

HCL OneDB 2.0.1

OneDB Explore



Contents

Chapter 1. OneDB Explore.....	3
Architecture.....	3
System Compatibility.....	3
Getting Started.....	4
Starting the OneDB Explore Server.....	4
Starting the OneDB Explore Agent.....	5
Logging in HCL OneDB™ Explore.....	7
HCL OneDB™ Explore Concepts.....	8
HCL OneDB™ Explore Server.....	10
HCL OneDB™ Explore Server Configuration.....	10
HCL OneDB™ Explore UI.....	14
HCL OneDB™ Explore Agent.....	39
OneDB Explore Agent Setup.....	39
OneDB Explore Agent Configuration Parameters.....	43
HCL OneDB™ Explore Server API Syntax and Examples.....	46
Upgrading OneDB Explore.....	46
Frequently asked questions (FAQs) about HCL OneDB™ Explore.....	49
High level architecture and functionality	49
Getting Started.....	50
Monitoring and the Repository Database.....	50
Security.....	51
Users and Permissions.....	54
Index.....	56

Chapter 1. OneDB Explore

OneDB Explore is a modern web console for visualizing, monitoring, and managing your OneDB server instances. It is purpose built for ease-of-use, scaling out, and optimizing DevOps needs. It provides critical performance management capabilities, monitoring how key performance metrics are changing over time and tracking how efficiently OneDB is running your workload even when you've stepped away from your screen. Its monitoring system feeds directly into a customizable alerting system so you can be immediately alerted via email, Twilio, or PagerDuty whenever an issue occurs on one of your OneDB database server instances. OneDB Explore is designed to be scalable to efficiently manage and monitor as many OneDB database server instances as you need. Moreover, it's a tool that can be shared by the DBAs, the app developers, the ops engineers, and management and accessed from any desktop, laptop, or mobile device. OneDB Explore is the centralized hub for graphical monitoring, alerting, and administration of your OneDB database servers.

Architecture

OneDB Explore consists of three distinct pieces that come together to give you a comprehensive monitoring and administering experience for OneDB.

OneDB Explore Server

- Java 8 based Jetty web server
- Monitors and administers many OneDB database servers
- Connects directly to OneDB databases servers to
 - Gather live data from the system
 - Perform administration
- Connects to the OneDB Explore agents regarding
 - Monitored data
 - Alerts/events

OneDB Explore Agent

- Lightweight Java 8 based monitoring agent
- Installed alongside each of your OneDB database instances
- Only needs read access to database server
- Can perform native command execution to gather OS statistics as well as database statistics

OneDB Explore User Interface (UI)

- Modernized web UI for monitoring, managing, visualizing, and assessing your OneDB database servers

System Compatibility

Before you install OneDB Explore, make sure that your computer meets the system requirements.

OneDB Explore Prerequisites

The following table lists the software prerequisites for OneDB Explore.

Software	Required Version
OneDB Database Server	1.0.0.0 or higher
Java	1.8

Supported Web Browsers

OneDB Explore supports all the latest browsers. Following table lists the web browser that OneDB Explore has been tested with.

Web Browser	Version
Google Chrome	81

Getting Started

This topic provides a brief tutorial to help you get started with HCL OneDB™ Explore.

Prerequisites

The following table lists the software prerequisites for HCL OneDB™ Explore.

Software	Required Version
HCL OneDB™ Database Server	1.0.0.0 or higher
Java	1.8

Starting the OneDB Explore Server

This topic provides a brief tutorial to help you get started with OneDB Explore Server.

1. Locate the `onedbe-explore-server.jar` and the `server.log4j.xml` file in the OneDBExplore package.
2. Create an [HCL OneDB Explore Server Configuration on page 10](#) file. You can refer to the `onedb-explore-server-example.properties` file as an example.

The OneDB Explore server configuration file must contain an `initialAdminPassword`. All other configuration properties are optional.

Sample configuration file

```
# required initial password for the admin user
initialAdminPassword=myPassword123
# optionally, uncomment these properties to have OneDB Explore encrypt its internal H2
database
```

```
#h2.encrypt.enable=true
#h2.encrypt.password=password
```



Note: The **initialAdminPassword** property is only required the very first time you start the OneDB Explore server, when it performs its start-up initialization and creates the first user named **admin**. Afterwards, you can remove the **initialAdminPassword** property from the **onedb-explore-server.properties** file.



Note: Password must be at least 8 characters and must contain at least one lowercase character, one uppercase character, and one number.

- Optionally, edit the *server.log4j.xml* file to [Logging in HCL OneDB Explore on page 7](#).



Note: By default OneDB Explore runs on 8080 port. For more information, see [HCL OneDB Explore Server Configuration on page 10](#).

- Start the OneDB Explore server using the following command:

```
java -Dfile.encoding=utf-8 -jar onedb-explore-server.jar -config
onedb-explore-server.properties -log4jFilePath onedb-explore-server.xml
```

where **onedb-explore-server.properties** is the name of the OneDB Explore server configuration file and **onedb-explore-server.xml** is the name of configuration file for logging.



Note: Providing `log4jFilePath` is optional.



Note: You can also start or stop the server using the script available in OneDBExplore package. For more information on this, see OneDB Explore help.

When the OneDB Explore server is started for the first time, a user with system administrator privileges for OneDB Explore is created with the username **admin** and the password as specified in the **initialAdminPassword** property from your configuration file.

- Using a web browser, go to the OneDB Explore UI at `http://localhost:8080/` and login with the user **admin** and the password specified in your configuration file.
- Once logged in, click **Add Server** to add an OneDB server that you want to monitor.

Starting the OneDB Explore Agent

This topic covers the two ways of starting the OneDB Explore Agent.

To get the most out of OneDB Explore, you should have an OneDB Explore agent running for each OneDB database server that you will be monitoring through the tool. While the OneDB Explore agent is not required to view information about your database server in the OneDB Explore UI, the agent is required if you want to gather monitoring data and configure alerts for that server.

There are two options for starting the OneDB Explore agent. You can use the OneDB Explore UI to automatically deploy and start the OneDB Explore agent or you can manually start the OneDB Explore agent on the command line.

Deploying and starting the OneDB Explore agent automatically from the UI:

Before you deploy and start the OneDB Explore agent automatically from the UI, you must meet the following prerequisites:

Prerequisites

1. SSH must be installed on the database server's host machine.
2. The *onedb-explore-agent.jar* file and the *onedb-explore-server.jar* file must be located in the same directory. For customized logging, the *agent.log4j.xml* file must also be included in the same directory.

To deploy and start the OneDB Explore agent automatically from the UI:

1. The OneDB database server that the agent will be monitoring must first be defined in OneDB Explore. If the database server has not been defined yet, in the UI, navigate to an OneDB Explore group, click **Add Server** and define the server's connection properties.
2. Navigate to the OneDB database server's **Setup > Agent** page and define a repository server and database. The repository database is where the monitored data will be stored.
3. From the server's **Setup > Agent** page, you can also try to deploy the agent with default configuration by clicking the **Deploy Agent** button. If the automatic deployment is successful, you are done. Otherwise, proceed to the next step.
4. Configure agent deployment:
 - Username: remote user which will own and run the agent
 - Password or Identity File/Passphrase: remote user's passphrase or remote user's identity file (e.g. private key) and its optional passphrase.
 - Remote directory: Directory to deploy the agent files to. This directory will be created if it does not exist.
5. If SSL is enabled, keystore configuration may be required as well. If keystore configuration is not provided, OneDB Explore will try to generate and deploy a keystore for you.
 - Keystore file: Location of an existing keystore on the remote machine (can be relative to the remote directory)
 - Keystore password: Password used to access the existing keystore
 - Keystore type: Existing keystore's type. Default is JKS
6. Click **Deploy Agent**
7. Once the agent is ready, use the OneDB Explore UI in your web browser to configure the monitoring profile and alerts for this server.

Starting an OneDB Explore agent manually on the command line:

1. The OneDB database server that the agent will be monitoring must first be defined in OneDB Explore. If the database server has not been defined yet, in the UI, navigate to an OneDB Explore group, click **Add Server** and define the server's connection properties.
2. Navigate to the OneDB database server's **Setup > Agent** page and define a repository server and database. The repository database is where the monitored data will be stored

3. Locate the *onedb-explore-agent.jar* and the *agent-log4j.xml* file in the OneDBExplore package of the OneDB Explore database server installation.
4. Copy the agent jar file and the log4j configuration file to the OneDB database server host machine.
5. Create an agent configuration file.

Sample agent configuration file:

```
# host and port of the OneDB Explore server
server.host=localhost
server.port=8080

# The id of the OneDB database server as defined in OneDB Explore
onedbServer.id=1
```



Note: You can find the id of your OneDB database server by navigating to the server's **Setup** page in the UI.

6. Optionally, edit the *agent-log4j.xml* file to [Logging in HCL OneDB Explore on page 7](#).
7. Start the OneDB Explore server using the following command

```
java -jar onedb-explore-agent.jar agent.properties
```

where **agent.properties** is the name of your OneDB Explore agent configuration file.

8. At this point the agent is ready and running. Use the OneDB Explore UI in your web browser to configure the monitoring profile and alerts for this server.

Logging in HCL OneDB™ Explore

This topic provides a brief tutorial on logging in HCL OneDB™ Explore.

The HCL OneDB™ Explore server and agent use the [log4j2](#) library for logging. By default, the HCL OneDB™ Explore server and agent will log messages at INFO level to an *onedb-explore-server.log* file and an *onedb-explore-agent.log* file respectively.

You can customize the logging behavior by providing a *server.log4j.xml* file in the current directory or classpath when starting the OneDB Explore server or a *agent.log4j.xml* file in the current directory or classpath when starting the OneDB Explore agent. Use these log4j2 configuration files to change the logging level (ERROR, WARN, INFO, or DEBUG), change the log file location, or enable rolling window logging. For more information, see the [log4j2 documentation](#).

Sample log4j2 configuration files are provided in *example-server.log4j.xml* and *example-agent.log4j.xml*.

ConsoleAppender

The ConsoleAppender writes its output to either System.out or System.err with System.out being the default target. A Layout must be provided to format the LogEvent.

RollingFile Appender

The RollingFileAppender is an OutputStreamAppender that writes to the file named in the fileName parameter and rolls the file over according to the TriggeringPolicy and the RolloverPolicy.

The CompositeTriggeringPolicy takes multiple triggering policies and returns true if any of the configured policies return true. The CompositeTriggeringPolicy is configured simply by combining other policies in a Policies element.

SizeBased Triggering Policy

Once the file reaches the specified size, the SizeBasedTriggeringPolicy causes a rollover. The size can be specified in bytes, with the suffix KB, MB or GB, for example 20MB. When combined with a time based triggering policy, the file pattern must contain a %i otherwise the target file will be overwritten on every rollover as the SizeBased Triggering Policy will not cause the timestamp value in the file name to change. When used without a time based triggering policy, the SizeBased Triggering Policy will cause the timestamp value to change.

TimeBased Triggering Policy

The TimeBasedTriggeringPolicy causes a rollover once the date/time pattern no longer applies to the active file. This policy accepts an interval attribute which indicates how frequently the rollover should occur based on the time pattern and a modulate boolean attribute.

Default Rollover Policy

The default rollover takes both date/time pattern and an integer specified in filePattern Attribute in RollingFileAppender. If the pattern contains an integer, it will be incremented on every rollover. If the date/time pattern is present, it will be replaced with current date and time values. If the file pattern ends with ".gz", ".zip", ".bz2", ".deflate", ".pack200", or ".xz", the resulting archive will be compressed using the compression scheme that matches the suffix.

This example shows a rollover strategy that will keep up to 20 files before removing them.

