

HCL Detect v12.1.8 Admin Guide



Contents

Chapter 1. Installing HCL Detect.....	3
Dependencies.....	3
Installing OS-managed packages.....	4
Configuring MariaDB.....	5
Configuring Redis for use by HCL Detect.....	8
Installation Process.....	9
Configuring HCL Detect's Runtime Environment.....	14
Configuring web proxy.....	14
Chapter 2. Running HCL Detect.....	21
Chapter 3. Accessing the interface for the first time.....	26
Chapter 4. Administration Interface.....	27
User & Role Management.....	27
System Health.....	37
Applications, Aggregates & Enrichments.....	40
Subscriber Segments.....	46

Chapter 1. Installing HCL Detect

HCL Detect is an on-premise next generation enterprise marketing automation product. Built for marketers, it enables marketing teams to plan and manage the customer lifecycle end-to-end and to deliver personalized campaigns quickly moving from micro segments to a segment of one.

It runs on RedHat 8 (or CentOS 8).

In the next sections, we will go over the software dependencies that must be satisfied prior to the installation of HCL Detect.

Dependencies

The server where HCL Detect is installed must be configured according to a few guidelines. The software also relies on a set of external dependencies that must be installed prior to setting it up for use. In this section, we discuss both of these issues.

Operating System Configuration

The server (or servers) where Detect runs must be properly configured as far as certain OS and shell resource limits.

While these limits are going to be checked during installation, we will provide certain helpful guidelines in this section.

First, the number of file descriptors that can be used by a process must be equal to or greater than 640,000.

The soft and hard limits are defined as part of the OS configuration (defined in `/etc/security/limits.conf`) and used/enforced by the (Bash) shell.

In the shell, the *soft* limits for each of these resources (including the maximum number of open files) can be inspected as follows:

```
ulimit -Sa
```

And the *hard* limits as follows:

```
ulimit -Ha
```

Naturally neither limit can go beyond the OS limit that applies to the server.

If the current limit is too low for the current shell, but sufficient as far as the OS is concerned, you can update your local `.bashrc` raising the limit as follows:

```
ulimit -n 640000
```

Nevertheless, in most cases, after a fresh OS install, it is necessary to update `/etc/security/limits.conf` to meet Detect's needs.

We recommend that you consult your OS documentation, but usually the following settings can be added towards the end of the file (assuming you want the limits applied to all users):

```
* soft nofile 640000
* hard nofile 640000
```

```
# End of file
```

After applying this change, a new login must be made with the userid that will be used by Detect (no reboot is necessary).

External Dependencies

The external software dependencies are required by Detect are specific to the operating system version and architecture where the software will run.

There are two types of external dependencies. The ones provided by the OS vendor and the ones from external vendors.

Starting with the dependencies provided by external vendors, requiring a manual installation, you will need to obtain:

- optional: IBM InfoSphere Streams, IBM SPSS, as well as Oracle WebLogic, if these are capabilities you have acquired in your particular HCL Detect installation. These are commercial products and you should consult their respective documentation to have them installed in your environment. Their integration with HCL Detect is described later in this section, where the `installer` will request additional information regarding their location in the file system.

Installing OS-managed packages

When it comes to OS-managed dependencies, i.e., dependencies that can be installed using the operating system package management, the installer will look for and warn you about missing dependencies.

The installation package comes with a utility, `dependency_checker`, that can be used to ensure that all dependencies are in place ahead of the installation.

This utility inspects the environment for RedHat- or Ubuntu-provided software (referred to as OS-provided software in the rest of this documentation) as well as for specific Python packages required by HCL Detect, which are provided as a `virtualenv` environment, pre-configured to match the HCL Detect needs.

Operating System-software packages must be installed using the regular mechanism employed to download and install them, usually `yum` on RedHat and `apt-get` on Ubuntu.

When using `dependency_checker` to extract the list of required dependencies, the output will be similar to (but not necessarily the same as) the following:

```
$ ./drive/bin/dependency_checker -l
List of OS package dependencies:

advance-toolchain-at8.0-runtime: 8.0 (installed)
mariadb: 5.5 (not installed)
mariadb-libs: 5.5 (installed)
mariadb-server: 5.5 (installed)

List of Python package dependencies (available in the HCL Detect virtualenv):
...
```

In this example, one external dependency (`mariadb`) is not currently installed. In this case, assuming this host is running RedHat Linux, `yum` must be used to install `mariadb`.

**Note:***Installing OS packages on a server without Internet connection*

In many cases, the server (or cluster) where HCL Detect is going to be installed is not directly connected to the Internet.

In such cases, the installation of additional OS-level packages can be accomplished by having access to the Operating System installation CD/DVD or, simply, to an .iso image with the OS installation.

If your installation is RedHat-based, use one of the following alternatives:

- if a DVD is available, please follow the [DVD-based yum repository directions](#) outlined by RedHat to create a locally available yum repository.
- if an .iso file is available, please follow these [.iso file directions](#) to create a locally available yum repository.

If your installation is Ubuntu-based, use one of the following alternatives:

- if a DVD is available, please follow the [DVD-based apt repository directions](#) outlined by Canonical to create a locally available aptitude repository.
- if an .iso file is available, mount it first and then use the mounting point in the steps above as the location when running `apt-cdrom`. To mount the .iso perform the following steps either `sudo`-ed or by logging in as `root`:

```
$ mkdir -p <mounting point location>
$ mount -o loop <file>.iso <mounting point> location
```

When installing external operating system-managed dependencies, as long as the major and minor version numbers match, the dependency is considered satisfied.

Configuring MariaDB

HCL Detect requires a relational database to store configuration as well as runtime data used by the applications as it runs.

Currently, the only supported database is [MariaDB](#).

Installing MariaDB

If you need to have it installed prior to installing HCL Detect, please refer to [MariaDB's online installation instructions](#) to become acquainted with both the installation and configuration process. The actual installation is done via `apt-get` or `yum`, depending on whether you are installing it on Ubuntu or RedHat.

MariaDB is executed as a service on both Ubuntu as well as on RedHat and its behavior is controlled by a configuration file (usually located in `/etc/my.cnf.d/server.cnf` on RedHat or `/etc/mysql/my.cnf` on Ubuntu). In this file, specifically in the `mysqld` section of the configuration, the following entry **must be commented out or removed**:

```
bind-address = 127.0.0.1
```

This change will ensure that other hosts in the cluster can interact with the MariaDB server.

If the version for the MariaDB server is older than 10.3.1, the InnoDB engine must also be appropriately configured with the following settings in the [mysqld] section:

```
innodb_file_format=Barracuda
innodb_file_per_table=ON
innodb_large_prefix=ON
```

The configuration modifications only become active after a server restart, which requires restarting the specific operating system service as follows. On RedHat:

```
$ sudo systemctl restart mariadb
```

And on Ubuntu:

```
$ sudo systemctl restart mysql
```

Finally, ensure that the MariaDB server is automatically started at boot time by appropriately configuring `init`, `systemd`, `cron` or whatever mechanism is in place for automating the startup of services.

Securing MariaDB

Once the installation is performed, it's also recommended that MariaDB's installation and configuration be [hardened](#):

```
$ sudo mysql_secure_installation
```

The simplest way to make the MariaDB server (`mysqld`) available to HCL Detect is to install it on the same host where HCL Detect is going to be placed. It is also necessary to ensure that the server is using its default port (3306):

```
$ netstat -lptn | grep 3306
tcp    0    0 0.0.0.0:3306    0.0.0.0:*    LISTEN  11430/mysqld
```

Note that placing the MariaDB server on another host as well as using a different port number can be done, but both of these options require additional configuration.

Populating the MariaDB timezone tables

Several tables in the MariaDB system database exist to store time zone information.

The MariaDB installation procedure creates the time zone tables, but does not load them.

To do so manually, as `root` run the following command (MariaDB's `root` password will be requested):

```
$ mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root -p mysql -D mysql
```

There will be one or two warnings similar to:

```
Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.
```

which you can ignore.



Note:



Loading the time zone information is not necessarily a one-time operation because the information changes occasionally.

When such changes occur, applications that use the old rules become out of date and you may find it necessary to reload the time zone tables (using the instructions above) to keep the information used by the MariaDB server current. Please refer to the MariaDB [documentation](#) for more information.

Setting up a MariaDB user

Once MariaDB is properly installed, it must be configured with a user and password to be used by HCL Detect when establishing connections with the database server. By default, both the user and password are set to `drive`.

To create a MariaDB user called `drive` with the password `drive`, start MariaDB's interactive shell using MariaDB's `root` user:

```
$ mysql -u root -p
```

And issue the following command:

```
MariaDB [(none)]> CREATE USER '<user>'@'%' IDENTIFIED BY '<password>';
```

Replacing `<user>` and `<password>`, with `drive`, specifically:

```
MariaDB [(none)]> CREATE USER 'drive'@'%' IDENTIFIED BY 'drive';
```

Once the user is created, it must be given privileges to create the HCL Drive databases:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON `drive_%` . * TO 'drive'@'%';
MariaDB [(none)]> FLUSH PRIVILEGES;
```

You can now close the MariaDB interactive shell by pressing CTRL-D (the `Control` and the `D` key, together) or by using the `exit` command.

At this point you should be able to login to MariaDB using the user `drive` and authenticate using the initial, default, password you just configured:

```
$ mysql -u drive -p
```

If the new user has been properly configured, you will once again be greeted by the MariaDB interactive shell:

```
$ mysql -u drive -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 39
...
MariaDB [(none)]>
```

Using a non-default MariaDB configuration

The use of MariaDB on a different host as well as the use of a different user name and/or password requires updating the HCL Detect configuration file `HCL/etc/drive.json`.

This JSON file can be edited using a regular text editor.

To change the password used by the HCL Detect database server user, it's necessary to first encrypt the new password so it's not stored in clear text in the configuration file. HCL Detect includes a tool that can be used to perform this operation:

```
$ $HCL_HOME/bin/password_encryptor -p <new_password>
```

Once you have the newly encrypted password, modify the `drive|database|{userName,userPassword}`, and, if necessary, the `solution|database|{userName,userPassword}` entries in the configuration file with the new password as well as a new user name.

As mentioned, some HCL Detect installations have optional features (i.e., a solution) that also make use of a relational database.

Note that the configuration file is nested, so the notation used in the prior sentence means, e.g., the entry `userName` as well as the entry `userPassword` are under the entry `database`, which is under the `solution` entry.

Note that if you are changing one or both of these settings, MariaDB itself must be updated with this new user/password information.

To change the host and port the MariaDB server uses, locate and update one or more of the following entries: `drive|database|address` and, if necessary, the `solution|database|{userName,userPassword}`. The address is a tuple with the following format: `hostname:port`, e.g., `foo.HCL.com:3306`.

Configuring Redis for use by HCL Detect

HCL Detect uses [Redis](#), an in-memory data structure store.

You might be required to address certain OS-level requirements to ensure that Redis runs efficiently. As the [Installation Process on page 9](#) is carried out, you might face warnings similar to the following:



Warning:

The operating system in this host is not optimally configured to run Redis, details:

- `overcommit_memory` is set to 0. Redis background save process may fail under low memory conditions. To fix this issue add `'vm.overcommit_memory = 1'` to `/etc/sysctl.conf` and then reboot or run the command `'sysctl vm.overcommit_memory=1'` for this to take effect. For more details, please refer to <http://redis.io/topics/admin>
- when Transparent Huge Pages (THP) support is enabled in the Linux kernel (as is the case now), it can lead to high latency when Redis forks to persist data to disk. THP support can be disabled by executing the command `'echo never > /sys/kernel/mm/transparent_hugepage/enabled'` as root, and adding it to your `/etc/rc.local` in order to retain the setting after a reboot. When making it permanent, make sure that the `rc.local` file has the execution permission for the owner (if this file is a symlink, ensure that the permission is set for actual file being linked). For more details, please refer to <http://redis.io/topics/admin>



- unable to enforce the TCP backlog settings of 511 yet-to-be-listened TCP connections in Redis. To fix this issue add 'net.core.somaxconn=511' to /etc/sysctl.conf and then reboot or run the command 'sysctl -w net.core.somaxconn=511' for this setting to take effect immediately, but only until the next reboot. For more details, please refer to <http://redis.io/topics/admin>

Ensure that, if/when these errors are displayed, the changes suggested by the error messages are put in place.

Installation Process

Installation Package

The HCL Detect installation tarball includes:

- The HCL Detect software platform itself, comprising all of the necessary components to run HCL Detect-supported applications.
- The pre-configured Python `virtualenv` environment, comprising all Python dependencies required by HCL Detect applications to run.
- The external open source software required by HCL Detect, e.g., Apache Tomcat, Apache Kafka, among others.

Installation Process

The installation process must be carried out using the userid that will manage the HCL Detect software.

It is recommended that this user be named `drive` or, if using a hybrid HCL Detect/InfoSphere Streams environment, `streams`.

Prior to starting the actual installation, if HTTPS-secured web access to HCL Detect will be made available, a (optional) DNS entry must be configured to provide a user-friendly URL to end users as well as a self-signed or commercial SSL certificate must be on-hand as it will be required to complete the product installation.

It might be helpful to become familiar with the infrastructure used to provide HTTPS access to Detect by reading the steps outlined in the section on web proxy configuration before attempting the installation steps.

The installation process begins by un-tarring the software tarball:

```
$ tar xvfz drive-2.2.0-rhel07.tar.gz
```

The suffixes `2.2.0` denote the version you are installing and `rhel07` the specific operating system (`rhel07` for RedHat 7).

Extract the software:

```
$ mkdir -p /opt/HCL/drive/rhel07
$ tar xzvf drive-2.2.0-rhel07.tar.gz -C /opt/HCL/drive/rhel07
```

Again, other locations are acceptable, but `/opt/HCL/drive/rhel07` is the recommended path. Now, you are ready to perform the configuration steps:

```
$ cd /opt/HCL/drive/rhel07/HCL/bin
$ ./installer
```

`installer` is an interactive program and will guide you through specific installation and configuration choices:

```
HCL Detect is a commercial product, subjected to End-User License Agreement terms. A paper-based or digital copy of these terms must have been signed and agreed by someone authorized to do so in your organization, prior to carrying out this configuration. A non-customer specific copy of these terms is included for your reference in this installation package (HCL/license/eula.pdf). Do you confirm that you are authorized to proceed with the configuration based on the terms specified in your organization's own license agreement with HCL Inc. (y/n)?
```

As a first configuration step, you will be asked about the directory that will be used to host the HCL Detect instance, i.e., the location in the file system that HCL Detect will use to host its services their logs as well as the data that will be kept by the HCL Detect data management services.

Next, you will be asked which network interface to use for external TCP/IP traffic. You should choose the interface that provides connectivity to other hosts in the cluster (if any) and/or external services the HCL Detect application will interface with:

```
Please select the network interface to use for external TCP/IP traffic
(default: 'eth0'):

[0] lo: 127.0.0.1
[1] eth0 10.0.0.123
[2] tun0: 10.8.0.45

Selecting the number corresponding to the interface you want (default: '1' for interface 'eth0'): 1

The 'eth0' network interface will be used for all external TCP/IP traffic.
```

Once a network interface is selected, the installer will proceed to ask which transport protocol should be used between HCL Detect's web-based frontend and its backend services:

```
Do you want to use HTTPS-based interactions between the HCL Detect browser-based interface and its backend
([n]o: HTTP will be used (in production, a proxy such as nginx must be used to secure the client/server
communication via HTTPS); / [y]es: HTTPS will be used with a self-signed SSL certificate (recommended only for
short-term testing) (default: HTTP) (y/n)?
```



Warning:

HCL Detect has access to and handles potentially sensitive information:

- **Authentication information:** the use of HCL Detect requires a user account. HCL Detect has its own directory of users or can, alternatively, defer to an enterprise-wide LDAP server. In either case, user passwords are employed to ensure that a user is both authenticated and authorized to use the system. While HCL Detect never stores user passwords in the clear, certain interactions require the transferring of passwords between the front end, web-based interface to the backend. Hence, encryption (through the use of HTTPS) is STRONGLY recommended.
- **Metadata and structural information about subscriber and corporate data feeds:** HCL Detect carries out analytics employing data that is often private and sensitive. Once, again, interactions between its web-based interface and its backend require manipulating such data and encryption, in the form of HTTPS interactions, is STRONGLY recommended to preserve end-to-end confidentiality.



While HCL Detect is generally hosted in an internal network, never facing non-corporate users, it does integrate with other segments of the enterprise computing environment. HCL recommends that administrators take every possible precaution to protect the integrity and confidentiality of the data consumed and produced by this platform.

There are multiple possible configurations to choose from and each one has certain advantages and risks:

- HTTP (without a proxy such as nginx securing the client/server interactions), available network-wide (STRONGLY DISCOURAGED): this is the simplest form of installing HCL Detect. Nevertheless, it is **insecure** as potentially sensitive information is transmitted in the clear over the network, flowing from the user's browser to the server without any encryption, including passwords (potentially, even the ones used for authenticating via enterprise-wide repositories such as a corporate LDAP server, if such an integration is eventually enabled).



Note:

SSL certificates

An SSL certificate is a data file that digitally binds a cryptographic key to an organization's identity. For instance, when installed on a web server, it activates the padlock used by the browser to indicate a `_secure_` connection and, hence, the HTTPS protocol in interactions between the browser and a server.

In general, an SSL certificate is obtained from a Certificate Authority (CA) and it attests the ownership of a public key by the named subject of the certificate, providing an assurance that an interaction is occurring between a client and a properly identified server. A CA acts as a third party, trusted both by the subject (owner) of the certificate and by the party relying upon the certificate.

The HCL Detect web service can make use of an SSL certificate to ensure that interactions between the web-based client and server are properly authenticated (i.e., to provide an assurance to the browser-based client that it is indeed speaking to an actual server) as well as encrypted, such that no sensitive information flows over between a client and server in clear text form.

SSL certificates can be purchased from several vendors or obtained for free from organizations such as [Let's Encrypt](#). Commercial SSL certificates are typically verified and accepted by mainstream web browsers such as Google Chrome and Mozilla Firefox.

SSL certificates can also be provided by any entity hosting a Public Key Infrastructure (PKI). For instance, an organization's IT department might host its own internal PKI and issue self-signed certificates. These certificates work just like commercial certificates, but they will typically produce a warning or an outright rejection from web browsers, which will not recognize the PKI's CA. In such cases, upon being directed by the organization's IT department, the certificate may be added to the browser and accepted as legitimate, thus quieting down the warnings that would, otherwise, be raised every time a web server presenting it is accessed.



HCL Detect can be configured with either type of certificate, but HCL strongly recommends that a certificate be obtained from an officially recognized commercial or non-profit CA.

- HTTPS, available network-wide (RECOMMENDED for testing only): this configuration is deemed safe, as it minimizes the chances of a sensitive data breach. In such a configuration, the interactions between the browser-based user interface and HCL Detect's backend is carried out via HTTPS interactions (encrypted). In this case, the HCL Detect backend service (hosted by Apache Tomcat) is the direct destination of calls performed by a user's browser to the server servicing the REST APIs.

The disadvantages of this approach are two-fold. First, it employs a self-signed certificate generated by HCL and not a commercial certificate issued by a proper Certification Authority. Second, the URL to access HCL Detect will be in the form `https://drive.company.com:<port>/drive`. In other words, the `port` where the service runs will be part of the URL. Typically, HTTPS servers bind and run on the privileged port 443 and such a port number can be omitted from the URL. While the Apache Tomcat-based Detect backend can be configured to use port 443, this configuration is not recommended since it requires running the web server as `root`, which opens up the possibility of a complete takeover of the host where the web server runs should an unknown (but possible) security vulnerability be exploited.

Alternatively, it is also possible to fiddle with the Operating System settings to allow non-`root`-owned applications to use a privileged port. Nevertheless, such a configuration is both non-standard and complex from a system administration standpoint.

If this configuration is chosen, the `installer` will ask for a non-privileged port to be used by the web server and will automatically create a certificate authority, will issue a CA certificate, and install a self-signed SSL certificate to be used by the web server as part of the installation process.

- HTTP, available only in the `localhost` interface, proxied by an HTTPS web proxy (STRONGLY RECOMMENDED): this configuration is deemed the safest, as it minimizes the chances of a sensitive data breach. As with the prior configuration, the interactions between the browser-based user interface and the web proxy in front of HCL Detect's backend are HTTPS-encrypted.

The web proxy (both Apache `httpd` and `nginx` are supported) employs a regular local HTTP connection to the HCL Detect backend and a client will interact with the proxy via HTTPS, which will in turn relay the requests to the web server.

While interception of end-user communication with the web server is possible, this can only occur when the host where the web server is installed is compromised and `root` access is available to the malicious party.

The benefit of this approach is that the web proxy (which is specifically hardened for this task), not HCL Detect's backend, runs as `root`. Furthermore, the web proxy's internal design is optimized for these types of interactions, the use of commercially-issued or locally-issued SSL certificates, specific SSL ciphers, as well as many other SSL-related configurations much easier to put in place.

In this configuration, the installation of an SSL certificate as well as the configuration of the HTTPS endpoint is done by installing and configuring the web proxy, a step described in the web proxy configuration section.

Finally, an HCL Detect installation can be optionally configured with additional integration points to external software packages including IBM SPSS Modeler Solution Publisher, IBM InfoSphere Streams, as well as Oracle WebLogic Server. Each of these integration points requires a prior licensed installation for each of these software packages. When installing an HCL Detect version enabled with one or more of these integration points, additional installation steps will take place. These steps, applicable only to the specific integration points enabled in the installation, will be carried out by the `installer` program to ensure that it can find the proper version for each of the external packages needed by them. For instance:

```
Please enter the path to an existing IBM SPSS Modeler Solution Publisher
installation to use: /opt/x86_64/ibm/spss/ModelerSolutionPublisher/17.1

Please enter the path to an existing IBM InfoSphere Streams
installation to use: /opt/rhel07/ibm/streams/4.0.1.0

ERROR: failed to validate IBM InfoSphere Streams
installation at '/opt/rhel07/ibm/streams/4.0.1.0'
Product information file '/opt/rhel07/ibm/streams/4.0.1.0/.product' does not exist
Please provide a valid path to an existing IBM InfoSphere Streams installation

Please enter the path to an existing IBM InfoSphere Streams
installation to use: /opt/rhel07/ibm/streams/4.0.1.0

Please enter the path to an existing Oracle WebLogic Server
installation to use: /opt/HCL/drive/noarch/weblogic-10.3.6.0
```

If everything is correctly configured, a success status message will be printed out:

```
HCL Detect environment was successfully installed...
```

Now that the configuration is complete, a test can be executed. To run any HCL Detect application, the shell where the application is going to run must be configured by invoking the `environment_setter` script (`<drive-install-location>/HCL/bin/environment_setter`):

```
$ ./environment_setter
```

Once the script is source'd, several environment variables are configured and the Python `virtualenv` environment is activated, indicating that we are ready to start an HCL Detect application:

```
[drive]]$ env | sort | grep HCL
HCL_ARCH=x86_64
HCL_HOME=/opt/HCL/drive/rhel07/HCL
HCL_JAVA_PATH=/usr/lib/jvm/java
HCL_OS=rhel07
HCL_PACKAGE_VERSION=2.2.0
HCL_PYTHON_PATH=/opt/HCL/drive/307/pyenv
HCL_READLINK=readlink
HCL_TOMCAT_PATH=/opt/HCL/drive/noarch/apache-tomcat-9.0.20
```

The preparation of the shell must be made for every shell and session where an HCL Detect application will run. Now, we can run a test application (`<drive-install-location>/HCL/bin`):

```
[drive]]$ cd HCL/bin
[drive]]$ ./installation_tester -c -sns
INFO: this application is using '/tmp/<user>/HCL' to output and store its
code-generated assets...
```

**Note:**

We are invoking the test application `installation_tester` using the `-c` flag to indicate that the application should be re-built (just in case, it has been executed before), thus ensuring that a fresh version, based on the current installation, is executed. Normally, HCL Detect applications are only rebuilt when needed and taking such a precaution is normally not necessary.

Configuring HCL Detect's Runtime Environment

The HCL Detect runtime environment can be configured based on the specific performance and reliability requirements of a customer deployment. These runtime environment settings are specified in the following file: `$HCL_HOME/etc/runtime_environment.json`.

An important runtime configuration is the persistence behavior of PinPoint, which is the Redis-based distributed in-memory store used by Detect to maintain subscriber profiles. As a primary mechanism, PinPoint relies on append-only logging to persist its data. As an additional mechanism, it also periodically checkpoints its data to disk.

The checkpointing can be configured using the `pinPoint | server | checkpointConfiguration | schedules` setting inside the `runtime_environment.json` file. The default checkpoint configuration is given as follows:

```
"schedules":
[
  {
    "operationCount": 1,
    "timeInSeconds": 86400
  },
  {
    "operationCount": 1000000,
    "timeInSeconds": 3600
  }
]
```

This configuration indicates that the state is checkpointed every hour if at least 1 million profiles are updated since the last checkpoint, and every day if at least one profile is updated since the last checkpoint. More schedules can be added or the existing schedules can be updated, as needed. Since the append-only logging already provides persistence, specifying frequent checkpointing is unnecessary and can adversely affect the CPU usage of the PinPoint servers.

Configuring web proxy

HCL Detect consists of a set of backend services, accessible via a web-based user interface.

Once installed, an instance of HCL Detect can be started. The communication between the web-based user interface and the backend employs REST APIs over HTTP or HTTPS.

HCL strongly recommends that HTTPS be used to ensure that the sensitive information that transits over the network is encrypted.

As seen in the [Installation Process on page 9](#) section, HCL Detect offers the choice of web transport protocols during its installation process and recommends HTTPS with the use of a web proxy as the means to access its private backend services.

This option requires installing HCL Detect with HTTP support, where the backend is bound only to the local `lo` network interface, allowing HTTP access only within the host where the HCL Detect web server runs.

Either of two commonly used web proxies, packaged and available in the Linux distributions supported by HCL Detect, can be configured to provide the HTTPS-based access to the users interacting with HCL Detect via their web browsers: [Apache httpd](#) and [nginx](#).

In either case, two steps must be carried out prior to the web proxy installation and configuration. These steps and the specific configuration of a web proxy are outlined next.

Configuring a DNS CNAME entry to point to HCL Detect (optional)

Ideally, the URL used to access HCL Detect will be in the form `https://HCL.acme.com/drive`, where `HCL` designates this locator as an HCL product installation at `acme.com` (the customer's Internet domain) and the context path, `drive`, indicates the name of the product.

Usually, the friendlier `HCL` name will map to an internal server hosting the HCL Detect installation, whose name will follow an (often less friendly) internal IT convention (e.g., `drive-cluster003-node001.acme.com`). Thus, we recommend that a DNS alias (specifically a `CNAME` entry) be obtained and registered prior to the HCL Detect installation.

Procedures to update the corporate DNS servers vary. We recommend that the system administrator installing Detect contacts a local IT specialist to understand the process and to carry out the actual DNS registration.

As an example, an organization using TinyDNS will need to add the following entry to its configuration:

```
CHCL.acme.com:drive-cluster003-node001.acme.com:120
```

And invoke the utility (`tinydns-data`) to activate this entry.

Obtaining an SSL certificate

SSL certificates come in different forms. A certificate can be obtained for a single host name, for multiple host names, or as a wildcard, accepting any name under a particular domain. While all of them ought to work with HCL Detect, a single-host certificate is sufficient and this option, if procuring a commercial certificate, is often the most economical.

Procedures to obtain a certificate vary both for commercial as well as for self-signed internal certificates.

Please consult a local IT specialist to understand which alternative is the most adequate in your environment.

In the rest of this document, it is assumed that a commercial certificate is on hand and available to be used during the configuration of the web proxy.

Regardless of how a certificate is obtained, in the end, a set of files related to the certificate must be available. In the example below, we are assuming that the certificates are being kept in a directory called `HCL.acme.com`:

- `HCL.acme.com/fullchain.pem`: this is the actual certificate plus the intermediate certificates (if any). See the documentation [the Apache httpd SSLCertificateFile configuration key](#) and the [the nginx ssl_certificate configuration key](#) for more information.
- `HCL.acme.com/privkey.pem`: this is the certificate private key for the server. See the documentation [the Apache httpd SSLCertificateKeyFile configuration key](#) and the [the nginx ssl_certificate_key configuration key](#) for more information.

Once the certificate has been obtained and these files are available either the Apache `httpd` or `nginx` must be installed and configured.

In the following sections, we will describe these steps assuming that the installation is being made on a RedHat server, where the HCL Detect web backend will eventually run.

Since the installation and configuration require updating system-owned resources, either `sudo` or `root` access is required.

Configuring Apache `httpd`

The installation requires downloading and installing the necessary OS packages:

```
$ yum install -y httpd mod_ssl
```

To ensure that the Apache `httpd` server is started on boot, the service must be enabled:

```
$ systemctl enable httpd
```

Once installed, the configuration enabling the proxying must be added to `httpd`'s main configuration file `/etc/httpd/conf/httpd.conf`. The following excerpt demonstrates how this is done, by creating a new virtual host entry:

```
...
<VirtualHost *:443>
    ServerName HCL.acme.com
    ## HCL Detect will accessible at https://HCL.acme.com/drive
    ProxyPass /drive http://[::1]:8081/drive
    ## HCL Detect must be configured with HTTP access restricted to localhost binding to port 8081
    ## Note that the IPv6 address being used, [::1], is the IP address corresponding to the loopback interface.
    ## HCL Detect's Tomcat instance by ## default will use the IPv6 protocol stack
    ProxyPassReverse /drive http://[::1]:8081/drive
    ProxyPreserveHost on
    ProxyRequests off
    Header always set Strict-Transport-Security "max-age=31536000; includeSubdomains; preload"
    SSLCertificateFile /etc/httpd/stash/HCL.acme.com/fullchain.pem
    SSLCertificateKeyFile /etc/httpd/stash/HCL.acme.com/privkey.pem
    SSLEngine on
    SSLProtocol TLSv1.2
    SSLProxyEngine on
</VirtualHost>
...
```



Note:



In our configuration, we chose to install the certificate-related files under `/etc/httpd/stash`. This is not necessary, but is convenient as far as applying the correct SELinux context to these files, which is carried out as follows:

```
$ restorecon -vr /etc/httpd/stash
```

Once this is done, `httpd` can be started (or restarted):

```
$ systemctl start httpd
```

If all is well, you should be able to start HCL Detect, open a browser and point it to `https://HCL.acme.com/drive`.

To start HCL Detect, see the [Starting and stopping HCL Detect on page 21](#) section. If a failure has happened when starting `httpd`, see the [Testing and troubleshooting the web proxy installation on page 18](#) section for additional help with troubleshooting.

Configuring `nginx`

The installation requires downloading and installing the necessary OS packages:

```
$ yum install -y nginx
```

To ensure that the `nginx` server is started on boot, the service must be enabled:

```
$ systemctl enable nginx
```

Once installed, the configuration enabling the proxying must be added to `nginx`'s main configuration file `/etc/nginx/nginx.conf`. The following excerpt demonstrates how this is done, by creating a new server entry:

```
http {
...
    server {

        listen 443;
        server_name HCL.acme.com;

        add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload";

        ssl on;
        ssl_certificate /etc/nginx/stash/HCL.acme.com/fullchain.pem;
        ssl_certificate_key /etc/nginx/stash/HCL.acme.com/privkey.pem;
        ssl_session_timeout 5m;
        ssl_protocols TLSv1.2;
        ssl_ciphers HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers on;

        ssl_session_cache shared:SSL:10m;
        ## HCL Detect will accessible at https://HCL.acme.com/drive
        location /drive {

            ## HCL Detect must be configured with HTTP access restricted to localhost binding to port 8081
            proxy_pass http://[::1]:8081/drive;
            ## Note that the IPV6 address being used, [::1], is the IP address corresponding to the loopback
            ## interface. HCL Detect's Tomcat instance by default will use the IPV6 protocol stack
            proxy_set_header X-Forwarded-Host $host;
            proxy_set_header X-Forwarded-Server $host;
```

```

    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_redirect off;
}
}
}

```

**Note:**

In our configuration, we chose to install the certificate-related files under `/etc/httpd/stash`. This is not necessary, but is convenient as far as applying the correct SE Linux context to these files, which is carried out as follows:

```
$ restorecon -vr /etc/nginx/stash
```

Once this is done, `nginx` can be started (or restarted):

```
$ systemctl start nginx
```

If all is well, you should be able to start HCL Detect, open a browser, and point it to `https://HCL.acme.com/drive`.
troubleshooting.

To start HCL Detect, see the [Starting and stopping HCL Detect on page 21](#) section. If a failure has happened when starting `httpd`, see the [Testing and troubleshooting the web proxy installation on page 18](#) section for additional help with troubleshooting.

Testing and troubleshooting the web proxy installation

Once the web proxy is configured and HCL Detect has been started, using the web interface is as simple as entering the access URL in the browser window.

There are typically two problems that might occur in a new installation, where either RedHat Linux or CentOS is being used:

- the firewall configuration: if the server running the proxy has `firewalld` installed, HTTPS access is typically blocked by default. While we provide some helpful directions below, we strongly recommend reading the `firewalld` documentation so the impact of these commands is fully understood.

To check on current status of `firewalld`, the following command can be executed:

```

$ firewall-cmd --list-all
public (active)
target: default
icmp-block-inversion: no
interfaces: enp0s3
sources:
services: dhcpv6-client https ssh
protocols:
masquerade: no
forward-ports:
sourceports:
icmp-blocks:
rich rules:

```

If `https` does not appear in the list of `services` (as it does above), it must be added, either temporarily:

```
$ firewall-cmd --zone=public --add-service=https
$ firewall-cmd --reload
```

Or permanently:

```
$ firewall-cmd --zone=public --permanent --add-service=https
```

At this point, re-running the status command will reveal whether HTTPS access to the server is now enabled.

- the SE Linux configuration SELinux is a set of kernel modifications and tools that have been added to RedHat and CentOS, providing support for access control security policies. It may affect the web proxy in the sense.

Often, if the SELinux access control is not properly configured for the web proxy, there will be two symptoms. First, an access from the browser will be rejected, often, with a "bad gateway" error message. Second, the SELinux log file (`/var/log/audit/audit.log`, `root` access is required both for search the logs as well as for reconfiguring SELinux policies) will include a rejection of an operation attempted by the web proxy (in the example below, `nginx`, but very similar for `httpd` as well), possibly, similar to the following:

```
$ grep nginx /var/log/audit/audit.log
type=AVC msg=audit(1490570804.119:555): avc: denied { name_connect } for pid=5725 comm="nginx" dest=8080
scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:http_cache_port_t:s0 tclass=tcp_socket
```

Fiddling with SELinux will have a direct impact on the security of a host as well as on the overall network where that host is located. HCL strongly recommends that changes being done on SELinux policies are undertaken by someone familiar with its configuration. The procedures we indicate below are not to be taken without understanding their potential repercussions.

This audit entry can be better understood by running the following command, which also prescribes the possible fixes:

```
$ ausearch -c nginx | audit2allow -m nginx

module nginx 1.0;

require {
    type httpd_t;
    type http_cache_port_t;
    class tcp_socket name_connect;
}

#===== httpd_t =====

#!!!! This avc can be allowed using one of the these booleans:
#     httpd_can_network_connect, httpd_can_network_relay
allow httpd_t http_cache_port_t:tcp_socket name_connect;
```

You can check the current state of these Booleans settings as follows, which at this point, is most likely `off`:

```
$ getsebool -a | grep httpd
```

Finally, the problem can be resolved by allowing network connections from the web proxy to the actual server (i.e., so it can act as a relay):

```
$ setsebool -P httpd_can_network_relay on
```

Alternatively, a new SELinux non-base policy can be put in place:

```
$cd /tmp  
$ ausearch -c nginx | audit2allow -M nginx  
$ semodule -i nginx.pp  
$ rm nginx.pp
```

Chapter 2. Running HCL Detect

Starting and stopping HCL Detect

Starting HCL Detect requires two steps.

First, we must ensure that the environment of the current shell is properly configured. This is accomplished by *invoking* the script that performs this task:

```
$ ./<installation-location>/drive/bin/environment_setter
```



Note:

Modifying the PATH environment variable in the user's .bashrc file

The `environment_setter` script loads the user's `.bashrc` file (if it exists) and, subsequently, validates that the execution environment is not modified inadvertently with respect to the needs of HCL Detect.

Specifically, the `environment_setter` validates that the executables for the Java Virtual Machine (`java`) and the Python interpreter (`python`) are consistent with the ones needed by HCL Detect.

In case an inconsistency is flagged due to a modification of the user's `PATH` environment variable in `.bashrc`, an error message will be emitted and two corrective alternatives are possible.

If it happens to be an irrelevant/unintended misconfiguration, you can simply eliminate it from your `.bashrc` file.

If the original behavior needs to be preserved, this can be accomplished by conditionally making the following modification to the `PATH` environment variable:

```
if [[ -z ${HCL_PRODUCT:-} ]]; then
    export PATH=<path_to_custom_python>:<path_to_custom_jdk>:$PATH
fi
```

`HCL_PRODUCT` is an environment variable defined by HCL Detect in the new instance of the Bash shell the `environment_setter` starts.

By checking for this environment variable and conditionally updating the `PATH` variable only when it is not defined, the conflicting settings will not affect HCL Detect's runtime environment. Yet, when a regular shell is instantiated, those `PATH` settings will still remain active.

Next, we start all of the HCL Detect services:

```
$ <installation-location>/drive/bin/services_manager start
```

At this point, it is possible to open a browser on any computer with access to the server where HCL Detect is installed and access its web user interface by pointing the browser to the appropriate URL: `https://HCL.acme.com/drive`.

If everything has worked correctly, you will be presented with the initial setup workflow, whereby an administration password will be configured as well as other initialization steps.

If you cannot access the HCL Detect UI, the [Testing and troubleshooting the web proxy installation on page 3](#) section provides a few troubleshooting instructions that will help you ensure that your configuration is correct.

Finally, we can stop the HCL Detect services:

```
$ <installation-location>/drive/bin/services_manager stop
```

Advanced options when starting and stopping Detect

HCL Detect can be started using the `services_manager` script located at `$HCL_HOME/bin`.

This script can be used to configure various parameters associated with the execution of the HCL Detect platform. A list of available parameters can be obtained as follows:

```
$ cd $HCL_HOME
$ ./bin/services_manager --help

usage: services_manager [-h] [-a] [-c path] [-i id] [-k] [-o [app [app ...]]]
                        [-r] [-p] [-s] [-t] [-u uri]
                        operation

The 'services_manager' application is part of HCL Detect 1.2.0 (build id:
HCLadmin@ubdev.HCL.com - commit id:
465f0374ffc53c20139e54ed9575ef81505c8ca3)

positional arguments:
  operation              specifies the operation to be performed, one of:
                        'start_applications', 'start', 'stop', 'check',
                        'stop_applications'

optional arguments:
  -h, --help            show this help message and exit
  -a, --disable-fastpast
                        indicates that FastPast should not be checked (when
                        checking the status of Detect services), started (when
                        starting the Detect services), or stopped (when
                        stopping the Detect services) (default: False)
  -c path, --drive-config-file path
                        specifies the path of the HCL Detect
                        configuration file (default: None)
  -i id, --instance-id id
                        specifies the instance identifier to use (default:
                        None)
  -k, --disable-kafka
                        indicates that Kafka should not be checked (when
                        checking the status of the services), started (when
                        starting the services), or stopped (when stopping the
                        services) (default: False)
  -o [app [app ...]], --applications-to-run [app [app ...]]
                        specifies the list of Detect applications to use (by
                        default, all of them are used)
  -r, --disable-pinpoint
                        indicates that PinPoint should not be checked (when
                        checking the status of the services), started (when
                        starting the services), or stopped (when stopping the
                        services) (default: False)
```

```

-p, --disable-campaign-actuator
    indicates that Campaign Actuator should not be checked
    (when checking the status of the services), started
    (when starting the services), or stopped (when
    stopping the services) (default: True)

-s, --disable-segment-updater
    indicates that Segment Updater should not be checked
    (when checking the status of the services), started
    (when starting the services), or stopped (when
    stopping the services) (default: False)

-t, --disable-tomcat
    indicates that Tomcat should not be checked (when
    checking the status of the services), started (when
    starting the services), or stopped (when stopping the
    services) (default: False)

-u uri, --instance-uri uri
    specifies the instance URI to use (default: None,
    indicating that a self-managed Name Service will be
    started and used)

```

The user responsible for managing HCL Detect (typically `HCLadmin`) can start HCL Detect using the following command:

```

$ ./bin/services_manager start
Starting Name Service
--- running as process id 27981
--- instance URI is 'redis://10.0.2.15:34549/HCLadmin'
Starting Kafka
--- running Kafka as process id 27991
--- running Zookeeper as process id 27983
Starting FastPast
--- running as process id 27994
--- waiting for FastPast to initialize (done)
Starting PinPoint
--- running as process id 27995
--- waiting for PinPoint to initialize (done)
Starting Tomcat
--- running as process id 28051
--- waiting for HCL Drive to initialize..... (done)
Starting application 'Segment Updater'
--- running as process id 28181
Starting feed applications: 'Topup Demo', 'Mobile Usage'
Starting application 'Topup Demo'
--- running as process id 28193
Starting application 'Mobile Usage'
--- running as process id 28209

```

After its execution, the HCL Detect web-based user interface is accessible by opening the `http://<hostname>:8080/Detect` URL in a browser.

HCL Detect can be stopped by invoking the following command:

```

$ ./bin/services_manager stop
Stopping feed applications: 'Topup Demo', 'Mobile Usage'
Stopping application 'Topup Demo'
--- waiting for application 'Topup Demo' to terminate.
--- stopped process id 28193
Stopping application 'Mobile Usage'
--- waiting for application 'Mobile Usage' to terminate.
--- stopped process id 28209

```

```

Stopping application 'Segment Updater'
--- waiting for application 'Segment Updater' to terminate.
--- stopped process id 28181
Stopping Tomcat
--- waiting for Tomcat to terminate.
--- stopped process id 28051
Stopping PinPoint
--- waiting for PinPoint to shutdown its servers.
--- waiting for PinPoint to terminate.
--- stopped process id 27995
Stopping FastPast
--- waiting for FastPast to shutdown its servers.
--- waiting for FastPast to terminate.
--- stopped process id 27994
Stopping Kafka
--- waiting for Kafka to terminate.
--- stopped process id 27991
--- waiting for Zookeeper to terminate.
--- stopped process id 27983
Stopping the Name Service
--- stopped process id 27981

```

Once the stop sequence is complete, the web-based user interface is not available anymore.

The sample `services_manager` invocation we used earlier starts up all applications available as part of your installation. The HCL Detect administrator user can, of course, make use of the other parameters associated with the script. For instance, in order to start the services only with a single application named `Topup Demo` (assuming it is available as part of your installation), the following command can be used:

```

$ ./bin/services_manager start -o 'Topup Demo'
Starting Name Service
--- running as process id 28730
--- instance URI is 'redis://10.0.2.15:57173/HCLAdmin'
Starting Kafka
--- running Kafka as process id 28741
--- running Zookeeper as process id 28735
Starting FastPast
--- running as process id 28743
--- waiting for FastPast to initialize (done)
Starting PinPoint
--- running as process id 28744
--- waiting for PinPoint to initialize (done)
Starting Tomcat
--- running as process id 28798
--- waiting for HCL Drive to initialize..... (done)
Starting application 'Segment Updater'
--- running as process id 28896
Starting feed applications: 'Topup Demo'
Starting application 'Topup Demo'
--- running as process id 28903

```

While the detailed status of the HCL Detect services can be examined via the web-based user interface, the `services_manager` script can also be used to get a brief status of all services. An example invocation is as follows:

```

$ ./bin/services_manager check
Kafka is UP
Zookeeper is UP

```



```
FastPast is UP
PinPoint is UP
Tomcat is UP
application 'Segment Updater' is UP
application 'Topup Demo' is UP
application 'Mobile Usage' is DOWN
```

While the services are up, the `services_manager` script can be used start and stop individual applications. For instance, the following command can be used to start the 'Mobile Usage' application:

```
$ ./bin/services_manager start_applications -o 'Mobile Usage'
Starting feed applications: 'Mobile Usage'
Starting application 'Mobile Usage'
--- running as process id 29058
```

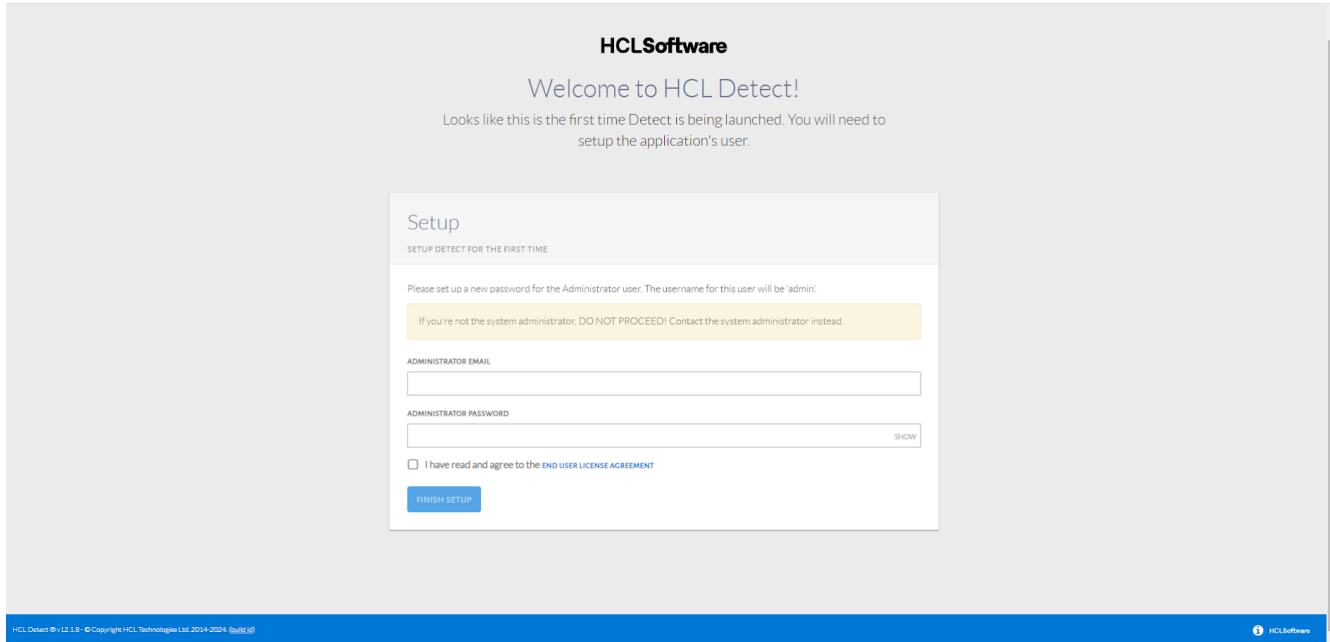
A running application can be stopped in a similar way. For instance, the following command can be used to stop the 'Mobile Usage' application:

```
$ ./bin/services_manager stop_applications -o 'Mobile Usage'
Stopping feed applications: 'Mobile Usage'
Stopping application 'Mobile Usage'
--- waiting for application 'Mobile Usage' to terminate.
--- stopped process id 29058
```

Chapter 3. Accessing the interface for the first time

The preferred browser for accessing the user interface is [Google's Chrome](#).

When accessing the user interface for the first time after the software installation, the user will be presented with a screen to setup the `admin` password as seen below:



The screenshot shows the HCL Software 'Welcome to HCL Detect!' screen. The main heading is 'HCLSoftware' followed by 'Welcome to HCL Detect!'. Below this, a message states: 'Looks like this is the first time Detect is being launched. You will need to setup the application's user.' The central focus is a 'Setup' form titled 'SETUP DETECT FOR THE FIRST TIME'. The form contains the following elements: a heading 'Setup', a sub-heading 'SETUP DETECT FOR THE FIRST TIME', a paragraph 'Please set up a new password for the Administrator user. The username for this user will be 'admin'.', a yellow warning box with the text 'If you're not the system administrator, DO NOT PROCEED! Contact the system administrator instead.', a text input field for 'ADMINISTRATOR EMAIL', a text input field for 'ADMINISTRATOR PASSWORD' with a 'SHOW' button to its right, a checkbox labeled 'I have read and agree to the END USER LICENSE AGREEMENT', and a blue 'FINISH SETUP' button. At the bottom left of the page, there is a footer: 'HCL Detect © v12.1.5 - © Copyright HCL Technologies Ltd. 2014-2024. (b) (d) (s)'. At the bottom right, there is an HCLSoftware logo.

Setting up the `admin` password.

This should only be done by the user responsible for managing HCL Detect.

The `admin` login provides unrestricted administrative access to HCL Detect and, among other tasks, can be used to add additional users.

Chapter 4. Administration Interface

User & Role Management

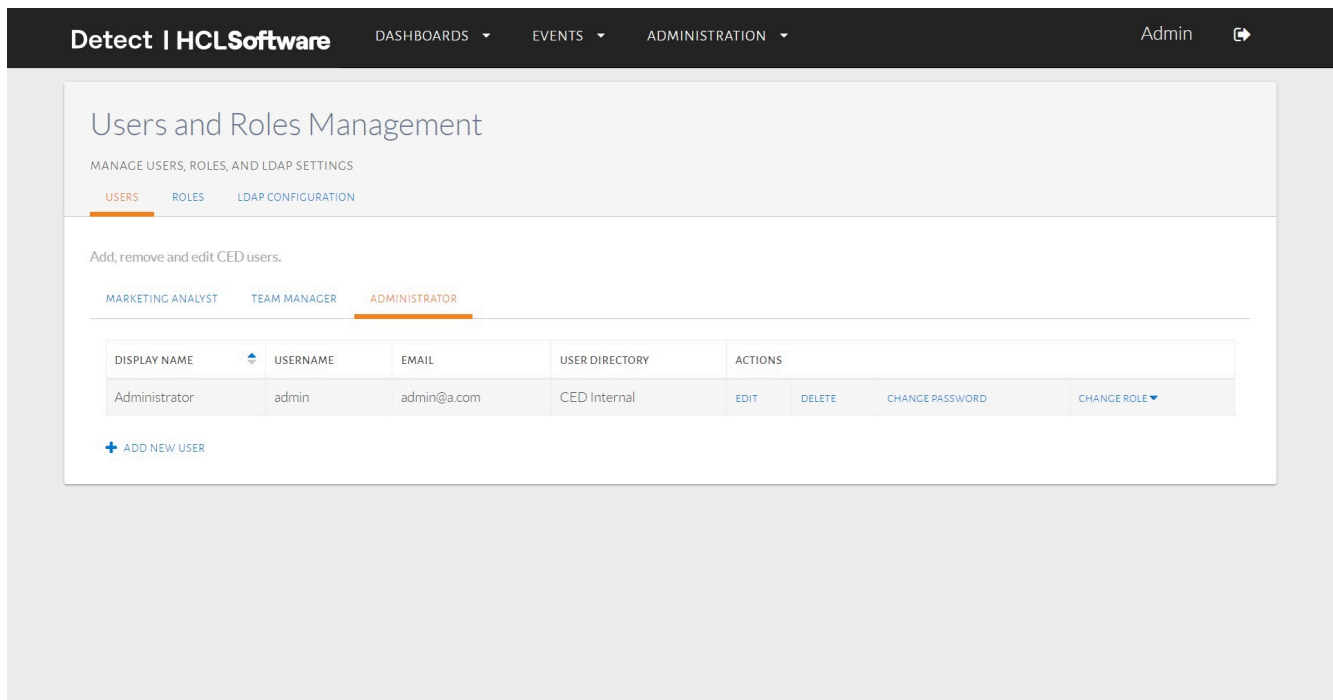
The only user that comes pre-configured with HCL Detect is the `admin` user.

The `admin` user can add additional users, including other administrators, as well as perform other user management tasks by accessing the

User And Role Management

link located under the `Administration` category in the top navigation bar.

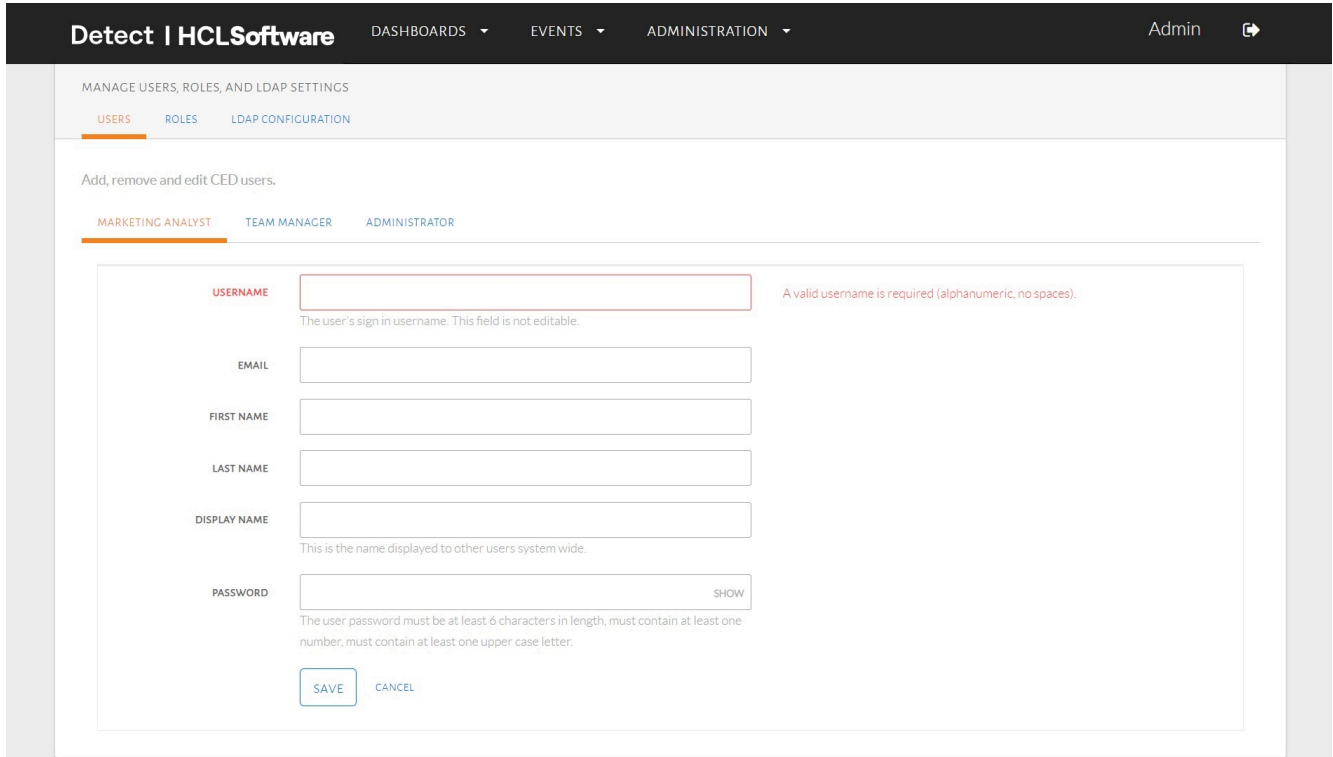
The user management screen is shown below:



The `UserAndRoleManagement` page.

In the `Users` tab, the various roles that are configured in the system and the users that exist for each such role are depicted.

To add a new user, the administrator must first decide the role for such a user, for example, the `MarketingAnalyst` role can be selected by picking the corresponding tab in the `User Management` page and by clicking on the `Add New User` link:



Adding a new user.

Subsequently, the Username, Email, First Name, Last name, Display name and Password attributes should be filled out. Finally, the Save button can be pressed to store this data.

Once completed, the newly added user should be visible in the user interface:

Detect | HCLSoftware DASHBOARDS ▾ EVENTS ▾ ADMINISTRATION ▾ Admin ↗

Users and Roles Management

MANAGE USERS, ROLES, AND LDAP SETTINGS

USERS ROLES LDAP CONFIGURATION

Add, remove and edit CED users.

MARKETING ANALYST TEAM MANAGER ADMINISTRATOR

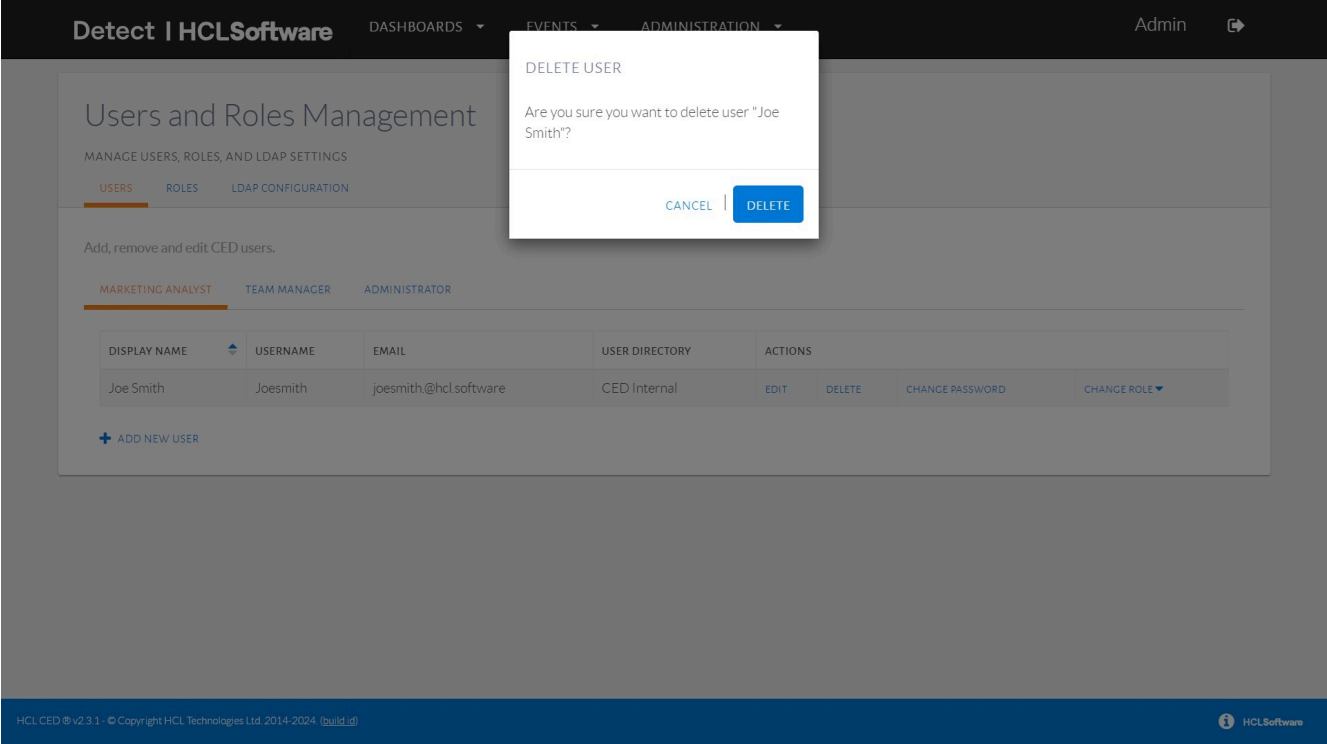
DISPLAY NAME	USERNAME	EMAIL	USER DIRECTORY	ACTIONS
Joe Smith	Joesmith	joesmith@hcl.software	CED Internal	EDIT DELETE CHANGE PASSWORD CHANGE ROLE ▾

[+ ADD NEW USER](#)

A newly added user.

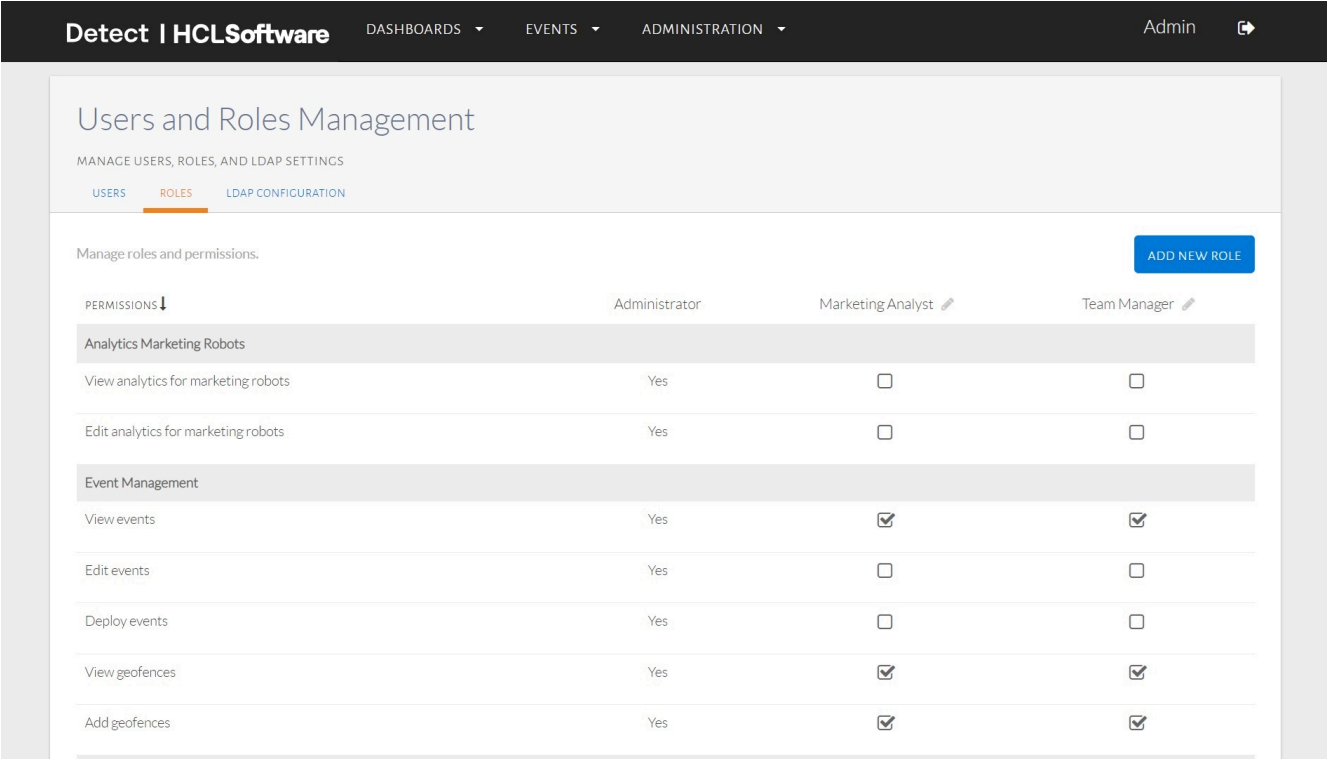
The data for an existing user can be modified and the user itself can be removed by another user with user management entitlement.

The deletion and editing are performed using `Delete` and `Edit` button under `Actions` head of the table:



Removing or updating a user's data.

The Roles page can be accessed by accessing the Roles tab in the navigation bar:



The `Roles` tab.

The HCL Detect, by default, comes with three default roles: the `Administrator`, the `Team Manager` and the `Marketing Analyst`:

- The `Administrator` has unrestricted permissions in HCL Detect.
- The `TeamManager` does not have the permissions associated with user and role management, but has all the other permissions.
- The `MarketingAnalyst` has permissions that are a notch below a `Team Manager`, excluding, for instance, the ability to edit the feeds.

Roles can be renamed and removed.

The removal of a role can only be accomplished if no user(s) with that role exists. The user interface shows an appropriate error when an attempt is made to delete a role that has user(s) assigned to it:

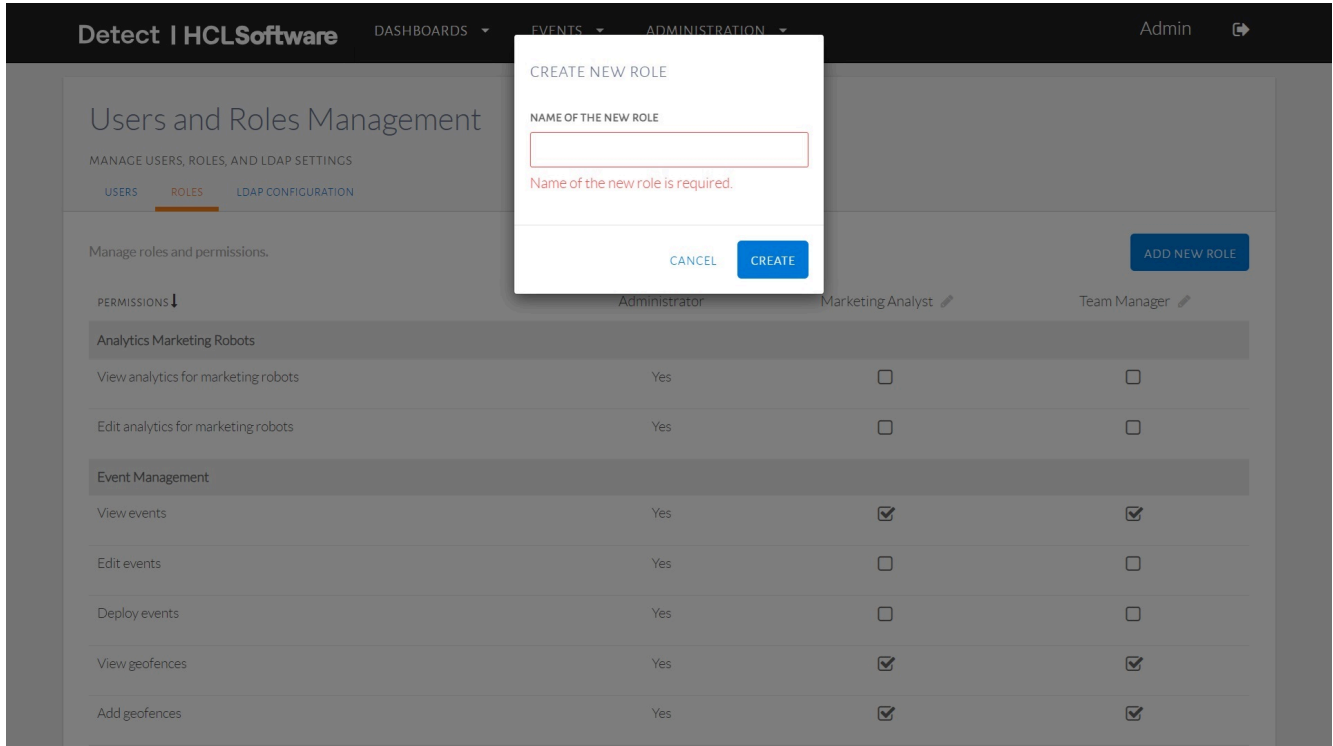
Manage roles and permissions. [ADD NEW ROLE](#)

PERMISSIONS ↓	Administrator	Marketing Analyst	Team Manager
Analytics Marketing Robots			
View analytics for marketing robots	Yes	<input type="checkbox"/>	<input type="checkbox"/>
Edit analytics for marketing robots	Yes	<input type="checkbox"/>	<input type="checkbox"/>
Event Management			
View events	Yes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Edit events	Yes	<input type="checkbox"/>	<input type="checkbox"/>
Deploy events	Yes	<input type="checkbox"/>	<input type="checkbox"/>
View geofences	Yes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Add geofences	Yes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Renaming and deletion options for a role.

