAction:
%1
(%2)
caught
Exception
in
postFinish()
for
%3
(%4)

RPXE0104W

KernelChannel
connect(),
exception
while
trying
to
bind
to
local
address
%1:
%2

RPXE2501E

An
error
occurred
while
attempting
to
handshake
with
the
server
using
protocol
%1
and
cipher
suite
%2.
This
type
of
failure
is
often
related
to
a
mismatch
between
the
requested
protocol
or
cipher
suite
and
the
ones
the
server
is
expecting
or
may
be
related
to

Explanation: An SSL connection between a client and server is set up by a handshake, the goals of which are: To satisfy the client that it is talking to the right server (and optionally visa versa). Also, for the parties to have agreed on a cipher suite, which includes which encryption algorithm they will use to exchange data. These goals were not achieved.

System action: Execution ends because a secure connection cannot be established with the server.

User response: If the server requires a client digital certificate work with the server administrator to obtain one. If the server requires strong ciphers work with customer support to obtain the required and restricted ciphers.

RPXE2550E

The
digital
certificate
RCS
file
'%1'
was
not
found
or
was
corrupt:
%2

RPXE2552I

digital
certificate
alias

RPXE2900E

The
server
rejected
the
client's
digital
certificate.

RPXE2901W

The server closed the connection abruptly. This is probably due to an overloaded server or to a problem negotiating a digital certificate or cipher suite. Check the web server's SSL error log for more details.

RPXE4000W

Schedule
or
Test
not
found.
May
not
have
compiled.
-
%1

RPXE4001E

Runner
Exception
occurred

RPXE4002E

Communications
Error:
Invalid
Logging
Level

RPXE4003E

Communications
Error:
Invalid
TestLog
Level
for
%1
events

RPXE4004E

Communications
Error:
Invalid
Statistics
Level
or
Interval

RPXE4005E

Runner
Exception
occurred
-
See
problem
determination
log

RPXE4006E

Communications
Error:
Invalid
Dataset
information

RPXE4007E

Communications

Error:

No

communication

from

the

workbench

in

%1

milliseconds.

For

more

information,

see

the

Troubleshooting

section

of

the

online

help.

RPXE4008E

Attempt

to

change

statistic

interval

ignored.

RPXE4008l

Think:
requested
time
%1
milliseconds,
actual
time
%2
milliseconds

RPXE4009l

Delay:
requested
time
%1
milliseconds,
actual
time
%2
milliseconds

RPXE4010I

Schedule
completed.
See
Performance
Report,
Verification
Points
Report,
and/
or
Percentile
Report
to
further
evaluate
the
results
of
this
run
according
to
your
success
criteria.

RPXE4011E

Communications
Error:
Invalid
Stop
timeout

RPXE4013I

Additional
events
from
%1

RPXE4014E

Communications
Error:
Invalid
RunStagger
information
for
%1
(pairCount)

RPXE4015E

Communications
Error:
Invalid
RunStagger
information
for
%1
(pair
%2)

RPXE4016E

Failed
to
start
users
for
user
group:
%1.

RPXE4017I

Additional
execution
history
events
from
%1
are
available,
but
they
have
been
stored
separately
upon
user
request.
See
file
%2.
Refer
to
the
most
current
version
of
the
product
release
notes
for
information
on
how
to
access
and
view
them.

RPXE4018E

Failed
to
write
message
to
workbench
[%1]

RPXE4019E

Failed
to
remove
users
for
user
group:
%1.

RPXE4020E

Failed
to
add
users
for
userGroup:
%1
numUsers=[%2]
startId=[%3]

RPXE4021E

Failed
to
add
users
because
the
runner
is
not
in
a
runnable
state.

RPXE4022E

failed
to
add
desired
number
of
users

RPXE4023E

failed
to
reach
target
number
of
users
ramping
down

RPXE4024E

not
runnable
or
command
failed

RPXE4025E

failed
to
set
the
DataView
state
of
user
%1[%2]
to
%3.

RPXE4026E

DataView
command
%1
is
not
yet
implemented.

RPXE4027E

DataView
command
%1
is
not
recognized.

RPXE4028E

MessageEventFilter
command
parsing
error
in
token
%1[%2]
of
command
[%3]

RPXE4029E

The
testLog
message
event
filter
specified
by
[%1]
cannot
be
constructed.
This
filter
element
will
be
ignored.

RPXE4050I

Operating
System
Info:
name
[%1]
architecture
[%2]
version
[%3]

RPXE4100W

Cannot
open
execution
history
cache
file
[%1],
execution
history
will
not
be
cached

RPXE4101E

Error
closing
execution
history
cache
file
[%1]

RPXE4102E

Error
reading
%1
bytes
from
execution
history
cache
file
[%1]

RPXE4103E

Error
writing
%1
bytes
to
execution
history
cache
file
[%2]

RPXE4104E

Error
opening
execution
history
cache
file
[%1]
for
reading

RPXE4105E

Error
testing
execution
history
cache
file
[%1]
for
available
input

RPXE4106E

Unexpected
EOF
reading
%1
bytes
from
execution
history
cache
file
[%2]

RPXE4107E

Exception
processing
execution
history
event

RPXE4108E

Error
reading
execution
history
cache
file
[%1]

RPXE4109E

Error
writing
to
TestLog
cache
file
[%1]

RPXE4110E

Error
closing
TestLog
cache
file
[%1]

RPXE4111W

Cannot
open
testLog
cache
file
[%1]
for
random
access
writing,
the
testLog
may
contain
bad
data.

RPXE4112W

Error
removing
testLog
event
from
cache
file
[%1].
Writing
%2
bytes
at
offset
%3.

RPXE4120E

Error
writing
to
TestLog
[%1]

RPXE4150E

Error
opening
execution
history
annotation
file
[%1]

RPXE4151E

Error
writing
%1
bytes
to
execution
history
annotation
file
[%2]

RPXE4152E

Error
flushing/
closing
history
annotation
file
[%1]

RPXE4153E

Error
deleting
history
annotation
file
[%1]

RPXE4200W

Warning:
Statistics
delivery
thread
running
behind
statistics
interval
by
%1
milliseconds

RPXE4201W

Warning:
Statistics
delivery
thread
over
slept
by
%1
milliseconds

RPXE4202E

Error:
Statistics
delivery
thread
over
slept
by
%1
milliseconds

RPXE4203E

Error:
Statistics
collection
time
too
long:
%1
bytes
%2
milliseconds

RPXE4204W

Warning:
Statistics
collection
time
too
long:
%1
bytes
%2
milliseconds

RPXE4205E

Error:
Statistics
write
time
too
long:
%1
bytes
%2
milliseconds

RPXE4208E

Error:
Could
not
create
agent
measurements
file
%1.

Explanation: It is not possible to create a file on the file system.

System action: Unable to create a file. The agent measurements will not be available.

User response: You do not have the permissions on your file system or it is full.

RPXE4209I

Error:
Statistics
collection
thread
was
interrupted

Explanation: An error occurred which caused the interruption of the statistics collection.

System action: Statistics may be incomplete.

User response: Run the test again.

RPXE4210E

Error:
A
severe
error
occurred
when
processing
statistics.

Explanation: An exception occurred on the agent while processing statistics and/or sending them to the server.

System action: Statistics will be incomplete.

User response: Contact support.

RPXE4211E

Error:
A
severe
error
occurred
when
sending
statistics.

Explanation: An exception occurred on the agent while sending statistics to the server.

System action: Statistics will be incomplete.

User response: Contact support.

RPXE4212E

Error:
A
severe
error
occurred
when
closing
statistics.

Explanation: An exception occurred on the agent while completing the statistics processing.

System action: Statistics may be incomplete.

User response: Contact support.

RPXE4213E Statistics
 sub-
 system
 error:
 %1

Explanation: A severe error occurred during writing to the agent measurements file.

System action: The agent detailed measurements will not be available.

User response: Start a test execution again.

RPXE4214W Statistics
 sub-
 system
 warning:
 %1

Explanation: Warning message to the user during writing to the agent measurements file.

System action: The agent detailed measurements may be affected by a problem.

User response: Fix the problem given by the message or contact support.

RPXE4215E Statistical
 counter
 descriptors
 file
 not
 found:
 %1.

Explanation: Unable to find the counter descriptors file in the deployment directories.

System action: Statistics will not be available.

User response: Start the test again, contact support if the problem persists.

RPXE4215I

Statistics
sub-
system
message:
%1

Explanation: This message is displayed in debug mode.

System action: No action.

User response: You can report this message to the support.

RPXE4216E

Problem
in
statistical
counter
descriptors
file:
%1.

Explanation: The counter descriptors file has a problem.

System action: Statistics will not be available.

User response: Start the test again, contact support if the problem persists.

RPXE4217E

Submitted
value
%1
is
out
of
range
of
allowed
values
for
the
counter
type
%2.

Explanation: The value is out of the limits of the counter.

System action: Measurements and statistics for the specified counter will not be available.

User response: If you are a protocol developer, fix the problem. Otherwise, contact support.

RPXE4218E

In
order
to
use
this
method,
the
runtime
type
of
the
counter
must
be
either
STATIC
or
RATE.

Explanation: A protocol is using a legacy API to change the value of a counter.

System action: Measurements and statistics for the specified counter will be inaccurate.

User response: If you are a protocol developer, use a runtime counter type to STATIC or RATE. Otherwise, contact support.

RPXE4219E

Mismatch
between
runtime
type
%1
and
static
counter
type
%2.

Explanation: The type of the counter in runtime and in the statistic definition do not match.

System action: The runtime type will be applied.

User response: If you are a protocol developer, change the declared counter type, or the runtime type. Otherwise, contact support.

RPXE4220E

No
static
declaration
found
for
counter
%1.

Explanation: Unable to find a definition for the counter.

System action: The counter values will be ignored.

User response: Add a definition for the counter, or use an undeclared counter.

RPXE4221E

Attempt
to
create
an
undeclared
counter
%1
(type
%2)
over
a
declared
counter
of
a
different
type
(%3).

Explanation: An attempt to create an undeclared counter was made, but a counter declaration with another type already exists.

System action: The undeclared counter values will be ignored.

User response: Use another path for the undeclared counter that does not conflict with the existing declared counter.

RPXE4900I Test
 execution
 completed
 with
 no
 reported
 problems

RPXE4901I %1
 ERROR
 verdicts
 reported

RPXE4902I %1
 FAIL
 verdicts
 reported

RPXE4903I %1
 INCONCLUSIVE
 verdicts
 reported

RPXE4904I All
 reported
 verdicts
 PASSed

RPXE4905l

%1
ERROR
verdict
reported

RPXE4906l

%1
FAIL
verdict
reported

RPXE4907l

%1
INCONCLUSIVE
verdict
reported

RPXE4908l

%1
FAIL
verdict
roll-
up

RPXE4909l

%1
ERROR
verdict
roll-
up

RPXE4910l

%1
INCONCLUSIVE
verdict
roll-
up

RPXE4911l

%1
PASS
verdict
roll-
up

RPXE4912l

%1
ERROR
verdicts
reported
from
driver
%2

RPXE4913l

%1
FAIL
verdicts
reported
from
driver
%2

RPXE4914l

%1
INCONCLUSIVE
verdicts
reported
from
driver
%2

RPXE4915l

%1
ERROR
verdict
reported
from
driver
%2

RPXE4916l

%1
FAIL
verdict
reported
from
driver
%2

RPXE4917l

%1
INCONCLUSIVE
verdict
reported
from
driver
%2

RPXE4918l

duration

RPXE4920I

%1
was
successfully
invoked.
This
does
not
indicate
the
pass/
fail
verdict
of
the
test
itself,
only
that
the
invocation
of
the
test
was
successful.
Expand
to
inspect
verdicts.

RPXE4921I

%1
was
invoked.
This
does
not
indicate
the
pass/
fail
verdict
of
the
test
itself,
only
that
the
invocation
of
the
test
was
successful.
No
verdicts
will
be
reported
from
the
test.

RPXE4930I

The
%1
testLog
level
was
pushed
from
%2%3
to
%4%5.

RPXE4931I

The
%1
testLog
level
was
popped
from
%2%3
to
%4%5.

RPXE4932I

The
%1
testLog
level
was
changed
from
%2%3
to
%4%5.

RPXE4940I

Transaction
[%1]
started
%2
milliseconds
after
start
of
test
run.

RPXE4941I

Transaction
[%1]
stopped
%2
milliseconds
after
start
of
test
run.
Elapsed
time:
%3
milliseconds.

RPXE4942I

Transaction
[%1]
aborted.

RPXE4944W

Transaction
[%1]
is
already
started.

RPXE4945W

Transaction
[%1]
has
not
been
started.

RPXE4948W

Execution
Variables
-
Input

RPXE4950I

Null
user
group
name.

RPXE4952E

Unable
to
find
target
loop
named
'%1'.
Error
handler
did
not
complete
properly.

Explanation: The loop name specified in the loop handler does not exist.

System action: The user will not follow the loop error handler and will continue execution at the next action.

User response: Change the loop handler to point to an existing loop.

RPXE5301E

Error
encountered
while
loading
Native
Library:
%1

RPXE5305E

A
required
customer-
supplied
file
was
not
found.
Please
check
the
"external_files"
folder
and
your
installation
instructions
for:
%1

RPXE5330E

Unable
to
apply
dataset
swap:
%1

Explanation: An error occurred attempting to parse the data set swap command-line option.

System action: The data set swap will not occur.

User response: See the command-line usage to ensure the command syntax is correct.

RPXE5500W

Unable
to
apply
Open
Tracing
context.
The
root
Jaeger
span
will
be
unparented.
%1

Explanation: An error occurred when attempting to create an Open Tracing span context from the properties starting with OPENTRACING_CTX_.

System action: Jaeger logging will still occur but the root span will be linked to a parent span.

User response: Make sure the content of properties starting with OPENTRACING_CTX_ is correct.

RPXE5501W

Transaction
times
for
this
run
do
not
include
failing
transactions,
according
to
workbench
Test
Execution
preferences.

Explanation: A failing transaction will not be added to stats. This will only be logged once per transaction, but multiple instances may have failed.

System action: Execution will continue as normal. This is not an error condition.

User response: If this behavior is not desired, uncheck preferences at Test > Test Execution.

RRIT0001E

Environment
variable
INTEGRATION_TESTER_AGENT_HOME
not
set
to
HCL
OneTest™
API
Agent
installation
location,
or
does
not
contain
expected
RunTests(.exe)
program.

Explanation: The test execution cannot find the HCL OneTest™ API Agent.

System action: None.

User response: Set the environment variable INTEGRATION_TESTER_AGENT_HOME to point the root installation directory of the HCL OneTest™ API agent. This must be done on each location used in a schedule.

RRIT0002E

Error
unzipping
__IT_PRODUCT_NAME__
project.

Explanation: The __IT_PRODUCT_NAME__ project cannot be deployed.

System action: None.

User response: Verify that there is enough disk space on the executing location.

RRIT0003E

__IT_PRODUCT_NAME__
library
not
found.

Explanation: The library required to communicate with the __IT_PRODUCT_NAME__ agent is missing in the installation.

System action: None.

User response: Contact your support.

RRIT0004E

Error
processing
messages
received
form
the
__IT_PRODUCT_NAME__
client.

Explanation: A communication error has occurred with the __IT_PRODUCT_NAME__ agent.

System action: None.

User response: Try again, contact your support if problem persist.

RRIT0005E

Some
tag
values
are
missing.

Explanation: A value cannot be assigned to a tag defined in an __IT_PRODUCT_NAME__ test during execution.

System action: None.

User response: Verify that each tag of each __IT_PRODUCT_NAME__ test maps to a variable in the schedule or compound test.

RRITUI1002W

Open
__IT_PRODUCT_NAME__
resources
has
been
disabled
in
Test
>
__IT_PRODUCT_ACRONYM__
Integration
preferences

Explanation: User as disabled __IT_PRODUCT_NAME__ resource but want to open this kind of resource.

System action: None.

User response: Open Test > __IT_PRODUCT_ACRONYM__ Integration preference and enable open resources by checking __IT_PRODUCT_NAME__ is installed on this machine .

DCRC0001E

Missing
message
for
log
entry
'{0}'
in
class:
{1}

DCRC0002E

Cannot
get
Log
key
'{0}':
SecurityException
raised

DCRC0003E

Cannot
initialize
Log
key
'{0}'

DCRC0008W

Warning:
field
'{0}'
is
not
defined
in
class:
{1}

DCRC0009W

Warning:
cannot
get
check
message
versus
log
key
mapping
for
'{0}'
of
class
{1},
SecurityException
raised

DCRC0010E Unexpected
exception,
please
check
Error
Log
view:
{0}

DCUI0001E unexpected
exception

Explanation: An exception that could not be handled occurs during processing.

User response: Close rule editor and report exception to product support.

DCUI0003E Error
getting
persistent
property
'{0}'

DCUI0004E Error
setting
persistent
property
'{0}'

DCUI0006E Cannot
reload
resource
'{0}'

DCUI0007W

Failed
to
encode
model
to
clipboard.

DCUI0008W

Failed
to
decode
model
from
clipboard.

DCUI0009E

None
of
the
attribute
providers
own
attribute
id
'{0}'.

Explanation: Rule file refer to an unknown rule attribute id. File may be edited on a system having more protocol extension rather than current one.

User response: Rule file should not be edited on this product installation.

DCUI0010E

Missing
IRuleUIProvider
extension
point
for
'{0}'

Explanation: Rule file refer to a rule that is unknown on this product installation.

User response: Rule editor is able to display that rule on the tree but not able to edit it contents.

DCUI0011E

Missing
IConditionUIProvider
extension
point
for
'{0}'

Explanation: Rule file refer to a rule condition that is unknown on this product installation.

User response: Rule editor is able to display that rule condition on the tree but not able to edit it contents.

DCUI0012E

Cannot
save
editor
'{0}'

DCUI0013E

Missing
IRulePassUIProvider
extension
point
for
'{0}'

Explanation: Rule file refer to a rule pass that is unknown on this product installation.

User response: Rule editor is able to display that rule pass on the tree but not able to edit it contents.

DCUI0014E

Missing
IRuleArgumentUIProvider
extension
point
for
'{0}'

DCUI0015E

Missing
IRuleArgumentContainerUIProvider
extension
point
for
'{0}'

DCUI0016E

Try
rule
failed

DCUI0017E

Try
rule
failed:
'{0}'

DCUI0998E

Cannot
load
file
'{0}'

Additional error messages

You can find the additional error messages.

Address already in use

Address
already
in
use.

Explanation: Typically, this error message is displayed when all available TCP/IP ports have been exhausted.

System action: None.

User response:

- If the A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule. contains loops, move the loops into tests. If a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule. contains loops, at the beginning of a loop iteration each virtual user closes existing connections and opens new connections. This can cause the agent computer to exhaust all available TCP/IP ports. If a test contains loops, virtual users attempt to re-use existing connections. Re-use of existing connections can take advantage of keep-alive connections.
- Increase the number of TCP/IP ports available. The number of TCP/IP ports on a Windows™ computer is limited to 5000 by default. To increase the number of TCP/IP ports available on a Windows™ computer:
 1. Create the following Registry key: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\MaxUserPort`. This key does not exist by default.
 2. Specify the type as `DWORD`.
 3. Set the value to `65000`.
 4. Reboot the computer.

Browser profile in use

Your browser profile is already in use. You must close the browser or select another browser from Application to Record in Preferences.

Explanation: The browser configured for recording is already running, or a lock file exists that prevents the browser from running.

System action: The test run does not start.

User response:

- Close the browser configured for recording, and record again or select another web browser in the **Test From Recording** wizard.
- Search for and remove any `lock` files from the browser profile directory.

Cannot open test

Cannot
open
the
test
testname
because
it
contains
the
following
features
that
are
not
installed:
featurename.

Explanation: The test that you are attempting to open requires the installation of one or more protocol extensions.

System action: The test is not opened.

User response: To install a protocol extension:

1. Close the product.
2. Launch IBM® Installation Manager.
3. Click **Modify**.
4. Navigate to the **Modify Packages** window.
5. Select the required protocol extensions.
6. Follow the steps in the wizard to complete the installation of the protocol extensions.

A protocol license key is required to use protocol extensions other than HTTP. See the installation guide for more information on managing protocol license keys.

Connection closed

Web
server
computername
unexpectedly
closed
the
connection
while
in
the
process
of
retrieving
URI
URIname

Explanation: This message is displayed when the HTTP server being tested has become too busy and has closed the connection without completing the response.

System action: None.

User response: If you receive this error message multiple times for a particular HTTP request or when the server is not under load, contact the administrator of the server under test to determine potential causes of the behavior.

Dataset accessed using different modes

Dataset
datapoolname
is
accessed
using
different
dataset
modes
by
different
tests.

Explanation: Two tests are accessing the same dataset. These two tests use different access methods for the dataset.

System action: None.

User response: Examine your tests to find the dataset mentioned. Check the access mode for the dataset by double-clicking the dataset name in the test. The access mode will be **Random**, **Shuffled**, or **Sequential**. Ensure that the access mode is the same for all tests that use the dataset.

Error binding to port

Error
Binding
to
Port:*portnumber*
Exception:
java.net.SocketTimeoutException:
Accept
timed
out.

Explanation: The listener port for HTTP recording is in use by another application.

System action: None.

User response: Change the listener port for HTTP recording to a port that is not in use by another application. See [Changing HTTP recording preferences on page 865](#) for more information. On the **Browsers Recording** preferences page, edit the value of the **Proxy Recorder Local Port** field.

Error connecting to license server

Error
connecting
to
license
server.

Explanation: The computer could not connect to the license server.

System action: None.

User response: HCL OneTest™ Performance version 8.2 or later requires IBM® Rational® License Server version 8.1.1 or later. If you have used a previous version of HCL OneTest™ Performance with a license server, this error message can be displayed when the IBM® Rational® License Key Administrator points to an incompatible version of the license server. Ensure that the IBM® Rational® License Key Administrator has been configured to point to the correct version of the license server. Use Rational® License Key Administrator version 8.1.1 or later. On Microsoft™ Windows™, Rational® License Key Administrator 8.1.1 is installed in the IBM® Rational® program group. For more

information on licensing, see the *Installing and upgrading* section of the product documentation. If the error message persists, specify the port on which the license server is listening as the **Port ID** value. By default, the **Port ID** on the license server is 27000. HCL OneTest™ Performance comes with a compatible version of Rational® License Server. If you install another version of Rational® License Server on the same computer, this error can occur. Use the version of Rational® License Server provided with HCL OneTest™ Performance.