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <RollingFile name="RollingFile" fileName="logs/app.log"
filePattern="logs/${date:yyyy-MM}/app-%d{MM-dd-yyyy}-%i.log.gz">
      <PatternLayout>
        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
      </PatternLayout>
      <Policies>
        <TimeBasedTriggeringPolicy />
        <SizeBasedTriggeringPolicy size="250 MB" />
      </Policies>
      <DefaultRolloverStrategy max="20" />
    </RollingFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingFile" />
    </Root>
  </Loggers>
</Configuration>
```

HCL OneDB™ Explore Concepts

This topic covers some of the conceptual aspects of OneDB Explore.

Group

HCL OneDB™ Explore provides the ability to create groups of servers to make them easier to manage and monitor. HCL OneDB™ Explore's groups are based on a hierarchy. The base "root" group is the top level group for all HCL OneDB™ Explore groups and servers. From this root group, you can add as many servers and sub-groups as you desire, nesting them to whatever level makes sense for your organization.

You can define monitoring and alerting profiles for groups, simplifying the task of managing monitoring for all of your database servers.

Agent

The HCL OneDB™ Explore agent is a lightweight Java based program that is designed to run alongside each OneDB database server, gathering data about the performance of the system. The data gathered by the agent is fully configurable and is defined by the list of **sensors** in the server's **monitoring profile**. The data gathered by the agent is stored in a **repository database**.

Repository Database

A repository database holds information collected by the HCL OneDB™ Explore agent about the OneDB database server. The repository database can either be local to the database server that is being monitored or it can be on a remote OneDB instance. You can define a common repository database shared by multiple OneDB database server instances, or you can define a separate repository database for each monitored instance.

The repository database must be an OneDB database and must exist before the agent is started. You must define the database server to be used as a repository in the HCL OneDB™ Explore UI, using the **Add Server** action on any group dashboard, before it can be defined as a repository.

To define a repository database for a particular OneDB database server, go to the server's **Setup** page in the UI and click on the **Agent** tab.

Sensor

A sensor defines a metric or set of metrics for the agent to gather. An example is the "DBSpace Usage" sensor that gathers metrics on used and free space for all database server spaces.

Monitoring Profile

A monitoring profile defines the list of sensors that the agent runs to gather data about and OneDB database server instance or about its host operating system.

For each sensor in a monitoring profile, you can configure the frequency at which that sensor will run and how long that sensor's data will be kept in the repository database.

Monitoring profiles can be configured for groups as well as servers. HCL OneDB™ Explore uses the concept of inheritance for determining a particular server's or group's monitoring profile. All servers and groups inherit monitoring profile information

from its parent group in the hierarchy. Servers or groups can also disable or override the configuration of any sensors inherited from a parent group.

Alert

An alert defines a condition that should trigger an alerting incident in HCL OneDB™ Explore. An example would be an alert defined for when the OneDB database server goes offline.

Alerting profile

An alerting profile defines the set of alerts configured for a particular server or group. Alerting profiles follow an inheritance model similar to monitoring profiles.

Alerting incident

While the monitoring data is collected by the HCL OneDB™ Explore agent, it is the HCL OneDB™ Explore server that is tasked with evaluating alerting conditions. As data is collected by the agent, the HCL OneDB™ Explore server evaluates each new incoming data point against the alert definitions in the server's alerting profile. Any data point that meets an alerting condition triggers an alerting incident.

Alerting incidents are shown in the HCL OneDB™ Explore UI for that server. Alerting incident counts are also aggregated and highlighted on the group dashboards. Alerting incidents can be acknowledged as read or even deleted from the Incidents page in the UI. Users of HCL OneDB™ Explore can configure their own alerting preferences to automatically receive alerting incidents directly via email, Twilio, or PagerDuty.

HCL OneDB™ Explore Server

The HCL OneDB™ Explore server is a Java 8 based Jetty web server. The server is the heart of HCL OneDB™ Explore. It manages the monitoring profiles for all instances, communicates with agents, handles all alerting activities, hosts the web UI, and provides the REST services that the web UI depends on.

HCL OneDB™ Explore Server Configuration

A properties file is required to run the HCL OneDB™ Explore server. When starting the HCL OneDB™ Explore server, you can pass the properties file name as part of the start command. Otherwise, HCL OneDB™ Explore will look for a properties file named `OneDB_Explore-server.properties` in the classpath.

An example configuration file documenting all supported HCL OneDB™ Explore server configuration properties can be found in `OneDB_Explore-server-example.properties`.

- Required configuration properties on initial startup
 - [initialAdminPassword](#) on page 11
- Optional configuration properties
 - [alert.numberConditionCheckThreads](#) on page 11
 - [alert.startNumberAlertSendThreads](#) on page 11
 - [dataSource.IFX_ISOLATION_LEVEL](#) on page 11

- [hostname](#) on page 12
- [httpPort](#) on page 12
- [httpsPort](#) on page 12
- [h2.encrypt.algorithm](#) on page 12
- [h2.encrypt.enable](#) on page 12
- [h2.encrypt.password](#) on page 12
- [pool.connectionTimeout](#) on page 12
- [pool.maximumPoolSize](#) on page 13
- [pool.minimumIdle](#) on page 13
- [pool.idleTimeout](#) on page 13
- [redirectHTTPtoHTTPS](#) on page 13
- [rest.session.timeout](#) on page 13
- [ssl.keystore.file](#) on page 13
- [ssl.keystore.password](#) on page 13
- [ssl.key.password](#) on page 14
- [user.password.maxAge](#) on page 14
- [user.password.minLength](#) on page 14
- [user.password.requireLowerCase](#) on page 14
- [user.password.requireNumber](#) on page 14
- [user.password.requireSpecialCharacterFromSet](#) on page 14
- [user.password.requireUpperCase](#) on page 14

initialAdminPassword

When starting the HCL OneDB™ Explore server for the first time, an **admin** user will be created with the password specified in the **initialAdminPassword** property. This user will have system administrative privileges on the HCL OneDB™ Explore server which includes the ability to create other users, grant privileges, and make configuration changes to the server.

This property is only required the very first time you start the HCL OneDB™ Explore server. For security reasons, it is recommended that you remove this property from your OneDB Explore server configuration file after the HCL OneDB™ Explore server has been initialized for the first time.

alert.startNumberAlertSendThreads

Configures the number of threads in the thread pool that processes and dispatches alert notifications (by email, Twilio, Pager Duty, etc.) when an alerting incident occurs. The default number of threads is 4.

alert.numberConditionCheckThreads

Configures the number of threads in the thread pool that checks whether alerting conditions have been violated whenever new monitoring data comes in. The default number of threads is 4.

dataSource.IFX_ISOLATION_LEVEL

Specifies the isolation level to set on JDBC connections to the various OneDB database servers. The default value is 1.

hostname

The host name of the HCL OneDB™ Explore server. The host name determines the network adapter or interface that the HCL OneDB™ Explore server binds the server socket to.

The default value is an empty string. When set to an empty string, the HCL OneDB™ Explore server will bind to all available network interfaces on the host machine.

httpPort

The HTTP port to run the HCL OneDB™ Explore server on. This port will serve both the OneDB Explore web UI and the HCL OneDB™ Explore REST API. Set this value to -1 to disable the HTTP protocol for HCL OneDB™ Explore. If httpPort is set to -1, make sure that httpsPort is set to something other than -1. The default value is 8080.

httpsPort

The HTTPS port to run the HCL OneDB™ Explore server on. This port will serve both the HCL OneDB™ Explore web UI and the HCL OneDB™ Explore REST API.

Set this value to -1 to disable the HTTPS protocol for HCL OneDB™ Explore. If httpsPort is set to -1, make sure that httpPort is set to something other than -1.

If httpsPort is something other than -1, you must set the **ssl.keystore.file** and **ssl.keystore.password** properties, and potentially also the **ssl.key.password** property if your key password is different from the keystore password.

The default value is -1 indicating that HTTPS is disabled by default.

h2.encrypt.algorithm

Sets the algorithm for H2 database file encryption. The encryption algorithms supported by H2 are AES, XTEA, and FOG. The default value is AES.

h2.encrypt.enable

Controls whether the H2 database file which holds HCL OneDB™ Explore server's internal metadata is encrypted. If you set this property to true, you must also set the h2.encrypt.password property. The default value is false.

h2.encrypt.password

Sets the password to use for H2 database file encryption. If h2.encrypt.enable is set to true, you must set the password for encryption.

pool.connectionTimeout

Specifies the number of milliseconds to wait for a JDBC connection to an OneDB database server to be established before it times out. The default value is 5000 (5 seconds).

pool.idleTimeout

Specifies the number of milliseconds that a JDBC connection can be idle in the connection pool before it is closed. The default value is 60000 (1 minute).

pool.maximumPoolSize

The maximum number of JDBC connections in each connection pool. The HCL OneDB™ Explore server will maintain a connection pool for each OneDB database that it needs to connect to. The **pool.maximumPoolSize** puts a cap on the total number of open JDBC connections that can be established to each database.

The default value is 5.

pool.minimumIdle

The minimum number of idle JDBC connections in each connection pool. The HCL OneDB™ Explore server will maintain a connection pool for each OneDB database server that it needs to connect to. Setting **pool.minimumIdle** to zero indicates that all JDBC connections in the connection pool should be closed when they have been sitting idle for longer than the **pool.idleTimeout** threshold. Setting **pool.minimumIdle** to a positive integer indicates the number of connections that should be kept open in the connection pool even when they exceed the **pool.idleTimeout**. The default and recommended value is 0.

redirectHTTPtoHTTPS

If set to true, HTTP traffic to HCL OneDB™ Explore will automatically be redirected to HTTPS. This will include web socket communication between the HCL OneDB™ Explore server and agent. If this value is set to true, you will be required to configure SSL in your agent configuration properties.

The default value is false.

rest.session.timeout

Specifies the number of milliseconds that a REST session can be idle before it is closed. The default value is 3600000 (60 minutes).

ssl.keystore.file

The path to the keystore file that contains the certificate to use for network encryption. This property must be set if **httpsPort** is set to something other than -1.

ssl.keystore.password

The password to unlock the keystore file for network encryption. If this property is not set and the HTTPS is configured, you will be prompted on the command line to enter the keystore password when starting the OneDBHCL OneDB™ Explore server.

ssl.key.password

The password to unlock the entry into the keystore. The default value is no password, which means to use the keystore password. If the entry into the keystore requires a password that is different from the keystore password, set this property to the entry password.

user.password.maxAge

Controls the maximum age (in days) of a user password. User passwords that are older than the max age will be considered as expired. Setting this property to zero, which is the default value, specifies that user passwords never expire. Setting this property to a value greater than zero specifies the maximum age (in days) of a user password before it expires. A user will start receiving notifications in the HCL OneDB™ Explore UI when the difference between the current date and the password expiration date is less than or equal to 15 days.

user.password.minLength

Controls the minimum length for a user password. The default value is 8.

user.password.requireLowerCase