A new role can be created by clicking on the `AddNewRole` button located at the top-right corner.

The resulting popup asks the user to enter the name of the new role and, upon clicking on the `Create` button, the corresponding new role is created.



Adding a new role.

A new role is created without any permissions. Permissions can then be added to the newly created role by clicking on the corresponding checkboxes:

Detect | HCLSoftware DASHBOARDS ▾ EVENTS ▾ ADMINISTRATION ▾ Admin ↗

Users and Roles Management

MANAGE USERS, ROLES, AND LDAP SETTINGS

USERS ROLES **LDAP CONFIGURATION**

Manage roles and permissions. ADD NEW ROLE

EDITING IN PROGRESS SAVE DISCARD CHANGES

PERMISSIONS ↓	Administrator	Event Manager ✎	Marketing Analyst ✎	Team Manager ✎
Analytics Marketing Robots				
View analytics for marketing robots	Yes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Edit analytics for marketing robots	Yes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Event Management				
View events	Yes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Edit events	Yes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Deploy events	Yes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
View geofences	Yes	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

A newly created role (Event Manager) with View events permissions only.

HCL Detect can be optionally configured to do LDAP based authentication. The `LDAPConfiguration` page can be used to configure the `LDAP Configuration` tab in the navigation bar:

Detect | HCLSoftware DASHBOARDS ▾ EVENTS ▾ ADMINISTRATION ▾ Admin ↗

Users and Roles Management

MANAGE USERS, ROLES, AND LDAP SETTINGS

USERS ROLES **LDAP CONFIGURATION**

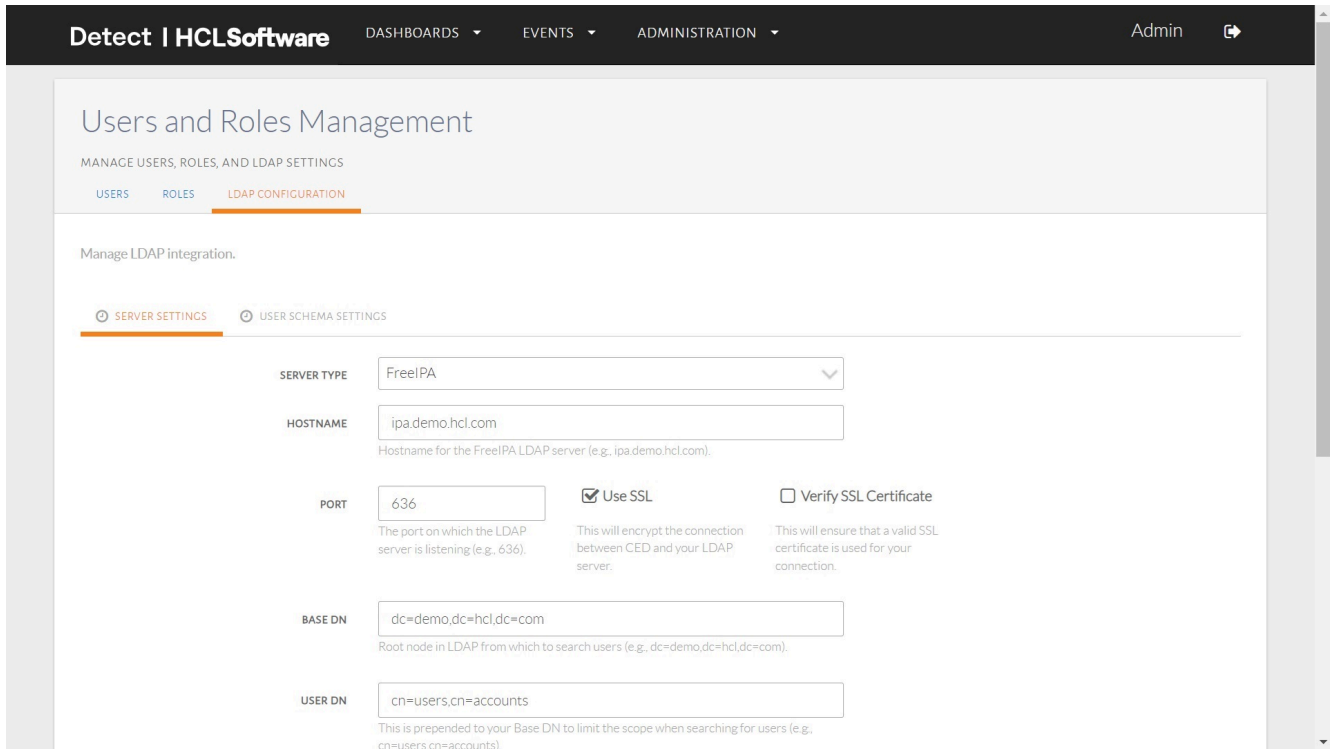
Manage LDAP integration.

LDAP integration has not been set up yet.

CONFIGURE AND ENABLE LDAP

The `LDAPConfiguration` tab.

To enable LDAP based authentication, we need to click on `ConfigureAndEnableLDAP` button:



Configuring the `LDAPConfiguration` tab.

Subsequently, the `ServerType`, `Hostname`, `Port`, `Base DN`, `User DN`, `Group DN`, `Search User Name` and `Search User Password` attributes should be filled out. Current two type are servers are supported, i.e., `Free IPA` and `Active Directory`. Finally, the `Test And Save` button can be pressed to test the server setting:

Detect | HCL Software DASHBOARDS ▾ EVENTS ▾ ADMINISTRATION ▾ Admin ↗

SERVER SETTINGS USER SCHEMA SETTINGS

SERVER TYPE ActiveDirectory ▾

HOSTNAME ipa.demo.hcl.com
Hostname for the ActiveDirectory LDAP server (e.g. ipa.demo.hcl.com).

PORT 389 Use SSL Verify SSL Certificate
The port on which the LDAP server is listening (e.g. 636). This will encrypt the connection between CED and your LDAP server. This will ensure that a valid SSL certificate is used for your connection.

BASE DN dc=unicindia,dc=com
Root node in LDAP from which to search users (e.g. dc=demo,dc=hcl,dc=com).

USER DN cn=Users
This is prepended to your Base DN to limit the scope when searching for users (e.g. cn=users,cn=accounts).

GROUP DN
This is prepended to your Base DN to limit the scope when searching for users (e.g. cn=groups,cn=accounts).

SEARCH USER NAME unicaidia\administrator
This is the user name that will be used for authentication of the user search (e.g. uid=test,cn=users,cn=accounts,dc=demo,dc=hcl,dc=com).

SEARCH USER PASSWORD *****
This is the password that will be used for authentication of the user search.

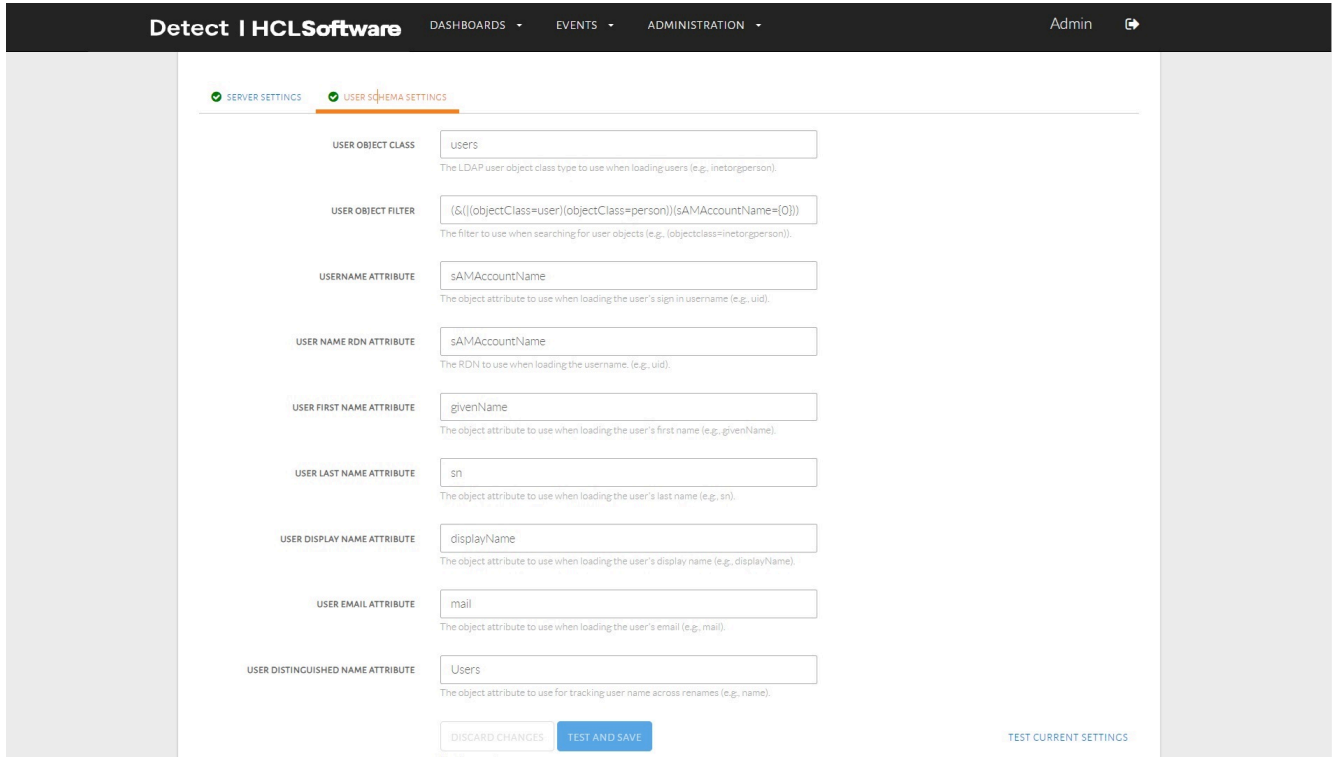
SHOW ADVANCED SETTINGS

DISCARD CHANGES TEST AND SAVE TEST CURRENT SETTINGS NEXT

No changes to save

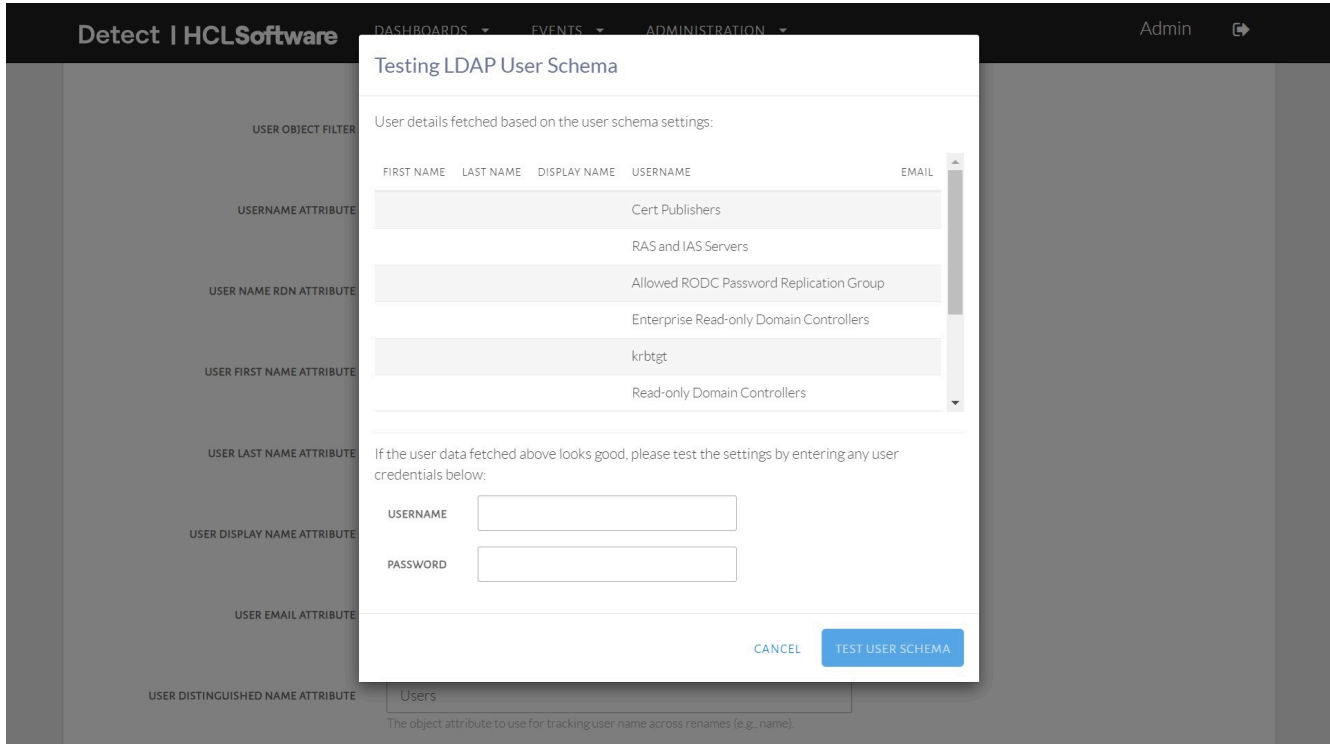
Testing the `ServerSettings`.

Press `Next` Button to start configuring the `User Schema Settings` tab. Now `user Object Class`, `User Object Filter`, `Username Attribute`, `User Name RDN Attribute`, `User First Name Attribute`, `User Lastname Attribute`, `User Display Name Attribute`, `User Email Attribute` and `User Distinguished Name Attribute` attribute needs to be filled based on your organization's LDAP configurations:



Configuring the `UserSchemaSettings`.

Now Click on `TestAndSave` button to test the configuration. Test it by filling `Username` and `Password` for any existing LDAP user and clicking `Test And Save` on the popup display.



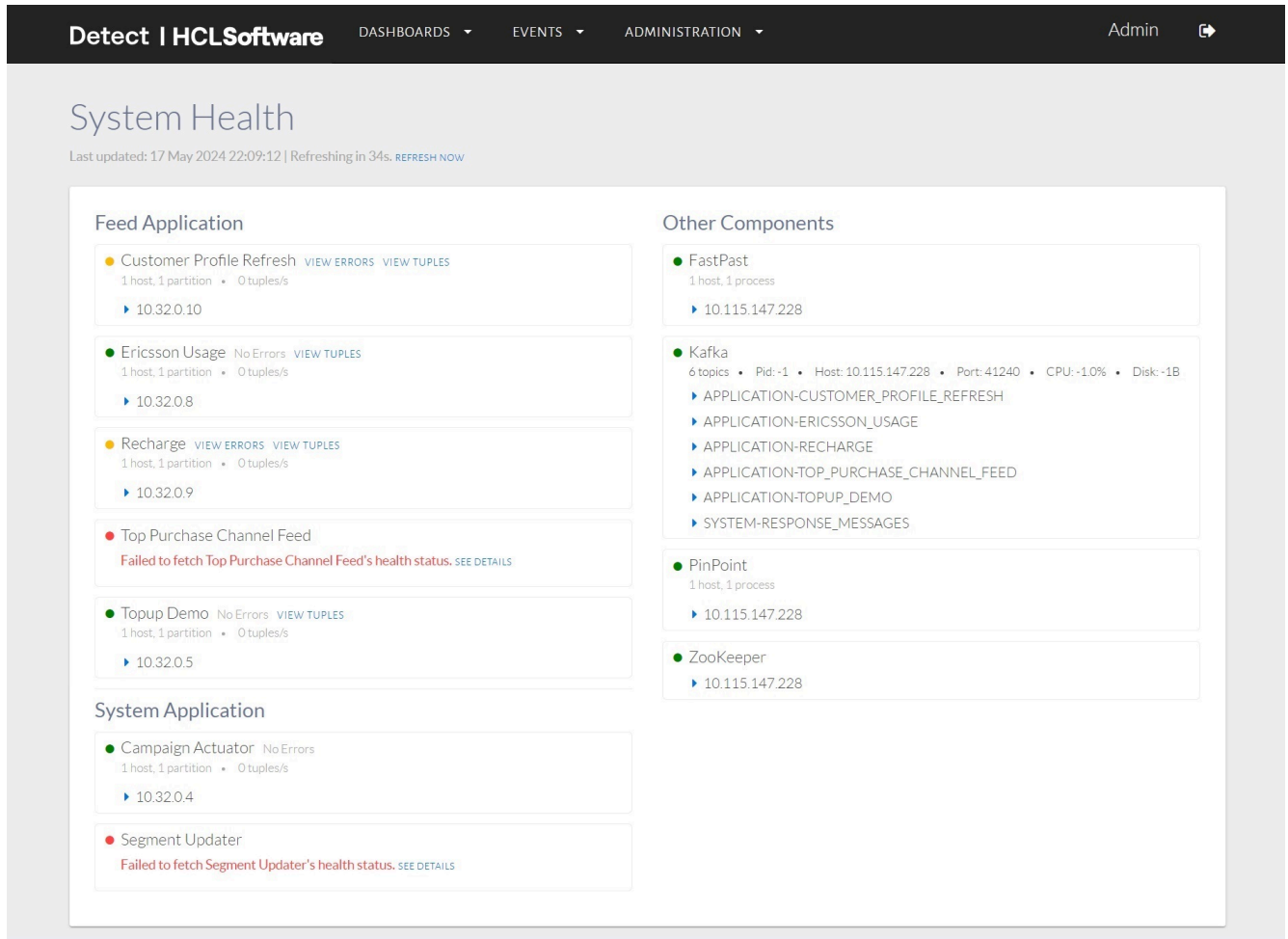
Testing the `UserSchemaSettings`.

System Health

The `SystemHealth` dashboard can be accessed by clicking on the `System Health` link in the top navigation bar.

The dashboard shows the status of the `AnalyticsApplications`, `Feed Applications`, `System Applications` and `Other Components`.

`FeedApplications` are used for ingesting the incoming streaming data, `Analytics Applications` are used for stream or Batch based analytics jobs, `System Applications` are built-in applications used for different important functions of application like detecting events or managing batch segment updates. `Other Components` are the HCL Detect supporting components, including the `FastPast`, the `PinPoint`, the `Kafka` and the `Zookeeper`:



The `SystemHealth` dashboard.

`FeedApplications` can also be used to see the last few data rows/tuples being processed by clicking `View Tuples` button:

Recent Tuples in the Customer Profile Refresh Feed Application Last 10 tuples. REFRESH

EXPAND / COLLAPSE Tuples Show only: attributes Sort Attributes by: NAME / TYPE

Tuple ID	Timestamp
TUPLE 1	17 May 2024 05:33:18