No local agent controller

Local
computer
is
not
running
the
Agent
Controller.

Explanation: The agent controller coordinates schedule playback between the workbench and agent computers. The agent controller is not running on the local computer.

System action: No tests run.

User response: Make sure that the agent controller is installed and running on the workbench computer. On Windows™ computers, the agent controller is a process called ACWinService. On Linux™ computers, the agent controller is a process called RAServer.

To start the agent controller, do one of the following steps:

- On Windows™, at a command prompt, enter this command:

```
net start "IBM Rational Agent Controller"
```

- On Linux™, change to the directory `AgentController/bin` in the product installation directory; then enter this command:

```
./RAStart.sh
```

Performance Test Errors were found in the project

Performance

Test

Errors

were

found

in

the

project.

Explanation: This error can occur when workspaces are shared on different computers or when the installation location of the product has changed since the project was created.

System action: None.

User response: To work around this error, clean up the Java™ build path and the generated Java™ source files.

1. Click **Window > Open Perspective > Resource** to open the **Resource** perspective.
2. Select the project in the **Project Explorer** view, and then right-click and select **Properties** to view the project properties.
3. Select the **Java Build Path** property.
4. Click the **Libraries** tab.
5. Select each entry that is displayed with a red X and remove that entry from the build path. If a referenced library is pointing to user-specific libraries that have moved or are not present, edit the entry so that it points to the correct location. Do not remove entries for user-specific libraries.
6. Click **OK**.
7. Expand the `src` folder in the project.
8. Remove all generated Java™ source files that are displayed with a red X. If the source file is user-specific, then edit the build path to point to the correct dependent libraries.

Test run aborted

Execution
failure.
No
status
received
from
location
computername
in
interval
seconds.

Explanation: The workbench has lost communication with one or more agent computers.

System action: The test run stops.

User response: For each agent computer:

- Check that the playback Java™ process is running. If it is still running, the problem might be on the workbench computer. Stop the Java™ process and all related typeperf and vmstat processes.
- Examine the problem determination log for error messages or exception messages.
- If the playback Java™ process is not running, search for `javacore.*` files. The contents of these files might help you determine the cause of the problem.
- Run tests again and monitor the memory size of the playback Java™ process. If the playback Java™ process is consistently running at its maximum heap size, the process might not have enough memory.

If the problem is not on an agent computer, check that the workbench computer has sufficient memory. To increase the available memory, either increase the workbench heap size or reduce the level and amount of execution history.

Test run aborted due to error

Virtual
users
have
exited
prior
to
stage
completion.

Explanation: In a schedule with multiple stages, at the end of a stage that is configured with n virtual users, there were fewer than n virtual users running. The schedule did not assign enough work to the virtual users to keep them active for the duration of the stage, or the virtual users stopped because of an a different error. The workload problem can occur if you use multiple stages but do not enclose the workload in an infinite loop.

System action: The test run stops.

User response: On the workbench computer, examine the workload for each user group in the schedule. Check that the entire workload is inside an infinite loop, so that the virtual users always remain active regardless of the stage duration. If the tests already use infinite loops, then the agent computers might have encountered errors while running tests. To determine why virtual users stopped before the stage completed:

- Examine the test log for error or exception messages.
- Increase the test log level and decrease the number of users. Try running the schedule again. Examine the test log for error or exception messages.
- Simplify the workload by running one user in one stage. Check to see whether the single user takes the expected amount of time to complete all actions. If not, examine the test log and problem determination log for error or exception messages.

Testgen completed with warnings

Testgen
completed
with
warnings.

Explanation: The response data has been truncated according to the setting in the HTTP Test Generation preferences.

System action: None.

User response: To adjust how response data is truncated, click **Window > Preferences > Test > Test Generation > HTTP Test Generation > Test Generation Options**, and edit the value of the **Save only the first 4KB of responses larger than** field.

Variable not initialized

Variable
variablename
has
not
been
initialized
for
this
test.

Explanation: A variable was used in a test, but the variable had never been initialized to a value.

System action: Depending on the value of the **Run-time error if variable not initialized** setting, the system will either do nothing, issue a warning, issue a test log error, or exit the test.

User response: Check the variable to determine where the initialization should have happened. Make sure that the test that contains the initialization of the variable occurs before the test trying to use the variable.

Chapter 11. Reference Guide

This guide describes, additional topics to gain more knowledge about HCL OneTest™ Performance.

Accessibility features

Accessibility features help users who have physical disabilities, such as visual and, hearing impairment, or limited mobility, to use the software products successfully.

Accessibility features are product dependent and might include one or more of the following aspects:

- Keyboard-only operation
- Screen reader usage
- Color and typeface preferences



Note: The accessibility features mentioned here apply to the Windows operating system. Some of these features might also work on Linux, but are not officially supported.

Accessibility compliance

The product documentation is published by using Oxygen XML WebHelp Responsive. To understand the accessibility compliance status for Oxygen XML WebHelp Responsive, refer to [WebHelp Responsive VPAT Accessibility Conformance Report](#).

Accessing UI elements

HCL OneTest™ Performance Rational® Service Tester for SOA Quality supports navigation in the UI by using different methods such as a mouse, keyboard, or touchpad.

You can use the keyboard keys such as **Tab**, arrow keys such as **UP**, **DOWN**, **LEFT**, and **RIGHT** to navigate to the different pages in the **Navigation** pane or to the different action labels in the right pane on the UI.

Keyboard shortcuts for performance and service testing

The keyboard shortcuts for performance and service testing are available when you record or edit a test or a schedule.

Key combination	Description
Ctrl+S	Save the test or schedule.
Alt+Shift+T, G	Generate a test from the selected recording (.recmodel) file.
Alt+Shift+T, R	Create a report (the test must be selected in the Test Navigator).
Alt+Shift+T, T	Test connection (a location must be selected in the Test Navigator).

Key combination	Description
Alt+Shift+X, B	Run the test (a test must be selected in the Test Navigator).
Alt+Shift+X, C	Run the schedule (a schedule must be selected in the Test Navigator).
Del	Delete the selection
Ctrl+Del	Delete the selection
Insert	Insert a new element (same as the Insert push button).
Ctrl+Insert	Add a new element (same as the Add push button).
Ctrl+Up Arrow	Move the element up.
Ctrl+Down Arrow	Move the element down.
Ctrl+Alt+<, Ctrl+Alt+>	Resize the test editor and schedule editor windows. The new size is retained when you reopen the window.
Ctrl+Shift+F1	During HTTP recording, insert a comment.
Ctrl+Shift+F2	During HTTP recording, insert a screen capture.
Ctrl+Shift+F3	During HTTP recording, insert a synchronization point.
Ctrl+Shift+F4	During HTTP recording, start a transaction.
Ctrl+Shift+F5	During HTTP recording, end a transaction.
Ctrl+Shift+F6	During HTTP recording, insert a split point.
Ctrl+Shift+F7	During HTTP recording, set the name of the current page.

The following keyboard shortcuts are available when you record Citrix performance tests:

Key combination	Description
Tab or Shift+Tab	Cycle the focus through the UI elements
Arrows	Select a push button
Space	Click a push button or toggle between selections

When you record Citrix performance tests and you work in image synchronization mode, you can use these keys:

Key	Description
Space	Set the origin of selection area
Arrows	Move the cursor
Enter	Select the image synchronization area and set the synchronization area (press twice)
Esc	Cancel the selection

General reference for performance testing

See these performance testing topics for general reference.

Data correlation rules

You can customize how data is correlated by using data correlation rules.

Rules that create elements

Create a built in data source

Inserts a built-in data source in the test.

Create a custom code

Inserts a custom code element in the test.

Create a dataset column

Creates a dataset column that can be used by substitution sites.

Create a reference

Creates a reference in data that matches a specified regular expression.

Create a substitution

Creates a substitution site in data that matches a specified regular expression.

Create a variable assignment

Inserts a variable assignment in the test.

Create a variable declaration

Creates a variable that can be used by substitution sites.

Rules that change elements

Encode a substitution

Specifies whether substitution fields are encoded or decoded.

Rename a data source site

Changes the name of a data source that matches a regular expression.

Rename a substitution site

Changes the name of a substitution site that matches a regular expression.

Replace reference regular expression

Changes the regular expressions that are used by references.

Unlink a substitution

Removes the links between substitution sites and references and other data sources.

Rules that find elements

Find a reference

Returns a reference. Add child conditions to specify the reference to find.

Find a substitution

Returns a substitution site. Add child conditions to specify the substitution site to find.

Find a variable

Returns a variable. Add child conditions to specify the variable to find.

Rules that remove elements

Remove a built in data source

Deletes data sources from the test. Add child conditions to specify the data sources to delete. If you do not add child conditions, this rule deletes all data sources in the test.

Remove a custom code

Deletes custom code elements from the test. Add child conditions to specify the custom code elements to delete. If you do not add child conditions, this rule deletes all custom code elements in the test.

Remove a reference

Deletes references from the test. Add child conditions to specify the references to delete. If you do not add child conditions, this rule deletes all references in the test.

Remove a substitution

Deletes substitution sites from the test. Add child conditions to specify the substitution sites to delete. If you do not add child conditions, this rule deletes all substitution sites in the test.

Remove a variable assignment

Deletes variable assignments from the test. Add child conditions to specify the variable assignments to delete. If you do not add child conditions, this rule deletes all variable assignments in the test.

Remove a variable declaration

Deletes variables from the test. Add child conditions to specify the variables to delete. If you do not add child conditions, this rule deletes all variables in the test.

Error conditions

Error conditions include verification point failures, connection failures, server timeouts, custom code alerts, custom code exceptions, and problems with data correlation. You can specify an action to take when the error condition occurs. The Errors report displays the error conditions and error behavior that occurred in a test or schedule.

Page Title Verification Point Failure [HTTP]

The returned title for the primary request for an HTTP page does not match the expected title. The default value of the expected page title is what is returned between the <title></title> tags during recording. See [Specifying the expected page title on page 292](#) for more information.

Response Code Verification Failure [HTTP]

The returned response code does not match the expected response code. You can specify an exact match or a relaxed match. See [Specifying the expected response code on page 293](#) for more information.

Response Size Verification Failure [HTTP]

The number of bytes returned does not match the expected number of bytes. You can control how closely the returned response size must match the recorded response size. See [Specifying the expected response size on page 294](#) for more information.

Content Verification Point Failure

The received data does not match the expected data. Content verification point controls are protocol-specific.

Connection Failure

The workbench or agent computers cannot connect to the server under test.

Authentication Failure

An attempt to log in to the server under test failed.

End of Dataset reached

The last row of the dataset is reached. See [Dataset overview on page](#) for more information.

Reference Extraction Failure

The response received during playback is different from the response received when the test was recorded. Data correlation failed because the regular expression that is associated with the reference did not match the expected value.

Substitution Failure

A reference for an expected data substitution is a null reference.

Server Timeout

The server under test does not respond before the timeout interval elapses.

Custom Verification Point Failure

A custom verification point did not return a Pass status after performing a verification written in Java™ code. See [Reporting custom verification point failures on page 683](#) for more information.

Custom Code Alert

Custom code reported an **RPTCondition.CustomCodeAlert** condition. The following code reports a custom code alert:

```
tes.getTestLogManager().reportErrorCondition(RPTCondition.CustomCodeAlert);
```

See the `ITestLogManager` Javadoc for more information.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**HCL OneTest Performance **API Reference**.

Custom Code Exception

The custom code in a test has an exception. By default, HCL OneTest™ Performance exits the user whenever there is an exception in custom code. For information on setting different actions, see [Error-handling behavior on page](#)

Related information

[Specifying error-handling behavior on page 297](#)

Resource monitoring data sources

Resource monitoring data can be captured or imported from a number of sources.

IBM® Tivoli® Monitoring

IBM Tivoli® Monitoring monitors and manages system and network applications on a variety of platforms and keeps track of the availability and performance of all parts of your enterprise network. IBM® Tivoli® Monitoring provides reports that you can use to track trends and troubleshoot problems.

Not all IBM® Tivoli® Monitoring agents are supported. Over 100 IBM® Tivoli® Monitoring agents are available from IBM® and non-IBM vendors. The following IBM® Tivoli® Monitoring agents are supported for resource monitoring data collection:

- Operating system agents
 - Monitoring Agent for Linux™ OS
 - Monitoring Agent for UNIX™ OS
 - Monitoring Agent for Windows™ OS
 - Monitoring Agent for z/OS®
- Application agents
 - Monitoring Agent for Citrix
 - Monitoring Agent for IBM® DB2®
 - Monitoring Agent for IBM® Tivoli® Composite Application Manager for WebSphere®
 - Monitoring Agent for IBM® WebSphere® Application Server
 - Monitoring Agent for IBM® WebSphere® MQ
 - Monitoring Agent for Oracle Database
 - Monitoring Agent for SNMP-MIB2 (only)

IBM® DB2® Monitoring

IBM DB2® collects information from the database manager, its databases, and any connected applications. The snapshot monitor captures the state of database activity at a particular point in time.

IBM® WebSphere® Performance Monitoring Infrastructure

IBM WebSphere® Application Server collects performance data and provides interfaces so that external applications can monitor that performance data. To help identify performance problems and help tune an environment that runs web applications, data is collected through the Performance Monitoring Infrastructure (PMI). The Performance Monitoring Infrastructure is the underlying framework in WebSphere® Application Server that gathers performance

data from various runtime resources, such as Java™ Virtual Machine (JVM) and Thread Pools, and application components, such as servlets and Enterprise JavaBeans™ (EJB) components.

Java™ Management Extensions

Java Management Extensions (JMX) can monitor performance characteristics of application servers and applications that are run on application servers. The following application servers support JMX monitoring:

- Apache HTTP Server
- Apache Tomcat
- JBoss Application Server
- Oracle WebLogic Server
- SAP NetWeaver

Java™ Virtual Machines also support JMX monitoring.

Oracle Database Metrics

Oracle Database collects metrics that are related to database health and workload.

UNIX™ rstatd

With the rstatd daemon, users can collect performance statistics remotely from networked UNIX™ (or Linux™) computers. The rstatd daemon collects statistics that are related to network, virtual memory, interrupt, disk, and processor usage.

Simple Network Management Protocol (SNMP) agents

The Simple Network Management Protocol (SNMP) is typically used to monitor network health, performance, and hardware. SNMP agents are software components that are installed on managed devices and collect management information.

Windows™ Performance Monitor

Windows Performance Monitor (PerfMon) collects data from performance objects. The Microsoft™ Windows™ operating system provides performance objects for the major hardware components: memory, processors, and so on. Each performance object provides specific performance counters. For example, the `Memory` object provides a `Pages/sec` counter that tracks the rate of memory paging. Other programs on the computer, including Internet Information Services (IIS) and Microsoft™ SQL Server, can install their own performance objects. For example, a mail server program might install a mail performance object. The specific counters depend on the version of the Windows™ operating system and on the additional programs that are installed on the computer.

Response time breakdown data sources

Response time breakdown data can be imported from a number of sources.

IBM® Tivoli® Composite Application Manager for Application Diagnostics

IBM Tivoli® Composite Application Manager for Application Diagnostics enables users to view the health of web applications and servers, then drill down to diagnostic information for specific application requests to identify the root cause of problems.

IBM® Tivoli® Composite Application Manager for Response Time Tracking

IBM Tivoli® Composite Application Manager for Response Time Tracking measures the level of service that the application delivers to users. It does this by monitoring the availability and response time that users experience at the client desktop. It works with a wide range of web-based, e-business, and Microsoft™ Windows™ applications that run in many different environments.

IBM® Tivoli® Composite Application Manager for WebSphere®

IBM Tivoli® Composite Application Manager for WebSphere® provides immediate problem determination, availability monitoring, and performance analysis for enterprise WebSphere® applications running on Windows™, UNIX™, OS/400®, and z/OS® environments. IBM® Tivoli® Composite Application Manager for WebSphere® monitors heterogeneous environments consisting of both mainframes and distributed systems.

IBM® Tivoli® Monitoring for Transaction Performance

IBM Tivoli® Monitoring for Transaction Performance is a centrally managed suite of software components that monitor the availability and performance of web-based services and Windows™ applications.

Related information

[IBM Tivoli Composite Application Manager for Applications](#)

UI preferences

Read the UI preferences topics.

HTTP preferences

You can change the product behavior by changing these HTTP-related settings.

HTTP protocol data view preferences

Preference settings control how protocol data is displayed when tests run.

Access the preference settings for the HTTP protocol data view. Click **Window > Preferences > Test > HTTP Protocol Data View**.

Render binary response data

Typically, you leave this box unchecked, because the data is generally unreadable and can cause temporary high processor usage when converted into text. If enabled, the Response and Browser pages of the Protocol Data view display unrecognized binary data.

Replay delay

During test debugging, when you replay one virtual user after the run is completed, specify the number of seconds that the Protocol Data view pauses between showing each page.

Enable real-time protocol data support for HTTP test

Typically, you leave this box checked and select whether you want to display the **Browser** tab or the **Event Log** tab by default; you can switch between these pages during playback.

Show the following page when launching HTTP test

Specifies which page is displayed when an HTTP test runs.

- **Browser:** Click to view rendered HTTP pages during playback, thus verifying that a test is behaving as expected. Because the protocol data is used, the **Browser** page might not render the contents exactly as a web browser would.
- **Event Log:** Click to see a line of summary information for each defined page of the currently running test. This summary includes a count of verdicts that did not pass, unexpected response codes, and other items of interest. Click an event to drill down for more detailed information.

Highlight Substitutions in Protocol Data View

This option visibly highlights substituted data in the **Request**, **Response Headers** and **Response Content** pages of the **Protocol Data View** when viewing test log or test editor elements that use data correlation.

HTTP recorder preferences

Preference settings control the behavior of the recording wizard.

Access the preference settings. Click **Window > Preferences > Test > Recording > HTTP Recording**. After changing a setting, click **Apply**.

Enable the HCL OneTest™ Performance toolbar in browsers

Click to install the annotation toolbar. This enables you to add comments and transactions, and to change page names during recording.

Verify annotation toolbar is installed before recording

Click to verify that the annotation toolbar is installed in the web browser before recording.

HTTP test editor preferences

The preference settings on the HTTP page of the test editor control how URLs are displayed in a test and how content verification occurs.

To access the preference settings for HTTP test editor, click **Window > Preferences > Test > Test Editor > HTTP Test**.

You can set the following preferences for the HTTP test editor:

Display decoded URLs whenever possible

Select to decode any encoded element in a URL. Decoding improves readability.

Hide HTTP request/response content larger than (kB)

Select to hide data larger than a specific size. The Content area in a response indicates the size of the hidden data and whether it is binary. To display hidden data, press Ctrl+Shift+Spacebar.

Show in all requests

Select to display the host and port information in every request in the **Test Contents** area of the test editor. A test often contains many server connections. When you clear this preference, it is easier to read a test.

Show on primary requests only

Select to display the host and port information in only the primary request for each HTTP page in the **Test Contents** area of the test editor.

Show when different from primary request

Select to display the host and port information in the **Test Contents** area of the test editor for requests that use a different connection than the primary request.

Skip responses with binary contents

Select to skip binary response data when you enable content verification points in a test. Content verification points verify whether specified strings are present in response data.

Create only in primary responses

Select to limit the creation of content verification points to primary responses when you enable content verification points in a test.

HTTP test generation preferences

Preference settings control how performance tests are generated, such as how tests will process verification points, data correlation, and generic protocols.

Test generation options

To access the preference settings for test generation options, click **Window > Preferences > Test > Test Generation > HTTP Test Generation**, and click the **Test Generation Options** tab.

Do not generate a new page if think time is less than

Enter the shortest time, in milliseconds, that the generator uses as a delay to emulate user think time for an HTTP page. If your tests contain fewer pages than expected, try a shorter interval.

Create a new page if delay between requests is greater than

Enter the longest delay, in milliseconds, that the generator allows between page requests. If this time is exceeded, a new page is generated. If your tests contain more pages than expected, try a longer interval.

Maximum request delay

Enter the longest delay, in milliseconds, that the generator allows before truncating HTTP requests. The requests are truncated on the generated test. The recorded test still contains the original values, and you can get them back by generating a new test.

Save only the first 4KB of responses larger than

Enter the limit of response data, in KB, that the generator saves. If a response is larger than the specified limit, only the first 4 KB of data is saved.

Suppress NSLookup() and use numeric IPs

Select this option to shorten test generation time. The disadvantage is that IP addresses in a test are less user-friendly than web page format (www.example.com).

Disable Page Cache Emulation during test generation

Select this option to disable page cache emulation. When page cache emulation is enabled, caching information in server response headers is honored. Additionally, requests are not submitted to the server for content that is confirmed by the client as fresh in the local cache. Page cache emulation is enabled by default.

Enable domain review before test generation

Clear the check box to not show the test generation page to select specific domains to be added to the test. By default, in addition to the domain that you intend to record, other domains linked to the original domain are also recorded.

Remove HTTP request delays from page response times

To not include the client delays in the page response times for the test or A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule., select this check box. By default, the page response times include delays to represent processing time caused by clients such as a web browser. Sometimes this delay could exceed the logical limit causing page response times to increase drastically.

Use Legacy Test Generator

Select this option if you have been instructed to use the legacy HTTP test generator.

Automatically include verification point of

Click to specify the types of verification points to be automatically included. If a check box for a verification point is selected, the code and edit controls for this type of verification point are generated in all tests. Verification points can also be enabled or disabled within specific tests.

Relaxed

Response codes that are in the same category (for example, 200, 201, 203, 209) are considered equivalent. An error is reported if the response code is not in the same category.

Exact

An error is reported if the response code does not match the recorded value exactly.