Controls whether user passwords are required to include at least one lowercase character. The default value is true.

user.password.requireNumber

Controls whether user passwords are required to include at least one number. The default value is true.

user.password.requireSpecialCharacterFromSet

Controls whether user passwords are required to include at least one special character. An empty string indicates that no special characters are required. Setting this value to "!@#%&*()?" would require user passwords to include at least one of those characters. The default value is an empty string.

user.password.requireUpperCase

Controls whether user passwords are required to include at least one uppercase character. The default value is true.

HCL OneDB™ Explore UI

This topic highlights important aspects of the HCL OneDB™ Explore user interface. The HCL OneDB™ Explore UI is run by the Jetty web server that is embedded in the HCL OneDB™ Explore server. The default URL for the HCL OneDB™ Explore server is `http://localhost:8080/` but the [HCL OneDB™ Explore server configuration on page 10](#) file can be used to change the port and configure whether https is enabled.

Adding Servers and Groups

After logging in to OneDB Explore for the very first time, you will be taken to a group dashboard for the root group. This dashboard will initially contain zero servers and zero groups.


Adding Servers

To add servers, click the **Add Server** button to add connection information for your OneDB database server instances.

When adding a server to OneDB Explore, you can provide two sets of credentials:

- Monitoring credentials
- Admin credentials

The **monitoring credentials** are used by the OneDB Explore server whenever a user navigates to any part of the UI that issues a REST query which needs to gather data from the live OneDB database server instance. The **monitoring credentials** are also used by the agent whenever it gathers data from your database server. The **admin credentials** are used whenever a user in the UI requests that an administration action be performed on the database server, for example create a dbspace, edit an onconfig parameter, or deploy an agent.

Required privileges for the monitoring user	<ul style="list-style-type: none"> • CONNECT access to the sysmaster database • CONNECT access to the sysadmin database
Optional privileges for the monitoring user	<ul style="list-style-type: none"> • select privileges on the sysconblock, syssqltrace, syssqltrace_info, syssqltrace_hvar, and syssqltrace_iter tables in the sysmaster database. • select privileges on the ph_task, ph_run, ph_alert, ph_threshold, ph_bg_jobs, and ph_bg_jobs_results tables in the sysadmin database. <p> Note: Providing these optional privileges to the monitoring user enables the UI to show information about the server's the storage pool, sql tracing, auto update statistics, and scheduler tasks.</p>
Required privileges for the admin user	<ul style="list-style-type: none"> • Part of the DBSA group. • CONNECT access to the sysadmin database. • Privilege to run SQL Admin API commands. • select/insert/update/delete privileges on ph_task, ph_run, ph_alert, ph_threshold, ph_bg_jobs, and ph_bg_jobs_results tables in the sysadmin database. • Execute privilege on the following functions: <ul style="list-style-type: none"> ◦ exectask(lvarchar) ◦ exectask(lvarchar,lvarchar)

	<ul style="list-style-type: none"> ◦ exectask(integer) ◦ exectask(integer,lvarchar) <p>If this server is used as a repository server storing monitoring data for one or more instances, the admin user must also have:</p> <ul style="list-style-type: none"> • RESOURCE access to the repository database.
--	--

It is required that you provide monitoring credentials when adding your OneDB database server information to OneDB Explore. Admin credentials need only be provided if you want to use OneDB Explore to run administrative actions on your database server or if you want the OneDB Explore server to automatically deploy your agents.

SSL Connections

If your database supports or requires SSL connections you can setup SSL using the connection properties on the **Add Server** page. Specify the following connection parameters:

SSLCONNECTION	true
SSL_TRUSTSTORE	Absolute or relative path to the truststore/keystore file from the perspective of the directory from which the agent and the OneDB Explore server start. The truststore/keystore file must be present where both OneDB Explore server and OneDB Explore agent are running
SSL_TRUSTSTORE_PASSWORD	Password to unlock the truststore/keystore file for both OneDB Explore server and OneDB Explore agent

Advanced Connection Properties

You can specify any of the advanced JDBC connection parameters when you setup your connection to alter the behavior of the underlying connections OneDB Explore makes to the OneDB database servers. For more information, see OneDB environment variables with the OneDB JDBC Driver

Adding Groups

From the dashboard, you can also click the **Add Group** button to create groups of servers to make it easier to monitor and manage multiple servers.

Exploring Groups

On a group dashboard, you will see all of the servers and sub-groups that belong to the group. For each sub-group, the UI will show a card indicating the number of servers in the group (the first number is the number of servers directly in the group;

the number in parenthesis is the total number of servers under it in the hierarchy). It will also show if and how many unread incidents exist for servers within the group.

There will also be a card for each server in the current group indicating the server and agent status. Since the agent monitors server status information, the server status will be unknown if the agent is offline.

From the group dashboard, you can use the **Add Server** and **Add Group** buttons to add servers or groups to the hierarchy. Each server or group also has additional actions to rename, edit connection information (for servers only), move, or delete that object. To drill down on any server or group, click on the server or group card.

From the group dashboard, you can also access the monitoring and alerting profiles for the group by selecting those menu items from the left-hand sidebar. If you are a System Administrator, you will also see a link for granting/revoking permissions on the group.

Exploring OneDB Database Servers

Clicking on any server card from a group dashboard will take you to the server's dashboard. The dashboard includes sections on server status and any alerting incidents that have occurred, as well as information on high availability, threads, storage performance metrics, host memory and CPU usage, etc. If the agent is not running or sensors are not enabled, you will see the latest measurement for these metrics queried directly from the live database server. If the corresponding sensors are running, the server dashboard will show the recent history of each of these metrics graphically to give you a visual indication of how the database server is performing.



Note: If you see a triangular exclamation icon in the top right of any dashboard, that is an indication that the dashboard uses sensors that do not exist in the server's monitoring profile. Clicking on that icon will open a pop-up detailing the sensors used by the dashboard and providing a one-click button to enable all of those sensors.

For a particular database server, use the left-hand sidebar menu to continue to explore the server.

Configuring Monitoring

After clicking on a server or group from the dashboard, you can use the Monitoring link on the left-hand sidebar menu to configure the monitoring profile for that server or group. The Monitoring page lists the sensors currently configured in the server or group's monitoring profile.

Clicking the **Add Sensors** button will open a pop-up displaying the list of sensors not yet configured for this server or group. You can add custom sensors to this list from the [Sensor Management on page 36](#) page.

Click the **Edit** (pencil) icon to modify the run interval or data retention interval.

For servers or groups that inherit sensors from parent groups, there will also be a list of Inherited Sensors. The buttons next to the inherited sensors allow you to disable or override properties of the inherited sensors.

Configuring Alerting

Alerts in OneDB Explore can be fully customized. You can configure not only what you want to be alerted on, but also the threshold or condition that should trigger that alert.

Alerts defined for a group are automatically inherited by the child groups and child servers of that group. Defining alerts at the group level simplifies the process of managing alerts for multiple servers.

To create alerts:

1. Click on a server or a group from the dashboard and then select the **Alerting** link on the left-hand sidebar menu. The **Alerting** page lists the alerts configured in the server or group's alerting profile.
2. Click the **Add Alert** to open a form that guides you through the process of defining an alert. Provide an alert name to identify it.
3. Select what you want to alert on: OneDB server status, agent status, or data from a sensor metric.
4. Define the alerting condition. For example, alert me when the OneDB server status is offline or alert me when the number of sessions connected to OneDB is greater than 200.

For servers or groups that inherit alerts from parent groups, there will also be a list of Inherited Alerts. The buttons next to the inherited alerts allow you to disable inherited alerts. While you cannot override parent alerts like you can do for sensors (due to the complexities of alerts), there is a clone button provided to make it easy to clone and modify an inherited alert if changes are required.

Once alerts are defined, the OneDB Explore server will evaluate the applicable alerting condition for each new data point collected by the OneDB Explore agent in the process of monitoring your database server. If an alerting condition is met, an **alerting incident** is created. Alerting incidents are automatically displayed in the OneDB Explore UI, both in the dashboard and the **Incidents** pages for the server and for the parent group(s) it belongs to.

You can enable alert notifications to be sent through email, PagerDuty, Twilio, or a custom alerting script. To enable alerting notifications, a system administrative user for OneDB Explore must enable the desired alerting notification service on the [Configuring Alerting Notification on page 35](#) page. For certain alerting notification services, including email and Twilio, each individual user who wants to receive alerting notifications must enable it for their user on the [Configuring Alerting Notification on page 35](#) page where they will provide user specific settings like their email address or phone number.

Custom Dashboards

Creating a custom dashboard

You can create custom dashboard of sensor metric or custom query for one or more OneDB database servers.

Dashboards can be created for a single server or for multiple servers. Multi-server dashboards can have up to 5 different servers. All dashboards will be saved in the current group. This allows you to open and view the dashboard for different database servers within that group, although you have the option of saving a default server or set of servers for any particular dashboard.

To create a custom dashboard:

1. Click on **Dashboards** from the sidebar menu for a server or a group.
2. Click the **New Dashboard** button.
3. Select a server or set of servers which will allow you to preview the dashboard as you build it.
4. Add one or more dashboard panels.

Each dashboard panel can be defined to graph one or more sensor metrics from the repository. You can also define a graph based on custom query.

You can graph multiple metrics from multiple different sensors on the same graph. For multi-server dashboards, you can choose to associate each panel to a single server or you can graph metrics for each of your servers in the same graph.

You can customize the graph types (bar, pie, line and table), attributes of the left and right y-axes, colors and labels of graphed metrics.

5. Arrange your dashboard panels. Panels can be re-sized and moved to different positions to create a custom layout for your dashboard.

As you resize your dashboard panels, it is automatically saved.

Viewing a custom dashboard

To view a custom dashboard:

1. Go to the **Dashboards** page of the group or server within the group.
2. Select the desired dashboard from the list of available dashboards.
3. Optionally, select or change the servers to show on the dashboard.

If the selected dashboard has a set of default servers defined, it will open with the defined servers. Once opened however, you can change the servers shown in the dashboard by clicking on the server drop-down at the top of the dashboard.