TUPLE 2	17 May 2024 05:33:18
TUPLE 3	17 May 2024 05:33:18
TUPLE 4	17 May 2024 05:33:18
TUPLE 5	17 May 2024 05:44:42
TUPLE 6	17 May 2024 05:44:42
TUPLE 7	17 May 2024 05:44:42
TUPLE 8	17 May 2024 05:44:42
TUPLE 9	17 May 2024 05:44:42
TUPLE 10	17 May 2024 05:44:42

attributesList: [9881153226, 3G, 1, Abigail, Hill, Female, 22, a.adams@randatmail.com, 9881153226, Bachelor, Social Worker, 6, 4, ...]
MSISDN: 9881153226 (String)

DOWNLOAD AS JSON CLOSE

`ViewTuples` from feed applications.

`FeedApplications` also shows the resource utilization of a feeds and parallel processing flow of each feeds.

Feed Application

- Customer Profile Refresh ● VIEW ERRORS VIEW TUPLES
1 host, 1 partition • 0 tuples/s
10.32.0.10
- Ericsson Usage ● No Errors VIEW TUPLES
1 host, 1 partition • 0 tuples/s
10.32.0.8
- Recharge ● VIEW ERRORS VIEW TUPLES
1 host, 1 partition • 0 tuples/s
10.32.0.9
- Top Purchase Channel Feed ●
Failed to fetch Top Purchase Channel Feed's health status. SEE DETAILS
- Topup Demo ● No Errors VIEW TUPLES
1 host, 1 partition • 0 tuples/s
10.32.0.5

System Application

- Campaign Actuator ● No Errors
1 host, 1 partition • 0 tuples/s
10.32.0.4

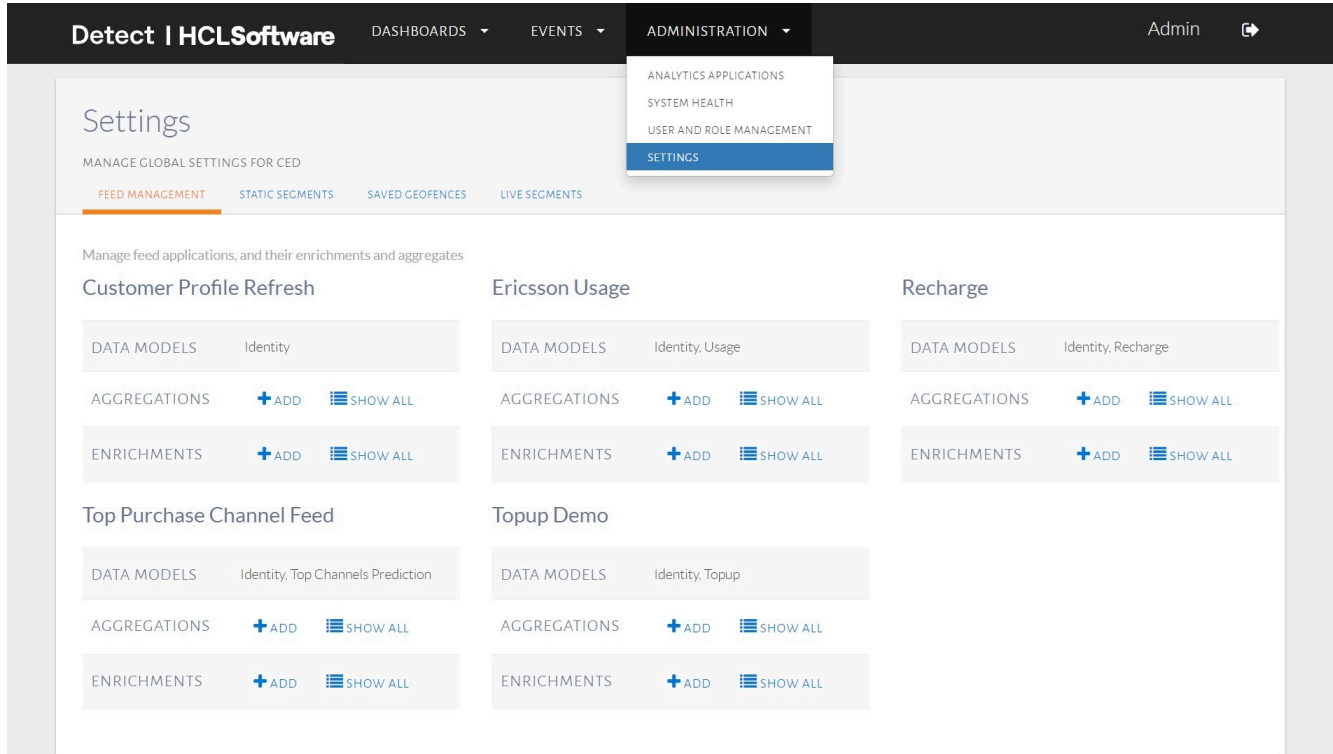
Other Components

- FastPast ●
1 host, 1 process
10.115.147.228
- Kafka ●
6 topics • Pid: -1 • Host: 10.115.147.228 • Port: 41240 • CPU: -1.0% • Disk: -1B
APPLICATION-CUSTOMER_PROFILE_REFRESH
APPLICATION-ERICSSON_USAGE
APPLICATION-RECHARGE
APPLICATION-TOP_PURCHASE_CHANNEL_FEED
APPLICATION-TOPUP_DEMO
SYSTEM-RESPONSE_MESSAGES
- PinPoint ●
1 host, 1 process
10.115.147.228
- ZooKeeper ●
10.115.147.228

partitions and statistics of a feed application.

Applications, Aggregates & Enrichments

All of the feed applications configured in HCL Detect can be seen by clicking on the **Administration** link in the top navigation bar and then by clicking the **Settings** menu item. We have to use **Feed Management tab** to see feed applications.



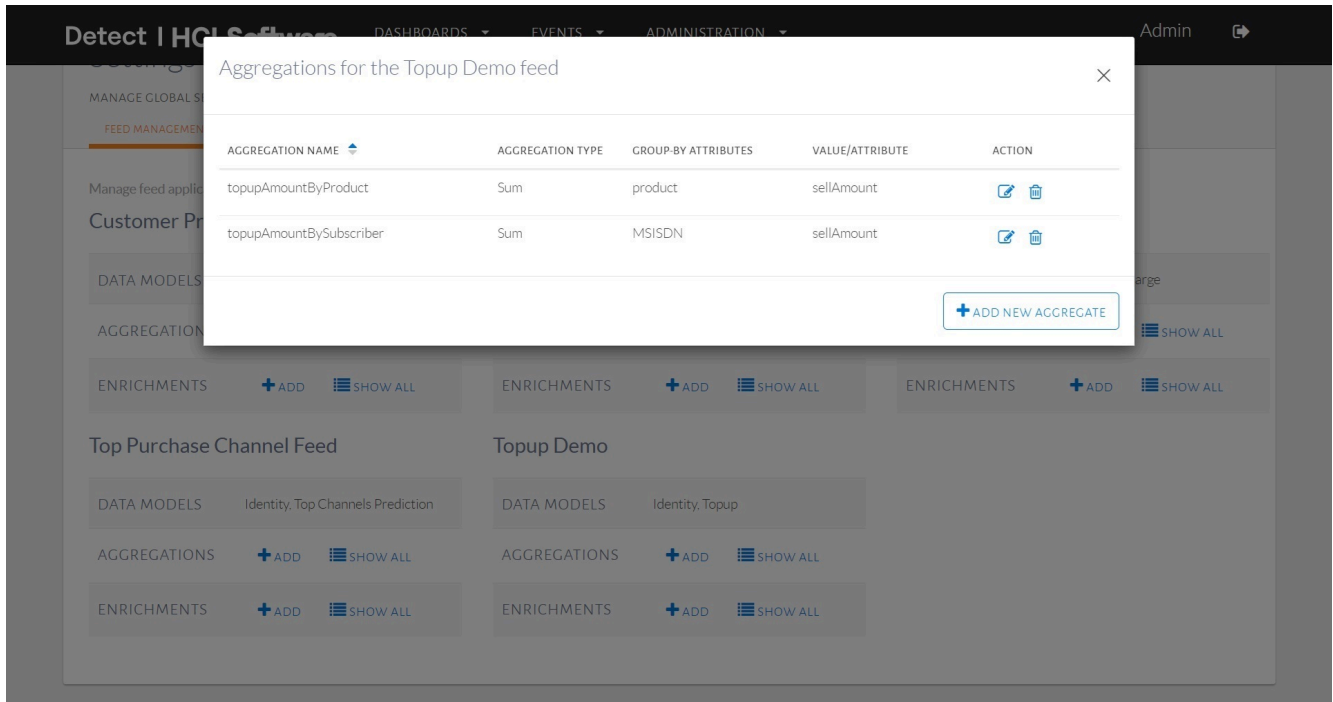
The applications configured in HCL Detect.

This dashboard shows the applications and their associated data model, e.g., 'Topup', 'Usage', etc.

Users with the appropriate permissions can use this page to view, edit, add and/or delete the aggregates and enrichments associated with a given feed application.

The existing aggregates configured for a feed application can be seen by clicking on the **ShowAll** button in the **Aggregations** row for that application.

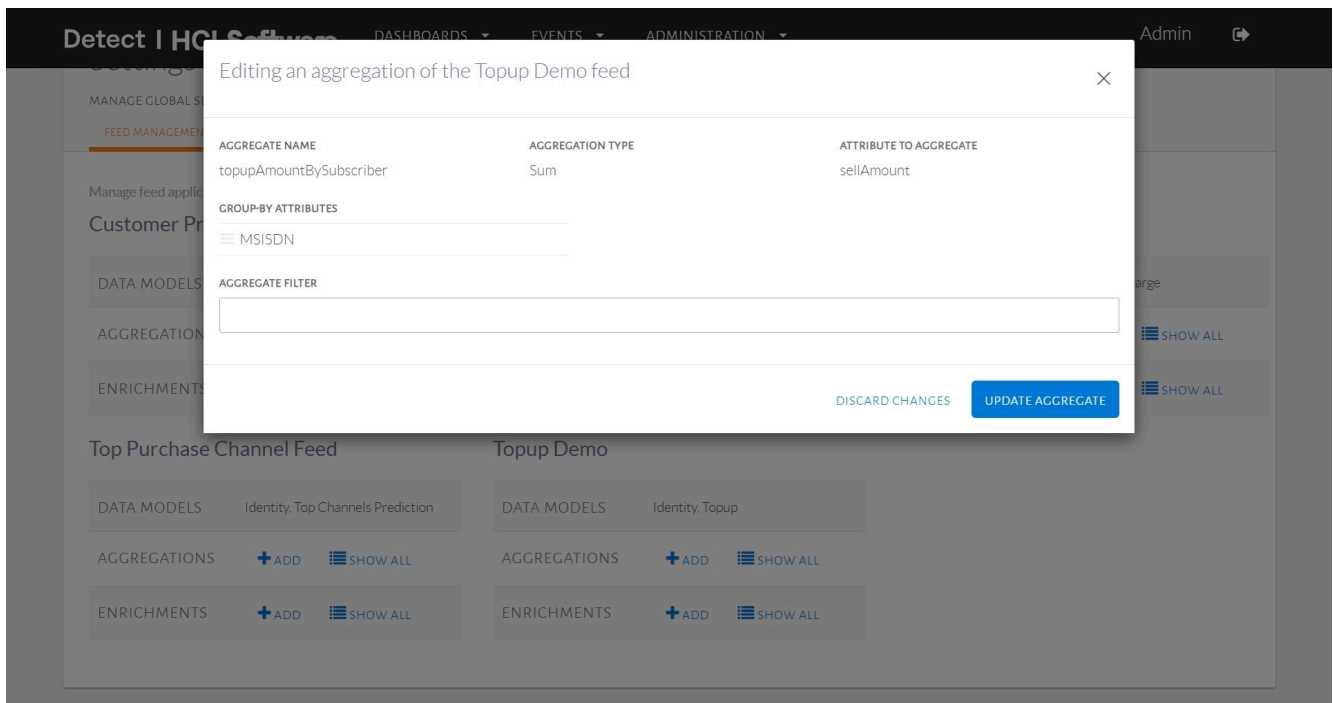
The dialog showing the aggregates associated with the **TopupDemo** application is below:



The aggregates associated with the `TopupDemo` feed.

The aggregates associated with a feed can be edited and removed.

To explain the aggregates a bit further, let us consider the aggregate called `topupAmountBySubscriber`. The dialog that opens up upon clicking on the edit button is shown below:



Editing an aggregate.

This aggregate is designed to count the number of calls by subscriber over various time windows (e.g., `currentDay`, `lastDay`, etc.).

`topupAmountBySubscriber`, i.e. the aggregate `Name`, is the name that is used to refer to the aggregate when using it in a trigger.

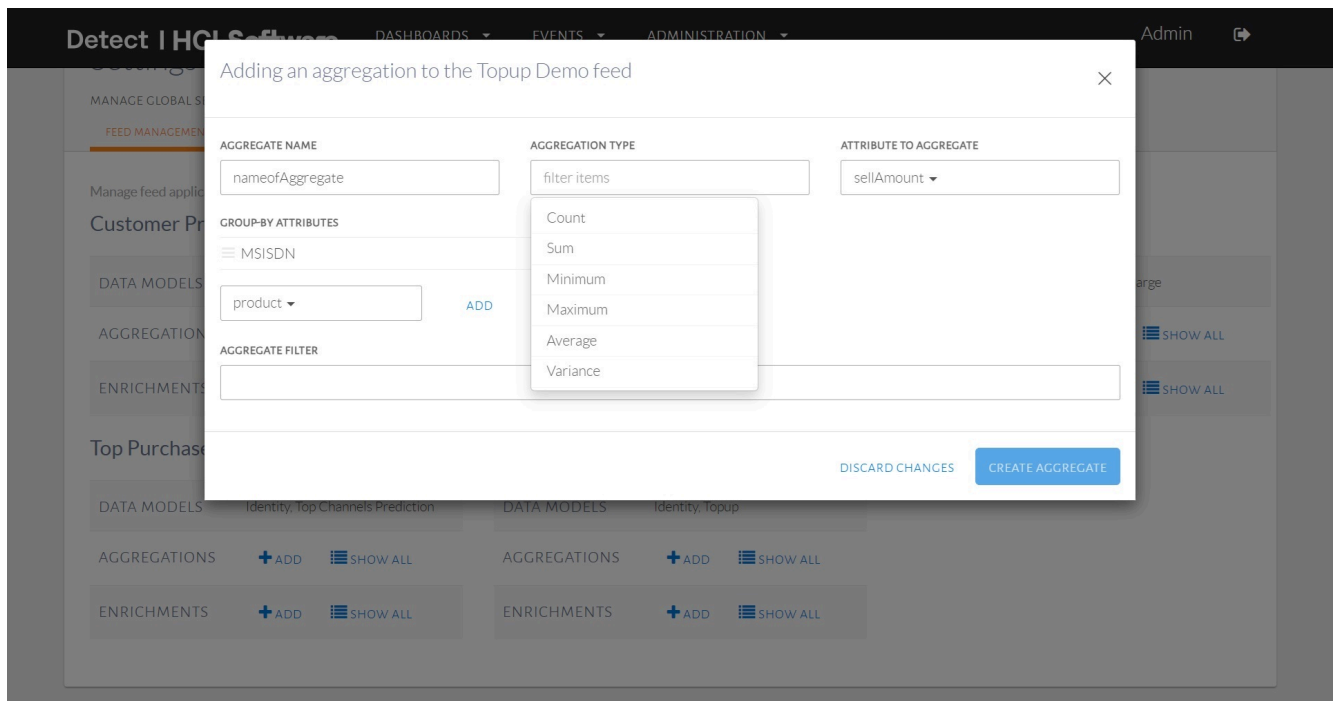
The `AggregationType`, in this case `Sum`, is used to accumulate the topup amounts, and the group-by attribute, in this case, `MSISDN`, is used to produce per-user tallies, since it identifies individual subscribers.

Additional group-by attributes can be added in a manner similar to the *group-by* clause used in traditional relational database management system (RDBMS) query languages.

A filter condition can also be used when only a subset of tuples is to be considered for an aggregation.

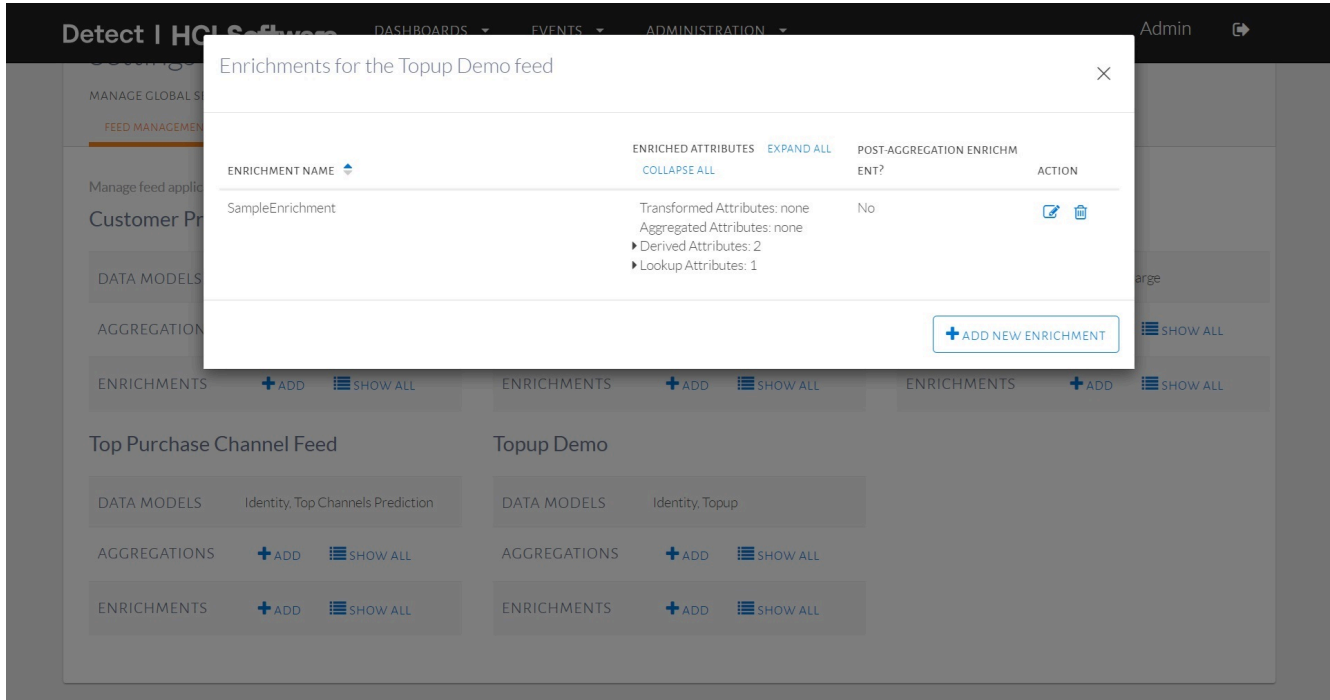
For instance, if the incoming tuples in the feed contain a Boolean `isDroppedCall`, indicating a dropped call, then one can calculate `droppedCallsBySubscriber` by using `isDroppedCall equals to True` as a filter condition with the `MSISDN` as the group-by attribute and `Count` as the aggregation type.

To add a new aggregate, we need to click on `Add` button for that feed application. A dialog with form needs to be filled by filling `Aggregate Name`, `Aggregation Type`, `Attribute to Aggregate`, `Group by Attribute` and `Aggregation Filter`:



Adding new aggregate to a feed.

The list of enrichments associated with a feed can be seen by clicking on the `Show` button in the `Enrichments` row. The enrichments associated with the `Topup Demo` feed can be seen below:



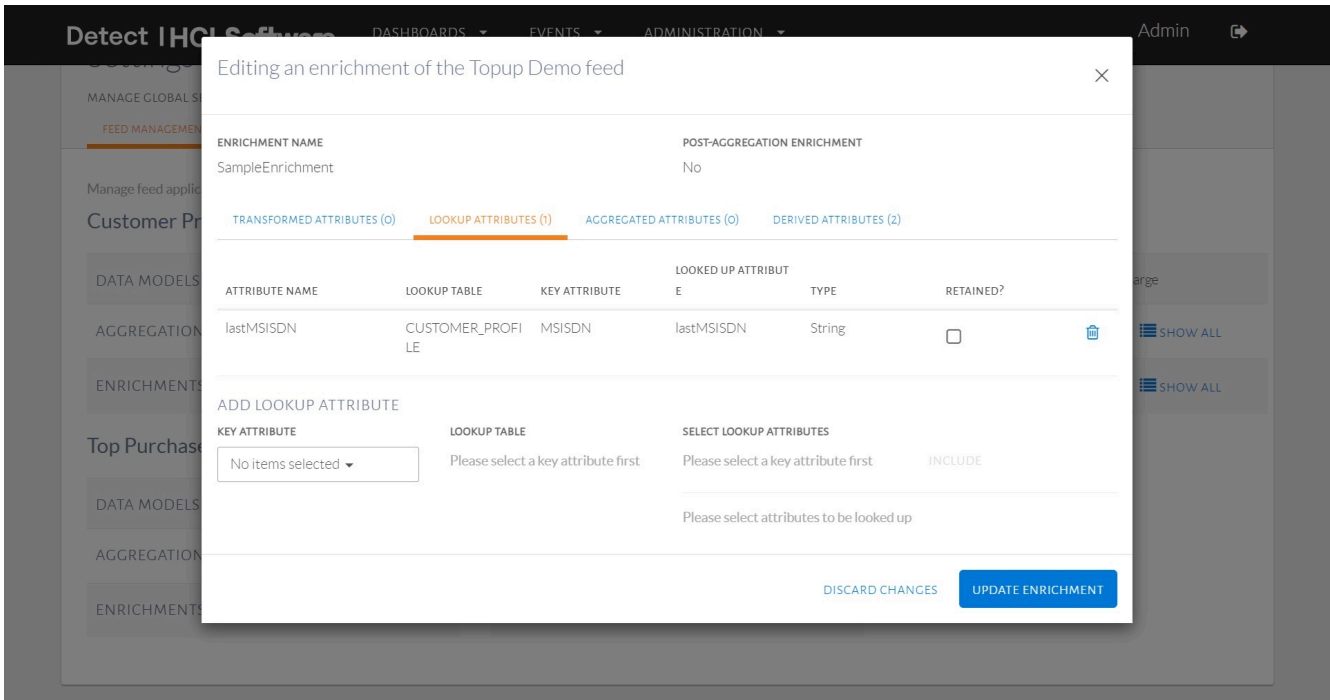
The enrichments associated with the `TopupDemo` feed.

HCL Detect supports 4 kinds of enrichments:

- The *transformed attributes* enrichment enables one to add attributes to the tuple that have either a constant value or a value that can easily be derived from an UEL expression. Such transformed attribute may be needed by other enrichments or retained as an attribute in the enriched tuple. Note that retaining and forwarding the constant attributes as part of the output tuple from the enrichment step is optional.
- The *lookup-based* enrichment, which retrieves attributes from the profile store and possibly add them to the outgoing tuple. Note that retaining and forwarding the looked up attributes as part of the output tuple from the enrichment step is optional. We have to first select a key attribute present in the tuple, the associate `Pinpoint` table will be listed in `Lookup Table` drop-down, finally we have to select the attribute from the lookup table to be looked up.