Accept sizes for primary request within

If you are automatically generating response size verification points, click to specify the acceptable size range for primary requests. No error is reported if a response is within the specified percentage above or below the expected size. By default, for primary requests, **HTTP response size** verification points use range matching.

Data correlation

To access the preference settings for data correlation, click **Window > Preferences > Test > Test Generation > HTTP Test Generation**, and click the **Data Correlation** tab.

Automatically correlate host and port data

By default, host and port data is correlated automatically. If tests in a previous release have significant manual correlations, or you are using proxies, the migration of the replace-host functionality feature is likely to fail during playback. In this situation, clear the check box. When you reopen your tests, they will not have the automatic correlation feature in them.

Automatically correlate URL pathname if redirected by response

Specifies whether URL path names are correlated if they are redirected by a selected response code. If a check box for a response code is selected, the test generator performs correlations for that response code. This option applies only to responses that are redirects, with a status code between 300 and 399.

Automatically correlate Referers

By default, the Referer field in an HTTP request header is correlated automatically. Clear the check box if you plan to correlate Referers manually. If you run tests against servers that do not require a Referer field, clearing this check box reduces the number of correlations performed when the test runs, and can increase user throughput.

Enable all other data correlation

By default, request and response data is correlated automatically. Clear the check box to disable automatic data correlation of request and response data. Consider clearing the check box if you create your own data correlation rules in the rules editor.

Create substitutions for empty strings

Select this check box to correlate empty strings. For example, strings such as spouse name or middle initial sometimes become important to correlate. However, correlating empty strings increases the time to generate a test.

Optimize automatic data correlation for execution

Specifies the characteristic that tests are automated for.

- With the **Accuracy** setting (the default), many references with an identical session ID value are created and the value of each session ID is substituted from the nearest previous reference.
- To make a test run faster by reducing the number of references that are created during automatic data correlation, change the optimization to **Efficiency**. For example, consider a

test where a session ID, which is assigned when a user logs in, is included in every subsequent request in the test. With the **Efficiency** setting, all session IDs are substituted from a single previous reference. The downside of this setting is that it can result in incorrect correlations. For example, a request that contains the `Joe Smith` string might be incorrectly correlated with a request that contains the `Joe Brown` string.

URL rewriting for execution

Specifies how web addresses (URLs) are rewritten during test execution. When correlating data, the test generator replaces part of a URL request string with a value that the server returned in response to a previous request.

- **Automatic** (default): The test generator automatically determines when rewriting the entire URL during substitution will facilitate test execution.
- **On**: Select to rewrite URLs in every instance of data correlation. This produces larger tests that take longer to run. Try this setting if your tests fail unexpectedly.
- **Off**: Select to manually correlate the instances where URL rewriting is needed. This setting might cause execution errors.

URL encoding for execution

With this option, you can control the encoding of the URLs. If you set it to Automatic, the tool detects the encoding that already exists in the test and applies it to the substitution site. If you set it to ON, the tool always encodes the substitutions according to the encoding standards. If you set it to OFF, no encoding occurs.



Note: To turn data correlation off entirely or to set whether names are automatically generated for data correlation references, click **Window > Preferences > Test > Test Generation > HTTP Test Generation**, and click the **Data Correlation** tab.

Data correlation types

To access the preference settings for types of data correlation, click **Window > Preferences > Test > Test Generation > HTTP Test Generation**, and click the **Data Correlation Types** tab.

Data Correlation Types

Specify when to generate data correlation constructs. With the **Automatic** setting, the test generator creates the required constructs where needed. If the test does not contain the required constructs, change the setting to **On**, which will always perform data correlation. If tests do not require a specific construct, select **Off**, which has the additional benefit of improving performance on subsequent test generation.

Jazz Foundation Services

The **On** and **Automatic** options enable data correlation for Jazz applications that use REST storage or query APIs from Jazz Foundation Services. An example of such an application is Rational DOORS Next

Generation. Although data correlation does not typically apply to browser-based Jazz web clients, it may be useful for other HTTP client-server applications that use REST services and the Atom Publishing Protocol for updating web resources.

Jazz Web Applications

The **On** and **Automatic** options enable data correlation for Jazz web applications that use the Jazz Foundation web UI framework. Examples of these web applications are the web interfaces for Rational Quality Manager and Rational Team Concert. Data correlation can also be useful for other web applications that contain javascript that employs JSON for client-server data exchange. This is a common practice with DOJO- and AJAX-based applications.

JSON

To perform data correlation on web applications that uses JSON framework, ensure that Automatic or ON is set to the JSON entry.

Prioritize correlation based on ID

Select **On** to correlate HTML response code based on its ID attribute. Generally, the HTML response code after the recording would appear as `<input type="username" name="User" id="aaa" value="John"/>`. Some applications dynamically update the *name* attribute. Therefore, when you play back the test, the HTML response code would appear as `<input type="username" name="idt020" id="aaa" value="John"/>`. Because the *name* attribute is changes dynamically, data correlation does not occur and the playback fails. When this option is turned on, the ID attribute is considered as the basis to correlate the *name* attribute in the request and to locate the *value* attribute.

SAP test preferences

You can change the product behavior by changing these SAP-related settings.

SAP test editor preferences

The SAP test editor preferences control the specific behavior of the test editor with SAP test suites.

To access the SAP test editor preferences, click **Window > Preferences**, expand **Test**, expand **Test Editor**, and click **SAP Test Editor**. After changing a setting, click **Apply**.

SAP Protocol Data View

These settings specify how the SAP Protocol Data view is displayed.

SAP GUI object highlight color

This setting specifies the color of the frame that highlights selected objects on the SAP GUI Screen page of the SAP Protocol Data view. By default, the highlight color is red.

Automatically set focus on SAP Protocol Data view

When enabled, this option automatically ensures that the SAP Protocol Data view is displayed each time an element is selected in the test editor. Disable this option if

you want to hide the SAP Protocol Data view or remove it from the Performance Test perspective. This option is enabled by default.

SAP recording preferences

Test recorder preferences control the default settings for recording SAP tests.

To access the SAP Test Recorder preferences, click **Window > Preferences**, expand **Test**, expand **Recording**, and click **SAP Recording**. After changing a setting, click **Apply**.

Screen capture options

These settings specify how the test recorder handles the screen captures that are shown in the **SAP Protocol Data** view.

None

No screen captures are recorded. This saves disk space, but disables the ability to create events or verification points from the SAP Protocol Data view.

On SAP screen entry

Screen captures are recorded each time a new screen is displayed in the SAP GUI. The recorded screen capture shows the initial state of the screen, before user input. This option is enabled by default.

On SAP screen exit

Screen captures are recorded each time a request is sent to the SAP server. The recorded screen capture shows the final state of the screen, after user input.

Both

Screen captures are recorded when a new screen is displayed in the SAP GUI and when the request is sent to the SAP server. The SAP Data Protocol view displays the final state on the send request elements and the initial state on all other events.

Select a SAPLOGON configuration file

The `saplogon.ini` configuration file provides a list of SAP system names that are displayed in the SAP recorder wizard. Use this setting to change the location of the `saplogon.ini` file.

SAP test generation preferences

Test generation preferences control how SAP tests are generated, such as how tests will process verification points, data correlation, and the default settings for generated test elements.

To access the SAP Test Generation preferences, click **Window > Preferences**, expand **Test**, expand **Test Generation**, and click **SAP Test Generation**. After changing a setting, click **Apply**.

Automatic Generation

The following settings specify test elements that are automatically generated after recording the test.

Use connection by string

When enabled, tests are generated with the connection by string launch method instead of using the SAP Logon program. This option is enabled by default.

Verification points for SAP screen titles

When enabled, this option generates verification points on screen titles with each SAP screen. This option is disabled by default.

Verification points for SAP request response time threshold

When enabled, this option generates verification points on the response time of the SAP server. If the server response time is above the specified threshold, the test produces a failed verification point. This option is disabled by default.

Calculate threshold from recorded (%)

This specifies the default response time threshold that is calculated when response time verification points are generated. The threshold value is calculated as a percentage of the actual response time that was measured during the recording. By default, the response time threshold is generated with a value of 120% of the recorded response time.

Default request timeout [ms]

Specify a timeout value for a request to ping the server. When the request is timed out, it no longer pings the server for that request.

GUI on execution

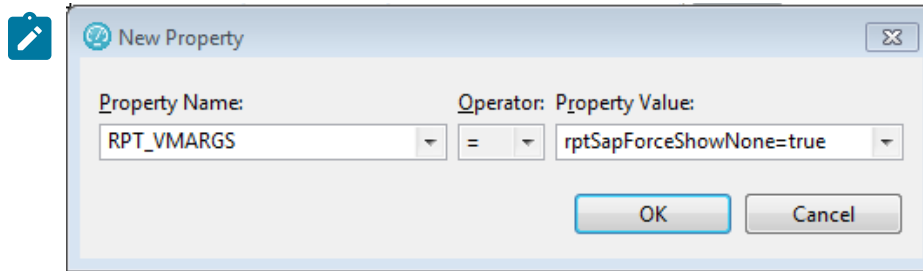
During test execution, it might not be desirable to display the SAP GUI. Hiding the SAP GUI improves the performance of the virtual users. This setting specifies the default behavior when the test is generated. However, you can change this setting in the test editor by selecting the SAP test element.

Hide GUI during execution

When selected, all instances of the SAP GUI are hidden. In some cases, modal dialog boxes from the SAP GUI can flash briefly on the screen. This is the default setting.



Note: If you run a test in the hidden mode and the test fails due to modal dialog boxes or pop-up windows in transactions, you must add the `RPT_VMARGS` property with value set to `rptSapForceShowNone=true` in the Location property.



Show GUI for only one virtual user

When selected, the SAP GUI is displayed only for the first virtual user. All other instances of the SAP GUI are hidden. This allows you to monitor the execution.

Show GUI for all virtual users

When selected, the SAP GUI is displayed for all virtual users.

Password prompt

Specifies behavior of the password request.

Prompt me for password when generating test

When enabled, a password is requested at the end of the recording session. If disabled, the password is recorded with an empty string. The recorder cannot record the password during the test. Therefore, if this option is disabled, the test uses an empty string for the password.

Citrix recorder preferences

Citrix recorder preferences control the behavior of the recording wizard.

To access the Citrix Recorder preferences, click **Window > Preferences**, expand **Test**, and click **Citrix Recording**. After changing a setting, click **Apply**.

Screen capture options

These settings specify how the test recorder performs screen captures of the Citrix desktop during recording.

No automatic screen capture

Select this option if you do not want the test recorder to record screen captures automatically. When this option is selected, you can still record screen captures manually. This option is selected by default.

Capture screen every

Select this option to automatically record a periodic screen capture and specify the time between captures.

Capture screen on window creation

Select this option to record a screen capture each time a window object is created in Citrix.

Exclude tooltips

When **Capture screen on window creation** is selected, enable this option to prevent creating a screen capture each time a tooltip event is displayed during the recording. If this option is disabled, screen captures are recorded when tooltips are displayed.

Capture screen on image synchronization

Select this option to ensure that a screen capture is recorded each time an image synchronization is recorded.

Citrix test editor preferences

Citrix test editor preferences control the test editor for Citrix performance tests.

To access the Citrix Test Editor preferences, click **Window > Preferences**, expand **Test**, expand **Test Editor**, and click **Citrix Test Editor**. After changing settings, click **Apply**.

Image Previews

These settings specify how screen captures are displayed in the test editor.

Fit screen to visible area

Select this option to automatically fit screen captures to the available area in the test editor. If disabled, the screen capture will be the actual size, which might require scrolling. This option is enabled by default.

Draw only last window

Select this option if you want to display only the current window in mouse sequence actions. When disabled, all recorded windows are displayed. This option is disabled by default.

Mouse Sequence

These settings specify how mouse sequences are displayed in the test editor.

Display mouse sequences for

This option specifies how you want to display previous, current, or all mouse sequences in the current mouse sequence.

Current® sequence

Only the current mouse sequence is displayed in the test editor. This option is selected by default.

Previous and current sequences

The current mouse sequence is displayed with any previous mouse sequences of the current window.

All sequences

All mouse sequences of the current window are displayed simultaneously.

Current® mouse sequence color

This option specifies the color of the currently selected mouse sequence.

Current® mouse sequence bold

Select this option if you want to display the current mouse sequence in bold. This option is selected by default.

Mouse move sequence color

This option specifies the color of mouse-move sequences when previous or all sequences are displayed.

Mouse drag sequence color

This option specifies the color of mouse-drag sequences when previous or all sequences are displayed.

Window color (when screen capture is not available)

This option specifies the color of a rectangle that represents the current window if there is no screen capture.

Citrix test generation preferences

Test generation preferences control how Citrix performance tests are generated, such as how tests will process verification points, data correlation, and options of the generated test elements.

Recording Optimization Options

These settings specify how mouse and window events are interpreted in the generated test.

Window activate recording

Specify whether to record no, last, or all window-activate actions when a sequence of similar actions is detected.

- **none** disables recording of window-activate events.
- **last** records only the last of an uninterrupted sequence of window events. This eliminates redundant window-activate actions from the recording.
- **all** records all events of the sequence.

Mouse move recording

This setting specifies which mouse move events are recorded. **Relevant** is the default setting.

- **All** records an uninterrupted sequence of mouse movements in the generated test.
- **Relevant** records only the mouse movements that generate a response, such as hover text.
- **First and last** records a simplified mouse-move action.

Automatic Generation

These settings specify test elements that are automatically generated after recording the test.

Verification point on every window title change

When enabled, this option generates a window title verification point whenever the caption changes. If this option is disabled, the window title is verified only when a new window is created. This option is disabled by default.

Response times for main windows

When enabled, this option generates response time measurements for all recorded main window-create events. A main window is a window that is created at the top level of the test contents tree and contains user actions. The generated response time measurement starts with the keyboard or mouse action that immediately precedes the window-create event. This option is enabled by default.

Window event synchronization criteria

Use this option to disable window recognition on the window position, size, or title. Disable any of these options if the test produces synchronization timeouts because a window changes its position, size, or title between or during test runs.

Default Test Execution Delays

This page specifies the default keyboard and mouse delays for the test client. Do not change these settings unless you are experiencing problems with events that do not run correctly.

Synchronization timeout delay

This is the delay after which a timeout error is produced when a window event or an image synchronization element is not recognized during test runs. The default value is 15000 milliseconds. The specified delay is for synchronizations that are set as conditional. Mandatory synchronizations use a delay of three times the specified delay. Optional synchronizations use a fixed delay of 2 seconds.



Note: In the generated test, the **Override synchronization timeout** for a particular window creation event will be enabled with the corresponding recorded time only if it is greater than what is specified in this preference.

If think time is under x ms, then replace with

If the delay between two events is above the specified limit, then it is handled as a think time. If the delay is below the limit, then the test generator replaces the think time with one of the following delays. The think time is the delay spent by a virtual user before performing an action. The default limit is 20000 milliseconds.



Note: In the generated test, the think time for a particular user action will be enabled only when the recorded think time is greater than the value specified for this preference.

Delay between mouse down and mouse up in a click

This is the default delay used to generate a mouse click action using a mouse down and a mouse up action. The default value is 20 milliseconds.

Delay between two mouse clicks in a double click

This is the default delay used to generate a double-click action using two mouse clicks. The default value is 50 milliseconds.

Delay between key down and a key up in a stroke

This is the default delay used to generate a key-stroke action using a key-down and a key-up action. The default value is 20 milliseconds.

Delay between two keyboard strokes in a text input

This is the default delay used to generate a text input action using multiple key stroke actions. The default value is 50 milliseconds.

Default OCR settings

This page specifies the settings for text extraction by optical character recognition in image synchronizations. You might need to experiment with various settings to obtain good results. These settings define the default behavior for new image synchronizations. You can change the behavior for individual image synchronization elements by changing the **OCR settings** in the test editor.

OCR default language

This is the language of the dictionary that is used to recognize words for the application that you are testing. This setting defines the subset of languages that will be available in image synchronization elements in the test editor.

OCR default zoom factor

This is the enlargement factor that is applied to the image. The default setting is medium for standard font sizes. Increase the zoom factor to improve recognition of smaller fonts or decrease for larger fonts.

OCR default brightness

This is the brightness level from 0 to 250 that is applied to the image. The default setting is 70 for text with normal contrast. Increase the brightness setting to improve recognition of darker images or decrease for lighter images.

OCR default recognition rate

This is the rate of recognition that is required for the extracted string to match the expected text. Decrease the recognition rate to tolerate a proportion of mismatching characters in the recognized text. The default is 100%, which means that an exact match is required.

Socket Test Generation preferences

With the socket test generation preferences, you can change how the test recorder generates new socket tests.

Filters

Select connections in this list to be *excluded* from the generated test. Click **Add Application** to filter connections from a specific program. Click **Add Host** to filter connections to a specific host.

Strategies

Select a change strategy or click **New** to add a change strategy. Click **Settings** to open the **Socket Strategy Settings** editor.

Socket Strategy Settings Editor

Use this editor to define rules for merging and handling large numbers of send and receive elements in a socket test.

Send elements**Merge consecutive send elements**

Select this option to merge together all the consecutive socket send elements that use the same connection.

Manipulate data with custom code

Select this option to force all the selected send elements to enable the **Manipulate data with custom code** setting with the specified **Class name** of a custom Java™ class that uses the API to process data in the socket send element.

Receive Actions**Do not merge**

Select this option to keep receive elements unmodified as they are initially recorded.

Merge consecutive receive elements

Select this option to merge together all the consecutive socket receive elements that use the same connection.

Keep only last receive element

Select this option to discard all multiple consecutive receive elements except the last one recorded.

Response timeout

The maximum delay (in seconds) to receive the first byte of the response. If no data is received before the end of the response timeout delay, the receive action produces an error in the test log. The response timeout counter starts when the receive action starts after the think time; the counter is interrupted when the first byte is received.

End policy

This option specifies when to stop receiving data and to move to the next test element.

- **Receives exact number of bytes:** The receive action stops when the recorded number of bytes is received. Specify a **Timeout** (in seconds) after which the receive action produces an error in the test log, if the correct number of bytes is not received. If **Link data size** is enabled, the receive action expects the number of bytes displayed in the **Data** area. If **Link data size** is disabled, the receive action expects the number of bytes displayed in **Bytes**. This is the default setting
- **Receives until end of stream:** The receive action stops when the connection is closed by the remote computer. If **Accepts empty response** is selected, then the reception of a single byte is *not* required and the **Response Timeout** is ignored. Specify a **Timeout** (in seconds) after which the receive action produces an error in the test log, if the correct number of bytes is not received.
- **Matches a string:** The receive action stops when a specified sequence of bytes is received. Specify a **Timeout** (in seconds) after which the receive action produces an error in the test log, if the correct number of bytes is not received.
- **Recognizes a regular expression:** The receive action stops when a sequence of bytes that matches a regular expression is received. Specify a **Timeout** (in seconds) after which the receive action produces an error in the test log, if the correct number of bytes is not received.
- **Delegated to custom code:** The receive action stops when a condition is met in a custom Java™ class. This setting allows great flexibility, but requires coding of a custom Java™ class following the HCL OneTest™ Performance extension API. Click **Generate Code** to generate a template based on the API or **View Code** to open the specified class in the Java™ editor.

Except when the **Receives until end of stream** policy is in force, receive actions produce an error in the test log when the connection is closed by the remote computer.

Timeout

For end policies that have a **Timeout** setting, this setting specifies a delay (in seconds) after which the receive action produces an error in the test log if the end policy criteria is not met. The timeout counter starts when the first byte is received.

Citrix monitoring panel reference

The Citrix monitoring panel is an optional panel that displays detailed information and control commands for each virtual user during the run of a schedule. When enabled, the Citrix monitoring panel is available during the run of a schedule.

Monitoring Panel

This panel displays information about the execution of each virtual user.

Pool Name

Displays the name of the virtual user pool. There is one pool per location and user group.

Active Virtual Users

Displays the number of virtual users currently active. This value is updated permanently during the run.

User Action Rate

Displays the number of Citrix user key or mouse actions that were simulated during the last 5 second interval.

Total Elapsed Time

Displays the total time elapsed since the start of the schedule run.

Current® Action

Displays the last user action executed in the test.

Timeouts

Displays the number of synchronization timeouts for the virtual user. The color represents the status of the timeout:

- Green: ok.
- Yellow: a timeout occurred on a conditional synchronization.
- Red: a timeout occurred on a mandatory synchronization.

Elapsed Time

Displays the time elapsed since the start of the virtual user run.

Status

Displays the execution status of the virtual user.

Go To

Click to display the Citrix session of the selected virtual user.

Pause or Play

Click to pause or resume the execution of the selected virtual user. You can also pause the execution by setting breakpoints in the test.

Step

When the test is on pause, click to execute each user input action in the test, step by step. To pause test execution, you can either click the **Pause** button or set breakpoints in the test. Click **Play** to resume the test.

Interact

When the test is on pause, click to allow manual actions in the virtual user session. Use this feature if a test fails to synchronize or gets stuck in an unexpected state. To pause test execution, you can either click the **Pause** button or set breakpoints in the test. Click **Play** again to resume the test execution at the point where it was paused.

Stop

Click to stop the execution of the selected virtual user. When all virtual users are stopped, the schedule ends.

Related information

[Enabling and disabling the Citrix monitoring panel on page 654](#)

[Debugging tests with the Citrix monitoring panel on page 655](#)

Proxy recording preferences

Use these preferences to specify a list of endpoints to ignore when recording a test with the proxy recorder.

To access the proxy recording preferences, click **Window > Preferences**, expand **Test**, expand **Recording**, and click **Proxy Recording**. After changing a setting, click **Apply**.

Ignore traffic to the following destinations during recording

Select this option to enable filtering of HTTP or HTTPS requests to specific endpoints. Click **Add** to add a hostname or IP address to the list of filtered endpoints. You can use the asterisk (*) character to specify a wild card.

Test editor preferences

The General preferences control what happens when you move test elements, and how digital certificates are substituted. The Colors and Fonts preferences control how the editor displays dataset and correlation data. The Search and Replace preferences control the behavior of search and replace.

General

Click **Window > Preferences > Test > Test Editor**, and click the **General** tab.