If the dashboard has no default servers defined, your context within OneDB Explore will determine which servers get loaded into the dashboard when it opens. If you open a dashboard from the context of a server, then that dashboard will be automatically loaded for the current server. If you open a dashboard from the context of a group, then you will be prompted to select one or more servers from that group to show on a dashboard.

Importing a custom dashboard

To import a custom dashboard:

1. Go to the **Dashboards** page of the group or server within the group.
2. Click the **Import** button and select the JSON file that has been exported from Dashboards page in OneDB Explore. New dashboard will be created with the same configuration that has been provided while exporting dashboards.

Exporting a custom dashboard

To export a custom dashboard:

1. Go to the **Dashboards** page of the group or server within the group.
2. Select the desired dashboard from the list of available dashboards.
3. Click the **Export** button. JSON file will be downloaded to system's Downloads folder.

Custom Manager

. The **Connection Manager** page allows you to visualize and manage CM unit, SLA and FOC for any CM.

Use the **Connection Manager** to:

- View all connection units
- View/add/modify/delete SLA within connection units
- View/add/modify FOC within connection units
- View/get alerts for number of SLA connections in any CM

Viewing Database Information

You can view detailed information about database using different tabs like Stored Procedures, Sequences, User Defined Types, Data Blades etc.

To view database information:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select any table from the list.
3. Click on each tab to view the information like Stored Procedures, Sequences, User Defined Types, Data Blades.

Viewing Table Information

You can view detailed information about tables using different tabs like Indexes, References, Constraints, Triggers etc

To view information about table:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select any table from the list.
3. Click on each tab to view the information like Indexes, References, Constraints, Triggers.

Creating a Database

To create a database:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select any database from the list.

3. Click ... and – Select **Create Database**
4. Enter the required details.
5. Click the **Finish** button. The new database will be created.

Creating a Demo Database

To create a demo database:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select any database from the list.
3. Click ... and – Select **Create Demo Database**
4. Enter the required details.
5. Click the **Create** button. The demo database will be created.

Dropping a Database

To drop a database:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select the database you want to drop.
3. Click ... and – Select **Drop Database**
4. Click **Yes** to confirm. The database will be dropped.

Creating an Index

To create an index:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select the table.
3. Click ... and – Select **Create Index**
4. Enter the required details.
5. Click the **View SQL** button to review the SQL statement.
6. Click the **Create** button, index will be created.



Note: You can also **enable** or **disable** the index.

Deleting an Index

To delete an index:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select the table for which you want to delete the index.
3. Click the **Indexes** tab.
4. Select the index you want to delete.

5. Click the **Delete** icon.
6. Click **Yes** to confirm. Index will be deleted.

Schema Manager

The **Schema Manager** page allows you to browse and view detailed information about the various tables and indexes in each of your databases.

Use the **Schema Manager** to:

- View detailed information about Databases
- View detailed information about Tables
- Create a Database
- Create a Demo Database
- Drop a Database
- Create a Table
- Drop a Table
- Create an Index
- Delete an Index

Viewing Database Information

You can view detailed information about database using different tabs like Stored Procedures, Sequences, User Defined Types, Data Blades etc.

To view database information:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select any table from the list.
3. Click on each tab to view the information like Stored Procedures, Sequences, User Defined Types, Data Blades.

Viewing Table Information

You can view detailed information about tables using different tabs like Indexes, References, Constraints, Triggers etc

To view information about table:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select any table from the list.
3. Click on each tab to view the information like Indexes, References, Constraints, Triggers.

Creating a Database

To create a database:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select any database from the list.
3. Click ... and – Select **Create Database**
4. Enter the required details.
5. Click the **Finish** button. The new database will be created.

Creating a Demo Database

To create a demo database:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select any database from the list.
3. Click ... and – Select **Create Demo Database**
4. Enter the required details.
5. Click the **Create** button. The demo database will be created.

Dropping a Database

To drop a database:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select the database you want to drop.
3. Click ... and – Select **Drop Database**
4. Click **Yes** to confirm. The database will be dropped.

Creating a Table

About this task

This topic explains the steps to create a table using OneDB Explore UI.

There are five types of tables:

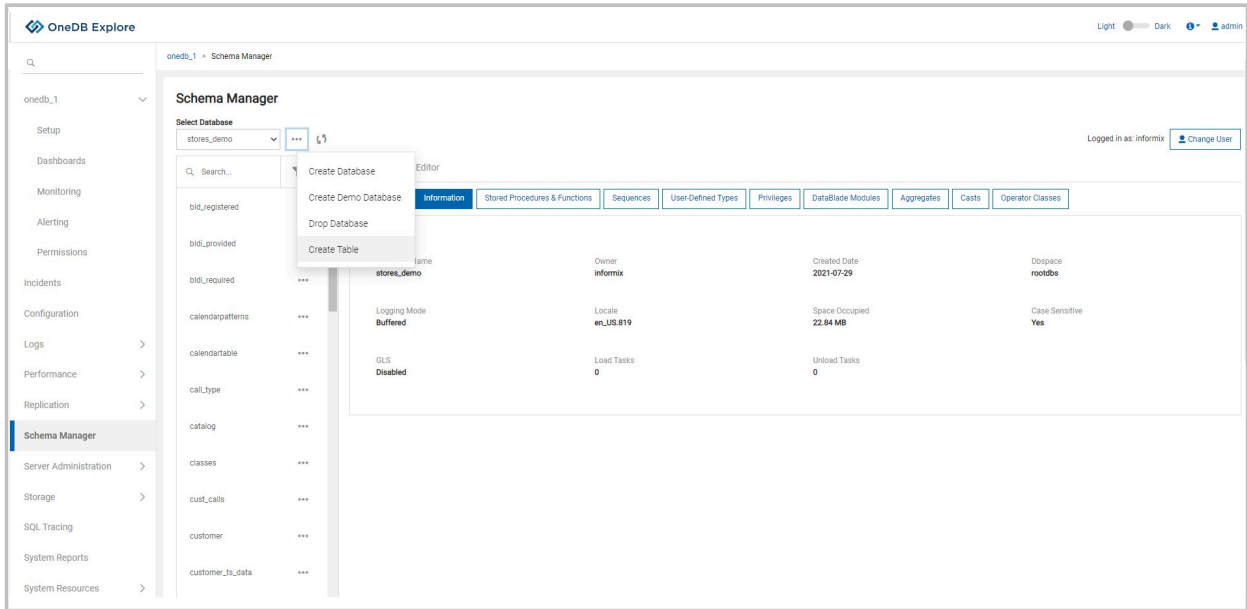
1. Standard table
2. Raw table
3. External Fixed table
4. External Delimited table
5. External Informix table

Standard & Raw table types are almost similar & three external table types are almost similar. This topic explains how to create both table types.

To create a table, follow the steps given below:

1. Go to the **Schema Manager** in OneDB Explore.
2. Select desired database and click on menu option (3 dots) next to **Select Database** dropdown.

3. Click on **Create Table** option from drop-down to create a table

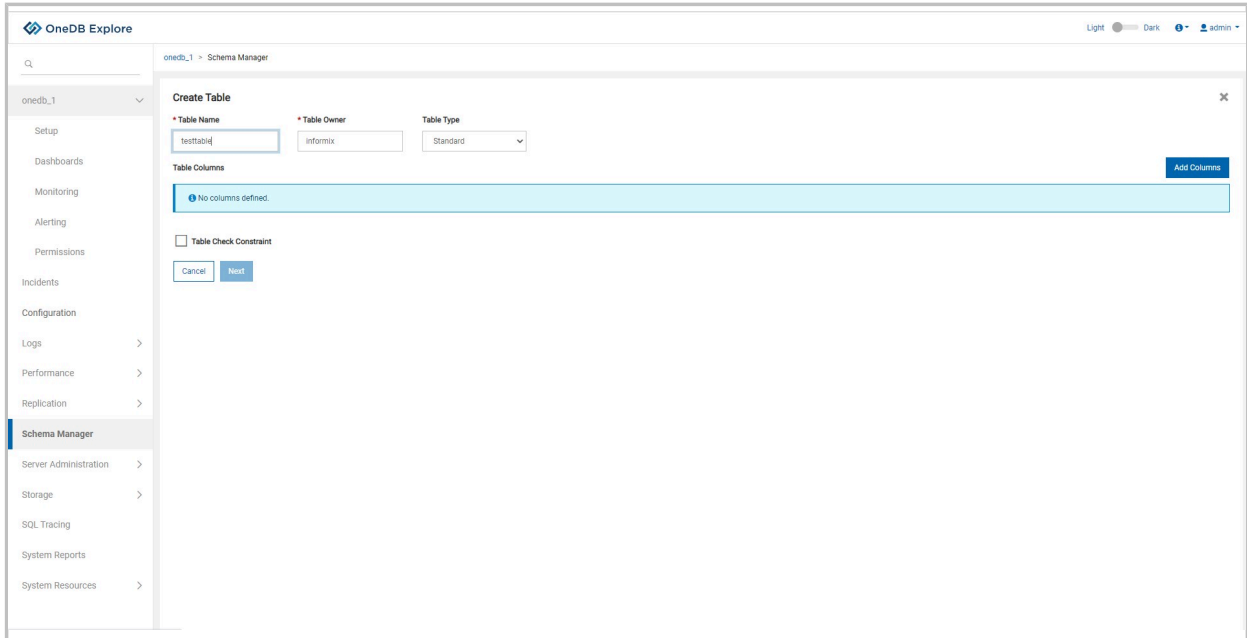


Creating a Standard or Raw type table

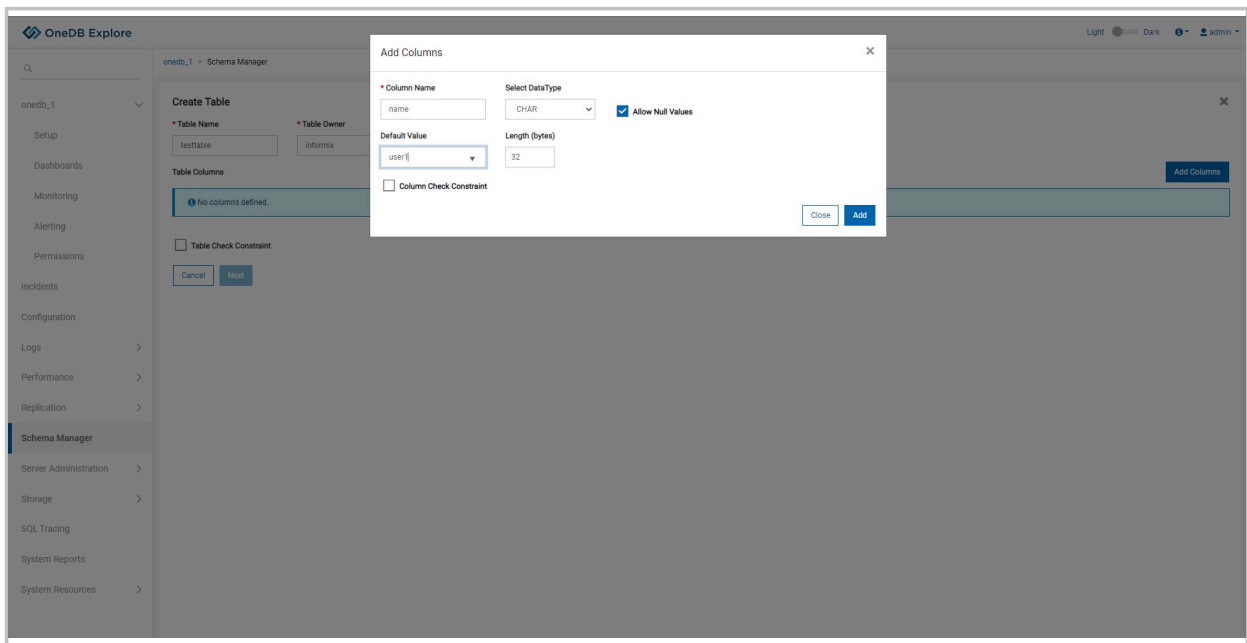
About this task

This topic explains the steps to create a standard or raw table.

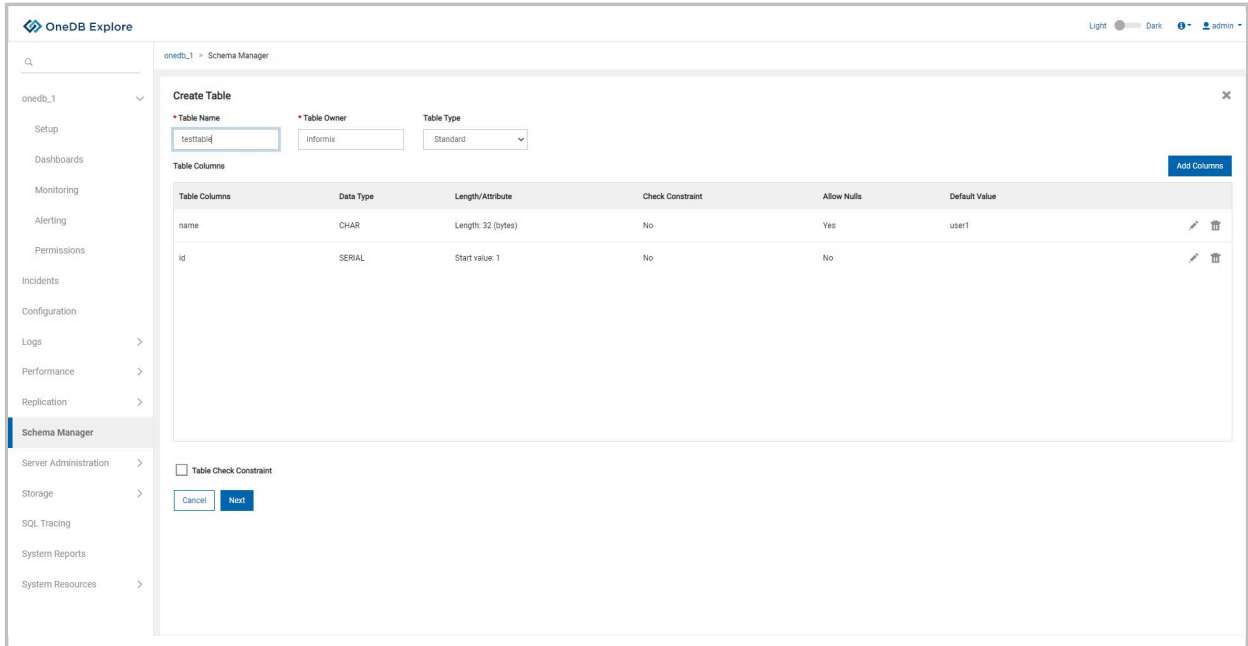
1. Enter mandatory fields such as **Table Name** and **Table Owner**.
2. Select Standard(default) or Raw type from **Table Type** dropdown.
3. Click on **Add Column** for adding columns to the table.
4. To cancel **Create Table** operation click on **Cancel** button.
5. **Next** button will be enabled after user is done adding columns to the table.



6. By clicking on **Add Columns** button pop up will appear to add column details.
7. Enter values for all mandatory fields(*).
8. To give a constraint on any column, click on **Column Check Constraint** checkbox.
9. Multiple columns can be added using the same pop up.
10. To go back to the main screen, click on **Close** button or **cross** icon in the upper right corner.

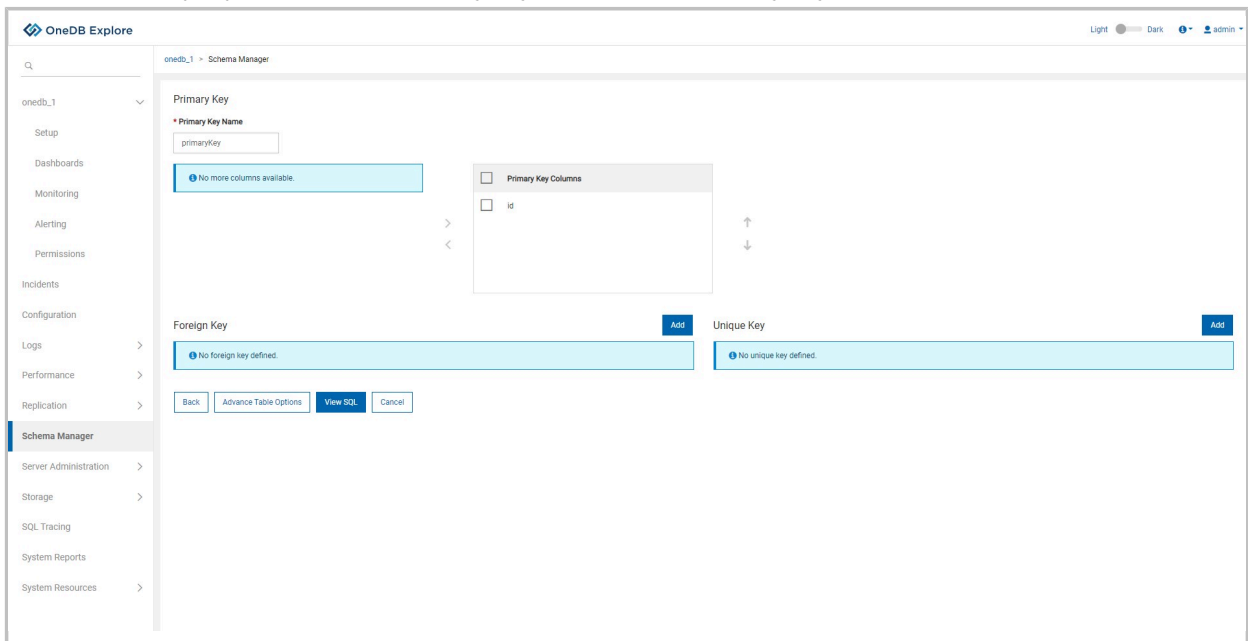


11. Once columns are added, user can view, edit, delete any of the columns.
12. Table level constraint can be added on this screen by clicking on **Table Check Constraint** checkbox.
13. Once details related to the columns are finalized, clicking on **Next** button takes user to add constraint page.



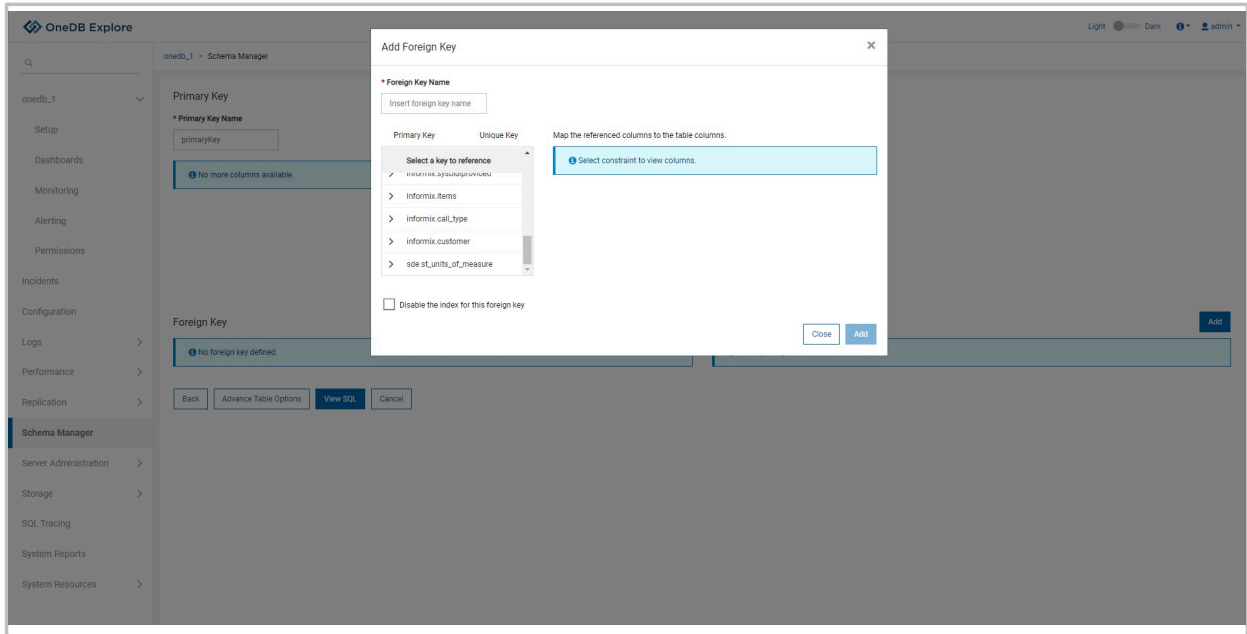
14. This screen is for adding a constraint like primary key, foreign key, unique key to a table.

15. To assign primary key to a table, give **Primary Key Name** & select **Primary Key Columns**.



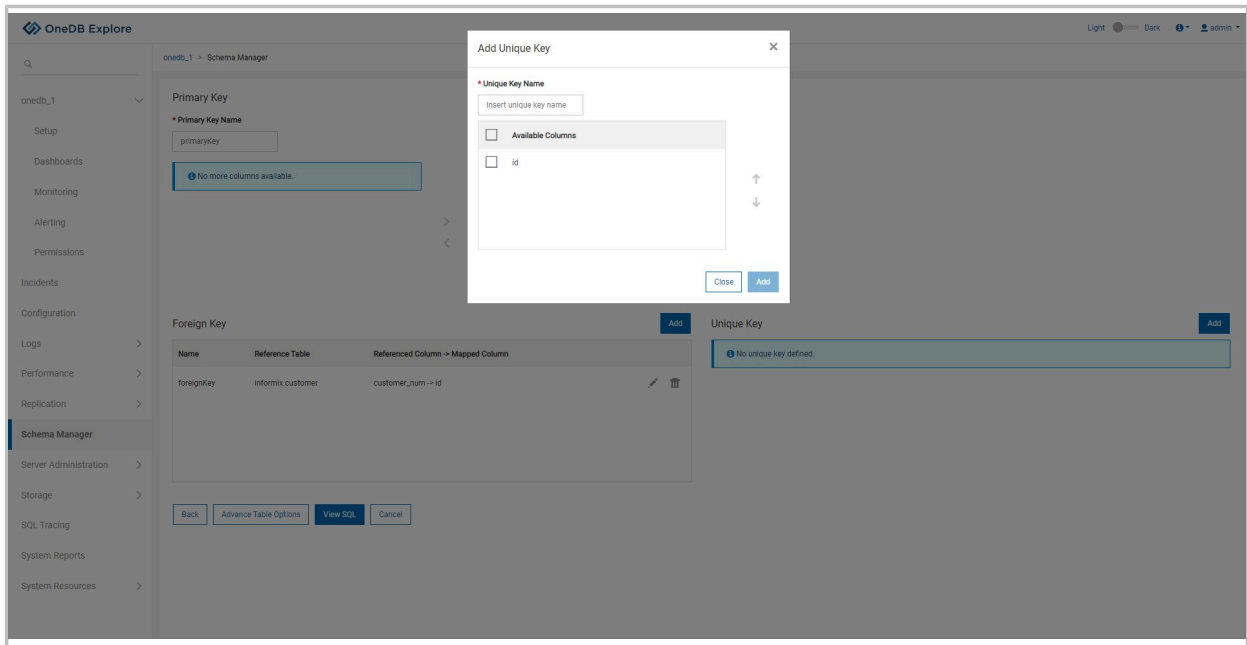
16. To add Foreign key constraint, click on **Add** button in Foreign key section.

17. To assign a foreign key for a table, give **Foreign Key Name** & map **Referenced Column** with **Table Column**.



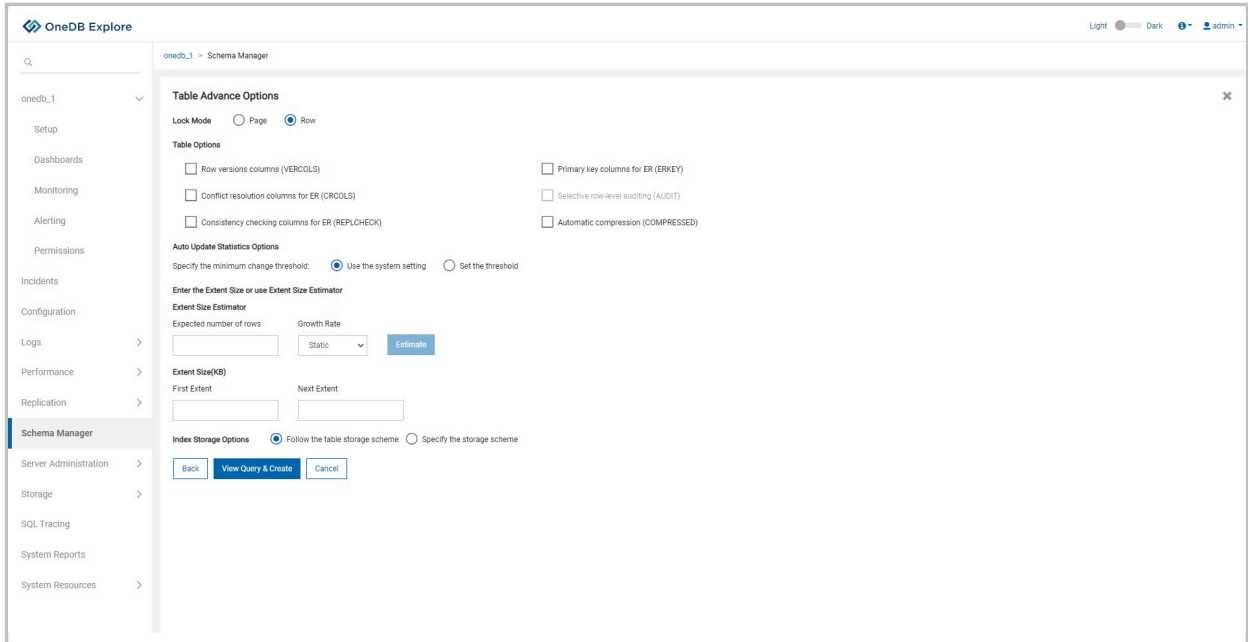
18. To add a Unique key constraint, click on **Add** button in **Unique key** section.

19. To assign a unique key for a table, give **Unique Key Name** & select **Available Columns** from the list. This will enable the **Add** button.

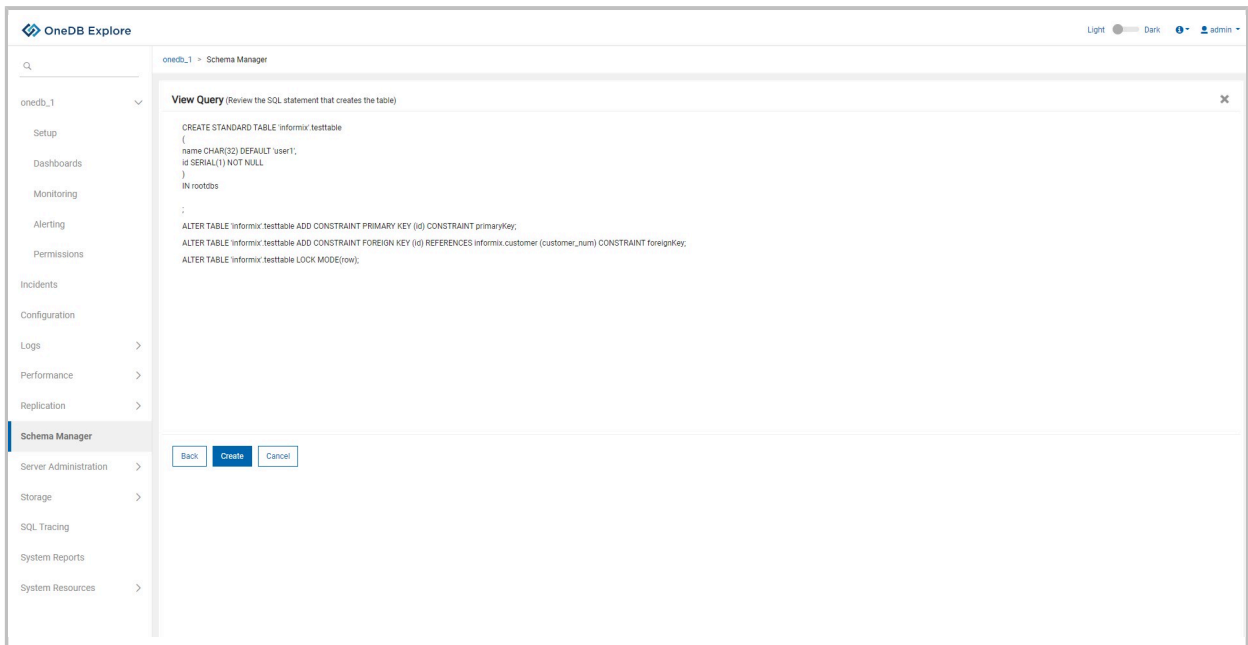


20. Once constraints are added user can go to either **View SQL** or **Advance Table Options**

21. Modify advance table level options using the screen given below. For example, changing lock mode, storage scheme, update statistics, etc.



22. Once advance table option are set, click on **View Query & Create** button to view SQL query for creating the table.
23. After clicking on **View Query & Create** button from Advance Table Options or on **View SQL** button from **Add Constraints** screen, user will be able to view 'create table' query as shown in the screen given below.
24. Click **Create** button to create the table.
25. If table is created successfully, information status message will be shown and user will be taken back to **Schema Manager** page.
26. If table creation fails, error status message is displayed and all the create table related queries will be rolled back.
27. To go back to modify any properties click **Back** button and to cancel the operation of create table click on **Cancel** button.

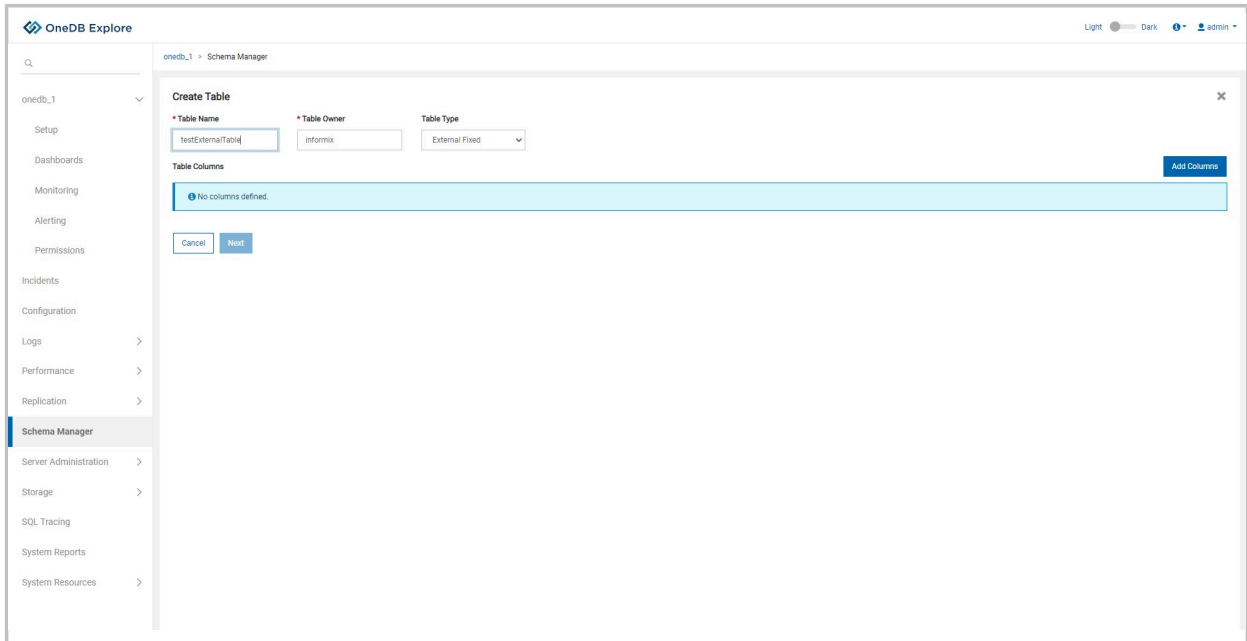


Creating an external type table

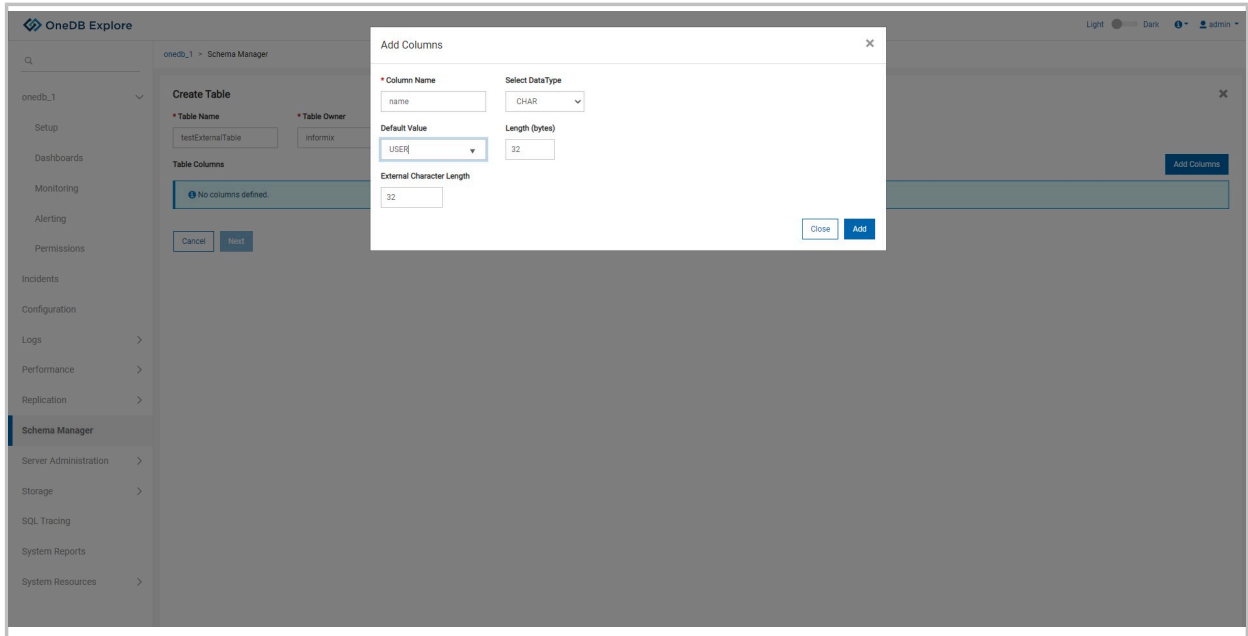
About this task

This topic explains the steps to create an external type table.

1. Enter mandatory fields such as **Table Name** and **Table Owner**.
2. Select one of **External Fixed**, **External delimited**, **External Informix** table type from **Table Type** dropdown.
3. Click on **Add Columns** for adding columns to the table.
4. To cancel **Create Table** operation, click on **Cancel** button.
5. **Next** button will be enabled once user adds columns to the table.

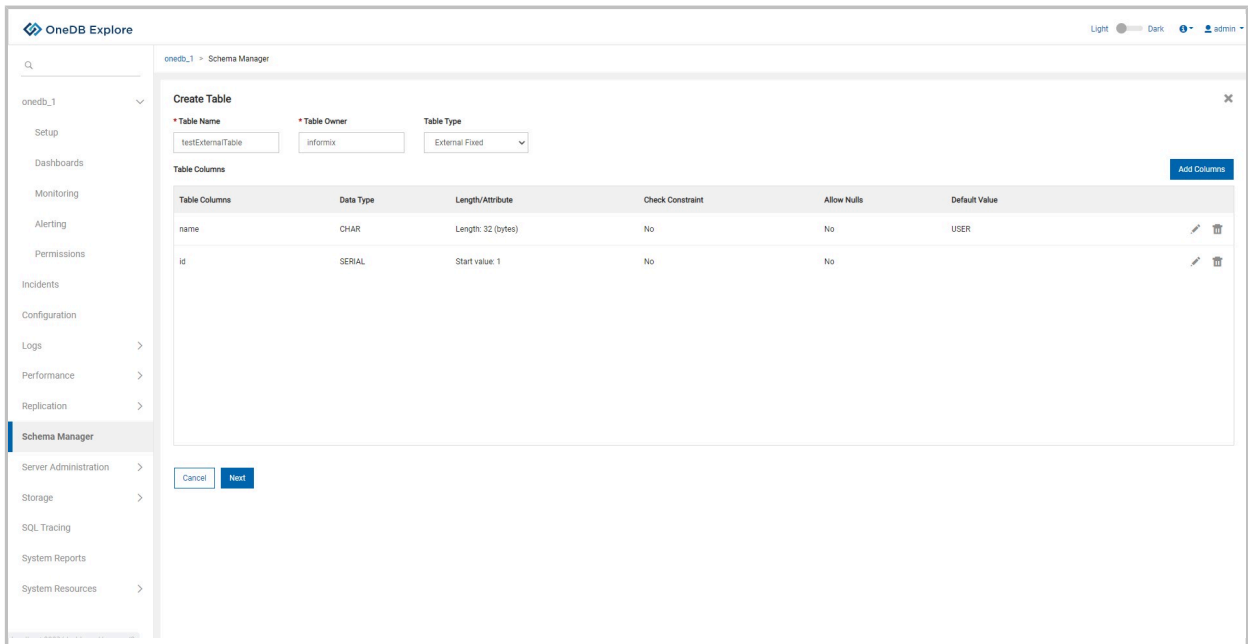


6. By clicking on **Add Column** button, a pop up will appear to add column details.
7. Enter all the mandatory field values(*).
8. User can add multiple columns using the same pop up.
9. To go back to the main screen, click on **Close** button or **Cross** icon.

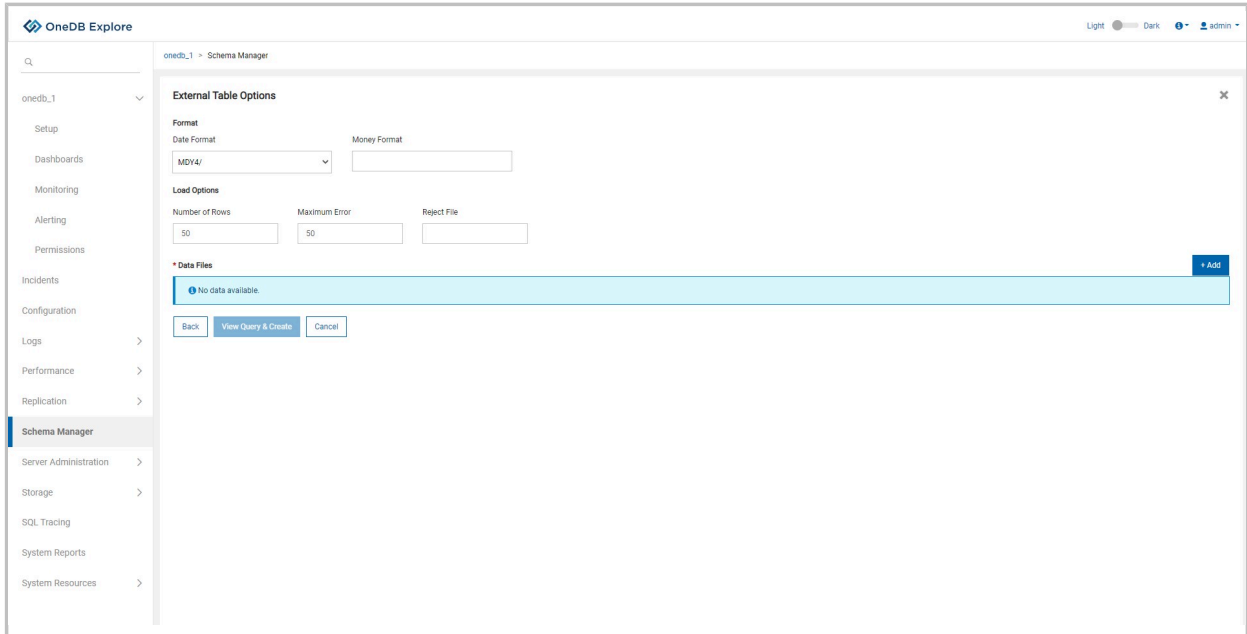


10. Once columns are added, user can view, edit , delete any of the columns.

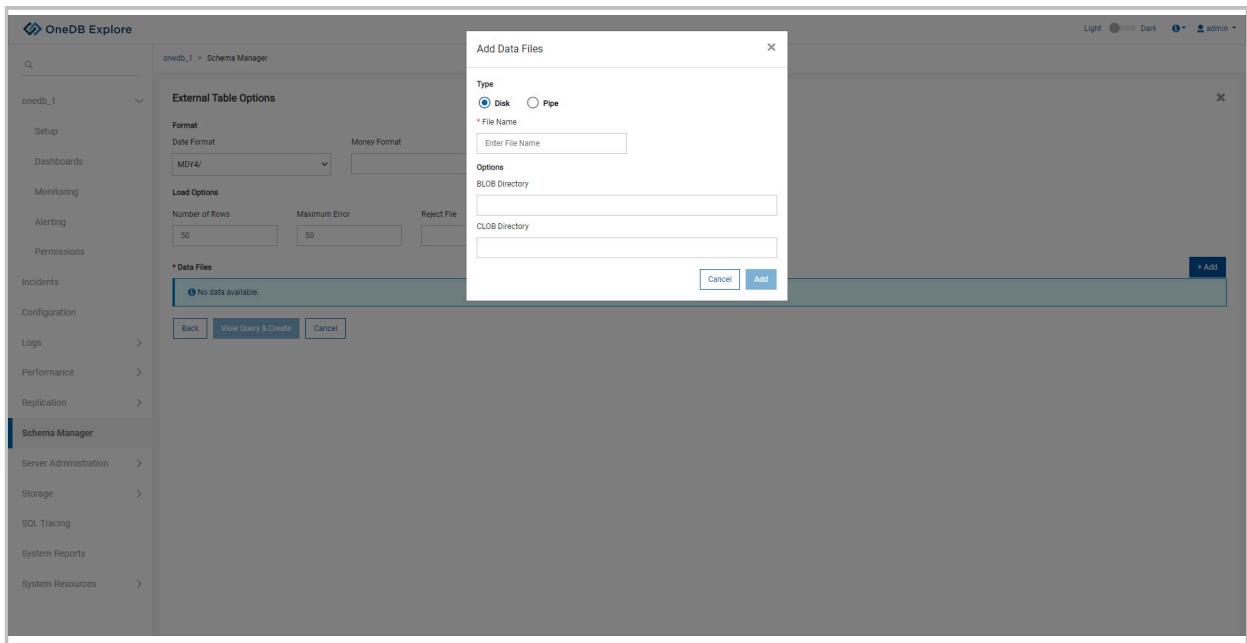
11. Once column details are finalized click on **Next** button to go to **External Table Options**.



12. Provide information for external table options & add mandatory data file by clicking on **+Add** button.



13. Following pop up is used to add data files to an external table.



14. Once data files are added, user can view, edit , delete any of the data files.

15. Once External table options are finalized click on **View Query & Create** button to view SQL query for creating the table.

16. User can either go **Back** or **Cancel** the operation using respective buttons.

External Table Options ✕

Format

Date Format Money Format

MDY4/ ▼

Load Options

Number of Rows Maximum Error Reject File

50

50

*** Data Files** + Add

Type	File Name	
disk	PATH:diskname;BLOBDIR:/temp	✎ ✖

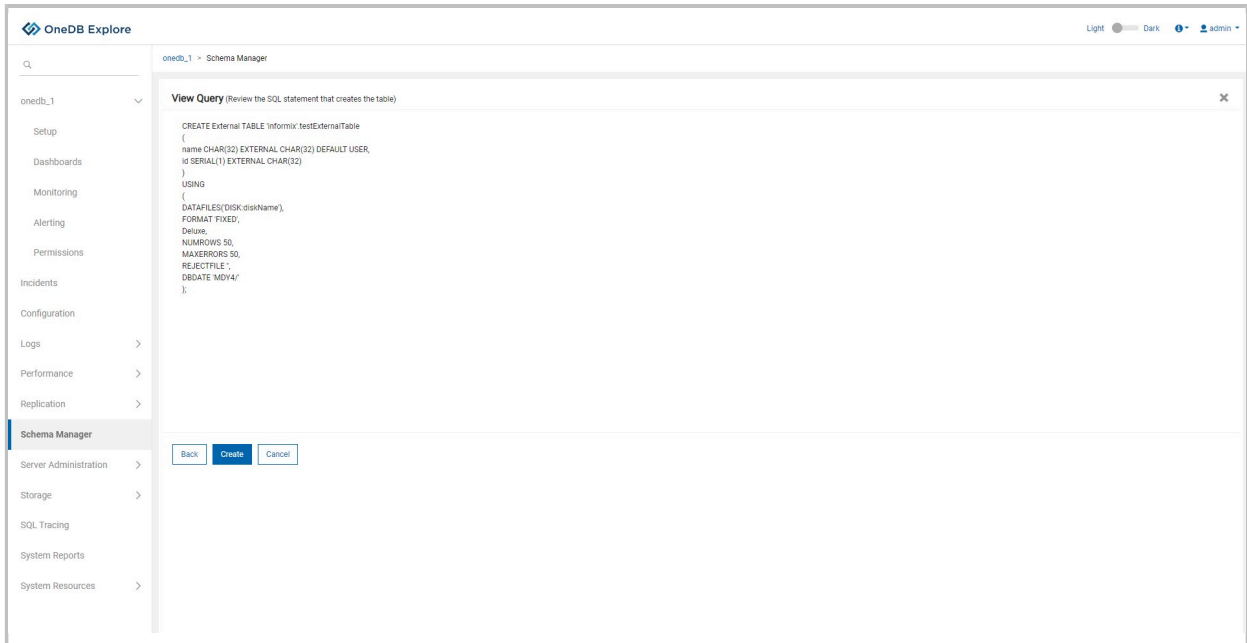
Back

View Query & Create

Cancel

17. After clicking on **View Query & Create** button from the external table option, user will be able to view create table query as shown in the screen below.
18. Click **Create** button to create the table.
19. If table is created successfully information status message will be shown and user will be taken back to **Schema Manager** page.
20. If table creation fails, error status message will be displayed and all the create table queries will be rolled back.

21. To go back to modify any properties, click **Back** button and to cancel the operation of create table click on **Cancel** button.



Dropping a Table

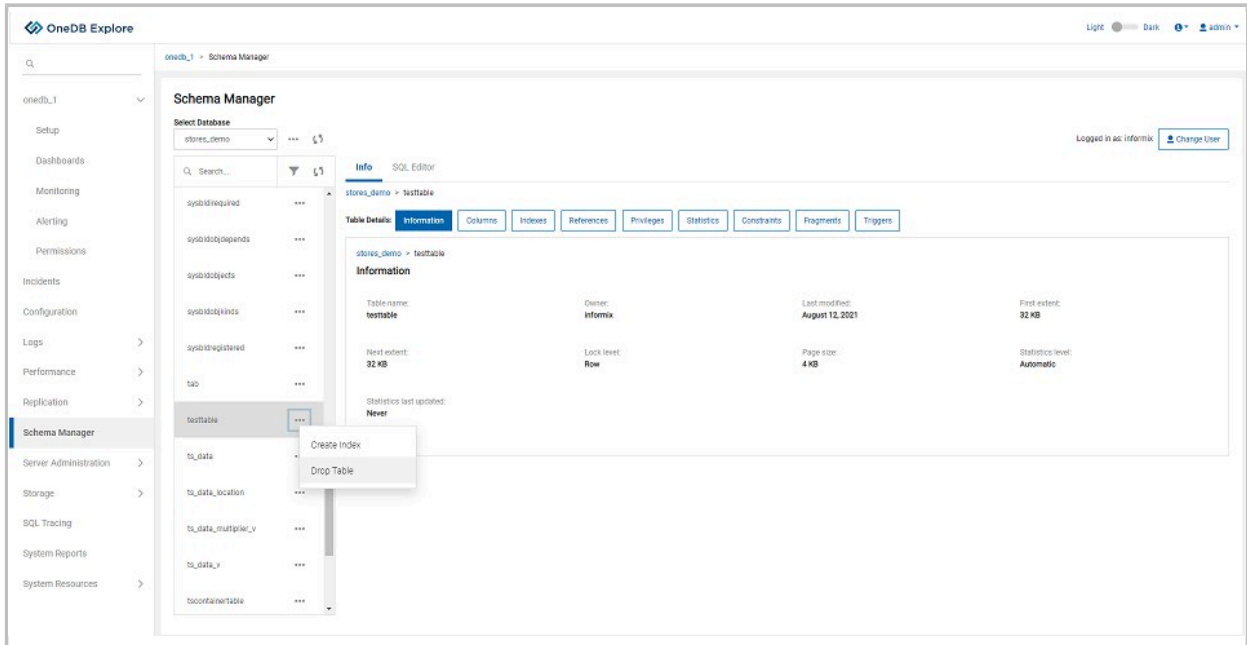
About this task

This topic explains the steps to drop a table using OneDB Explore UI.

To drop a table, follow the steps given below:

1. Click on **Schema Manager** in OneDB Explore.
2. Select the desired database from **Select Database** dropdown which contains the table to be dropped.
3. From the table list shown, locate the table to be dropped.
4. Click on menu option (3 dots) next to this table and select **Drop Table** from the dropdown menu to drop a table.

5. Confirm action on pop over (confirmation pop up for dropping a table).



Creating an Index

To create an index:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select the table.
3. Click ... and – Select **Create Index**
4. Enter the required details.
5. Click the **View SQL** button to review the SQL statement.
6. Click the **Create** button, index will be created.



Note: You can also **enable** or **disable** the index.

Deleting an Index

To delete an index:

1. Go to the **Schema Manager** page in OneDB Explore.
2. Select the table for which you want to delete the index.
3. Click the **Indexes** tab.
4. Select the index you want to delete.
5. Click the **Delete** icon.