- The *aggregate-based* enrichment, which fetches data from `FastPast` and possibly adds such attributes to the outgoing tuple. As part of configuring an aggregated attribute, it is necessary to (1) select an aggregate from the list of available aggregates for the feed, (2) select the window length unit, which can be `Minute`, `Hour`, `Day`, `Month` or `Year`, (3) select the period which can be one of `Current` or `Last` and (4) optionally select `Window length`, which can be used to indicate that only a subset of an aggregate is to be used. For instance, when the `Day` window length unit and the `Last` period with `Window Length` set to 7 is chosen, the aggregated value for the last 7 days can be computed. Note that when using the `Minute` window length unit, `Window Length` needs to be multiple of 10-minute i.e., backend configured sub-hour buckets length. Similarly to lookup-based enrichment, aggregate-based enrichment attributes may or may not be retained and forwarded as part of the output tuple.
- The *derived attributes* enrichment, which enables the execution of an external Python function. One such a function takes in a tuple (as well as any other required additional parameters, if any), performs a user-defined computation, and produces a result. This function invocation's return value can then be retained and forwarded as part of an

outgoing tuple. This capability can be used, for instance, to invoke a scoring function on an incoming tuple, which computes a score (e.g., based on previously mined data) that can then be added as an attribute to the corresponding outgoing tuple. The *derived attributes* enrichment are basically of three types, i.e., *enrichment function based*, *expression based* and *scorer function based*. *expression based* takes an UEL expression explained in Miscellaneous » UEL section.

The interface for editing enrichments is shown below:



Interface for editing enrichments.

The interface for editing derived attribute type enrichments is shown below:

The screenshot shows the 'Detect IH' Administration Interface. The top navigation bar includes 'Admin' and a refresh icon. Below it, there are tabs for 'TRANSFORMED ATTRIBUTES (0)', 'LOOKUP ATTRIBUTES (1)', 'AGGREGATED ATTRIBUTES (0)', and 'DERIVED ATTRIBUTES (2)'. The main content area is divided into two sections: a table of existing derived attributes and a form to add new ones.

ATTRIBUTE NAME	TYPE	FUNCTION/EXPRESSION/SCORER	STORE BACK	RETAINED?
deviceChanged	Bool	ENRICHMENT FUNCTION device_change (acme.application_helpers.to_pup_demo.enrichment_helpers)	Key: MSISDN Table: CUSTOMER_PROFILE Attribute: deviceChanged	<input type="checkbox"/>
thisMSISDN	String	EXPRESSION MSISDN	Key: MSISDN Table: CUSTOMER_PROFILE Attribute: lastMSISDN	<input type="checkbox"/>

ADD DERIVED ATTRIBUTE:

KIND OF DERIVED ATTRIBUTE:

- Enrichment function-based
- Expression-based
- Scorer-based

ATTRIBUTE NAME:

ENRICHMENT FUNCTION:

REQUIRED ATTRIBUTES: Please select an enrichment function

FUNCTION PARAMETERS: Please select an enrichment function

STOREBACK?: Yes No

KEY ATTRIBUTE:

STOREBACK TABLE: Please select a key attribute first

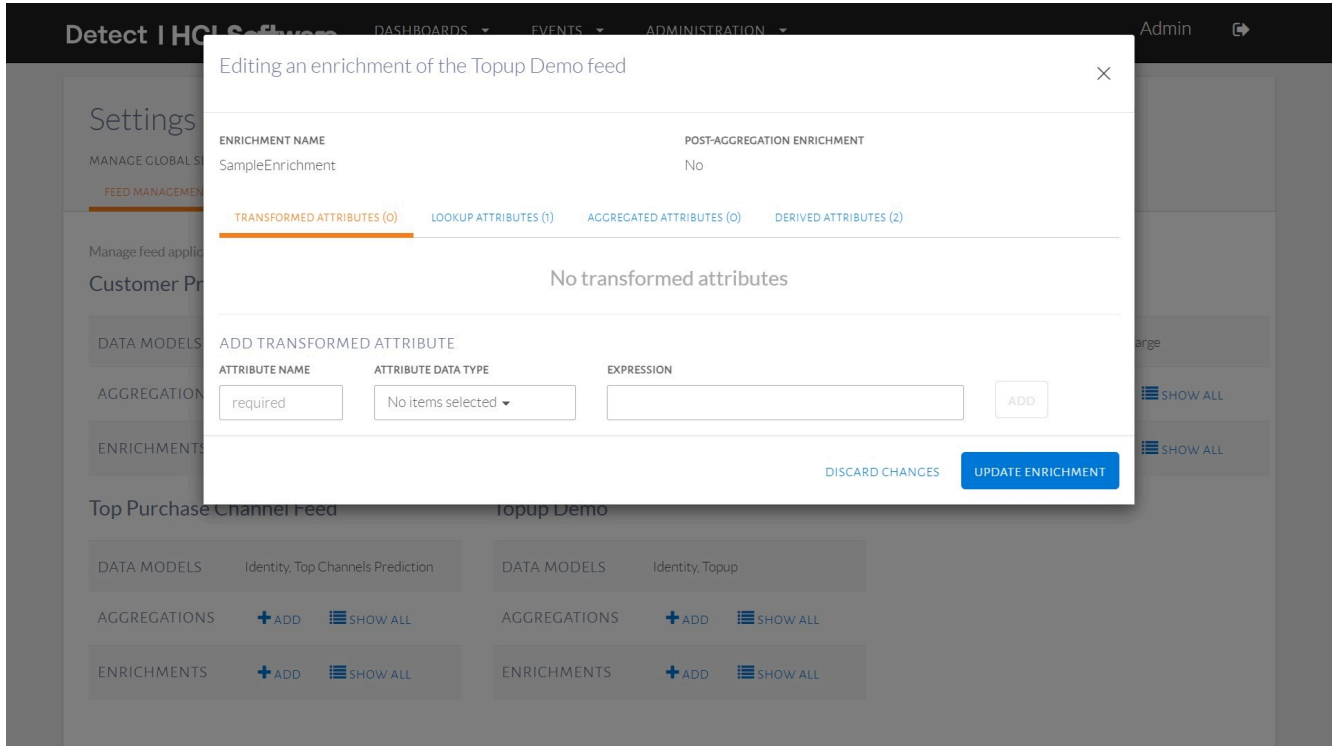
STOREBACK ATTRIBUTE: Please select a key attribute first

HCL Detect: #v2.3.1 - © Copyright HCL Technologies Ltd. 2014-2024. (build)

HCL Software

Interface for editing derived attribute type enrichments.

Enrichment operations can be configured to before aggregations are performed or after aggregations are performed. The value of a enriched *aggregate-based* attribute may change based on whether it configured to run before or after the aggregation operation. We can choose this while creating an enrichment by checking the `Post-Aggregation Enrichment` check-bit as shown in the below image:



Interface for creating a new enrichment.

Subscriber Segments

Segments are groups of customers with some similarities, setup segments based on preset conditions (Live), or through user selection list (Static). Grouping customers into Segments helps target them more appropriately or continuously across events with Detect. Detect supports 2 types of segments, Live & Static HCL Detect can make use of *subscriber segments*. Such segments can be used to include or exclude a group of subscribers from being considered as subjects for a trigger.

Static Segments

The set of `StaticSegments` can be inspected by clicking on `Settings` from the `Administration` menu and then selecting the tab titled `Static Segments`.

The example below shows a view of the page with 2 segments: the `DataUsers` and the `High Value` users.

Uploading a segment file.

To add, edit or delete a segment, a text file with its contents specified in a particular format can be uploaded to HCL Detect using the `UploadSegmentFile` button.

Two formats are supported. The first is a plain-text file with a simple linear organization as shown below:

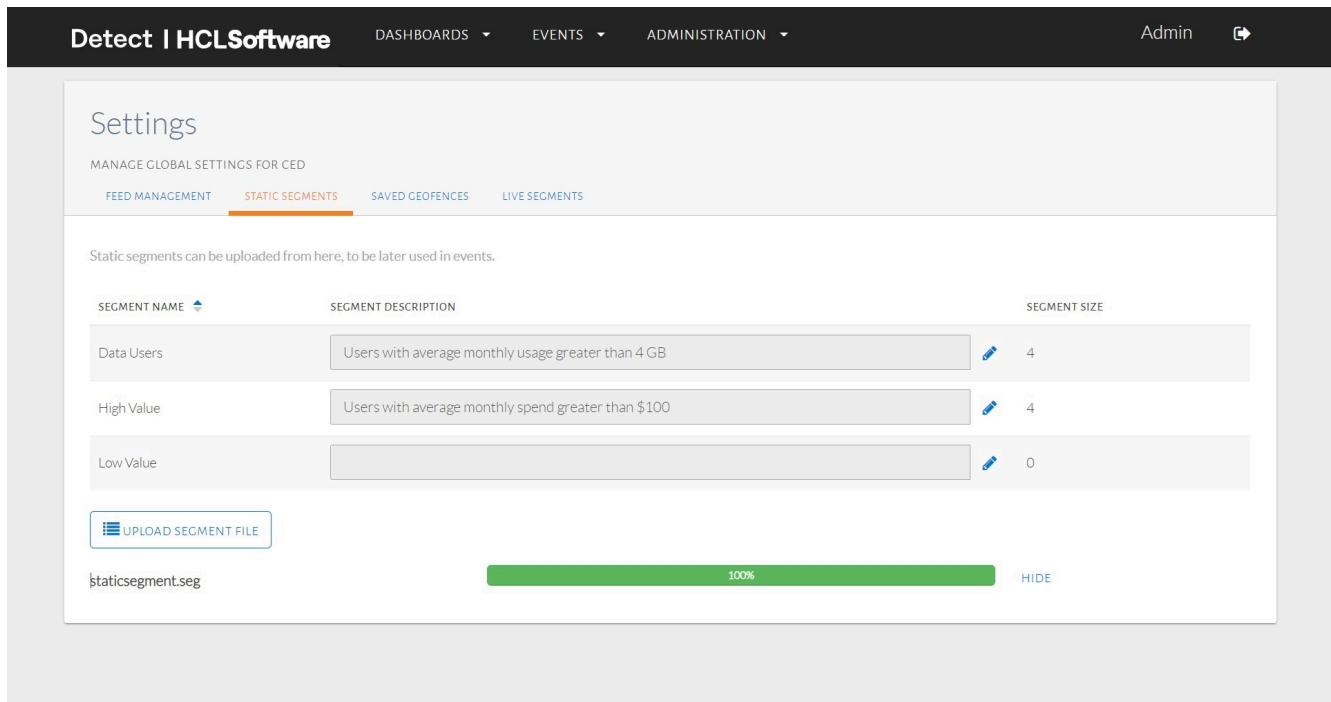
```
[Data Users]: add
911323232323
918787879988
918787989021
919898990906
[High Value]: add
912323989899
912367367676
912398982337
919828738787
[High Value]: remove
912388728787
[Low Value]: remove_all
```

Alternatively, the file can also be specified in JSON format, in which case it must have a `.json` extension. An example is given below (note that the order of the fields is important):