Move selected items into new transactions

Controls whether selected elements are automatically moved into new transactions, or whether you are prompted each time to move them.

Move selected items into new IF/ELSE blocks

Controls whether selected elements are automatically moved into new conditional blocks, or whether you are prompted each time to move them.

Move selected items into new loops

Controls whether selected elements are automatically moved into new loops, or whether you are prompted each time to move them.

Move selected items into new random selectors

Controls whether selected elements are automatically moved into new random selectors, or whether you are prompted each time to move them.

Keep children elements

When you delete a test element that has children elements, controls whether the test element children are also deleted, or whether you are prompted each time to delete them.

Automatically dataset certificate names

Controls whether and how digital certificate names are included in dataset.

Automatically adjust "once per user" dataset option

Controls whether and how digital certificates in dataset return dataset rows to a particular virtual user.

Make user-defined strings available to all tests

Click to save user-defined strings for content verification points in the workspace (and thus make them available to other tests). Click **Clear saved** to delete all saved strings.

Colors and Fonts

Click **Window > Preferences > Test > Test Editor**, and click the **Colors and Fonts** tab.

Here are some tips for working with these settings:

- To change a color, click the color box. From the color palette that opens, choose a different color.
- To save the current settings without closing the window, click **Apply**.
- To restore the settings to their factory defaults, click **Restore Defaults**.
- To save and close, click **OK**.

Information background

Click to change the highlighting shade that distinguishes page requests that contain dataset candidates, data in dataset, or correlated data.

Disabled Elements

Disabled Elements: Enables you to select the prefix and the color for the test elements that you manually disable. Click **Display nested** to precede nested elements with a double prefix (//). If you first disable a test element and then disable an entire test, the test element prefix is //. Clear **Display nested** to display all elements, whether nested or not, with a single prefix (/).

Use color to mark test elements with errors

Click to display in red test elements that have errors.

Appearance color options

Shows the current color settings. Select an element, and then click **Foreground** or **background** to change the settings.

Inline highlighting preview

Enables you to inspect the settings before you actually set them.

Search and Replace

Click **Window > Preferences > Test > Test Editor**, and click the **Search and Replace** tab.

Save search text

Click to retain the search text from session to session.

Save selected search types

Click to retain the test elements to search (pages, requests, responses, loops, and so on) from session to session.

Enable decorations on search results

Click to differentiate between visited and unvisited search results. For example, if you search for a name, and then click the text in the search result to locate it in the test, the search result will be marked in the color and the text that you specify.

Report preferences

Report preference settings apply to all protocols.

Test report preferences

The preference settings for test reports control such preferences as the typeface, color, and graph style of reports, and whether a Compare report is automatically launched when a staged run completes. You can also display a warning when changing Page Percentile report options will cause data to be lost.

To access the Performance Test Reports preference settings, click **Window > Preferences > Test > Performance Test Reports**.

You can set the following preferences for test reports:

Default Result Action

Set the report or log viewer to be displayed after a test or schedule is run, or when a prior result is opened from the Test Navigator view.

Default report preferences

Use this page to select the default report that opens during a run. Typically, you select **Determine default report based on protocols in test**, which determines the protocols that you are testing, and automatically opens the appropriate protocol-specific reports. Select a specific default report to display a customized report or if the default reports do not meet your needs. Note, however, that you will have to change this setting when you record other protocols.

Open the Default Report Preferences page. Click **Window > Preferences > Test > Performance Test Reports > Default Report**.

Export report preferences

Use this page to automatically export reports to a comma-separated-values (CSV) file at the end of a run. The CSV file is useful when you run a schedule from the command line because you can automatically export results without opening the workbench. The CSV file contains metadata about the test run, a blank line, and the report counter data. Simple CSV format contains only the last data value in the run. Full CSV format contains all data values for every sample interval during a run.

Open the Export Report Preferences page. Click **Window > Preferences > Test > Performance Test Reports > Export Reports**.

Web report preferences

Preference settings control access of reports from external web browser.

To access the preference setting for web report, click **Window > Preferences > Test > Performance Test Reports > Web Reports**.

You can set the following preferences for the web reports:

Allow remote access from a web browser

Select this check box to allow access to reports from a web browser.

Allow control of schedule execution from the web browser

Select this check box to control schedule execution from a web browser.

No security is required to access reports

Click this option to allow access to reports without login credentials. Specify a port number.

Security is required to access reports

Click this option to provide an authentication layer for accessing reports. Specify a port number and provide the login credentials.

Percentile analysis preferences

Use this view to customize the percentiles that are reported in the Page Percentile report or to customize the requirements on a percentile response. The defaults, 85, 90, and 95, are sufficient for most purposes. However, if you need to report on a different percentile set, or to set a different percentile requirement, you can edit the percentiles or add new percentiles.

Open the **Percentile Analysis Targets preference** page. Click **Window > Preferences > Test > Percentile Analysis Targets**.

Test editor references

To understand the relevance of different UI fields in the test editor of various test extensions, read these topics.

HTTP test editor reference

In HTTP testing, the test editor information is divided into five categories. This section describes the fields in each category that can be edited manually.

About this task

This section focuses on low-level editing tasks that experienced performance testers do. For information about the layout of the test editor and the more commonly performed, high-level editing tasks, see [Editing HTTP tests on page 276](#).

HTTP test details

Test detail fields apply to the entire test.

Common options

Datasets

Lists details about each dataset that the test uses: the name of the dataset, the columns that are used, and the location in the test where the dataset column is referenced. Click an item in the **Location** column to go to that location.

Add Dataset

Click to add a reference to a dataset for test to use. Clicking this option is the same as clicking **Add > Dataset** with the test selected.

Delete

Select a dataset reference, and then click to delete the reference from the test. The dataset is still available to other tests.

Show Dataset Candidates

Click to open the **Show Dataset Candidates** window, where you can review and change data correlation.

Digital Certificates

Lists details about the certificate stores that the test uses. Click **Add** to add a certificate store for the test to use. HTTP and SOA support digital certificates. Other protocols do not support digital certificates.

Enable response time breakdown

Enables collection of response time breakdown data. With response time breakdown, you can see statistics on any page element. The statistics show how much time was spent in each part of the system under test. You can use response time breakdown to identify code problems. You can see which application on which server is the performance bottleneck, and then drill down further to determine exactly which package, class, or method is causing the problem.

This option is displayed in multiple test elements. Enabling this option in an element also enables it in the element's children. For example, enabling monitoring at the test level also enables monitoring at the page and request levels. You can enable monitoring for a specific page; doing so enables monitoring for the requests of that page, but not for other pages or their requests.

HTTP and SOA support response time breakdown. Other protocols do not support response time breakdown.

Security

Digital Certificates

Lists details about the certificate stores that the test uses. Click **Add** to add a certificate store for the test to use. Not all protocols support digital certificates.

Enable Kerberos authentication

Select to enable Kerberos authentication. The user ID, password, and realm are supplied when a Kerberos authentication challenge occurs during playback. If you record a test using no authentication, and then enable Kerberos authentication on the system under test, select this check box.

User ID

Type the user principal name. The user principal name format consists of the user name, the "at" sign (@), and a user principal name suffix. Do not use the domain\username format. User IDs are case-sensitive.

Password

Type the password for the User ID. Passwords are case-sensitive.

Client realm

Type the realm of the client application. In Windows environments, the client realm is the Windows domain name for the computer sending the request to the server. Typically, the client realm is all uppercase.

Client KDC

Type the name of the client key distribution center. In Windows environments, the client key distribution center is the hostname of the domain controller for the client realm. By default, the client key distribution center is set to the domain controller of the computer where the test was recorded. Verify the default value with your system administrator.

Server realm

Type the realm of the server under test. The client and server might share the same realm. Type the server realm only if the server realm is different from the client realm. Contact your system administrator for more information about the server realm.

Server KDC

Type the name of the server key distribution center. In Windows environments, the server key distribution center is the hostname of the domain controller for the server domain. Type the server key distribution center only if the server is in a different domain than the client.

Enable response time breakdown

Select to enable the collection of response time breakdown data. You can enable response time breakdown collection at the parent or page level. Not all test elements support response time breakdown data collection.

Performance Requirements

Requirements

The table displays the performance and functional requirements that are defined in the test. To edit a requirement definition, double-click a table row. To return to this table, click the root name of the test in the **Test Contents** area.

Clear

Select one or more requirements and click to remove the definition. The requirement is still available and can be redefined.

Enable response time breakdown

Select to enable the collection of response time breakdown data. You can enable response time breakdown collection at the parent or page level. Not all test elements support response time breakdown data collection.

HTTP options

Timeout action

Specifies what the test does if the primary request for a page does not succeed within the **Timeout** interval. If you select **Log error and continue execution**, the test logs the error and proceeds to the next page. If you select **Try to reload the page**, the test attempts to reload the page one more time. If that attempt fails, the test logs an error and proceeds to the next page.

Timeout

Specifies the time threshold for initiating the action that you select for **Timeout action**.

Clear cookie cache when the test starts

This option resets the cookie cache when looping in the schedule or when a test follows another test in the schedule. By default, the cookie cache for a virtual user is not reset, which is consistent with browser behavior. If you want each loop iteration to behave as a new user, select this option. Otherwise, the cookies in the cache might alter the server responses and verification points might fail. To reset the cookie cache from one loop iteration to the next when looping *within* a test, add custom code and call an API.

Clear page cache when the test starts

This option deletes the page cache when a test starts. Typically, when a test follows another test in the schedule or when you anticipate an out-of-memory exception due to overload, you can delete the cache.

Disable page cache emulation in this test

This option disables page cache emulation. When page cache emulation is enabled, caching information in server response headers is honored. Additionally, requests are not submitted to the server for content that is confirmed by the client as fresh in the local cache. Page cache emulation is enabled by default.

Remove HTTP request delays from page response times

To not include the client delays in the page response times for the test or A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule., select this check box. By default, the page response times include delays to represent processing time caused by clients such as a web browser. Sometimes this delay could exceed the logical limit causing page response times to increase drastically.

Playback speed

Move the slider to increase or decrease the speed at which the HTTP requests are sent. You can specify a range from no delays to twice the recorded length. This scale is applied to the **Delay** field of each request in the test. If you speed playback up dramatically, requests might occur out of order. To fix this problem, reduce playback speed until the test runs correctly again.



Note: To set a maximum request delay, click **Window > Preferences > Test > Test Generation > HTTP Test Generation**. Click the **Protocol** tab, and enter a value for **Maximum Request Delay**.

Secondary request behavior

Click **Modify** to disable or reenable requests that occur within a page. You can disable all secondary requests, images, host-based or port-based requests, or user-defined requests.

Enable response time breakdown

Select to enable the collection of response time breakdown data. You can enable response time breakdown collection at the parent or page level. Not all test elements support response time breakdown data collection.

HTTP page details

Page detail fields apply to the page that is currently selected.

General tab

Page title

Specifies the display name for the page. If the primary request returned a title, the display name for the page is the content between the <title></title> tags. If the primary request returned no title or an empty title, a name for the page is constructed from the first node in the web address for the primary request URL, for example, www.site.com/displayname/.... If two pages have the same page title but are at different web addresses (for their primary request), then a number might be appended to indicate that they are different (for example, displayname {1}, displayname {2}). The pages are included in reports as separate pages, with their unique appended names.

Pages with the same title and web address appear in the test editor with the same page title and in reports as the same page. Rename any pages that you want to be reported on under a different name. Renaming a page neither changes the value (if any) between the <title></title> tags nor affects how the test runs.

Primary request

Displays a hyperlink to the primary request for the page. This request is highlighted and is the request from which the display name for the page is derived.

Think time

Specifies the programmatically calculated time delay that is observed for each user when this test is run with multiple virtual users. Think time is a statistical emulation of the amount of time actual users spend reading or thinking about a page before requesting another page from the server.

Test data

Summarizes data substitutions and potential matches in the page. Right-click a row, or select a row and then click **Options**, to perform common operations. Double-click a row to navigate to the location where

a substitution or potential match occurs. To associate a dataset candidate with a dataset, click the row, and then click **Substitute**. To remove a dataset substitution, click the row, and then click **Remove Substitution**. To find more locations in the test that have the same value as the selected row, click **Find More**. Click the icons to the left of the preview area to switch between an inline view and a hierarchical view of the selected data.

URL Encode

Indicates whether a test value contains special characters such as spaces or commas. With this option, special characters are encoded when variable data is substituted from a dataset.

Page title verification point

Indicates whether the page title verification point is enabled for this page. If so, **Enable verification point** is selected.

When **Enable verification point** is selected, the value between the <title></title> tags, if any, is copied to the **Expected page title** field on the properties page of the verification point. Click **Edit Properties** to change the **Expected page title**. The value between the title tags is different from the display page title (the value in the **Page title** field) that is used for reporting. Changing the **Page title** does not change the value between the title tags, and therefore does not affect what is initially copied to the **Expected page title** field.

If **Enable verification point** is selected, the test verifies whether the page returns the value in the **Expected page title** field. An error is reported in the test log if the title returned by the primary request for the page does not contain the expected title. Although the comparison is case-sensitive, it ignores multiple white-space characters (such as spaces, tabs, and carriage returns).

Enable response time breakdown

Enables collection of response time breakdown data. With response time breakdown, you can see statistics on any page element. The statistics show how much time was spent in each part of the system under test. You can use response time breakdown to identify code problems. You can see which application on which server is the performance bottleneck, and then drill down further to determine exactly which package, class, or method is causing the problem.

This option is displayed in multiple test elements. Enabling this option in an element also enables it in the element's children. For example, enabling monitoring at the test level also enables monitoring at the page and request levels. You can enable monitoring for a specific page; doing so enables monitoring for the requests of that page, but not for other pages or their requests.

HTTP and SOA support response time breakdown. Other protocols do not support response time breakdown.

Advanced tab

Enable Requirements

Select to enable the use of performance and functional requirements for this test.

Name

Specifies the name of this set of enabled requirements. By default, it is the URL of the page. Although you can change the name to improve readability, only the Requirements report uses the changed name. Other reports use the default name. Click **Use Defaults** to reset **Name** to the default value.

Requirement

All performance and functional requirements are displayed in the table. Shaded requirements indicate that they are undefined. To define a requirement, set an **Operator** and **Value**. To apply the defined requirement to multiple pages, select the pages in the test, right-click the requirement row in the table, and click **Copy Requirements**.

Operator

Click this field to display a list of mathematical operators. Select an operator for this requirement.

Value

Click this field to set a numeric time value in milliseconds.

Standard

Select to enable this requirement to be processed by the report as a standard requirement. Standard requirements can cause a test to fail. Requirements that are not listed as standard do not cause the test to fail.

Hide Undefined Requirements

Select to prevent undefined requirements from appearing in the table. This hides the shaded rows.

Clear

Select one or more requirements and click to remove the definition. The requirement is still available and can be redefined.

Error Handling

Click to open the error condition table. You can use error handling to specify an action to take and a message to log when a specific condition occurs. Conditions include verification point failures, server timeouts, custom code alerts, and data correlation problems. All conditions are displayed in the table, along with the action to take and the message to log when the error occurs. To define an error handler, select a **Condition**, and then click **Edit**.

Hide unselected conditions

Click to display only the selected error handlers. Hiding a condition does not deactivate the condition.

HTTP request details

Page request fields apply to the page that is currently selected.

General tab**Version**

Indicates the HTTP version.

Method

Indicates the HTTP request method that was used during recording. Typically, you do not change this value unless you are adding a new request to a test. GET, POST, PUT, HEAD, PATCH, and DELETE are supported.

Primary request for page

Displayed for the primary request, and cannot be modified. A page can contain only one primary request.

Click to set as primary

Displayed for all secondary requests. Because each page can have only one primary request, if you select this option, the **Primary request for page** option is moved to this request, and the **Click to set as primary** option is moved to the original primary request. To undo your change, select **Click to set as primary** on the original primary request.

Connection

Specifies the connection to the web server. The connection includes the host name, which is typically the fully qualified domain name, and the listener port on the web server. Click the name of the connection to navigate to the server access configuration where the connection is defined. Click **Change** to change the connection used for this request.

Total number of requests

Applies to HTTP Secondary Request Generator. Specify the number of requests to send to the server. If there is an array variable assigned to the request, the number of requests set in the Test editor takes precedence.

URL

Specifies the path to a resource (such as a page, graphics file, or stylesheet file). When the method is GET, the **URL** field typically includes query strings that are designated as dataset candidates.

Data

Specifies additional content data that might be needed to clarify the request. When the method is POST, the data frequently includes values that are designated as dataset candidates.

Request Headers

Lists each request header and its value. To change the value of a header, click the row, and then click **Modify**. To add a new header, click **Add**. To delete a header, click **Remove**.

Enable response time breakdown

Select to enable the collection of response time breakdown data. You can enable response time breakdown collection at the parent or page level. Not all test elements support response time breakdown data collection.

Use the **Advanced** page to configure performance requirements, error handling, and delay behavior for the request.

Advanced tab

Always log details

To ensure that the details about the request is always logged, select this check box.

Use substituted URL in performance reports

Use this option to view the substitutions in Page Elements report.

Requirement

All performance and functional requirements are displayed in the table. Shaded requirements indicate that they are undefined. To define a requirement, provide details in **Operator** and **Value**. To apply the defined requirement to multiple requests, select the requests in the test, right-click the requirement row in the table, and click **Copy Requirements**.

Enable Requirements

Select to enable the use of performance and functional requirements for this test.

Name

Specifies the name of this set of defined requirements. By default, the name is the URL of the request. Although you can change the name to improve readability, only the Requirements report uses this name. Other reports use the default name. Click **Use Defaults** to reset **Name** to the default value.

Operator

Click this field to display a list of mathematical operators. Select an operator for the performance requirement.

Value

Click this field to set a value for the requirement.

Standard

Select to enable this requirement to be processed by the report as a standard requirement. Standard requirements can cause a test to fail. Requirements that are not listed as standard do not cause the test to fail.

Hide Undefined Requirements

Select to prevent the table from including undefined requirement candidates. Selecting this check box hides all shaded rows.

Clear

Select one or more requirements, and click to remove the definition. The requirement is still available and can be redefined.

Error Handling

Click to open the error condition table. You can use error handling to specify an action to take and a message to log when a specific condition occurs. Error conditions include verification point failures, server timeouts, custom code alerts, and data correlation problems. All error conditions are displayed in the table, beside the action to take and the message to log when the error occurs. To define an error handler, select a **Condition**, and then click **Edit**. The Errors report lists the number of errors and the corresponding actions that occurred in the test or schedule. You can also specify whether an error contributes to the health of the page. To set the health parameter, select a Condition and select the **Override contribution to health status** check box. The Page Health report shows the health of each page.

Hide unselected conditions

Click to display only the selected error handlers. Hiding a condition does not deactivate the condition.

Applied Transform

Indicates the data transformation that is applied to the request. Click **Change** to select a data transformation to apply to the request.

Character set

Indicates the character set to be used for the page request. Click **Change** to see the valid character sets.

Pre / Postprocessing

To modify and inspect certain aspects of the action before and after it is executed, specify pre and post processors. Click **Create** to create the Java file that will contain the skeleton of the Java file needed. Click **Browse** to navigate to a Java processor that has already been created.

Client Processing Delay

Previous versions of tests support only waiting for primary requests. **Wait for** and **Release when** are not available. The additional delay in previous versions of tests is measured from the first character received of the primary request.

Wait for

Indicates the associated request that must start or finish before this request is issued. Click **Request to** select a different request. Click the **Clear request association** icon to remove the association.

Release when

Select **Last Character Received** or **First Character Received** to indicate when this request is issued in relation to the associated request.

Additional delay (ms)

Indicates the additional delay, in milliseconds, to wait before this request is issued. Delays are statistical emulations of user behavior. You can scale this delay at the test level to make a test play back faster (or slower) than it was recorded.

Delay Between Requests

Applies to HTTP Secondary Request Generator. Use the delays to control the flow of the requests to the server. In **Release When**, select when exactly to release the request. For instance, First Character Sent indicates to release the second request after the first character in the first request is sent.

In **Additional delay**, specify a value in milliseconds to indicate additional delay to send the request.

Digital Certificates

Lists details about the certificate stores that the test uses. Click **Add** to add a certificate store for the test to use. HTTP and SOA support digital certificates. Other protocols do not support digital certificates.

Enable response time breakdown

Enables collection of response time breakdown data. With response time breakdown, you can see statistics on any page element. The statistics show how much time was spent in each part of the system under test. You can use response time breakdown to identify code problems. You can see which application on which server is the performance bottleneck, and then drill down further to determine exactly which package, class, or method is causing the problem.

This option is displayed in multiple test elements. Enabling this option in an element also enables it in the element's children. For example, enabling monitoring at the test level also enables monitoring at the page and request levels. You can enable monitoring for a specific page; doing so enables monitoring for the requests of that page, but not for other pages or their requests.

HTTP and SOA support response time breakdown. Other protocols do not support response time breakdown.

HTTP response data details

Response data fields apply to the response data that is returned by each page request.

General tab

Status

Indicates the status code for the HTTP response, such as 200, 201, 203, or 302.

Version

Indicates the HTTP version, such as 1.1.

Reason

Indicates the code for the HTTP response, such as OK, Found, or Not Found.

Response Headers

Lists each response header and its value. To change the value of a header, click the row, and then click **Modify**. To add a new header, click **Add**. To delete a header, click **Remove**.

Content

Shows the content (such as tagged HTML, graphics files, or stylesheet files) that the web server returned, based on the corresponding request.

Advanced tab**Applied transform**

Indicates the data transform that is applied to the response. Click **Change** to select a data transform to apply to the response.

Character set

Indicates the character set to be used for the response. Click **Change** and select the encoding to change the character set.

HTTP server access configuration details

Server access configurations store HTTP connection information. By default, a connection does not remain open across test boundaries. Several connections can use the same server access configuration, and the same connection can be used by several other requests in the same test. If you change the host, port, or authentication for a server access configuration, those changes apply to all connections in the test that use the configuration.

Configuration name

Specifies the name of the server access configuration.

Host

Specifies the name of the host for the web server. Usually, this is the fully qualified domain name, but it can be an IP address or other name.