6. Click **Yes** to confirm. Index will be deleted.

OneDB Explore Server Settings

Users with System Administrator privileges, including the initial admin user created when OneDB Explore is started for the very first time, will have a **System Settings** link in the drop-down menu shown when they click on the user icon in the top right corner of the application title bar. This link is used for making changes to the global OneDB Explore Configuration.

The OneDB Explore Configuration page includes settings for:

- [Alerting Configuration on page 35](#) – enable and configure OneDB Explore to be able to send alerts to users via email, Twilio, or PagerDuty
- Data Cleanup Configuration – configure the schedule and settings for when OneDB Explore runs its repository data cleanup jobs
- [Sensor Management on page 36](#) – create and manage custom SQL sensors for monitoring your OneDB database servers
- Server Configuration – configure system-wide settings for the OneDB Explore server, including the REST SQL session timeout.
- User Management – add users and edit their OneDB Explore privileges

Configuring Alerting Notification

System administrative users for OneDB Explore must enable which alerting notification services the OneDB Explore server should use when an [alerting incident on page 18](#) occurs. If desired, you can configure and enable multiple alerting notification services.

OneDB Explore supports the following alerting notification services, all of which can be configured and enabled on the **System Settings > Alerting Configuration** page.:

- **Email**

OneDB Explore can be configured to send emails through an external SMTP server when alerting incidents occur. To enable email notifications, the system administrative user must provide a SMTP server and port to use. Optionally, you can provide a user and password for authenticating to that SMTP server and a **from email address** that OneDB Explore should use when sending alerting notification emails.

Email notifications must first be enabled at the system level by the system administrative user. Then each individual OneDB Explore user who wants email notifications must enable it for their email address on their [My Settings->Alerting Configuration on page 38](#) page.

- **Twilio**

OneDB Explore can be configured to send alerting incidents through Twilio. To enable Twilio notifications, the system administrative user must provide the Twilio account SID, authorization token, and phone number to send alerts from.

Twilio notifications must be enabled at the system level by the system administrative user. Then each individual OneDB Explore user who wants Twilio notifications must enable it for their phone number on their [My Settings->Alerting Configuration on page 38](#) page.

• Pager Duty

OneDB Explore can be configured to send alerting incidents through Pager Duty. Pager Duty alerts are enabled globally by the system administrative user of OneDB Explore. To enable Pager Duty notifications, the system administrative user must provide a PagerDuty service key.

Pager Duty alerts do not need to be enabled by each individual user of OneDB Explore. When Pager Duty alerting notifications are enabled by the system administrative, all alerting incidents that occur in OneDB Explore will be sent to the specified Pager Duty service key.

OneDB Explore sends PagerDuty notifications through REST using Pager Duty's Events API v1.

• Run Script

The OneDB Explore server can be configured to run a local script whenever an alerting incident occurs. Script notifications in OneDB Explore provide an extensible way to integrate OneDB Explore's alerting with any alerting mechanism used by your organization.

Script notification is enabled globally by the system administrator on the **System Settings > Alerting Configuration** page. When script notification is enabled, the OneDB Explore server will run the specified script whenever an alerting incident occurs on any server or group managed by OneDB Explore.

Before the OneDB Explore server runs your script, it will set the following environment variables to contain information about the alerting incident that occurred:

- ALERT_ID – id of the alerting incident
- ALERT_TIMESTAMP – timestamp of the alerting incident
- ALERT_SUMMARY – summary text describing the alerting incident
- ALERT_MESSAGE – detailed message describing the alerting incident
- SERVER_ID – id of the OneDB server on which the alerting incident occurred
- SERVER_ALIAS – alias of the OneDB server on which the alerting incident occurred
- GROUP_ID – id of the parent group containing the OneDB server on which the alerting incident occurred
- GROUP_NAME – name of the parent group containing the OneDB server on which the alerting incident occurred
- EVENT_URL – an url link to view the alerting incident in OneDB Explore

A sample use case for the "Run Script?" alerting service would be, suppose your organization uses a third party alerting service that OneDB Explore does not have native support for. That service requires you to POST a JSON document to a specific URL to generate an alert. You can write a script that reads in the environment variables set by OneDB Explore, reformat that data into a JSON document as required, and then use curl to send a REST POST request to your organization's alerting service.

Creating Custom Sensors

Use the **System Settings > Sensor Management** page to create and manage custom SQL sensors. Custom sensors allow you to define data to be collected by the OneDB Explore agent. Data from custom sensors can be graphed on a [custom dashboard on page 18](#). You can also use custom sensors to define alerting conditions in OneDB Explore.

Each custom SQL sensor is based on a single SQL query to be run by the agent on the sysmaster database of the OneDB server being monitored. Any data that can be returned by a SQL query against sysmaster can be monitored by the agent.

Only System Administrative users of OneDB Explore can define custom SQL sensors. Once defined, custom sensors can be added to any server or group's monitoring profile.

To define a custom SQL sensor, go to the **System Settings > Sensor Management** page and click **Create Sensor**.

For each custom sensor, define the following:

- **SQL**

- Define the **SQL query** to be run against the sysmaster database on the OneDB server being monitored.
 - Select a sample server to run the query against to preview the data.
- Optionally, specify the **Transpose** option to have the agent transpose (or pivot) the data returned by your SQL query based on a selected column.
- Optionally, specify a **Primary Key** column for your query.
 - If your query returns multiple rows describing multiple objects (e.g. dbspaces), use the primary key column to define the unique identifier for each object.

- **Metrics**

- Define a metric for each column returned by your query. Each metric will have:
 - **Name:** A display name for the metric which will be used to display this metric's data in the OneDB Explore UI.
 - **Unit:** Optional. Defining the unit as percentage or bytes will direct the OneDB Explore UI to format the sensor data values before they are displayed in the UI.
 - **Default Value:** Optional. A default value to be used for the metric if the query returns null for that column.
 - **Calculate Delta:** Optional. If enabled, the agent will store the difference per second between the latest reading and the previous reading.

- **Sensor Metadata**

- A unique **id** for the sensor
- A display **name** for the sensor
- An optional **description** of the sensor
- A default **run interval**
- A default **data retention interval**

User Settings

Each user will have a Settings link in the drop-down menu shown when they click on the user icon in the top right corner of the application title bar. This link is where users can configure their own personal settings.

The user settings page includes links for:

- [Configuring Alerting Notification on page 35](#) – enable alerts for your username and choose how to get alerts (email or Twilio). This requires that the OneDB Explore administrator has configured the corresponding alerting services.
- Changing your password

Configuring User Alerting Notification

OneDB Explore provides two alerting notification options that can be enabled for each individual user: email and Twilio. To enable these notification services, the system administrative user for OneDB Explore must enable email and/or Twilio notifications at the system level on the OneDB Explore [System Settings > Alerting Configuration on page 35](#) page. Then each individual user must enable email or Twilio notifications for their user on the **My Settings->Alerting Configuration** page, providing their email address or phone number respectively.

InfomixHQ also supports [Pager Duty on page 35](#) and [Run Script on page 35](#) based alerting notifications, but these are configured globally in OneDB Explore and are not enabled for each individual user.

Users and Permissions

Only users with System Administrator privileges can add or delete users or modify their privileges. The initial **admin** user that is created when the OneDB Explore server starts for the first time has System Administrator privileges. Additional users with System Administrator privileges can be created.

To add a user, delete a user, or modify their OneDB Explore system privileges and permissions, a System Administrator user should click on their user icon in the top right corner of the title bar and go to the **System Settings > User Management** page.

When adding a user, you can optionally make the new user a System Administrator if you want them to manage users and configure OneDB Explore. System Administrators automatically have full access (**Read, SQL, and Admin** permissions) on all servers and groups. When creating a new non-System Administrator, you have the option to grant **Read, SQL, and/or Admin** permissions for that user to any servers and groups that have been added to OneDB Explore.

Read permissions allow a user to view information about a server in the UI, **Admin** permissions allow a user to execute administrative actions to make changes to a server, and **SQL** permissions allow a user to run SQL queries against that database server on the **Schema Manager** page.



Note: These three permissions are mutually exclusive. Granting **Admin** permissions does not automatically include **Read** permission.

In order to run administrative actions on an OneDB database server, a user must have **Admin** permissions on that server and admin credentials must have been provided when adding that server to OneDB Explore. You can review and edit server credentials by going to the server's **Setup** page or by going to the group dashboard, selecting the server's card, and clicking **Edit**.

Permissions for a server or a group in OneDB Explore are inherited. If a user is granted **Read** or **Admin** privilege on a group, the same privilege will be granted on all servers and sub-groups within that group.

While the **System Settings > User Management** page allows you to view and edit the complete permissions for each individual user of OneDB Explore, there is also a **Permissions** page specific to each server or group that allows you to see (and edit) all users who have permissions to that particular server or group. You can access this page by navigating to a particular server or group and clicking on **Permissions** in the side-bar menu.

HCL OneDB™ Explore Agent

The HCL OneDB™ Explore agent is a lightweight Java 8 based monitoring agent. It should be installed on the host machine for each OneDB database server that you want monitored by HCL OneDB™ Explore.

The agent runs alongside the database server, gathering database statistics through JDBC connections to sysmaster. The HCL OneDB™ Explore agent only needs read access to the database server.

By running the agent directly on the OneDB server host machine, the agent is also able to gather and monitor OS statistics which can be just as critical in evaluating and tuning the OneDB database server performance metrics.

OneDB Explore Agent Setup

This topic explains how to configure OneDB Explore agent from OneDB Explore UI.

Select repository database

Repository server is the server which contains Repository database, which will be used to store all the monitoring data collected by OneDB Explore agent. Without selecting a repository database, user is not allowed to save agent setup changes. Repository database server can be any OneDB server specified in OneDB Explore User Interface.

TIP: User should create a dedicated database to store metrics to be captured by OneDB Explore agent. A new database can easily be created from [Schema Manager on page 22](#) page. Similarly metrics can be defined by adding Sensors from Monitoring page.

Following are two scenarios if repository database is not available for any reason:

1. **OneDB Explore Agent is not configured yet:**