```
{
  "segmentUploads": [
    {
      "operation": "Add",
      "segment": "Data Users",
      "users": [
        "911323232323",
        "918787879988",
```

```
    "918787989021",
    "919898990906"
  ]
},
{
  "operation": "Add",
  "segment": "High Value",
  "users": [
    "912323989899",
    "912367367676",
    "912398982337",
    "919828738787"
  ]
},
{
  "operation": "Remove",
  "segment": "High Value",
  "users": [
    "912388728787"
  ]
},
{
  "operation": "RemoveAll",
  "segment": "Low Value"
}
]
```

Once uploaded, one can also add a description for the segments:



Segments and their descriptions.

Live Segments

Detect allows the user to enter a set of audience conditions and saves customers who meet those criteria in a Live Segment. Detect checks all customers against the defined conditions and maintains the segment over time. This means that Detect adds or removes customers from the segment as they fulfill or stop fulfilling the defined criteria. For example: if you'd like to frequently target customers who have the main balance atleast \$100 & who have installed a Mobile App i.e., Facebook, this can be saved as a Live Segment.

Detect will check all customers against the defined Main balance check & app installation status and sort the appropriate customers to the segment. Conditions can be setup as shown below:

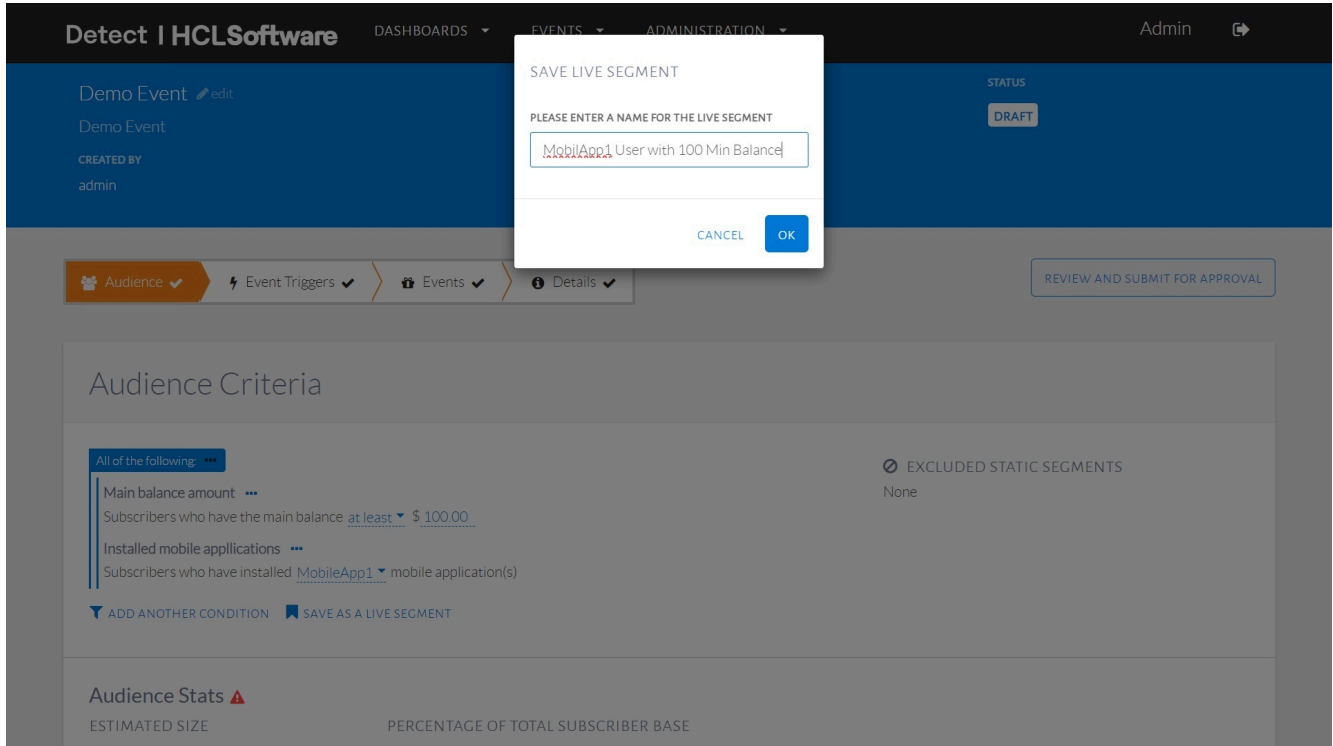
The screenshot displays the 'Detect | HCLSoftware' administration interface. The top navigation bar includes 'DASHBOARDS', 'EVENTS', and 'ADMINISTRATION'. The user is logged in as 'Admin'. The main header shows 'Demo Event' with an 'edit' link, 'PROJECT: Demo', and 'STATUS: DRAFT'. Below the header, a breadcrumb trail shows 'Audience', 'Event Triggers', 'Events', and 'Details'. A 'REVIEW AND SUBMIT FOR APPROVAL' button is visible on the right. The main content area is titled 'Audience Criteria' and features a configuration section with the following details:

- All of the following:**
 - Main balance amount:** Subscribers who have the main balance *at least* \$ 100.00.
 - Installed mobile applications:** Subscribers who have installed *MobileApp1* mobile application(s).
- Buttons: 'ADD ANOTHER CONDITION' and 'SAVE AS A LIVE SEGMENT'.
- EXCLUDED STATIC SEGMENTS:** None.

Below the configuration section, there is an 'Audience Stats' section with a warning icon, showing 'ESTIMATED SIZE' and 'PERCENTAGE OF TOTAL SUBSCRIBER BASE'.

Creating an audience condition.

To save above audience condition as reusable live segment, we need to click on `SaveasLiveSegment` button. A dialog will open to get the name of the live segment:



Saving a live segment.

Once the live segment is saved, the condition will be wrapped as a live segments. This live segment can be replaced and edited by clicking three dots on the live segment shown:

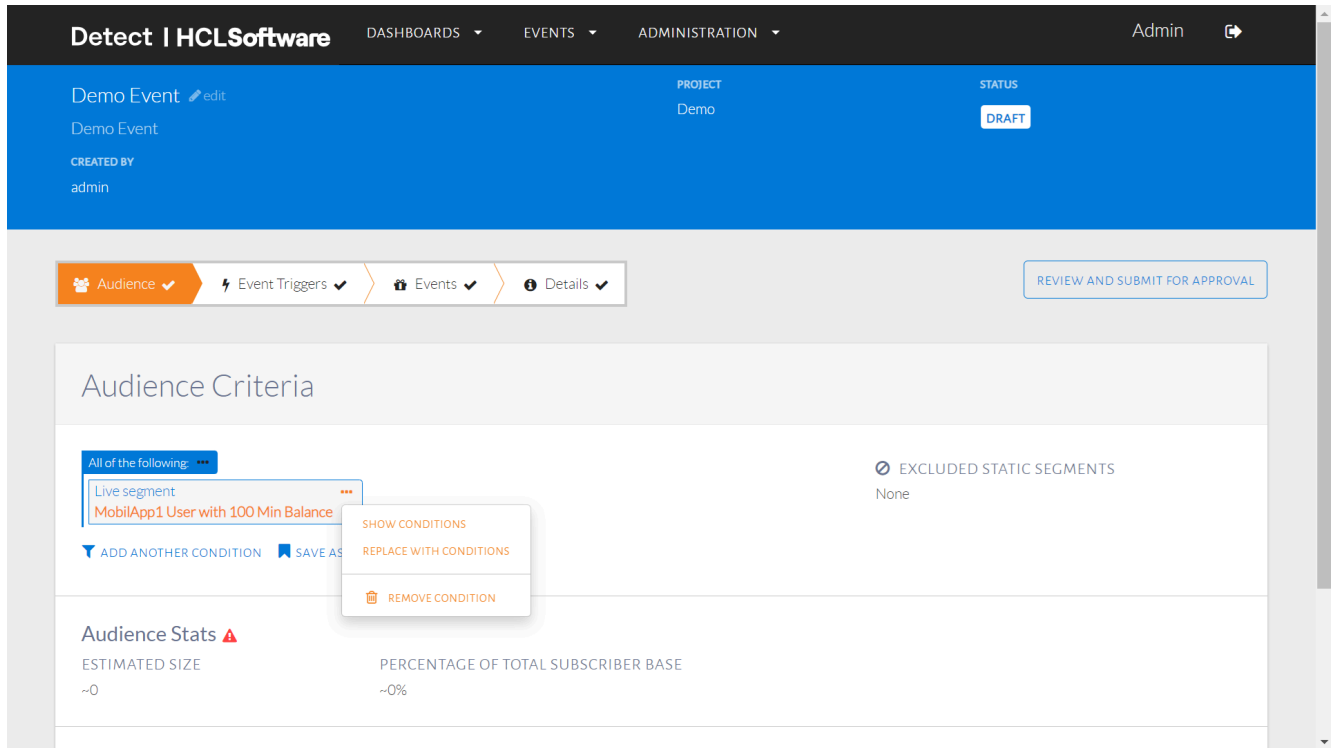
The screenshot displays the 'Detect | HCLSoftware' Administration Interface. The top navigation bar includes 'DASHBOARDS', 'EVENTS', and 'ADMINISTRATION' menus, along with a user profile 'Admin'. The main header area shows 'Demo Event' with an 'edit' link, 'PROJECT: Demo', and 'STATUS: DRAFT'. Below this, a breadcrumb trail shows 'Audience' (selected), 'Event Triggers', 'Events', and 'Details'. A 'REVIEW AND SUBMIT FOR APPROVAL' button is visible on the right.

The 'Audience Criteria' section is active, showing a list of conditions under 'All of the following:'. One condition is selected: 'Live segment: MobilApp1 User with 100 Min Balance'. A context menu is open over this condition, offering options: 'SHOW CONDITIONS', 'REPLACE WITH CONDITIONS', and 'REMOVE CONDITION'. Below the conditions, there are buttons for 'ADD ANOTHER CONDITION' and 'SAVE AS'. To the right, a section for 'EXCLUDED STATIC SEGMENTS' shows 'None'.

The 'Audience Stats' section at the bottom shows two metrics: 'ESTIMATED SIZE' with a value of '~0' and 'PERCENTAGE OF TOTAL SUBSCRIBER BASE' with a value of '~0%'. A warning icon is present next to the 'Audience Stats' title.

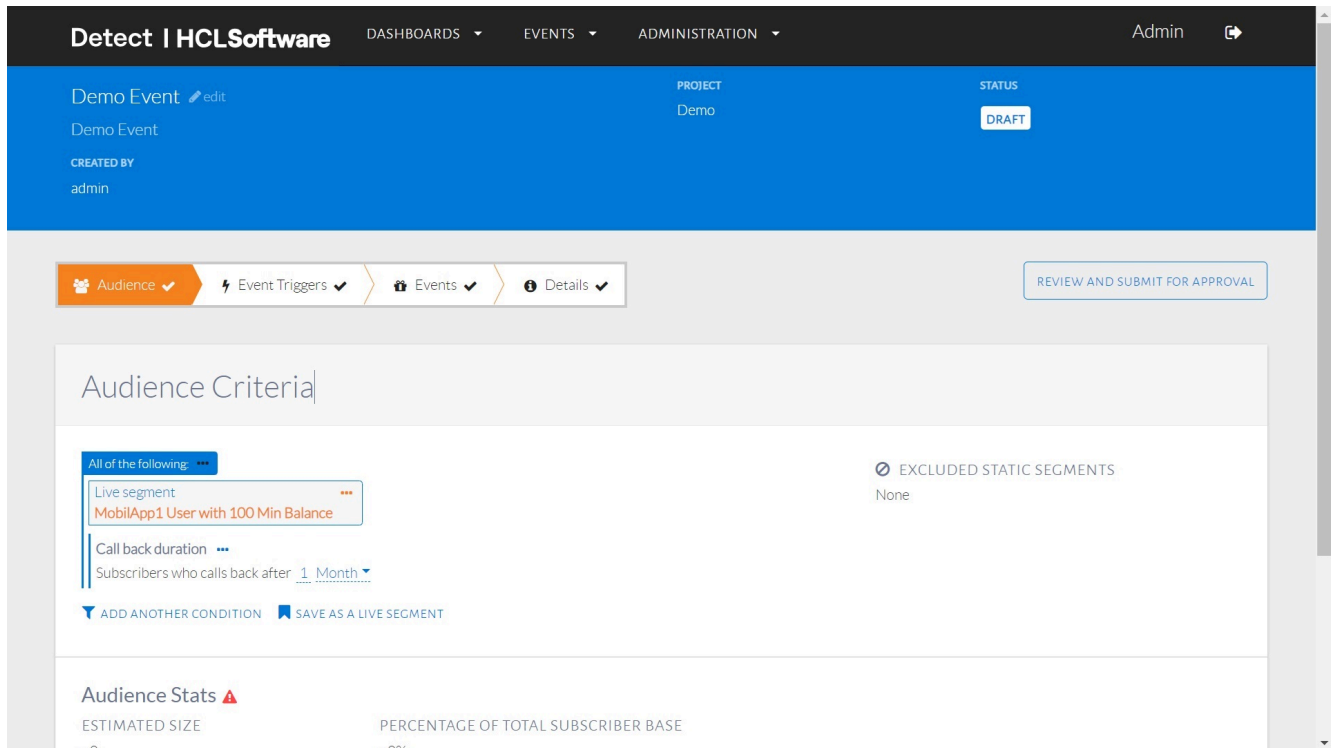
Using a live segment and editing it.

The set of `LiveSegments` can be inspected by clicking on `Settings` from the `Administration` menu and then selecting the tab titled `Live Segments`.



List of live segment.

Customers belonging to Live or Static Segments can be used as audience criteria by any user, in any event. Segments can be used along with other audience conditions as well, and can be added as shown below:



Audience condition using live segment and other audience conditions.