Port

Specifies the listener port on the web server.

Authentication and security

Indicates whether this connection uses the Secure Sockets Layer (SSL) protocol, the NT/LAN Manager (NTLM) authentication protocol from Microsoft™, or an HTTP proxy server. A blank field indicates that the connection is unauthenticated and not secure. To add proxy, SSL, or NTLM authentication, expand the request, click the connection, and then click **Add**.

Connections that use this configuration

Lists the connections that use this configuration.

SAP test editor reference

In SAP testing, the test editor information is divided into nine categories. This section describes the fields in each category that can be edited manually.

SAP test details

In the test editor, the test element is the first element in the test suite. These settings apply to the entire test.

SAP options

Display SAP GUI on execution

During test execution, it might not be desirable to display the SAP GUI. Hiding the SAP GUI improves the performance of the virtual users. This setting specifies the behavior for the current test suite. However, you can change the default setting for generated tests in the SAP Test Generation preferences.

Hide

When selected, all instances of the SAP GUI are hidden. In some cases, modal dialog boxes from the SAP GUI can flash briefly on the screen. This is the default setting.

Show

When selected, the SAP GUI is displayed for all virtual users.

Show only first virtual user

When selected, the SAP GUI is displayed only for the first virtual user. All other instances of the SAP GUI are hidden. This allows you to monitor the execution.

Common options

Datasets

Lists details about each dataset used by the test: the name of the dataset, the columns that are used, and the location in the test where the dataset column is referenced. Click the location to navigate there.

Add dataset

Adds a reference to a dataset that you want a test to use. Clicking this option is the same as clicking **Add > Dataset** with the test selected.

Remove

Removes the selected dataset. This option is not available if the dataset is in use.

SAP connection details

In the test editor, SAP connection elements are at the top of the test site and describe the connection to the SAP server. These settings apply to the entire test.

SAP system name

This is the description normally used by SAP Logon to identify the server. If the **Connection by string** option is selected this field is ignored.

Connection by string

Select this option to use the connection string that was returned by the server when recording to connect to the server without referring to the SAP Logon program. This is safer when deploying the test on remote computers. Advanced users can edit the connection string if necessary. You can use data correlation to substitute this value.

Get SAP GUI session statistics

Select this option to record session statistics from the SAP GUI client in the test results. These results are displayed on the **User Load** page of the test report.

Use new visual design

Select this option to run tests with a visual design theme when using SAP GUI 7.0 or later. In most cases, it is best to leave this option disabled, which causes tests to run with the default SAP GUI visual design and avoids compatibility issues.

Use recorded visual design theme

If **Use new visual design** is selected, select this option to use the visual design theme that was used during the recording o

Use other visual design theme

If **Use new visual design** is selected, select this option to use a specific visual design theme. Ensure that the name is correct and that the visual design theme is installed on the test computer. Unexpected results might occur if you specify a visual design theme name that cannot be located on the test computer.

SAP screen details

In the test editor, SAP screen elements are located in transactions and are the basic performance measurement unit for the test. These settings apply to the selected get event.

SAP element label

This is the name of the selected SAP test element as it is displayed in the Test Contents. Use this field to rename the test element, or click **Restore Default** to revert to the default name.

Title

This is the recorded name of the SAP screen. This field is read-only.

Do not measure performance on this screen

Select this option if you do not want to obtain response time results for the current SAP screen. Use this for SAP screens that are not meaningful for your test, such as the logon screen.

Optional screen

Select this option if you do not want to log an error when the current SAP screen is not displayed. Use this for SAP screens that are not always displayed.

Data Table

Summarizes data substitutions and substitution candidates in the SAP screen. Double-click a row to navigate to the location where a substitution or candidate occurs. To associate a dataset candidate with a dataset, click the row and then click **Dataset Variable**. To remove a dataset substitution, click the row and then click **Remove Substitution**.

Screen Title Verification Point

Enable Verification Point

When selected, the test verifies whether the SAP screen returns the value shown in the **Expected screen title** field. An error is reported in the test log if the screen title returned during the test does not match the expected title.

Expected screen title

This field allows you to specify the expected SAP screen title. By default, the expected title is the recorded title. The expected title can optionally be expressed as a regular expression.

Recorded screen title

This field displays the recorded title of the current SAP screen. This field is read-only.

Use Regular Expression

Select this option to express the expected title using the standard regular expression syntax.

SAP set details

In the test editor, SAP sets are located in SAP screen elements and describe a user input action in the SAP GUI client. These settings apply to the selected SAP set.

SAP element label

This is the name of the selected SAP test element as it is displayed in the Test Contents. Use this field to rename the test element, or click **Restore Default** to revert to the default name.

Think Time

Specifies the programmatically-calculated time delay that is observed for each user when this test is run with multiple virtual users. Think time is a statistical emulation of the amount of time actual users spend reading or thinking before performing an action.

SAP Set

Property name

This is the description of the GUI object related to the current SAP set as it appears to the user in the SAP GUI. This field is read-only.

Value

This is the value entered by the user in the current SAP set. You can use data correlation to substitute this value.

SAP GUI Object Information

Name

This is the recorded name of the GUI object related to the current element. This field is read-only.

Type

This is the recorded type of the GUI object related to the current element. This field is read-only.

Identifier

This is the recorded identifier of the GUI object related to the current element. This field is read-only.

SAP get details

In the test editor, SAP get events are located in SAP screen elements and provide a way to retrieve data from a SAP GUI object to implement verification points. These settings apply to the selected get event.

SAP element label

This is the name of the selected SAP test element as it is displayed in the Test Contents. Use this field to rename the test element, or click **Restore Default** to revert to the default name.

Think Time

Specifies the programmatically-calculated time delay that is observed for each user when this test is run with multiple virtual users. Think time is a statistical emulation of the amount of time actual users spend reading or thinking before performing an action.

SAP Get

Property name

This is the description of the GUI object related to the current event as it appears to the user in the SAP GUI client. This field is read-only.

Value

This is the value recorded during the test or during the last execution. You can use data correlation to reference this value. This field is read-only.

Verification Point

Enable Verification Point

When selected, the test verifies whether the screen returns the value specified in **Expected Value**. An error is reported in the test log if the value returned during the test does not match the expected value.

Expected Value

This field enables you to specify the expected value for the get event. The expected value can optionally be expressed as a regular expression. You can use data correlation to substitute this value.

Use Regular Expression

Select this option to express the expected value using standard regular expression syntax.

SAP GUI Object Information

Name

This is the recorded name of the GUI object related to the current element. This field is read-only.

Type

This is the recorded type of the GUI object related to the current element. This field is read-only.

Identifier

This is the recorded identifier of the GUI object related to the current element. This field is read-only.

SAP call details

In the test editor, SAP call elements are located in SAP screen elements and describe various recorded interactions with the SAP server. These settings apply to the selected SAP event.

SAP element label

This is the name of the selected SAP test element as it is displayed in the Test Contents. Use this field to rename the test element, or click **Restore Default** to revert to the default name.

Think Time

Specifies the programmatically-calculated time delay that is observed for each user when this test is run with multiple virtual users. Think time is a statistical emulation of the amount of time actual users spend reading or thinking before performing an action.

SAP Call

Method name

This is the internal method call used by the SAP GUI client. This field is read-only.

Parameter

If the method uses parameters, one **Parameter** line is displayed for each parameter. Advanced users can modify these parameters. Refer to SAP documentation for more information about the parameters used by SAP GUI method calls. You can use data correlation to substitute this value.

Return

If the method returns a value, a **Return** line is displayed, which can be used for data correlation or for a verification point. The value displayed is not the actual return value, but only represents the type of the parameter, for example, `string` for a string type or `o` for an integer. Refer to SAP documentation for more information about the parameters used by SAP GUI method calls.

Verification Point**Enable Verification Point**

When selected, the test verifies whether the **Return** value of the SAP call (if any) matches the value specified in **Expected Value**. An error is reported in the test log if the value returned during the test does not match the expected value.

Expected Value

This field enables you to specify the expected value for the call. The expected value can optionally be expressed as a regular expression. You can use data correlation to substitute this value.

Use Regular Expression

Select this option to express the expected value using standard regular expression syntax.

SAP GUI Object Information**Name**

This is the recorded name of the GUI object related to the current element. This field is read-only.

Type

This is the recorded type of the GUI object related to the current element. This field is read-only.

Identifier

This is the recorded identifier of the GUI object related to the current element. This field is read-only.

SAP server request details

In the test editor, server request elements are located at the end of every SAP screen and provide information that the server returns for the selected screen.

SAP element label

This is the name of the selected SAP test element as it is displayed in the Test Contents. Use this field to rename the test element, or click **Restore Default** to revert to the default name.

SAP Screen

Name

This is the name of the current SAP transaction code. This field is read-only.

Program

This is the name of the SAP source program that is currently running. This field is read-only.

Flushes

This is the count of the number of flushes in the automation queue during server communication. This field is read-only.

Response Time

This is the delay between the moment the SAP GUI client sends the request to the SAP server and the moment the server response arrives. The units are milliseconds. This field is read-only.

Interpretation Time

This is the delay between the moment the data is received by the SAP GUI client and the moment the screen is updated. It measures interpretation of data by the SAP GUI client, not SAP server performance. The units are milliseconds. This field is read-only.

Roundtrips

This is the count of token switches between the SAP GUI client and the SAP server to perform the request. This field is read-only.

Request Time Verification Point

Enable verification point

When selected, the test verifies whether the request time returned by the server is below the specified threshold value. An error is reported in the test log if the measured request time is above the threshold.

Response time threshold (ms)

This is the request time limit above which an error is reported in the test log.

Request Timeout

Timeout value (ms)

Select this option to change the default timeout value (3 minutes) for very long transactions.

Response time threshold (ms)

The test verifies that the request time returned by the server is below the specified threshold value. An error is reported in the test log if the measured request time is above the threshold.

SAP batch connection details

In SAP batch input tests, SAP batch connections contain the basic connection information for a batch input test to connect to the SAP server without a SAP GUI. In most cases, these details are the same as those used when you connect manually to SAP with the SAP GUI.

SAP element label

This is the name of the selected SAP test element as it is displayed in the Test Contents. Use this field to rename the test element, or click **Restore Default** to revert to the default name.

SAP Batch Input Connection

Client

This is the SAP client number that is used by the batch input test to connect to the SAP server. You can use data correlation to substitute this value.

User

This is the user name that the batch input test uses to connect to the SAP server. You can use data correlation to substitute this value.

Password

This is the password that the batch input test uses to connect to the SAP server. You can use data correlation to substitute this value.

Language

This is the two-letter language code. You can use data correlation to substitute this value.

Host

This is the IP address or computer name of the SAP server. You can use data correlation to substitute this value.

System Number

This is the system number of the SAP server. You can use data correlation to substitute this value.

Additional SAP Connection Properties

Use this list to specify any advanced SAP Java™ Connector (JCo) properties for advanced SAP router setup. Select the JCo property that you want to set in the **Property name** list, and type the required value in **Property value**. Click **Add** to add more properties.

Test Connection

Use this button to test the connection to the SAP server.

SAP batch input transaction details

In the SAP batch input tests, SAP batch input transactions are located in transactions and are recorded transactions that are to be run at a low level, without a SAP GUI, in order to produce a load on the SAP server.

SAP element label

This is the name of the selected SAP test element as it is displayed in the Test Contents. Use this field to rename the test element, or click **Restore Default** to revert to the default name.

SAP Batch Input Transaction

Code

This is the SAP transaction code of the recorded transaction.

Mode

This is the mode of the batch input transaction as it was recorded in the SAP GUI.

Data table

This is the data table of the batch input transaction as it was recorded in the SAP GUI. See the SAP documentation for details on the contents of the recording.

Citrix test editor reference

In Citrix testing, the test editor information is divided into eleven categories. This section describes the fields in each category that can be edited manually.

Citrix test details

In the test editor, the Citrix Test is the first element of a Citrix test. These settings apply to the entire Citrix test.

Citrix options

Synchronization timeout delay

This is the delay after which a timeout error is produced when a window event is not recognized during test execution. The default value is 6000 milliseconds.

Delay between mouse down and mouse up in a click

This is the default delay used to generate a mouse click action using a mouse down and a mouse up action. The default value is 50 milliseconds.

Delay between two mouse clicks in a double click

This is the default delay used to generate a double-click action using two mouse clicks. The default value is 200 milliseconds.

Delay between key down and a key up in a key stroke

This is the default delay used to generate a key stroke action using a key down and a key up action. The default value is 100 milliseconds.

Delay between two keyboard strokes in a text input

This is the default delay used to generate a text input action using multiple key stroke actions. The default value is 500 milliseconds.

Common options

Datasets

Lists details about each dataset used by the test: the name of the dataset, the columns that are used, and the location in the test where the dataset column is referenced. Click the location to navigate there.

Add dataset

Adds a reference to a dataset that you want a test to use. Clicking this option is the same as clicking **Add > Dataset** with the test selected.

Remove

Removes the selected dataset. This option is not available if the dataset is in use.

Citrix session details

In the test editor, the session is located at the top of the Citrix test. Session settings apply to connection with the server.

Session Attributes

Session Title

This is the name of the current session. By default, it is the same as the name of the test.

Server Address

This is the address of the Citrix server. The value can be a host name or an IP address. This value can be linked to a dataset.

Initial Program

This is the name of a published application on the Citrix server. Use this option to manually specify a published application if no Independent Computing Architecture (ICA) file is available. If no published application and no ICA file is specified, the session starts with the Windows™ desktop.

ICA File

If you recorded the test with an ICA file, this is the location and name of the file. The ICA file contains connection and application information to launch directly a published application with the Citrix XenApp client.

User name and Password

These fields allow you to specify user authentication information. These values can be linked to a dataset.

Color Depth

This is the recorded color depth for the Citrix XenApp client. This value is read-only.

Screen Size

This is the recorded screen resolution for the Citrix XenApp client. This value is read-only.

Response Time Definitions

This table defines the response time measurements that will be performed during the test. By default, response times are automatically generated on main create window events.

Response Time

This is the name of the response time measurement. To change a name, select a response time and click **Rename**. These names appear in the performance test report.

Started by

This is the user input action that triggers the start of the response time measurement. To navigate to the corresponding user input action in the test editor, click **Go to Start**.

Stopped by

This is the user window event that stops the response time measurement. To navigate to the corresponding user input action in the test editor, click **Go to Stop**.

Add, Rename and Delete

These buttons allow you to manually create, rename or delete a response time measurement.

Citrix window details

In the test editor, the Citrix window elements contain all user input actions and window events. These settings apply to the selected window element.

Window Title

This is the title of the window as displayed in the Citrix session. Some windows do not have titles, and the window ID is used for identification. This field is read-only.

Window ID

This is the window ID number attributed by the Windows™ operating system when the window is created during the recording session. This number changes each time the test is executed, but the ID remains the same throughout a session. This field is read-only.

Locations

This field displays the X and Y coordinates of the top left corner of the window and size of the window in pixels. This field is read-only.

Window recognition during execution uses

This option allows you to disable window recognition on window position or size. Disable any of these options if the test produces synchronization timeouts because a window changes its position or size between or during test runs.

Parent Window

This is a link to the window element that is the parent of the selected window.

Go to same occurrences of this window

Use these navigation buttons to navigate through the test to other occurrences of this window, for example if during a test the user switches back and forth between windows, or if the current window is modified in any way.

Styles**Window Styles**

This area lists the style properties that are enabled for the current window. These are read-only.

Window Extended Styles

This area lists the extended style properties that are enabled for the current window. These are read-only.

Verification Point**Enable Verification Point**

When selected, the test verifies whether the window returns the title shown in the **Expected title** field. An error is reported in the test log if the title returned during the test does not match the expected title.

Use Regular Expression

Select this option to express the expected title using standard regular expression syntax.

Expected title

This field allows you to specify the expected title for the window. The expected title can optionally be expressed as a regular expression.

Recorded title

This displays the title that was recorded for the current window. This field is read-only.

Citrix window event details

In the test editor, the Citrix window event elements are located inside window elements and describe any changes to the location or size of a window. These settings apply to the selected window event element.

Type of Event

This is the type of window event.

Window ID

This is the window ID number attributed by the Windows™ operating system when the window is created during the recording session. This number changes each time the test is executed, but the ID remains the same throughout a session. This field is read only.

Window Title

This is the title of the window as displayed in the Citrix session. Some windows do not have titles, and the window ID is used for identification. This field is read only. You can click the window title to select the window element in the test contents.

Window Title

This is the title of the window as displayed in the Citrix session. Some windows do not have titles, and the window ID is used for identification. This field is read only. You can click the window title to select the window element in the test contents.

Synchronization state

This describes the behavior of the test if a synchronization timeout occurs on the window event. The base timeout delay is specified in the Citrix test generation preferences, however the actual delay varies with the level of synchronization.

Conditional

The conditional timeout delay is the base timeout delay as specified in the Citrix test generation preferences. If the synchronization fails, the test tries to resume execution and a timeout is logged in the Citrix performance report and the test log.

Mandatory

The mandatory timeout delay is three times the base timeout delay. If the synchronization fails, the test exits with an error status and a timeout is logged in the test log.

Optional

The optional timeout delay is fixed at 2 seconds. If the synchronization fails, the test ignores the timeout.

Response Time

Stop response time for

Select this option to use the current window event to stop a response time measurement. When you select this option on an window event that is not already linked to a response time, a new response time is created with a default name. If there are response times that do not have a stop action, then these are also listed. Select the response time that you want to link to.

Go to response time definition

Click here to navigate to the session element to view the **Response Time Definitions** table.

Citrix key action details

Citrix key action fields apply to the selected key action element.

Type

This is the type of key action.

- Key Down: The key is pressed.
- Key Up: The key is released.
- Key Stroke: The key is pressed and released.

Key Code

This is the code of the key as interpreted by the Windows™ operating system.

Character

This field displays the actual key combination that is interpreted.

Modifiers

These options allow you to specify the standard keyboard modifiers: Control, Shift, Alt, or Extended.

Think Time

Enable Think Time

Select this option to specify a think time for the current user input action.

Think Time

Specifies the programmatically calculated time delay that is observed for each virtual user when this test is run with multiple virtual users. Think time is a statistical emulation of the amount of time actual users spend reading or thinking about an input before performing the action.

Character edition

Enter a character

This area allows you to enter any key combination to produce Unicode characters that are not normally available through single keystrokes. Select the input field and enter the character on your keyboard. The **Key Code** and **Character** fields display the corresponding character.



Note: The workbench uses some key combinations as keyboard shortcuts. Such combinations can be intercepted and cause undesirable actions instead of displaying a particular character in the Character field.

Response Time

Start response time for

Select this option to use the current input action to trigger the start of a response time measurement. When you select this option on an input action that is not already linked to a response time, a new response time is created with a default name. If there are response times that do not have a start action, then these are also listed. Select the response time that you want to link to.

Go to response time definition

Click here to navigate to the session element to view the **Response Time Definitions** table.

Citrix mouse action details

In the test editor, mouse action elements are located in window elements and describe mouse input. These settings apply to the selected mouse action element.

Type of Event

This is the type of mouse action:

- Mouse Down: The mouse button is pressed.
- Mouse Up: The mouse button is released.
- Mouse Click: The mouse button is pressed and released.
- Mouse Double Click: The mouse button is clicked twice.
- Mouse Move: The mouse is moved to a new location.

X Position and Y Position

These are the coordinates of the mouse action. In the case of a mouse move action, these are the coordinates at the end of the movement.

Buttons

These are the buttons that are activated, if any.

Think Time

Enable Think Time

Select this option to specify a think time for the current user input action.

Think Time

Specifies the programmatically calculated time delay that is observed for each virtual user when this test is run with multiple virtual users. Think time is a statistical emulation of the amount of time actual users spend reading or thinking about an input before performing the action.

Response Time

Start response time for

Select this option to use the current input action to trigger the start of a response time measurement. When you select this option on an input action that is not already linked to a response time, a new response time is created with a default name. If there are response times that do not have a start action, then these are also listed. Select the response time that you want to link to.

Go to response time definition

Click here to navigate to the session element to view the **Response Time Definitions** table.

Citrix text input details

In the test editor, text input action elements are located under window events and describe a series of key strokes. These settings apply to the selected text input element.

Value

Specify a string or portion of text that can be entered during the test. You can use references or dataset variables.

Think Time**Enable Think Time**

Select this option to specify a think time for the current user input action.

Think Time

Specifies the programmatically calculated time delay that is observed for each virtual user when this test is run with multiple virtual users. Think time is a statistical emulation of the amount of time actual users spend reading or thinking about an input before performing the action.

Response Time**Start response time for**

Select this option to use the current input action to trigger the start of a response time measurement. When you select this option on an input action that is not already linked to a response time, a new response time is created with a default name. If there are response times that do not have a start action, then these are also listed. Select the response time that you want to link to.

Go to response time definition

Click here to navigate to the session element to view the **Response Time Definitions** table.

Citrix mouse sequence details

In the test editor, Citrix mouse sequence elements are located under window elements and describe a series of mouse movements. These settings apply to the selected mouse sequence element.

Display mouse sequences for

This option specifies how you want to display previous, current, or all mouse sequences in the current mouse sequence:

Current® sequence

Only the current mouse sequence is displayed in the test editor. This option is selected by default.

Previous and current sequences

The current mouse sequence is displayed with any previous mouse sequences.

All sequences

All mouse sequences are displayed simultaneously.

Fit screen to visible area

Select this option to adjust the display of the mouse sequence to the available area in the test editor. If disabled, the screen capture will be the actual size, which might require scrolling. This option is enabled by default.

Screen capture area

This area represents the mouse movements on the screen. If a screen capture was recorded, it is displayed in the background. Mouse sequences are displayed as specified.

Citrix screen capture details

In the test editor, screen captures display a graphical overview of the state of the application at a given moment in the test, providing you with a point of reference for navigating through the test.

Screen captures are obtained by clicking the **Capture screen**  button in the **Citrix Recorder Control** window during recording.

Session Attributes

Locations

These are the screen coordinates and the size of the captured screen area.

Screen Capture Preview

This section displays a view of the screen or screen area that was captured during the recording.