In this scenario, OneDB Explore agent configuration is not set yet by the user, probably user is setting up OneDB Explore agent for the first time. OneDB Explore agent connects with monitoring server using existing OneDB server connection properties.

Setup

Server **Agent**

Repository Database Configuration

* Select Repository Server * Select Database

onedb server No database available.

Connection Properties

By default OneDB explore agent connects with sysmaster database using existing server connection properties. [See more...](#)

server1: Connection Properties

Use existing connection properties (Uncheck to modify properties)

2. OneDB Explore Agent is already configured:

In this scenario, previously saved OneDB Explore agent configuration will be shown in read only mode. Basically, whenever user runs agent.jar, it will connect using the already saved configuration.

Repository Database Configuration

* Select Repository Server * Select Database

onedb server No database available.

Connection Properties

OneDB explore agent connects with sysmaster database on 'onedb server1' and repository database (test1) on 'onedb server' [See more...](#)

onedb server1: Connection Properties

Use existing connection properties (Uncheck to modify properties)

SSLCONNECTION	true	x
SSL_TRUSTSTORE	C:\cert\client.p12	x
TRUSTSTOREPASSWORD	*****	x

onedb server: Repository Database Server Connection Properties

Use existing connection properties (Uncheck to modify properties)

SSLCONNECTION	true	x
SSL_TRUSTSTORE	C:\cert\ssl.keystore	x
TRUSTSTOREPASSWORD	*****	x

Add Connection Properties

Separate connection properties can be specified for the OneDB Explore agent. Users can specify connection properties for monitoring server and repository database server, respectively. Following are the two cases while specifying connection properties.

1. Repository database is located on monitoring Server:

In this scenario, OneDB Explore agent will connect with monitoring server and repository database using the same connection properties.

By default, OneDB Explore agent will use OneDB server connection properties (from OneDB server setup page) to connect with monitoring server and repository database. (When the checkbox is selected.)

The screenshot shows the 'Repository Database Configuration' page. Under the 'Connection Properties' section, the checkbox 'Use existing connection properties (Uncheck to modify properties)' is checked. The 'oneodb server' dropdown is set to 'onedb server' and the 'Select Database' dropdown is set to 'test1'. A 'Save' button is visible at the bottom.

If needed, user can add/modify OneDB Explore agent connection properties by unchecking the checkbox as follows:

The screenshot shows the 'Repository Database Configuration' page with the 'Use existing connection properties' checkbox unchecked. Below this, three custom connection properties are listed: 'SSLCONNECTION' with value 'true', 'SSL_TRUSTSTORE' with value 'C:\cert\ssl.keystore', and 'TRUSTSTOREPASSWORD' with a masked value. Each property has a red 'X' icon to its right. A '+ Add Connection Property' button is located below the list. A 'Save' button is at the bottom.

This will be specifically useful for providing different SSL keystore path for OneDB Explore agent.

2. **Repository database server is located on a different server:**

In this scenario, OneDB Explore agent will connect with monitoring server and repository database using different connection properties.

By default, OneDB Explore agent will use connection properties of respective OneDB servers to connect with monitoring server and repository database, respectively. (When checkbox is checked)

If needed, user can add/modify OneDB Explore agent connection properties for monitoring server and repository database separately by unchecking respective checkboxes as follows:

This will be specifically useful for providing different SSL keystore path for monitoring server and repository database, respectively. See ([Compatibility matrix for Java with SSL Keystore format on page 43](#)).

In both scenarios, if checkbox is unchecked, at least one connection property should be added to enable save button.

**Note:**

If Repository database is not configured or could not be connected due to any reason, then user will not be able to add/modify agent connection properties.

If already configured Repository server is removed from OneDB Explore, then such server will not be visible in agent setup page under **Select Repository server**. In this scenario, all the custom agent connection properties previously saved, will reset and user needs to re-enter all the custom connection properties in Agent Setup page once new repository server is selected.

Compatibility matrix for Java with SSL Keystore format

Following are keystore formats supported based on Java providers:

Java Provider	Type
IBM Java(v1.8)	JKS
Oracle Java(v1.8)	JKS, PKCS

Related information

[Schema Manager on page 22](#)

OneDB Explore Agent Configuration Parameters

A properties file is required to run the OneDB Explore agent.

When starting the agent, you can pass the properties file name as part of the start command. Otherwise, the agent will look for a properties file named **agent.properties** in the classpath.

An example configuration file documenting the supported OneDB Explore agent configuration properties can be found at `OneDB Explore-agent-example.properties`.

- Required configuration properties
 - [ONEDB_SERVER.id on page 44](#)
 - [server.host on page 44](#)
 - [server.port on page 44](#)
- Optional configuration properties
 - [dataSource.IFX_ISOLATION_LEVEL on page 44](#)
 - [pool.connectionTimeout on page 44](#)
 - [pool.idleTimeout on page 44](#)
 - [pool.maximumPoolSize on page 44](#)
 - [pool.minimumIdle on page 45](#)
 - [ssl.enable on page 45](#)

- [ssl.keystore.file](#) on page 45
- [ssl.keystore.password](#) on page 45
- [target.ONEDB_HOME](#) on page 45
- [target.onlinelog](#) on page 45
- [target.ping.frequency](#) on page 46
- [user.password.minLength](#) on page 46
- [user.password.requireLowerCase](#) on page 46
- [user.password.requireUpperCase](#) on page 46
- [user.password.requireNumber](#) on page 46
- [user.password.requireSpecialCharacterFromSet](#) on page 46

ONEDB_SERVER.id

The id of the OneDB database server in OneDB Explore. You find the server's id on the server's Setup page in the OneDB Explore UI.

server.host

The host name on which the OneDB Explore server is running.

server.port

The port on which the OneDB Explore server is running.

dataSource.IFX_ISOLATION_LEVEL

Specifies the isolation level to set on JDBC connections to the target and repository OneDB database servers.

The default value is 1 (DIRTY READ).

pool.connectionTimeout

Specifies the number of milliseconds to wait for a JDBC connection to the target or repository OneDB database server to be established before it times out. The default value is 5000 (5 seconds).

pool.idleTimeout

Specifies the number of milliseconds that a JDBC connection can be idle in the connection pool before it is closed.

The default value is 60000 (1 minute).

pool.maximumPoolSize

The maximum number of JDBC connections in each connection pool. The OneDB Explore agent will maintain a connection pool for the target database server and another a connection pool for the repository database server. The **pool.maximumPoolSize** puts a cap on the total number of open JDBC connections that can be established to each database. The default value is 3.

pool.minimumIdle

The minimum number of idle JDBC connections in each connection pool. The OneDB Explore agent will maintain a connection pool for the target database server and another a connection pool for the repository database server. Setting **pool.minimumIdle** to zero indicates that all JDBC connections in the connection pool should be closed when they exceed the **pool.idleTimeout**. Setting **pool.minimumIdle** to a positive integer indicates the number of connections that should be kept open in the connection pool even when they exceed the **pool.idleTimeout**.

The default and recommended value is 0.

ssl.enable

Whether SSL should be enabled to secure web socket communication between the agent and the OneDB Explore server. Set this value to true if the OneDB Explore server port specified in **server.port** is an HTTPS port.



Note: if **redirectHTTPtoHTTPS** is set to true in the OneDB Explore server configuration file, you must set this value to true in the agent configuration file.

If **ssl.enable** is set to true, you must also configure the **ssl.keystore.file** and **ssl.keystore.password** configuration properties.

The default value is false.

ssl.keystore.file

The path to the keystore file that contains the certificate to use for encrypting web socket communication between the agent and the OneDB Explore server. This property must be set if **ssl.enable** is set to true.

ssl.keystore.password

The password to unlock the keystore file for used for encrypting web socket communication between the agent and the OneDB Explore server.

This property must be set if **ssl.enable** is set to true.

target.ONEDB_HOME

Optionally, specify the directory on the local machine where OneDB is installed. If left empty, the agent will query the server for the ONEDB_HOME property. This property is used by sensors that gather data from onstat or other OneDB utilities

target.onlinelog

Optionally specify the path to the **online.log** file for the target OneDB database server.

If this is left empty and the Online Log monitoring sensor is enabled for this server, the agent will lookup the **online.log** file path by querying the database server.

There is no default value.

target.ping.frequency

Specifies the interval, in seconds, between pings to the target database server to see if it is still online. When the agent is running, it will regularly monitor whether the target database server is online or offline. This property controls the duration between these checks.

The default value is 1, indicating to check the server status every second.

user.password.minLength

Controls the minimum length for a user password. The default value is 8.

user.password.requireLowerCase

Controls whether user passwords are required to include at least one lowercase character. The default value is true.

user.password.requireUpperCase

Controls whether user passwords are required to include at least one uppercase character. The default value is true.

user.password.requireNumber

Controls whether user passwords are required to include at least one number. The default value is true.

user.password.requireSpecialCharacterFromSet

Controls whether user passwords are required to include at least one special character. An empty string indicates that no special characters are required. Setting this value to "!@#%&*()?" would require user passwords to include at least one of those characters. The default value is an empty string.

HCL OneDB™ Explore Server API Syntax and Examples

This topic document the syntax of REST requests supported by the HCL OneDB™ Explore Server.

HCL OneDB™ Explore Server jar file comes bundled with its own OpenAPI documentation of the REST API included in that jar file. You can access the OpenAPI documentation for your Explore server by accessing `http://<hostname>:<port>/dashboard/groups/0/swagger` in your browser, where <hostname> and <port> reflect the host and port where you are running the explore server.

For example, you might find your OpenAPI documentation at `http://localhost:8080/dashboard/groups/0/swagger`.

You can find an online version of this documentation here: [Explore API documentation for OneDB](#).

Upgrading OneDB Explore

OneDB Explore uses an embedded database called H2 to store custom configuration details. OneDB Explore 2.0.1.2 has upgraded its H2 version due to security vulnerabilities present in the older versions of H2. To upgrade OneDB Explore to

2.0.1.2 from the older versions, direct upgrade is not possible and the data needs to be migrated manually. For new users, who are using OneDB Explore 2.0.1.2 for the first time, there is no impact, and no upgrade is needed as OneDB Explore will create new H2db with latest version.

Before you begin

The following table lists the requirements for OneDB Explore 2.0.1.2 upgrade:

Table 1. Prerequisites	
Software	Required Version
OneDB Explore Server (onedb-explore-server.jar) or H2 Database Engine (h2-1.4.192.jar)	2.0.1.1 or earlier 1.4.192
OneDB Explore Server (onedb-explore-server.jar)	2.0.1.2
Java	1.8

1. Make a copy of the existing database:

Find the existing H2 database file path. For OneDB Explore, database can be found at `<installation_path>/explore/server/h2db.mv.db`. Make a copy of the **db** file before starting the migration. This back up file will be useful if any issues are encountered during migration.

2. Export existing data using current version of OneDB Explore or H2:

- a. Create a SQL source file **migrate.sql** . Store the file in the same folder where H2 database file is present.

migrate.sql:

```
ALTER TABLE IF EXISTS USERS ALTER COLUMN ID SET NOT NULL;
ALTER TABLE IF EXISTS USERS ADD PRIMARY KEY (ID);

ALTER TABLE IF EXISTS ONEDB_SERVERS ALTER COLUMN ID SET NOT NULL;
ALTER TABLE IF EXISTS ONEDB_SERVERS ADD PRIMARY KEY (ID);

ALTER TABLE IF EXISTS ONEDB_SERVER_groups ALTER COLUMN ID SET NOT NULL;
ALTER TABLE IF EXISTS ONEDB_SERVER_groups ADD PRIMARY KEY (ID);

ALTER TABLE IF EXISTS ALERTING_INCIDENTS ALTER COLUMN ID SET NOT NULL;
ALTER TABLE IF EXISTS ALERTING_INCIDENTS ADD PRIMARY KEY (ID);
```

Run the java command from the path where H2 db is present. Replace `<path>` with the path where onedb-explore-server.jar (2.0.1.1) is present.

```
java -cp <path>/onedb-explore-server.jar org.h2.tools.RunScript -url jdbc:h2:./h2db -script migrate.sql
```

- b. Export the database into a zip file.

Run the java command from the path where h2 db is present. Replace <path> with the path where onedb-explore-server.jar (2.0.1.1) is present.

```
java -cp <path>/onedb-explore-server.jar org.h2.tools.Script -url jdbc:h2:./h2db -script h2db.zip -options compression zip
```



Note: h2-1.4.192.jar can be used if onedb-explore-server.jar (2.0.1.1) is not available. If you are using h2-1.4.192.jar, replace "<path>/onedb-explore-server.jar" with "<h2_path>/h2-1.4.192.jar" in the above commands.

3. Create new H2 database and import the data:

Rename the existing h2db.mv.db database:

```
rename h2db.mv.db h2db_old.mv.db
```

Run the java command from the path where H2 database file is present. Replace <path> with the path where onedb-explore-server.jar (2.0.1.2) is present.