Fit screen to visible area

Select this option to resize the screen capture to the available space in the test editor.

Citrix image synchronization details

In the test editor, the Citrix image synchronization allows Citrix performance tests to keep track of the contents of a screen area during the replay. These settings apply to the image synchronization element that is selected.

Image synchronization attributes

Locations

These are the coordinates of the top left corner of the image synchronization area, and the size of the image synchronization area in pixels. This field is read only.

Synchronization state

This describes the behavior of the test if a synchronization timeout occurs on the image. The base timeout delay is specified in the Citrix test generation preferences, however the actual delay varies with the level of synchronization.

Conditional

The conditional timeout delay is the base timeout delay as specified in the Citrix test generation preferences. If the synchronization fails, the test tries to resume execution and a timeout is logged in the Citrix performance report and the test log.

Mandatory

The mandatory timeout delay is three times the base timeout delay. If the synchronization fails, the test exits with an error status and a timeout is logged in the test log.

Optional

The optional timeout delay is fixed at 2 seconds. If the synchronization fails, the test ignores the timeout.

Image synchronization preview

This is the screen capture of the image synchronization area as it was recorded. Select **Fit screen to visible area** to limit the size of the screen capture in the test editor.

Synchronization**Bitmap hash code**

This specifies that the synchronization will be evaluated on the bitmap hash code. A hash code is a unique number that is calculated from the image of the selected area. When an image synchronization is encountered during test execution, the test calculates the hash code on the selected area and synchronizes the test if the hash code of the screen area matches the expected hash code before a timeout occurs.

Optical character recognition

This specifies that the synchronization will be evaluated on a recognized text value. Optical character recognition extracts a text string from the selected image area. When an image synchronization is encountered during test execution, the test continually applies text recognition to the selected area and synchronizes the test as soon as the extracted text value matches the expected text value before a timeout occurs.

Value

This page specifies the expected value depending on the specified recognition technique. You can add alternate values by clicking Add so that the image synchronization can succeed in multiple conditions. Alternate values are evaluated in the same way as the main expected value.

Bitmap hash code

When **Bitmap hash code** is selected, this is the hash code that was calculated on the selected image area during the recording. After executing a test, you can create alternate hash code values by copying the resulting hash codes from the **Citrix image synchronization** view.

Expected text

When **Optical character recognition** is selected, this is the expected text value that was extracted by the optical character recognition from the selected image area. Click **Extract text** to extract a text string from the selected image area.

If the text extraction is unsuccessful, try changing the text recognition settings on the **Options** page. However, accuracy of the recognized text is not essential. It is only important that the recognized text is consistent each time the test is executed for the test to synchronize.

Use regular expression

Select this option to express the expected text string using standard regular expression syntax.

Options

This page specifies the settings for text extraction by optical character recognition. You might need to experiment with various settings to obtain good results. After changing a setting, click the **Value** tab and click **Extract text** to see if the text recognition has improved. Note that because optical character recognition is used for verification purposes, consistency of the results is more important than the accuracy of the extracted text.

Zoom factor

This is the enlargement factor that is applied to the image. The default setting is medium for standard font sizes. Increase the zoom factor to improve recognition of smaller fonts or decrease for larger fonts.

Language

This is the language of the dictionary used by the text recognition synchronization. Select the language of the application you are testing. If the language of your application is not available in the list, change the language setting in the **Default OCR settings** of the **Citrix Test Generation** preferences.

Brightness

This is the brightness level from 0 to 250 that is applied to the image. The default setting is 70 for normally contrasted text. Increase the brightness setting to improve recognition of darker images or decrease for lighter images.

Recognition rate

This is the rate of recognition required for the extracted string to match the expected text. Decrease the recognition rate to tolerate a proportion of mismatching characters in the recognized text. The default is 100%, which means that an exact match is required.

Verification Point

Enable verification point on synchronized element

When selected, the test verifies whether the image synchronization succeeds. If the synchronization produces a timeout, the verification point returns a *fail* status in the Citrix performance test report.

Response Time

Stop response time for

Select this option to use the current image synchronization to stop a response time measurement. When you select this option on an image synchronization that is not already linked to a response time, a new response time is created with a default name. If there are response times that do not have a stop action, then these are also listed. Select the response time that you want to link to.

Go to response time definition

Click here to navigate to the session element to view the **Response Time Definitions** table.

Citrix logoff details

In the test editor, the logoff element is located at the end of the Citrix test. The logoff element is created only when the recording is stopped by clicking

Stop Recording



in the **Recorder Control** window. Other methods of ending a recording, such as closing the Citrix XenApp client or closing the Windows™ session, do not create a session logoff element in the generated test.

Session Logoff Attributes

Session Title

This is the name of the current session. By default, it is the same as the name of the test.

Type of Event

Select whether the logoff element performs a **Logoff** or a **Disconnect** event.

Think Time

Enable Think Time

Select this option to specify a think time for the current user input action.

Think Time

Specifies the programmatically calculated time delay that is observed for each virtual user when this test is run with multiple virtual users. Think time is a statistical emulation of the amount of time actual users spend reading or thinking about an input before performing the action.

VU Schedule editor reference

Most VU Schedule editor settings apply either to the entire schedule or to individual user groups.

Schedule properties

When you open a schedule, you can set its properties.

User Load page

Right-click in the table, and select **Add** to add a stage. To modify a stage, select the row, and then click **Edit** or click the user icon in the first column.

Users

Enter the total number of users to be active in the stage (not the number of users to add or subtract to those currently running).

Run for specified period of time

Enter the length of time (and the time units) for the stage to run. When the specified number of users is achieved, the users will run for up to this time. When the time expires, the users continue to run if they are required for the next stage; otherwise, they are stopped gracefully.

Click **Show Advanced** to set further options to prepare the system under test before the users actually enter the stage:

Change Rate

Enter a number to set a delay between adding or removing each user, rather than adding them or subtracting them all at once. Staggering users avoids overloading the system, which can cause connection timeouts. The **User Load Preview** shows this delay in black.

Settle Time

A system under test might react to a sudden change in user population. With a defined settle time, which starts when the target number of users is reached, the system under test can settle into a steady state so that it can accurately reflect the user population. The **User Load Preview** shows this time in black.

Time limit for a user to respond to a stop request

Optionally enter a value. When a virtual user is asked to stop, it completes its current action (such as an HTTP request) and then finishes. If a virtual user has not finished within the specified time limit, the user is forced to finish.

User Load Preview

Previews the user population stages over time. The red line segments indicate that the total number of users has been achieved for the state.

Think Time page

Use the recorded think time

Select to play back a test at the same rate that it was recorded. This option has no effect on the think time.

Specify a fixed think time

Each user's think time is exactly the same value: the value that you specify. Although this does not emulate users accurately, it is useful if you want to play a test back quickly.

Increase/decrease the think time by a percentage

Type a percentage in **Think time scale**. Each user's think time is multiplied by that percentage. A value of 100 causes no change in think times. A value of 200 doubles the think times, so the schedule plays back half as fast as it was recorded. A value of 50 reduces the think times by half, so the schedule plays back twice as fast. A value of 0 indicates no delays.

Vary the think time by a random percentage

Each user's think time is randomly generated within the upper and lower bounds of the percentages that you supply. The percentage is based on the recorded think time. For example, if you enter 10 in **Lower limit** and enter 90 in **Upper limit**, the think times will be between 10 percent and 90 percent of the original recorded think time. The random time is uniformly distributed within this range.

Maximum think time

Setting a maximum think time is useful with tests that emulate actual think times. By setting a maximum, you do not have to search for and edit each long think time within a test. Numerous factors can generate long think times, for example, you might be interrupted while recording. To restore the original think times, clear this check box.

Resource Monitoring page

Enable resource monitoring

Select to activate resource monitoring. The available data sources are captured from these sources:

- Apache HTTP Server Managed Beans
- Apache Tomcat Managed Beans
- IBM® Tivoli® monitoring agents
- IBM® DB2® snapshot monitors
- The IBM® WebSphere® Performance Monitoring Infrastructure
- JBoss Application Server Managed Beans
- Java™ Virtual Machine Managed Beans
- Oracle Database
- Oracle WebLogic Server Managed Beans
- SAP NetWeaver Managed Beans
- the UNIX™ rstatd monitor

- Simple Network Management Protocol (SNMP) agents
- Windows™ Performance Monitor

Resource monitoring data can provide a more complete view of a system to aid in problem determination.

Ignore invalid resources when executing the schedule

Select this setting to suppress any error messages that invalid resources cause, such as unreachable hosts or invalid host names. If you select this option, you must view logs to see error messages.

Statistics page

Statistics log level

These options are listed in order of the increasing amount of data that they collect for the test log.

None

Collects minimal statistical data. Use this option to run a schedule quickly for testing purposes.

Schedule Actions

Reports the number of active and completed users in the run.

Primary Test Actions

For HTTP tests, this option reports page-related actions (attempts, hits, and verification points). For SAP tests, this option reports information on SAP screens.

Secondary Test Actions

For HTTP tests, this option reports information that is related to page elements. This option does not apply to SAP tests.

All

Provides statistics for all actions.

Statistics sample interval

Sets the sampling interval for reports. When you run a schedule, the reports show such information as response time during a specific interval, the frequency of requests being transferred during an interval, and the average response trend during an interval. You set this sampling interval here.



Note: In HCL OneTest™ Performance V9.1.1 and later, during a test or schedule run, the **Elapsed Time** value is updated every 5 seconds irrespective of a value set in the **Statistics sample**



interval field. You can view the **Elapsed Time** value that is changing in the **Run Summary** section on the **Performance Summary** tab of the report.

Only store All Hosts statistics

Select this option unless you are running a performance test over different WANs, and you are interested in seeing the data from each remote computer.

Variable Initialization

Use this page to initialize variables at the schedule level. When you initialize variables at the schedule level, all the user groups in the schedule use the variable initial values, except those for which a specific value is defined.

Add

Add a variable and initialize a value. The **Used by** column displays the test name that uses the corresponding variable. A warning icon is displayed for a variable that overrides the value specified at the schedule level or user group level and uses the value defined at the test level with the visibility set to **This test only**. Hover the cursor over the warning icon to view the tests that override the variable initial values.

Export

Export the variables defined at the schedule level to a file.

Use variable initial values file

Select this check box to use the variable values from a file. Click **Browse** to select an existing file or click **New** to create a file.

Performance Requirements page

Enable Requirements

Select to enable the use of performance and functional requirements for this schedule.

Name

Specifies the name of this set of requirements. This name is used in the Requirements report. By default, the name is `Performance Schedule -schedule_name`.

Use Defaults

Click to reset **Name** to the default value.

Requirement

All the requirements are displayed in the table. Shaded requirements are not defined for this schedule. To define a requirement, set an **Operator** and **Value**.

Expand the **Custom** section, and then double-click the row to add the counter information generated by using the custom code to the requirement.

Operator

Click this field to display a list of mathematical operators. Select an operator for this requirement.

Value

Click this field to set a value for the requirement.

Standard

Select to mark the requirement as standard. If a standard requirement is not met, the schedule run will have a verdict of fail, and this verdict will roll up to the entire run, like a verification point failure. Clear to make the requirement *supplemental*. In general, supplemental requirements are those that are tracked internally. A supplemental requirement cannot cause a run to fail, and its results are restricted to one page of the Requirements report.

Hide Undefined Requirements

Select to see only the requirements that you have defined. This hides the shaded rows.

Clear

Select one or more requirements and click to remove the definition. The requirement is still available and can be redefined.

Test Log page

The default setting, to log all errors and warnings and primary test actions, fits most purposes. However, you can log any type of information, from no information to all information from all users, although neither is typical.

- To see only errors and warnings, set the first two **What to Log** check boxes to **All**; then clear the third check box, **And also show all other types**, to avoid logging successful events.
- To check the structure of a schedule, when you are not interested in the test execution results, set all three **What to Log** check boxes to **Schedule Actions**.

Both choices and the default setting limit the size of the test log and reduce the total time to run the schedule by significantly reducing the test log transfer time at the end of a test.

If you are debugging a test, you might set all three **What to Log** fields to **All** or **Action Details**. These settings produce large test logs, especially if your tests are long or you are running a large number of users. Large test logs, in turn, increase the test log transfer time, and might even cause your computer to run out of disk space.

To reduce transfer times and the likelihood of running out of disk space, sample information from a very small subset of users; smaller even than the default of 5 users per user group. A fixed sampling rate samples the same number of virtual users from each group. A percentage sampling rate samples a percentage of virtual users from each group, but guarantees that at least one user is sampled from a group.

Response Time Breakdown page

Enable collection of response time data

Select to activate response time breakdown collection. This data shows you the response time breakdown for each page element.

Detail level

Select **Low** or **Medium** to limit the amount of collected data.

Only sample information from a subset of users

If you set the detail level to **High** or **Medium**, set a sampling rate to prevent the log from getting too large.

Fixed number of users

The number that you select is sampled from each user group. Unless you have specific reasons to collect data from multiple users, select **Fixed number of users**, and specify one user per user group.

Percentage of users

The percentage that you select is sampled from each user group, but at least one user is sampled from each user group.

Problem Determination page

Problem determination log level

In general, change the problem determination level only when asked to by IBM® Software Support. However, under certain conditions, you might want to change the problem determination level. For example, if problems occur when a run reaches a certain number of users, you might increase the level to **Config**, which is the most detailed level to use without consulting IBM® Software Support.

Only sample information from a subset of users

Select this option to set a sampling rate.

Fixed number of users

Specify the number of users to sample from each user group.

Percentage of users

The percentage that you select is sampled from each user group, but at least one user is sampled from each group.

User group properties

When you open a user group, you can set these properties.

Group size

Specifies either an absolute number of users, or a percentage of users, which you control dynamically.

Locations

Run this group on the local computer

Indicates that the user group should be run on your computer.

Run this group at the following locations

Indicates that the user group should be run on one or more remote computers, at the indicated locations. Typically, you run a user group at a remote location if you are running a large number of virtual users.

Options

Use the Options page to override the think time behavior of your schedule for a specific user group and to specify protocol specific options.

Override think time option

Select this check box to specify a think time behavior for the current user group.

Use the recorded think time

Select to play back a test at the same rate that it was recorded. This option has no effect on the think time.

Specify a fixed think time

Each user's think time is exactly the same value: the value that you specify. Although this does not emulate users accurately, it is useful if you want to play a test back quickly.

Increase/decrease the think time by a percentage

Type a percentage in the **Think time scale**. Each user's think time is multiplied by that percentage. A value of 100 causes no change in think times. A value of 200 doubles the think times, so the schedule plays back half as fast as it was recorded. A value of 50 reduces the think times by half, so the schedule plays back twice as fast. A value of 0 indicates no delays.

Vary the think time by a random percentage

Each user's think time is randomly generated within the upper and lower bounds of the percentages that you supply. The percentage is based on the recorded think time. For example, if you select a **Lower limit** of 10 and an **Upper limit** of 90, the think times will be between 10 percent and 90 percent of the original recorded think time. The random time is uniformly distributed within this range.

Limit think times to a maximum value

Setting a maximum think time is useful with tests that emulate actual think times. By setting a maximum, you do not have to search for and edit each long think time within a test, if, for example, you are interrupted during recording. No think time used will be greater than the maximum limit you set, even if you have chosen to vary the think time by a percentage that would exceed this maximum. To restore the original think times, clear this box.

Protocol-specific options

Click **Edit options** to set protocol-specific options for all tests in the user group. These settings override the protocol-specific options of the schedule.

Variable Initialization

Use this page to initialize variables at the user group level. When you initialize variables at the user group level, all the tests in the user group use the variables. If the same variable is defined at the schedule level, precedence is given to the variable at the user group level.

Add

Add a variable and initialize a value. The **Used by** column displays the test name that uses the corresponding variable. A warning icon is displayed for a variable that override the value specified at the schedule level or user group level and uses the value defined at the test level with the visibility set to **This test only**. Hover the cursor over the warning icon to view the tests that overrides the variable initial values.

Export

Export the variables defined at the user group level to a file.

Use variable initial values file

Select this check box to use the variable values from a file. Click **Browse** to select an existing file or click **New** to create a file.

WSDL security editor reference

With the Web Service Description Language (WSDL) security editor you can create and edit security configurations for a WSDL file.

Keystores

In this page, you can edit the keystores that are used for the WSDL file. The keystore contains the public and private keys that are required for the specified security protocol.

Defined Keystores

Click Add or Remove to add or remove keystore files from the workbench.

Keystore Details

This specifies the location and file name of the selected keystore. Click **Browse** to select a different file.

Name

This specifies the name of the keystore. This name is used throughout the test instead of the file name.

File

Click **Browse** to specify a keystore file containing a valid server certificate. The following formats are supported:

- KS
- JKS
- JCEKS
- PKCS12 (p12 or PFX)
- PEM

Password

If the keystore file is encrypted, type the required password.

Security Stacks

In this page you can edit the security algorithm stacks that the security protocol uses. Security stacks are a set of algorithms that are executed in a given order.

Security Stacks

Click **Add**, **Remove**, or **Rename** to add, remove, or rename the security stacks that are associated with the WSDL file.

Security Algorithm Details

Click **Add**, **Insert**, or **Remove** to add or remove security algorithms in the stack. Click **Up** and **Down** to change the order of a selected algorithm in the security stack. The following security algorithms can be added to the security stack:

Time Stamp

The time stamp security algorithm adds time stamp information to the XML document in the response. For details on security algorithms, refer to the web service security specification.

Actor / Role name

Specify the name of the recipient of the algorithm header element, if required.

Must understand

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

Expiration delay

Specify the delay after which the time stamp expires.

Millisecond precision

Select this option to produce a time stamp that uses millisecond precision instead of the default (1/100th second).

User name token

The user name token security algorithm adds a user name token to the XML document in the message. For details on security algorithms, refer to the web service security specification.

Actor / Role name

Specify the name of the recipient of the algorithm header element, if required.

Must understand

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

Name

Type the name of the user.

Password

Type the password of the user.

Password type

Specify the password type for the security algorithm as defined in the Web Services Security UsernameToken profile.

Use nonce

Select this check box to add the Nonce element to the User Name Token XML code. In most cases, the Nonce ID is required.

Use created

Select this check box to add current timestamp to the Created XML element in the User Name Token XML.

XML Encryption

The XML encryption security algorithm specifies how the XML document is encrypted. For details on security algorithms, refer to the web service security specification.

Actor / Role name

Specify the name of the recipient of the algorithm header element, if required.

Must understand

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

Identifier type

Select the type of key identifier to be used for the encryption. The following key identifiers are available, as defined in the Web Services Security (WSS) specification X509 profile and the OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- SKI_KEY_IDENTIFIER
- EMBEDDED_KEYNAME
- THUMBPRINT_IDENTIFIER
- ENCRYPTED_KEY_SHA1_IDENTIFIER

User XPath part selection

This enables you to specify an XPath query that describes parts of the XML document that can be subjects of the algorithm. By default, the body is the subject.

Key

Select the key used for the encryption. The details of each key vary.

- **x509 key**: This specifies the name and password of the x509 key and the keystore where it is located.
- **Raw key**: This specifies the name and the byte value of your SecretKey in hexadecimal.
- **Encrypted key**: This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

Encoding Algorithm Name

Specify the encryption method to be used as defined in the XML Encryption Syntax and Processing specification.

Key Encoding Algorithm

Specify the standard algorithm for encoding the key as defined in the XML Encryption Syntax and Processing specification.

XML Signature

The XML signature security algorithm specifies how the XML document is signed. For details on security algorithms, refer to the web service security specification.

Actor / Role name

Specify the name of the recipient of the algorithm header element, if required.

Must understand

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

Security token

Select the type of key identifier to be used for the signature. The following key identifiers are available, as defined in the the Web Service Security (WSS) specification X509 profile and OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- SKI_KEY_IDENTIFIER
- KEY_VALUE
- USER_NAME_TOKEN
- CUSTOM_SYMM_SIGNATURE

In addition, the following identifiers are available when the signature is based on a UsernameToken profile:

- USER_NAME_TOKEN
- CUSTOM_SYMM_SIGNATURE

User XPath part selection

Specify an XPath query that describes parts of the XML document that can be the subjects of the algorithm. By default, the body is the subject. Click the **XPath Helper** button to build the Xpath expression.

Key

Select the key used for the encryption. The details of each key vary.

- **x509 key:** This key specifies the name and password of the x509 key and the keystore where it is located.
- **User name token key:** This specifies a user name and password for the signature.
- **Encrypted key:** This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

Signature algorithm name

Specify the signature method algorithm as described in the XML Signature Syntax and Processing specification.

Canonicalization

Specify the canonicalization method to be used as described in the XML Signature Syntax and Processing specification.

Digest algorithm method

Specify which digest method to be used based on the algorithm method used on the server side.

Inclusive namespaces

Specify whether the canonicalization is exclusive as described in the Exclusive XML Canonicalization specification.

Custom Security Algorithm

If you want to use a Java™ class as a custom security algorithm, then use this stack element to apply the custom algorithm to the service.

Java™ Project

If you have not implemented a custom Java™ class, select **Java Project**, type a name for the new project, and click **Generate** to create a new Java™ class with the default structure for custom security implementations.



Note: If you are using IBM® Security AppScan®, this field is not available.

Implementation class

Specify the name of the class that implements the custom security algorithm. Click **Browse Class** to select an existing Java™ class from the workspace.

Properties

Use this table to send any specific properties and associated values to the custom security algorithm.

Security Considerations

This document describes the actions that you can take to ensure that your installation is secure, customize your security settings, and set up user access controls.

- [Enabling secure communication between multiple applications on page mccxxxix](#)
- [Ports, protocols, and services on page mccxxxix](#)
- [Customizing your security settings on page mccxxxix](#)
- [Privacy policy considerations on page mccxxxix](#)
- [Security limitations on page mccxxxix](#)

Enabling secure communication between multiple applications

The workbench computer that controls the execution of the test communicates with the remote agent computers. The agents apply load for HCL OneTest™ Performance. The communication can be secure or nonsecure. By default, the tool use nonsecure communication. Also, if a workbench computer uses a self-signed certificate, it cannot be changed. Agent computers are trusting.

- [Configuring port numbers for agents on page 554](#)

Ports, protocols, and services

The Majordomo service running on remote agents must run with administrator or super user credentials, which means that the test execution it supports has full privileges on the test computer where it resides. Product communication uses HTTP and HTTPS. Ports are configurable.

- [Configuring port numbers for agents on page](#)

Customizing your security settings

Datasets can be encrypted and access controlled by password that is difficult, but not impossible, to break.

- [Encrypted datasets overview on page](#)

Privacy policy considerations

This software offering does not use cookies or other technologies to collect personally identifiable information. For additional information on cookies, see the [Notices on page mccxl](#) topic.

Security limitations

Passwords are stored using Eclipse mechanisms that are difficult but not impossible to break.

Workbench and agent communication is encrypted but not absolutely safe from impersonation attack.

Notices

This document provides information about copyright, trademarks, terms and conditions for the product documentation.

© Copyright IBM Corporation 2001, 2016 / © Copyright HCL Technologies Limited 2016, 2020

This information was developed for products and services offered in the US.

HCL® may not offer the products, services, or features discussed in this document in other countries. Consult your local HCL® representative for information on the products and services currently available in your area. Any reference to an HCL® product, program, or service is not intended to state or imply that only that HCL® product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any HCL® intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL® product, program, or service.

HCL® may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

For license inquiries regarding double-byte character set (DBCS) information, contact the HCL® Intellectual Property Department in your country or send inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. HCL® may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-HCL® websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this HCL® product and use of those websites is at your own risk.

HCL® may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

HCL

330 Potrero Ave.

Sunnyvale, CA 94085

USA

Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by HCL® under terms of the HCL® Customer Agreement, HCL® International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL® products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL® has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL® products. Questions on the capabilities of non-HCL® products should be addressed to the suppliers of those products.

Statements regarding HCL®'s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to HCL®, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. HCL®, therefore, cannot guarantee or imply reliability,

serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. HCL® shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).

Portions of this code are derived from HCL Ltd. Sample Programs.

© Copyright HCL Ltd. 2000, 2017.

Trademarks

HCL®, the HCL® logo, and hcl.com® are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. Other product and service names might be trademarks of HCL® or other companies.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the HCL® website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of HCL®.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of HCL®.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

HCL® reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by HCL®, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

HCL® MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Index

A

- Access web reports remotely 800
- accessibility
 - keyboard shortcuts (testing) 1166
- adapters
 - Rational Quality Manager 132, 136, 137, 140
- agent
 - configure port numbers 554
- agent computers
 - deleting locations 403
 - renaming locations 402
- agent status 553
- AIX
 - configuring IP aliases 568
- Apache HTTP Server
 - resource monitoring 578
- Apache Tomcat
 - resource monitoring 579
- APIs
 - code generation (SDK) 755
 - data correlation package (SDK) 721
 - evaluating results (SDK) 771
 - performance test object interfaces (SDK) 718
 - runtime (SDK) 762
 - schedules (SDK) 748
 - verification points packages (SDK) 724
- application servers
 - instrumenting for data collection 108, 111
 - removing instrumentation 113
 - removing the data collection infrastructure 112
- assets
 - testing shared assets 141
- asynchronous calls
 - adding to web service tests 370
- asynchronous requests
 - adding to web service tests 369
- asynchronous service 367
- attachment verification points
 - adding 344, 346
- authentication
 - SSL 506
- authentication folders
 - adding to requests 287

B

- batch input transactions
 - SAP 322
- binary content
 - viewing 519
- binary data 473
 - creating transformations 481
 - transforming 478, 478
- breakpoints
 - setting in Citrix 657
- browsers
 - recording preferences 1174
- byte counters
 - types 847

C

- call counters
 - services 883
- callback
 - adding to web service tests 369

- calls
 - web service tests 348
- captured data
 - defining 691
- Citrix
 - adding elements to tests 329
 - counters
 - image synchronization 875
 - overview 873
 - user action 880
 - window 880
 - window synchronization 873
 - editing text inputs 332
 - image synchronization 334, 335
 - overview 1222
 - screen areas 333
 - inserting new recordings 220
 - key action details 1218
 - logoff details 1225
 - mouse action details 1220
 - mouse sequence details 1221
 - mouse sequences 330
 - performance testing
 - overview 53
 - performance testing guidelines 209
 - recorder preferences 1182
 - recording tests 212, 215, 220
 - reports 832
 - screen capture details 1222
 - session details 1215
 - synchronization overview 325
 - test details 1214
 - test editor
 - preferences 1183
 - test editor overview 324
 - test generation preferences 222, 1184
 - test input details 1220
 - timer counters 878
 - verification points 326, 327
 - overview 326
 - window titles 328
 - window events
 - event details 1217
 - windows
 - details 1216
 - Citrix breakpoints
 - setting 657
 - Citrix monitoring panel
 - disabling 654
 - enabling 654
 - Citrix Performance Report 832
 - Citrix performance tests
 - response times 329
 - Citrix tests
 - custom Java code 336
 - debugging 654
 - ClearCookies class 671
 - clients
 - generated clients in service tests 231
 - clock skew
 - correcting 791
 - close elements
 - socket 373
 - code
 - generating (SDK) 751
 - code generation
 - extending (SDK) 750
 - extension points (SDK) 751
 - extensions (SDK) 754
 - public APIs (SDK) 755
 - templates (SDK) 754
- coexistence
 - product installation 90
- colors
 - preferences 1190
- command line
 - create config file 624
 - schedule runs 627
- comments
 - adding (HTTP) 383
 - performance testing 383
- Compare report
 - launching automatically 1192
- Compound tests
 - adding tests 485
 - adding to Test Workbench projects 488
 - creating 483
 - modifying 485
 - overview 483
 - running 486
 - viewing 484
- ComputerSpecific class 672, 673
- conditional statements
 - adding to tests 384
 - preferences 1190
- conditions
 - error handling 1169
- configuration
 - SSL authentication 506
 - WebSphere MQ transport 500
- configurations
 - reusing tests 307
 - SAP 105
 - SAP batch input tests 106
 - service testing
 - SOAP security 228, 499
 - service tests 225
 - socket 250
 - TN3270 testing 251
 - web service
 - Custom security algorithms 363
 - SOAP security 353
 - WS-Addressing 366
 - web service security 352
 - web services
 - WS-Policy 354
- configure port number for agents 554
- connection details
 - SAP 1206
- contain verification points
 - web service 340
- conventions
 - installation 88
- cookies
 - clearing during run 299
 - setting and clearing for virtual users 671
- CountAllIterations class 669
- counter
 - manage counters 790
- counters
 - Citrix 873
 - image synchronization 875
 - user action 880
 - window 880
 - window synchronization 873
 - Citrix timers 878

- extending (SDK) 765
 - HTTP 847
 - byte 847
 - page performance 848
 - run 859
 - test 861
 - SAP 866
 - screen (SAP) 866
 - services
 - call 883
 - overview 883
 - verification points 894
 - socket API 904, 904
 - standard deviation (SDK) 765
 - transaction 862
 - CountUserIterations class 669
 - creating
 - Service stubs 491
 - CSV format
 - exporting results 794
 - Custom code
 - debug 683
 - custom counters
 - test execution services 678
 - custom Java code
 - code execution counts 669
 - controlling loops 663
 - creating 657
 - custom counters 678
 - deleting 403
 - determining where a test is running 672, 673
 - extracting strings 674
 - migrating 687
 - overview 657
 - performance 662
 - printing input arguments to a file 668
 - renaming 402
 - retrieving the maximum JVM heap size 676
 - retrieving virtual user IP address 668
 - running a program with a test 677
 - setting and clearing cookies for virtual users 671
 - statistics 680
 - transactions 680
 - verification points 683
 - Custom security algorithms
 - web service 363
- D**
- data collection
 - response time breakdown 593
 - data collection infrastructure
 - configuring 107
 - instrumenting 108, 111
 - Oracle WebLogic Application Server 110
 - overview 107
 - removing 112, 113
 - WebSphere Application Server 110
 - data correlation
 - automatic 448
 - custom code example 674
 - description 444
 - disabling 472
 - extending (SDK) 724
 - extending for test execution (SDK) 726
 - extending for test generation (SDK) 725
 - manual 460
 - overview 443, 458
 - packages (SDK) 721
 - preferences 1175, 1190
 - problem identification 469
 - regenerating tests 473
 - rules 1168
 - Siebel 313
 - substituting 461
 - tests
 - data correlation 443
 - troubleshooting 472
 - viewing 446
 - data correlation rules
 - creating references 448
 - creating substitutions 450
 - description 447, 456
 - generating 457
 - linking references to substitutions 453
 - linking substitutions to built-in data sources 454
 - linking substitutions to references 453
 - recording tests 456
 - reкорrelating test data 455
 - saving 457
 - viewing logs 458
 - data sources
 - configuring 578, 579, 580, 583, 587, 587, 590
 - resource monitoring 577, 1171
 - data transformation
 - binary data 473
 - encoded data 473
 - datasets
 - creating in test 406
 - creating in workspace 410
 - deleting 403
 - digital certificates 427
 - editing 420
 - encrypting 426
 - encryption 425
 - navigating to tests 428
 - options 414
 - organizing 275
 - overview 405
 - removing encryption 427
 - renaming 402
 - segmented
 - row assignment 414
 - substitution modifications 1198
 - test references 415
 - test value associations 417
 - typical 414
 - viewing in tests 419
 - Debug custom code 683
 - debugging 645, 649
 - Citrix tests 654, 657
 - debugging 655
 - default reports
 - changing 790
 - defects
 - submitting 144
 - delays
 - adding to schedules 540
 - HTTP tests 618
 - performance test generation preferences 1175
 - deployment directory
 - deleting deployment directory 555
 - digital certificates
 - authentication in tests 266
 - creating 261, 263
 - overview 258
 - types 259
 - using in tests 266
 - using with datasets 427
 - disconnect agent 553
- E**
- Eclipse
 - fully-enabled
 - start-up 61
 - installing with Eclipse instance 90
 - streamlined 61
 - editing tests
 - generating data correlation rules 457
 - saving data correlation rules 457
 - editors
 - common editor framework (SDK) 734
 - contributing menu actions (SDK) 738
 - creating actions (SDK) 739
 - extending (SDK) 728
 - extending menus (SDK) 738
 - extension points (SDK) 740
 - methods to extend (SDK) 741
 - structure (SDK) 733
 - element labels
 - batch connection details 1213
 - batch input transaction details 1213
 - get (SAP) 1209
 - SAP 1210
 - screen details 1207
 - set (SAP) 1208
 - encrypting
 - recording session data 272
 - Entrust
 - overview 266
 - equal verification points
 - socket tests 375, 376, 377
 - web service 339
 - error handlers
 - code generation (SDK) 745
 - defining (SDK) 741
 - displaying (SDK) 742, 743
 - error handling
 - conditions 1169
 - extending (SDK) 741
 - HTTP 297
 - run time (SDK) 746
 - error types
 - creating (SDK) 744
 - errors
 - viewing 799
 - ExecTest class 677
 - Export
 - Event Console Output 802
 - exporting 1193
 - performance test assets 396
 - performance test projects 397
 - reports
 - to .view file 795
 - to HTML format 793
 - results
 - to CSV format 794
 - extending (SDK) 738
 - extension points
 - code generation behavior (SDK) 751
 - common editor framework (SDK) 734
 - RPTReport (SDK) 769
 - schedules (SDK) 748
 - test editors (SDK) 740
- F**
- file attachments
 - opening 522
 - files

- printing input arguments to 668
 - Firefox
 - browser settings for recording 182
 - recording tests 1174
 - fonts
 - preferences 1190
 - functional test reports
 - generating 775
- G**
- generated clients
 - recording service tests 231
 - generic service client 495
 - overview 56
 - graphs
 - customizing 788
- H**
- HCL
 - OneTest
 - Performance
 - migrating from earlier releases 104
 - supporting new protocols (SDK) 688
 - HCL
 - OneTest
 - Performance
 - Protocol Extensibility SDK
 - advanced annotation concepts 710
 - APIs 762
 - code generation
 - APIs 755
 - extension points 751
 - extensions 750, 754
 - templates 754
 - code generation for error handlers 745
 - common editor framework 734
 - contributing error handlers 741
 - contributing menu actions 738
 - creating a script class 752
 - creating error types 744
 - data correlation
 - execution 726
 - extending 724
 - packages 721
 - test generation 725
 - defining captured data 691
 - defining clients 693
 - defining error handlers 741
 - defining packet converters 704
 - defining recorder abilities 695
 - defining recorder wizards 696
 - defining recorders 692
 - defining test generation 702
 - defining test-generation wizards 706
 - displaying error handlers 742, 743
 - editor extension points 740
 - extending counters 765
 - extending default reports 769
 - extending editor 728
 - extending evaluation results 764
 - extending reports 769
 - extending test generation 702
 - extending test recorder 690
 - extending the annotation toolbar 709
 - extending the log viewer (SDK) 763
 - extending the test recorder 708, 708
 - generating elements from annotations 710
 - generating test code 751
 - guidelines 688
 - implementing runtime error handling 746
 - initialization and finalization 761
 - load test behavior model APIs 718
 - LTBM extension points 713, 717
 - LTBM model element registration 713
 - LTBM overview 711
 - LTBM protocol
 - constructs 716
 - LTBM required attributes 713
 - LTBM updates 712
 - menu actions 739
 - migrating recorder implementations 699
 - migrating test generation 707
 - overview 688
 - performance test object (SDK) 718
 - protocol extension structure 688
 - recording without a UI 695
 - registering LTBM elements (example) 715
 - results evaluation APIs 771
 - run-time extensions 756
 - schedules
 - APIs 748
 - component extensions 747
 - extension points 748
 - subsystem management 757
 - test editor
 - structure 733
 - test generation without a UI 706
- headers
- adding to tests 200
 - editing contents in tests 202
- high-resolution timer
- configuring 555
- host names
- reusing 307
- HTML format
- exporting reports 793
- HTTP 190
- counters overview 847
 - performance testing
 - overview 52
 - proxy server enablement 1205
 - run counters 859
 - test editor overview 278
 - test generation preferences 205
- HTTP data
- exporting as text 652
- HTTP endpoint
- service call 510
- HTTP proxies
- recording web service tests 229
- HTTP tests
- debugging 644
 - Kerberos protocol 267
- HTTP traffic
- recording for tests 183
- HTTP transport
- services 495
- I**
- IBM DB2
 - resource monitoring 580
 - IF-THEN statements
 - adding to tests 384
 - common interfaces package (SDK) 719
 - image synchronization 325
 - adding values to 335
 - Citrix 1222
 - counters 875
 - manually adding 334
 - tests 333
- importing
- reports
 - to .view file 795
- installation
- extending an Eclipse instance 90
 - launchpad program 92
 - locations 89
 - terminology 88
- Installation Manager
- overview 88
- installing packages
- Installation Manager 88
- installing products
- coexistence 90
- instrumenting
- application servers for data collection 111
- integrating
- Rational Team Concert 142
- Integrating
- Micro Focus ALM 176
- Internet Explorer
- browser settings for recording 182
- invoke
- HTTP service call 510
 - JMS service call 511
 - service call with a WSDL file 518
- IP addresses
- retrieving from virtual user 668
 - virtual users 567
- IP aliases
- configuring (AIX) 568
 - configuring (Linux) 568
 - configuring (Windows) 567
- IP aliasing
- enabling 534
- ITCAM
- response time breakdown 1172
- J**
- Jaeger
 - 167
 - Jaeger logs
 - testlogs 167
 - Java
 - test execution services 657
 - Java code
 - custom code in Citrix tests 336
 - Java Virtual Machine
 - resource monitoring 583
 - JBoss Application Server
 - resource monitoring 584
 - JMS endpoint
 - service call 511
 - JMS transport
 - services 499
 - JVM heap size
 - retrieving maximum 676
 - JVM_Info class 676
- K**
- Kerberos
 - browser configurations 269, 270
 - editing tests 383
 - generating tests 270
 - overview 267
 - recording applications 269, 270
 - key actions
 - Citrix 1218
 - keyboard actions
 - editing (Citrix) 331
 - keyboard shortcuts

- testing 1166
- L**
 - launch configurations
 - default 608
 - test settings 621
 - launchpad program
 - installing the product 92
 - line speeds
 - delaying 537
 - Linux
 - default browser 1174
 - IP alias configurations 568
 - open file limit 180
 - listeners
 - setting for recording 1174
 - load test behavior model (LTBM)
 - data correlation package (SDK) 721
 - extending classes (SDK) 717
 - extension points (SDK) 713
 - overview (SDK) 711
 - performance test object interfaces (SDK) 718
 - protocol constructs (SDK) 716
 - public APIs 718
 - registering elements (SDK example) 715
 - registering model elements (SDK) 713
 - required attributes (SDK) 713
 - updates (SDK) 712
 - verification points packages 724
 - locations
 - deleting 403
 - organizing 275
 - remote 538, 569
 - renaming 402
 - user groups 1231
 - log viewer
 - extending (SDK) 763
 - logoff details
 - Citrix 1225
 - logs 797
 - exporting HTTP data as text 652
 - exporting test events 801
 - levels
 - overview 597
 - problem determination 604
 - sampling rates 599
 - services performance tests 225
 - setting for maximum performance 180
 - socket performance tests 250
 - statistics 597
 - TN3270 performance tests 251
 - problem determination levels 642
 - stub server activity 494
 - viewing 651
 - viewing adjustments 802
 - viewing data correlation rules usage 458
 - viewing test events 799
 - long duration tests 610
 - loops
 - adding to schedules 560
 - adding to tests 387
 - controlling 663
 - iteration rates 563
 - preferences 1190
 - searching tests 394
 - setting up efficiently 180
 - virtual user memory allocation 299
 - LTTest packages
 - SDK 718
- M**
 - manage counters 790
 - memory
 - increasing 638
 - setting agent 180
 - menus
 - contributing actions (SDK) 738
 - creating actions (SDK) 739
 - message content
 - viewing 519
 - methods
 - extending common editor framework (SDK) 741
 - Microsoft .NET transport
 - services 503
 - migration
 - custom Java code 687
 - LTBM updates (SDK) 712
 - performance testing assets 104
 - mobile native application testing 190
 - modifying packages
 - installation manager 88
 - monitoring panel
 - debugging Citrix tests 655
 - mouse
 - actions
 - adding to tests (Citrix) 330
 - details (Citrix) 1220
 - test editor preferences (Citrix) 1183
 - sequences
 - Citrix 1221
 - viewing (Citrix) 330
 - Mozilla
 - browser settings for recording 182
 - recording 1174
 - mySAP
 - test support 1175
- N**
 - network traffic
 - emulating slower 537
 - notification-based services
 - testing 367
 - NT/LAN Manager (NTLM)
 - enabling 1205
 - NTLM
 - see NT/LAN Manager 1205
- O**
 - obfuscating
 - recording session data 272
 - OpenSSL
 - digital certificates 261
 - optical character recognition
 - Citrix configuration guidelines 209
 - Oracle Database
 - resource monitoring 587
 - Oracle WebLogic Server
 - resource monitoring 587
- P**
 - package groups
 - coexistence 90
 - installation locations 89
 - packages
 - common interfaces (SDK) 719
 - data correlation (SDK) 721
 - LTTest (SDK) 718
 - runtime (SDK) 762
 - verification points (SDK) 724
 - page counters
 - types 848
 - page elements/pages
 - response time breakdown 595
 - page percentile reports
 - performance test preferences 1194
 - performance testing 827
 - pages
 - error handling 297
 - merging in tests (HTTP) 304
 - performance test generation preferences 1175
 - protocol data view preferences 1173
 - searching tests 394
 - splitting in tests (HTTP) 303
 - test report elements 824
 - title specifications in tests 292
 - title verification points 1198
 - viewing test data 419, 428
 - viewing test request data 428
 - ParseResponse class 674
 - percentile reports
 - performance testing 836
 - performance profiling data
 - data collection infrastructure 107
 - Performance report 814
 - performance requirements
 - customizing percentiles 1194
 - defining in schedule 558
 - defining in tests 285
 - requirements
 - defining in schedule 558
 - defining in tests 285
 - Performance Requirements report
 - performance testing 805
 - performance test preferences 1194
 - performance test report export preferences 1193
 - performance test report preferences 1193
 - performance testing
 - Citrix configuration guidelines 209
 - overview (Citrix) 53
 - overview (HTTP) 52
 - overview (SAP) 52
 - reports
 - overview (SAP) 829
 - setting up efficiently 180
 - socket API 58
 - socket guidelines 250
 - supporting new protocols (SDK) 688
 - TN3270 applications 59
 - TN3270 guidelines 251
 - web service
 - overview 55, 490
 - performance testing sample Daytrader 86
 - performance testing sample installation 85
 - performance testing sample overview 84
 - performance testing sample Plants 85
 - performance testing sample Snoop 85
 - performance tests
 - socket reports overview 846
 - ports
 - configuring for different locations 622
 - setting the recorder listener 1174
 - preferences
 - changing recording (Citrix) 221
 - changing recording (HTTP) 865
 - destination filtering 1190
 - editor 1174
 - endpoint
 - endpoint filtering
 - filtering 1190

- recording 1190
- performance tests reports 1193
- recording (Citrix) 1182
- recording (HTTP) 1174
- reports 1192
- socket test generation 1187
- test editor 1190
- test editor (Citrix) 1183
- test generation
 - changing Citrix preferences 222
 - changing HTTP preferences 205
 - changing web service preferences 249
 - Citrix overview 1184
 - overview 1175
 - SAP overview 1180, 1180
- prerequisites
 - Siebel testing 311
- PrintArgs class 668
- problem determination levels
 - changes during schedule runs 642
 - setting 604
- problem identification
 - data correlation 469
- problems
 - troubleshooting 908
- profiling data
 - test options 1194
- projects
 - creating 181
 - deleting 403
 - renaming 402
- properties
 - schedules 1226
- protocol data
 - preferences 1173
- Protocol Data view
 - SAP
 - test editor preferences 1179
 - viewing SAP GUI data 323
 - viewing test in 309
 - watching virtual user in 647
- protocol extensions
 - creating constructs (SDK) 716
- protocols
 - supporting new (SDK) 688
- proxies
 - enabling servers 1205
 - setting the local port 1174