```
java -cp <path>/onedb-explore-server.jar org.h2.tools.RunScript -url jdbc:h2:./h2db -script h2db.zip -options compression zip
```

4. **Optional:** Clean the migration files by deleting h2db.zip , h2db_old.mv.db and migrate.sql files.



Note: If **h2.encrypt.enable** & **h2.encrypt.password** parameters are set in **onedb-explore-server.properties**, then all the URL needs to be modified to include those values.

Example 1: If h2.encrypt.enable=true and h2.encrypt.password=password123, H2 database URLs should be modified as below:

(Add a space after the password, since the format is <file_password> space <user_password>)

```
java -cp <path>/onedb-explore-server.jar org.h2.tools.RunScript -url "jdbc:h2:./h2db;CIPHER=AES" -password "password123 " -script migrate.sql
```

Example 2: If h2.encrypt.enable=true , h2.encrypt.password=password123 and h2.encrypt.algorithm=XTEA, h2 database URLs should be modified as below:

```
java -cp <path>/onedb-explore-server.jar org.h2.tools.RunScript -url "jdbc:h2:./h2db;CIPHER=XTEA" -password "password123 " -script migrate.sql
```

Results

Once all the above steps are completed without any error, data is successfully migrated to h2db.mv.db.

Frequently asked questions (FAQs) about HCL OneDB™ Explore

These topics provide short answers to some frequently asked questions about HCL OneDB™ Explore.

High level architecture and functionality

This topic provides answers to some frequently asked questions about high level architecture and functionality.

- [What is the difference between the HCL OneDB Explore server and agent? on page 49](#)
- [Is it necessary to start the agent to use HCL OneDB Explore? on page 49](#)
- [I have multiple OneDB database instances running on the same host machine. Do I need one agent per host or one agent per database server? on page 49](#)

What is the difference between the HCL OneDB™ Explore server and agent?

The OneDB Explore server is a Java 8 based Jetty web server that hosts both the web user interface (UI) portion of the tool and the REST web services. The HCL OneDB™ Explore server also connects directly to the OneDB database server instances to gather live data and run administration commands, manages connections to all agents, and evaluates and dispatches alerts when new monitored data comes in, among other things. You only need to run a single instance of the HCL OneDB™ Explore server to manage and monitor all of your OneDB database server instances.

The HCL OneDB™ Explore agent is a lightweight Java program that runs alongside each of your OneDB database server instances and gathers monitoring data. The agent is intended to be installed on the same host machine as the OneDB database server that you want monitored by HCL OneDB™ Explore, which allows it to also gather operating system statistics about the host machine. Unlike the HCL OneDB™ Explore server, you will have one instance of the HCL OneDB™ Explore agent running for each OneDB database server that you want the tool to monitor.

For information, see [OneDB Explore Architecture on page 3](#).

Is it necessary to start the agent to use HCL OneDB™ Explore?

The agent is not required to use HCL OneDB™ Explore. However, it is important to note that the agent process is the one responsible for gathering all of the monitoring data. Therefore, if you choose not to connect the agent, the tool will not be monitoring your OneDB database servers when you close your web browser. You will not be able to see graphs in the UI of how various performance metrics are trending over time and you will not be able to configure alerting conditions if you are not using the agent.

I have multiple OneDB database instances running on the same host machine. Do I need one agent per host or one agent per database server?

There is a one-to-one relationship between the HCL OneDB™ Explore agent and the OneDB database server. Each agent monitors just one OneDB database server instance. If you have multiple OneDB database server instances on the same host and you want to monitor each of them, then you will need to have multiple agents on that same host machine, one for each database server instance.

Getting Started

This topic provides answers to some frequently asked questions on getting started with HCL OneDB™ Explore.

- [Where can I get HCL OneDB Explore? on page 50](#)
- [What should I do before upgrading to the latest version of HCL OneDB Explore? on page 50](#)
- [Where can I find sample configuration files for the server and agent? on page 50](#)
- [How can I configure the logging for the server or agent? How do I change the logging level? on page 50](#)

Where can I get HCL OneDB™ Explore?

HCL OneDB™ Explore is available as part of the HCL OneDB™ database server installation.

For information, see [Getting Started on page 4](#).

What should I do before upgrading to the latest version of HCL OneDB™ Explore?

Before upgrading to the latest version, you must make a backup copy of the H2 database file. This is required in case if you ever want to revert to the previous version.

Where can I find sample configuration files for the server and agent?

Sample configuration files for both the HCL OneDB™ Explore server and agent are available in the OneDBExplore package of your OneDB database server installation.

For information, see [OneDB Explore Server Configuration on page 10](#) and [OneDB Explore Agent Configuration on page 43](#).

How can I configure the logging for the server or agent? How do I change the logging level?

OneDB Explore uses the [logback](#) library for logging. By default, the HCL OneDB™ Explore server and agent will log messages at INFO level to an HCL OneDB™ Explore-server.log file and an HCL OneDB™ Explore-agent.log file respectively.

You can customize the logging behavior by providing a server-logback.xml file in the current directory or classpath when starting the HCL OneDB™ Explore server or a agent-logback.xml file in the current directory or classpath when starting the HCL OneDB™ Explore agent. Use these logback configuration files to change the logging level (ERROR, WARN, INFO, or DEBUG), change the log file location, or enable rolling window logging. For more information, see [logback](#) and [Logging in HCL OneDB™ Explore on page 7](#).

How do you stop the server and how do you stop the agent?

You can stop both the HCL OneDB™ Explore server and agent by terminating the java process that is running them.

Monitoring and the Repository Database

This topic provides answers to some frequently asked questions on monitoring and the repository database in HCL OneDB™ Explore.

- [Where is the monitored data stored? on page 51](#)
- [Does the repository database need to exist ahead of time? Do I need to run any DDL statements to initialize the repository database? on page 51](#)
- [If I add a new database server instance to a group and start an agent for that new server, do I have to do anything to enable all of the group's sensors on the new server? on page 51](#)

Where is the monitored data stored?

The monitored data is stored in the [repository database on page 8](#) as defined in the HCL OneDB™ Explore UI. The repository database must be an OneDB database, but it can be located wherever you choose. You can store the monitored data within the same OneDB instance that is being monitored or you can choose to use a central repository database to store all of the monitored data for all of your OneDB database server instances.

Does the repository database need to exist ahead of time? Do I need to run any DDL statements to initialize the repository database?

The repository database must exist before you define it as a repository database. You define a repository database in the HCL OneDB™ Explore UI on a server's **Setup > Agent** page.

You do not need to run any DDL statements to initialize the schema for the repository database. The agent will automatically create the tables it needs to store the monitored data.

If I add a new database server instance to a group and start an agent for that new server, do I have to do anything to enable all of the group's sensors on the new server?

No, this happens automatically. Sensors defined in a group's monitoring profile are automatically applied to all database servers within the group. When a new server is added to the group, it will automatically inherit all of the sensors enabled in the group's monitoring profile. No additional step is needed to make this happen. The same thing applies to a group's alerting profile.

Security

This topic provides answers to some frequently asked questions on HCL OneDB™ Explore security.

- [Do I need to keep the initialAdminPassword in the properties file after the HCL OneDB Explore server is started for the first time? Isn't it a security issue to keep the password in plain text in the properties file? on page 51](#)
- [How can I configure HTTPS and/or SSL for HCL OneDB Explore? on page 52](#)
- [How can I encrypt the internal H2 database that the HCL OneDB Explore server uses? on page 53](#)
- [How can I configure HCL OneDB Explore to use SSL when connecting to my database server? on page 53](#)

Do I need to keep the initialAdminPassword in the properties file after the HCL OneDB™ Explore server is started for the first time? Isn't it a security issue to keep the password in plain text in the properties file?

The [initialAdminPassword on page 11](#) property is only required in the HCL OneDB™ Explore server properties file the first time it is started. When the server is started for the very first time, it initializes its internal H2 database and creates the initial admin user. For all subsequent starts of the HCL OneDB™ Explore server, the admin user will already exist and therefore the

initialAdminPassword will be ignored if it is present in the properties file. This means that after the server is started for the first time, you can safely remove the initialAdminPassword property from the properties file. This allows you to avoid having that password continue to sit around in plain text in your properties file.

How can I configure HTTPS and/or SSL for HCL OneDB™ Explore?

To use the Secure Sockets Layer (SSL) protocol to encrypt communication with HCL OneDB™ Explore, you will need a keystore and certificate. You can use the method that best fits your environment for creating the keystore and certificate, for example Java keytool, OpenSSL.

- **Configuring HTTPS in the HCL OneDB™ Explore server**

Once you have the keystore, secure the HCL OneDB™ Explore web user interface and REST API by configuring HTTPS in the HCL OneDB™ Explore server. To configure HTTPS in the OneDB Explore server, in your HCL OneDB™ Explore server properties file, set the [httpsPort on page 12](#), [ssl.keystore.file on page 13](#), and [ssl.keystore.password on page 13](#) properties and potentially also the [ssl.key.password on page 14](#) property if your key password is different from the keystore password.

Additionally, if you want to disable HTTP access to the HCL OneDB™ Explore so that all communication to and from the HCL OneDB™ Explore server uses HTTPS, set the [httpPort on page 12](#) to -1 in your properties file. If instead you would like the HCL OneDB™ Explore server to automatically redirect all HTTP traffic to the HTTPS port, set the [redirectHTTPtoHTTPS on page 13](#) property to true.

Sample HCL OneDB™ Explore server properties file with HTTPS enabled:

```
# The initialAdminPassword is only required the first time the OneDB Explore server is started
initialAdminPassword=myAdminPassword

# configure ports
httpPort=-1
httpsPort=8088
redirectHTTPtoHTTPS=false

# configure keystore
ssl.keystore.file=/opt/OneDB Explore/mykeystore.jks
ssl.keystore.password=myStorePassword
# uncomment the following line if a separate key password is required for your keystore
#ssl.key.password=myKeyPassword
```

- **Configuring the HCL OneDB™ Explore agent to encrypt its web socket communication with SSL**

Once you have HTTPS enabled in the HCL OneDB™ Explore server, you must configure your OneDB Explore agents to encrypt their web socket communication with the HCL OneDB™ Explore server. If you use the **Deploy Agent** button in the UI to have the HCL OneDB™ Explore server automatically deploy the agent, it will automatically configure the agent to use SSL if the OneDB Explore server has HTTPS enabled.

If you are starting your agents manually to enable SSL, set the [ssl.enable on page 45](#) property to true in your agent configuration file and then set the [ssl.keystore.file on page 45](#) property, the [ssl.keystore.password on page 45](#) property.

Sample agent configuration file with SSL enabled:

```
# host and port of the OneDB Explore server
server.host=localhost
server.port=8088

# The id of the OneDB database server as defined in OneDB Explore
ONEDB_SERVER.id=1

# SSL configuration
ssl.enable=true
ssl.keystore.file=/opt/OneDB Explore/mykeystore.jks
ssl.keystore.password=myStorePassword
```

How can I encrypt the internal H2 database that the HCL OneDB™ Explore server uses?

The OneDB Explore server creates an H2 database to store its internal metadata. The H2 database file, **h2db.mv.db** will be created in the directory where you start the HCL OneDB™ Explore server. It will store information about the groups and servers you define in the tool (including the database server connection credentials), the monitoring and alerting profiles, and alerting incidents.

You can configure encryption for this H2 database file by setting the following properties in your OneDB Explore server configuration file.

```
h2.encrypt.enable=true
h2.encrypt.password=some_password
```

Optionally, you can also set the `h2.encrypt.algorithm` property if you want to set the encryption algorithm to something other than AES.



Note: If you want to encrypt the H2 database, you must set these properties the first time you start the OneDB Explore server when the H2 database is first created and initialized. You cannot change your H2 encryption configuration after the H2 database has been created. If you want to encrypt an H2 database that has already been created, you can use H2's ChangeFileEncryption tool as described in http://www.h2database.com/html/features.html#file_encryption or you can delete your `h2db.mv.db` file and have the OneDB Explore server recreate it from scratch the next time you start it.

How can I configure HCL OneDB™ Explore to use SSL when connecting to my database server?

If your database supports or requires SSL connections, you can setup SSL using the connection properties on the **Add Server** page when adding the server or on the server's **Setup** page after it is created.

You must add the following connection properties in order to use SSL on HCL OneDB™ Explore's JDBC connections to your database server:

```
SSLCONNECTION=true
SSL_TRUSTSTORE=/path/to/truststore
SSL_TRUSTSTORE_PASSWORD=password
```

The truststore/keystore file that you specify must be present both where HCL OneDB™ Explore server is running as well as the machine where the OneDB Explore agent is running.

For more information, see [Adding Servers and Groups on page 14](#).

Users and Permissions

This topic provides answers to some frequently asked questions on HCL OneDB™ Explore users and permissions.

- [What's the difference between monitoring/admin credentials and Read/SQL/Admin permissions? on page 54](#)
- [What privileges are required on the HCL OneDB database server? on page 55](#)
- [Is there any relationship between the users I create in HCL OneDB Explore and OS users? on page 55](#)

What's the difference between monitoring/admin credentials and Read/SQL/Admin permissions?

The monitoring and admin credentials are used by the HCL OneDB™ Explore server itself as part of the JDBC connection whenever it needs to connect to the database server. The Read, SQL, and Admin permissions are those that are assigned to users in the tool and therefore control what users can see and do in the UI.

• Monitoring and admin credentials

When you click **Add Server** from a group dashboard page to add a new OneDB database server to OneDB Explore, you are asked to provide the HCL OneDB™ Explore tool with not only the host and port, but also user and password information that will be used when establishing a JDBC connection to that database server instance.

The monitoring credentials are used by the HCL OneDB™ Explore server whenever it needs to query for data to be displayed in the UI or by the HCL OneDB™ Explore agent data it monitors your database server instance.

The admin credentials are used whenever a user in the UI requests that an administration action be performed on the database server, for example creating a dbspace, editing an onconfig parameter, or deploying an agent. The user provided for the admin credentials should be a DBSA and needs access to the sysadmin database and permissions to run SQL Admin API commands on the database server.

The required privileges for the monitoring and admin users can also be found in [Adding Servers and Groups on page 14](#).

It is required that you provide monitoring credentials when adding your OneDB database server information to HCL OneDB™ Explore. Admin credentials need only be provided if you want to use HCL OneDB™ Explore to run administrative actions on your database server or if you want that server to be used as a repository database.

• Read, SQL, and Admin permissions

The Read, SQL, and Admin permissions are the permissions assigned to the users that are created in HCL OneDB™ Explore itself. These permissions determine what kinds of access that a user has on various OneDB servers and groups in the UI and also which REST services that user is authorized to run. Therefore, these permissions control what each user can see or do in the tool.

Read permission provides the ability to view information about a server. SQL permission provides the ability to run any SQL query or statement (including DML) against the database server on its Schema Manager page in the UI. Admin permission provides the ability run administrative actions on the server, for example creating a dbspace or editing and onconfig parameter. Read, SQL, and Admin permissions are mutually exclusive, so be sure to grant each user the full set of permissions that they will need.

Read, SQL, and Admin permissions can be granted to OneDB Explore users by a System Administrator user in HCL OneDB™ Explore on the **System Settings > User Management** page or on the **Permissions** page for any server or group.

What privileges are required on the HCL OneDB™ database server?

The required privileges for the monitoring and admin users can be found in [Adding Servers and Groups on page 14](#).

Is there any relationship between the users I create in HCL OneDB™ Explore and OS users?

There is no relationship between the users you create in HCL OneDB™ Explore and the operating system users on the host machine it is running on. Users that are created in OneDB Explore are users that are specific to the HCL OneDB™ Explore tool. They have no relationship to operating system users or even HCL OneDB™ database server users.

Index

F

- FAQ 49
- Frequently asked questions 49

O

- OneDB Explore
 - API 46
 - examples 46
 - syntax 46
- OneDB Explore migration)
 - upgrading 46
- OneDB Explore Upgrade 46

T

- Troubleshooting
 - FAQ 49
 - frequently asked questions 49