R

- Rational Application Performance Analyzer
 - response time breakdown 592
- Rational Performance Tester Protocol Extensibility SDK
 - common interfaces for extending model elements 719
 - test editor 738
 - framework 741
 - verification points packages 724
- Rational Quality Manager
 - adapter 140
 - configuring the adapter 132
 - running the adapter 137
 - starting the adapter 136
 - testing shared assets 141
- Rational Service Tester for SOA Quality
 - migrating from earlier releases 104
- Rational Team Concert
 - defect tracking 144
 - integrating 142

- raw transaction data
 - viewing 519
- receive elements
 - merging 256
 - socket 374
- recording
 - changing preferences
 - Citrix 221
 - HTTP 865
 - Citrix tests
 - inserting new recordings 220
 - overview 209
 - web interface 215
 - extending (SDK) 690, 708, 708
 - HTTP tests 183
 - HTTP tests (overview) 182
 - sensitive data 272
 - service tests
 - generated clients 231
 - overview 225
 - setting preferences
 - Citrix 1182
 - HTTP 1174
 - socket tests
 - procedures 252, 254
 - web service tests
 - BPEL resources 242
 - creating manually 244
 - HTTP proxies 229
 - WebSphere MQ tests
 - creating manually 245
 - XML call tests
 - creating manually 248
- recording tests
 - Citrix tests 212
 - data correlation rules 456
- recordings
 - regenerating tests 275
- recorelating test data
 - data correlation rules 455
- redirects
 - correlating URL paths 1175
 - HTTP tests 276
- reference links
 - data correlation rules 453
- references
 - creating with rules 448
 - test datasets 428
 - viewing 466
- regular expressions
 - creating data correlation rules 448, 450
- Reliable messaging 521
- remote locations
 - configuring differing ports for tests 622
 - declaring 534
 - memory allocation increases 638
 - running user groups 538
 - setting IP aliases 569
- remote WSDL files
 - synchronization 520
- replace
 - preferences 1190
- replacing text
 - searches 394
- reports 1193
 - .view format exports 795
 - changing colors 1192
 - changing default 790
 - Citrix
 - performance report 832
 - types 832

- comparisons 773, 773
- customizing graphs 788
- displaying 800
- extending (SDK)
 - counters 765
 - default 769
 - evaluation results 764
 - RPTReport extension point 769
- filtering results 781
- functional test 775
- HTML format exports 793
- HTTP 814
- migration 104
- page elements 824
- performance (SAP) 829
- Performance Requirements 805
- performance test percentile preferences 1194
- performance test preferences 1193
- performance testing percentiles 827, 836
- preferences 1192
- reports
 - HTTP performance report 814
- response time breakdown
 - filtering 596
- SAP Performance 829
- service
 - performance report 837
- socket performance 846
- socket performance tests 846
- synchronization point 806
- Transaction 809, 811
- Transaction Net Server Time 812
- verification points
 - citrix 835
 - HTTP 828
 - SAP 832
 - web service 842
 - web service
 - types 837
- requests
 - adding authentication folders 287
 - adding headers 200
 - disabling HTTP 305
 - editing header contents 202
 - enabling HTTP 305
 - primary 1198
- requirements
 - customizing percentiles 1194
 - software installation 87
- resource monitoring
 - adding data sources 577
 - Apache HTTP Server 578
 - Apache Tomcat 579
 - data sources 1171
 - enabling 576
 - IBM DB2 580
 - IBM Tivoli Monitoring 582
 - IBM WebSphere Performance Monitoring Infrastructure 582
 - Java Virtual Machine 583
 - JBoss Application Server 584
 - Microsoft Windows Performance Monitor 585
 - Oracle Database 587
 - Oracle WebLogic Server 587
 - overview 570
 - SAP NetWeaver 589
 - Simple Network Management Protocol 590

- UNIX rstatd 590
- resources
 - monitoring 570
 - organizing 275
- response codes
 - specifying expected 293
- response size
 - specifying 294
- response time
 - adjustments 802
 - disabling 803
 - Citrix 1218
- response time breakdown
 - configuring 107
 - data sources 1172
 - enabling in tests 593
 - enabling on Windows 7 593
 - enabling on Windows Server 2008 593
 - enabling on Windows Vista 593
 - filtering 596
 - logging levels 594
 - overview 592
 - page element data 595
- response time breakdown data
 - instrumenting application servers 108, 111
- response times
 - measuring in Citrix tests 329
 - reports 812
 - verifying (SAP) 318
- responses
 - adding headers 200
 - adding to service tests 350
 - editing header contents 202
 - searching tests 394
 - skipping binary responses 1174
- results
 - APIs for results evaluation (SDK) 771
 - CSV format exports 794
 - deleting 403
 - extending (SDK) 764
 - filtering 781
 - organizing 275
 - renaming 402
- results evaluation
 - extending (SDK) 764
 - public APIs (SDK) 771
- RPT Protocol Extensibility Tester SDK
 - test recorder 690
- RPT SDK
 - defining captured data 691
- RPT software development kit
 - defining recorder abilities 695
- rstatd
 - resource monitoring 590
- rule sets
 - data correlation 448
- rules
 - data correlation 447, 1168
- run counters
 - types 859
- run-time environments
 - extending (SDK) 756
 - extending initialization and finalization (SDK) 761
 - public APIs (SDK) 762
- runs
 - displaying reports after 800
 - setting durations 527
 - stopping 643

S

- SAP
 - adding elements to tests 319
 - adding sequence elements 320
 - batch connection details 1213
 - batch input transaction details 1213
 - batch input transactions
 - adding to tests 322
 - call details 1210
 - configuration 105
 - configuring environment 106
 - connection details 1206
 - get details 1209
 - GUI data 323
 - Performance reports 829
 - performance testing 52
 - screen counters 866, 866
 - screen details 1207
 - set details 1208
 - set events
 - adding to tests 319
 - test details 1206
 - test editing overview 313
 - test editor
 - preferences 1179
 - test editor overview 313
 - test generation preferences 1180, 1180
 - tests
 - splitting 322
 - verification points
 - adding 317
 - overview 316
 - screen title 318
- SAP NetWeaver
 - resource monitoring 589
- schedule runs
 - changing stage duration 641
 - configuring differing ports for tests 622
 - launch configurations 608
 - out-of-memory errors 638
 - overview 608
 - problem determination levels 604, 642
 - random order test elements 400
 - random order tests 564
 - statistics display settings 597
 - test log level settings 599
- schedules
 - adding
 - loops 560
 - transactions 566
 - adding delays 540
 - adding tests 556
 - adding user groups 534
 - adjusting user groups 537
 - command-line starts 627
 - creating 524
 - deleting 403
 - disabling portions of 399
 - extending (SDK) 747
 - extension points (SDK) 748
 - keyboard shortcuts 1166
 - launch configuration settings for tests 621
 - long run mode 610
 - migration 104
 - modeling workloads over time 527
 - organizing 275
 - overview 523
 - properties 1226
 - public APIs (SDK) 748
- remote locations 534, 537
- renaming 402
- resizing 523
- run configurations 622
- setting line speeds 537
- setting run durations 527
- setting user loads 527
- synchronization points 540
- screen captures
 - Citrix
 - details 1222
 - recording preferences 1182
 - test editor preferences 1183
- screen counters
 - SAP 866
- script classes
 - generating code (SDK) 752
- scripts
 - extending (SDK) 752
- SDK
 - HCL
 - OneTest
 - Performance
 - Protocol Extensibility SDK
 - 744
 - see
 - HCL
 - OneTest
 - Performance
 - Protocol Extensibility SDK
 - 741, 741, 742, 743, 745, 746
 - see software development kit 702, 707
 - extending the test recorder 708
 - See software development kit 690, 691, 692, 693, 695, 695, 696, 699, 702, 704, 706, 706, 709, 710, 710
 - extending the test recorder 708
 - search
 - preferences 1190
 - Secure Sockets Layer
 - enabling 1205
 - security
 - converting tests 308
 - keystores 1233
 - services 1233
 - stacks 1233
 - Web Service Description Language
 - editor 1233
 - security alerts
 - suppressing 183
 - see software development kit
 - extending test generation 702
 - See software development kit
 - defining packet converters 704
 - defining test generation 702
 - test generation without a UI 706
 - selectors
 - adding to schedules 564
 - adding to tests 400
 - send elements
 - merging 256
 - socket 373
 - sending
 - service request with a WSDL file 508
 - WebSphere MQ service request 512
 - sequence elements
 - adding (SAP) 320
 - service calls 56
 - service requests
 - viewing content 519
 - service responses

- adding to service tests 348
- services
 - configuring environment 228, 499
 - counters
 - call 883
 - overview 883
 - verification points 894
 - file attachments 522
 - HTTP call 510
 - HTTP transport configuration 495
 - invoking calls 495
 - JMS call 511
 - JMS transport configuration 499
 - Microsoft .NET transport configuration 503
 - recording tests
 - generated clients 231
 - security editor reference 1233
 - WebSphere MQ request 512
 - WSDL file 508, 518
- session details
 - Citrix 1215
- set events
 - SAP 319
- SetCookieFixedValue class 671
- share agent 553
- shared assets
 - testing 141
- shared resources directories
 - installation locations 89
- shortcuts
 - keyboard
 - testing 1166
- Siebel
 - tests
 - HTTP testing comparison 312
 - prerequisites 311
 - request value correlations 313
 - support settings 1175
- Simple Network Management Protocol
 - resource monitoring 590
- simulating
 - services 491
- SNMP
 - see: Simple Network Management Protocol 590
- SOAP security
 - creating configurations 353
- socket
 - configuration 250
 - recording tests
 - performance tests 254
- socket API
 - counters 904
 - performance testing 58
 - screen counters 904
 - test editor overview 371
 - tests
 - splitting 379
- socket API tests
 - performance reports 846
- socket close elements
 - adding to tests 373
- socket receive elements
 - adding to tests 374
- socket send elements
 - adding to tests 373
- socket tests
 - generation preferences 1187
 - recording
 - performance tests 252

- verification points
 - checking received data 375, 377
 - checking received data size 376
- software development kit
 - advanced annotation concepts 710
 - defining a new client 693
 - defining a new recorder 692
 - defining new types of captured data 691
 - defining recorder abilities 695
 - defining recorder wizards 696
 - defining test-generation wizards 706
 - extending RPT 690
 - extending the annotation toolbar 709
 - generating elements from annotations 710
 - migrating recorder implementations 699
 - migrating test generation 707
 - recording without a UI 695
- software installation
 - requirements 87
- split points
 - inserting during recording 194, 204, 273
- splitting
 - tests
 - SAP 322
 - socket API 379
- SSL
 - see Secure Sockets Layer 1205
- SSL authentication
 - configuration 506
- stage duration
 - changing during a run 641
- stages 773
- statements
 - conditional
 - adding to tests 384
 - IF-THEN (SDK) 719
- statistics
 - CSV format exports 794
- statistics log level
 - setting 597
- statistics sample interval 225
 - long socket performance test settings 250
 - TN3270 performance test settings 251
- stress testing 387, 540
- strings
 - extracting from input arguments 674
- stub servers
 - logging activity 494
- stubbing
 - services 491
- substitution links
 - data correlation rules 453, 454
- substitutions
 - creating with data correlation rules 450
 - data correlation 461
- subsystem management
 - extending (SDK) 757
- synchronization
 - Citrix
 - overview 325
 - states 1217
 - Synchronization Point report
 - performance testing 806
- synchronization points
 - coordinating virtual users in schedules 540
 - coordinating virtual users in tests 387
- synchronization points in schedules 540, 540

- synchronization points in tests 387, 387
- synchronizing
 - remote WSDL files 520

T

- templates
 - code generation (SDK) 754
 - creating empty tests 197
 - new elements 198
- terminology
 - product installation 88
- test assets
 - importing into Rational Quality Manager 140
 - organizing 275
- test counters
 - types 861
- test data
 - reccorrelating 455, 473
 - sources 458
- test editor
 - generating data correlation rules 456
- test editors
 - common editor framework (SDK) 734
 - contributing actions (SDK) 738
 - creating actions (SDK) 739
 - extending (SDK) 738
 - API classes 741
 - overview 728
 - extension points (SDK)
 - editor layer 740
 - structure (SDK) 733
- test elements
 - running in random order 400
 - selecting multiple types 393
- test execution services
 - code execution counts 669
 - custom counters 678
 - determining where a test is running 672, 673
 - extracting strings 674
 - migrating Java code 687
 - overview 657, 657
 - printing input arguments to a file 668
 - retrieving the maximum JVM heap size 676
 - retrieving virtual user IP address 668
 - running a program with a test 677
 - setting and clearing cookies for virtual users 671
 - statistics 680
 - transactions 680
 - verification points 683
- test generation
 - changing preferences
 - web service 249
 - extending (SDK) 702
 - Kerberos 270
 - preferences 256
- test generator
 - generating test code (SDK) 751
- test recorder
 - defining (SDK) 692
 - defining clients (SDK) 693
 - extending (SDK) 690, 708, 708
- test runs
 - extending subsystem management (SDK) 757
- testing
 - keyboard shortcuts 1166
 - overview 367

- services
 - guidelines 225
- tests 645, 649
 - adding
 - comments 383
 - elements 383
 - elements (Citrix) 329
 - loops 387
 - mouse actions (Citrix) 330
 - templates 198
 - transactions 383
 - adding custom Java code 657
 - adding headers 200
 - adding to schedules 556
 - annotating during recording 271
 - annotations
 - adding during recording 271
 - automating 623, 623
 - binary data 473
 - browser settings for recording 182
 - changing displays 1174
 - Citrix
 - image synchronization 333
 - input details 1220
 - options 1214
 - Citrix preference settings 1184
 - client delays 618
 - conditional statements 384
 - configuring different ports for 622
 - connection settings 1205
 - connection timeout 619
 - converting to SSL 308
 - creating transformations 481
 - dataset column associations 417
 - dataset references 415
 - datasets 425, 426, 427
 - SAP options 1206
 - debugging 644, 645, 649
 - HTTP overview 644
 - declaring variables 431
 - delays 618
 - deleting 403
 - disabling HTTP requests 305
 - disabling portions of 399
 - editing 284
 - Citrix 324
 - Citrix overview 324
 - HTTP overview 276, 278
 - overview 276
 - SAP 313
 - service overview 337
 - socket API 371
 - web service 338
 - web service security 352
 - editing header contents 202
 - editing Kerberos tests 383
 - editor
 - preference settings (Citrix) 1183
 - enabling HTTP requests 305
 - exporting HTTP data 652
 - extending
 - controlling loops 663
 - custom Java code 657
 - failures
 - cookie caches 299
 - virtual user memory allocation 638
 - forcing logons 287
 - generating
 - preference settings (Citrix) 222
 - preference settings (SAP) 1180, 1180
 - HTTP redirect support 276
- inserting new recordings
 - Citrix 220
- keyboard shortcuts 1166
- logs 225
 - exporting 801
 - settings 599
 - socket performance test settings 250
 - TN3270 performance test settings 251
 - viewing 799
- loops
 - iteration rates 563
- manual data correlation 460
- merging pages 304
- migrating custom Java code 687
- migration 104
- modular 300
- mySAP 1175
- options (HTTP) 1194
- organizing 275
- overview 644
- page searches 394
- protocol data 649
- receive elements
 - adding 374
- recording Citrix 215
- recording Citrix tests 212
- recording HTTP traffic 183
- recording socket transactions 252, 254
- regenerating from recordings 275
- renaming 402
- results
 - settings 621
- running in random order 564
- running locally 608
- SAP
 - adding batch input transactions 322
 - adding elements 319
 - adding sequence elements 320
 - adding set events 319
 - batch connection details 1213
 - batch input transaction details 1213
 - call details 1210
 - editing overview 313
 - get details 1209
 - screen details 1207
 - set details 1208
- scaling HTTP playback rate 620
- searching overview 393
- services
 - recording with generated clients 231
 - setting up efficiently 180
- Siebel 312, 1175
- socket
 - adding close elements 373
 - adding send 373
- splitting
 - HTTP 300
- splitting during recording 194, 204, 273
- splitting pages 303
- synchronization points 387
- templates 197
- test execution services
 - code execution counts 669
 - custom counters 678
 - determining where a test is running 672, 673
 - extracting strings 674
 - printing input arguments to a file 668
 - retrieving the maximum JVM heap size 676
- retrieving virtual user IP address 668
- running a program with a test 677
- setting and clearing cookies for virtual users 671
- statistics 680
- transactions 680
- test generation preferences (HTTP) 205
- transforming binary data 478, 478
- using custom transformations 482
- using on multiple hosts 307
- variables 429
- verifying HTTP data 651
- viewing errors 799
- viewing HTTP data 309
- watching virtual user actions 647
- web service
 - adding calls 348
 - adding responses 350
 - creating manually 244
 - recording 242
- web services
 - adding asynchronous calls 370
 - adding asynchronous requests 369
 - adding callbacks 369
 - adding responses 348
 - recording with HTTP proxies 229
- WebSphere MQ
 - creating manually 245
- XML call
 - creating manually 248
- text inputs
 - editing Citrix 332
- think time
 - Citrix
 - action details 1218
 - input details 1220
 - SAP 1210
 - get details 1209
 - set details 1208
- think times
 - overview 549
 - setting maximum values 550, 551
 - settings
 - behavior 549
 - page details 1198
- time offset
 - correcting 791
- timer
 - configuring for high resolution 555
- timer counters
 - Citrix 878
- Tivoli Composite Application Manager
 - response time breakdown 1172
- Tivoli Monitoring for Transaction Performance
 - response time breakdown 1172
- TN3270
 - configuration 251
- TN3270 applications
 - performance testing 59
- tokens
 - extracting from input arguments 674
- transaction counters
 - types 862
- Transaction Net Server Time Percentile report
 - performance testing 812
- Transaction Percentile report
 - performance testing 811
- Transaction report
 - performance testing 809

- transactions
 - adding to schedules 566
 - adding to tests 383
 - preferences 1190
 - searching tests 394
- transformations
 - binary data in requests 478
 - binary data in tests 478
 - creating 481
 - using 482
- troubleshooting
 - data correlation 472
 - performance testing 908

U

- updating packages
 - Installation Manager 88
- URLs
 - displaying decoded URLs 1174
 - performance test generation preferences 1175
- user behavior
 - emulating 524
- user groups
 - adding to schedules 534
 - adjusting 537
 - locations 1231
 - overview 533
 - properties 1231

V

- variable
 - initialize variable from XML 433
- variables
 - assigning 431
 - in tests 429
 - initializing 431
- verification point counters
 - services 894
- verification points
 - Citrix
 - details 1216
 - enabling globally 326, 327
 - overview 326
 - reports 835
 - window titles 328
 - counters in socket API tests 904
 - expected response codes 293
 - HTTP
 - reports 828
 - packages (SDK) 724
 - page title test specifications 292
 - page titles 1198
 - performance test generation preferences 1175
 - performance testing
 - global enablement 292
 - specifications for expected contents 296
 - specifying expected contents 296
 - response size specifications 294
 - SAP
 - adding 317
 - adding screen titles 318
 - counters 866
 - enabling 1209
 - overview 316
 - reports 832
 - screen title 1207
 - searching tests 394
 - socket tests
 - checking received data 375, 377

- checking received data size 376
- web service
 - adding attachment verification points 344, 346
 - adding Xpath query 343
 - checking returned messages 339, 340
 - overview 339
 - reports 842
- view file
 - exporting reports 795
 - importing reports 795
- virtual users 387, 540
- counting code runs 669
- datasets 415
- emulating different IP addresses (AIX) 568
- emulating different IP addresses (Linux) 568
- emulating different IP addresses (Windows) 567
- memory allocation problems 299
- retrieving IP addresses 668
- setting and clearing cookies 671
- supplying different IP addresses 567
- user groups 533
- watching in real time 647

W

- WCF transport
 - services 503
- Web reports
 - access remotely 800
- web service
 - adding calls to tests 348
 - attachment verification points 344, 346
 - configuration 225
 - creating tests manually 244
 - Custom security algorithms 363
 - performance testing 55, 490
 - reports 837
 - security configurations 352
 - security editor overview 352
 - SOAP security configurations 353
 - test editor overview 338
 - verification points
 - checking returned messages 339, 340
 - overview 339
 - Xpath query verification points 343
- web Service
 - WSDL syntax compliance 227
- Web Service Description Language
 - security editor 1233
 - syntax compliance for JMS web services 227
- Web Service Performance Report 837
- web service responses 343
- web services
 - asynchronous service testing 367
 - recording tests
 - HTTP proxies 229
 - WS-Addressing 366
 - WS-Policy 354
- WebSphere Application Server
 - data collection infrastructure 110
- WebSphere MQ
 - creating tests manually 245
- WebSphere MQ endpoint
 - service request 512
- WebSphere MQ transport

- services 500
- WebSphere Performance Monitoring Infrastructure
 - resource monitoring 582
- weighted blocks
 - renaming 402
- window counters
 - Citrix 880
- window details
 - Citrix 1216
- window event details
 - Citrix 1217
- window event synchronization
 - Citrix 325
- window synchronization
 - counters 873
- window title verification points
 - specifying 328
- window titles
 - expected criteria in Citrix performance tests 328
- Windows
 - configuring IP aliases 567
 - default browser 1174
- Windows 7
 - enabling response time breakdown 593
- Windows Performance Monitor
 - resource monitoring 585
- Windows Server 2008
 - enabling response time breakdown 593
- Windows Vista
 - enabling response time breakdown 593
- workbench
 - efficient heap size 180
- workloads
 - emulating 522
- workspaces
 - copying projects 397
 - copying test assets 396
- WS-Addressing 521
 - creating configurations 366
- WS-Coordination 521
- WS-Policy
 - creating configurations 354
- WS-RM 521
- WSDL
 - sending a service request 508
- WSDL files
 - remote WSDL files 520
- WSDLfile
 - invoking a service call 518

X

- XML call
 - creating tests manually 248
- XML headers 521
- Xpath query verification points 343