Unica Insights Reports Installation and Configuration Guide

Contents

Chapter 1. Unica Insights Reports Installation and Configuration Guide

This guide provides information on how to install and configure Unica Insights reports.

Overview

An installation of the HCL Unica reports is complete when you install IBM Cognos® BI or Unica Insights reports and configure it for HCL Unica applications. This guide provides detailed information about configuring Unica Insights reports with HCL Unica. .

For details on IBM Cognos® BI reports, see the Cognos Reports Installation and Configuration guide.

Plan your Unica Insights reports installation

When you plan your Unica Insights reports installation for HCL Unica products, you must ensure that you correctly set up your system and configure your environment. Review the prerequisites carefully.

Prerequisites

Before you install or upgrade any HCL Unica product, you must ensure that your computer complies with all the prerequisite software and hardware.

System requirements

For information about system requirements, see the Recommended Software Environments and Minimum System Requirements guide.

Network domain requirements

The HCL Unica products that are installed as a suite must be installed on the same network domain to comply with the browser restrictions that are designed to limit the security risks that can occur with cross-site scripting

JVM requirements

HCL Unica applications within a suite must be deployed on a dedicated Java™ virtual machine (JVM). HCL Unica products customize the JVM that is used by the web application server. If you encounter errors that are related to the JVM, you must create an Oracle WebLogic or WebSphere® domain that is dedicated to the HCL Unica products. Insights war is to deploy on any one application server – Weblogic, WAS, Tomcat or JBOSS.

Note: If you are unable to view Unica Insights reports with JBOSS application server on Unix, set all permissions (777) to Platform/Insights/Reports/Insights.war file before Insights.war deployment.

Knowledge requirements

To install HCL Unica products, you must have a thorough knowledge of the environment in which the products are installed. This knowledge includes knowledge about operating systems, databases, and web application servers.

Access permissions

Verify that you have the following network permissions to complete the installation tasks:

- Administration access for all required databases.
- Read and write access to the relevant directory and sub-directories for the operating system. account that you use to run the web application server and HCL Unica components.
- Write permission for all files that you must edit.
- Write permission for all directories where you must save a file, such as the installation directory and backup directory if you are upgrading.
- Appropriate read, write, and execute permissions to run the installer.

Verify that you possess the administrative password for your web application server. The following additional permissions are required for UNIX:

- The user account that installs and Unica Platform must be a member of the same group as the Unica Campaign users. This user account must have a valid home directory, and have write permissions for that directory.
- All installer files for HCL products must have full permissions, for example, `rwxr-xr-x`.

If a JAVA_HOME environment variable is defined on the system, where you install an HCL Unica product, verify that the variable points to a supported version of JRE. For information about system requirements, see the HCL Unica Recommended Software Environments and Minimum System Requirements guide.

If the JAVA_HOME environment variable points to an incorrect JRE, you must clear the JAVA_HOME variable before you run the HCL Unica installers.

You can clear the JAVA_HOME environment variable by using one of the following methods:

- Windows: In a command window, enter set JAVA_HOME= (leave empty) and press Enter.
- UNIX: In the terminal, enter export JAVA_HOME=(leave empty) and press Enter.

The HCL Unica installer installs a JRE in the top-level directory for the HCL Unica installation. Individual HCL Unica application installers do not install a JRE. Instead, they point to the location of the JRE that is installed by the HCL Unica installer. You can reset the environment variable after all installations are complete.

For more information about the supported JRE, see the Recommended Software Environments and Minimum System Requirements guide.

Unica Platform requirement

You must install or upgrade Platform before you install or upgrade any HCL Unica products. For each group of products that work together, you must install or upgrade Platform only once. Each product installer checks whether the required products are installed. If your product or version is not registered with Platform, a message prompts you to install or upgrade Platform before you proceed with your installation. Unica Platform must be

deployed and running before you can set any properties on the Settings > Configuration page.

Supported locales

Currently, only English language is supported.

Prerequisites for Unica Deliver Reports

If you use Unica Deliver reports, you must comply with specific prerequisites in addition to the system requirements for installing reports.

For better performance, you might need 40% of the data size as temporary table space. Work with your database administrator to regularly fine-tune your database. For best results, you can store Unica Deliver system tables in a separate table space that is mounted on a separate, unshared disk.

Settings for DB2®

If you use DB2®, you must use version 9.7.8 or higher.

Important: You must set the following value before you begin applying the Unica Deliver reports:

```
db2set DB2_COMPATIBILITY_VECTOR=ORA
```

Sample sizes for DB2®

For an DB2® Unica Campaign database of approximately 600 GB with most of the data population in the reporting tables (UCC_*), you might use the following settings:

- Tablespace page size: 16K
- Temp tablespace: 250 GB
- db2 update db cfg using auto_reval DEFERRED_FORCE;
- db2 update db cfg using decflt_rounding ROUND_HALF_UP;
- db2 update db config using LOGFILSIZ 102400;

- db2 update db config using logprimary 13;
- db2 update db config using LOGSECOND 25;
- db2stop force
- db2start

Important: The size of your transaction log can affect reports processing. Review your database environment requirements, especially the size of the transaction log, with your database administrator.

Settings for Oracle

If you use Oracle, you must use version 11g or higher. Review your environment requirements with your database administrator.

Sample sizes for Oracle

For an Oracle Unica Campaign database of approximately 650 GB with most of the data population in the reporting tables (UCC_*), you might use the following settings:

- Temp tablespace: 250 GB
- Size for REDO logs: 2 GB
- Number of REDO logs: 4

Settings for Microsoft™ SQL Server

If you use Microsoft™ SQL, you must use SQL Server 2008 or higher. Review your environment requirements with your database administrator.

Sample sizes for Microsoft™ SQL Server

For Microsoft™ SQL Server database of approximately 520 GB with most of the data population in the reporting tables (UCC_*), you might use the following settings:

- Temp tablespace: 250 GB

Settings for Internet Explorer

If you use Internet Explorer, make sure that the browser security settings allow automatic prompting for file downloads. Complete the following steps to verify that the browser allows automatic prompting for file downloads:

1. Open Internet Explorer and go to **Tools > Internet Options**.
2. In the **Security** tab, click **Custom Level**.
3. Scroll down to the **Downloads** section.
4. Make sure that the **Automatic prompting for file downloads** option is set to **Enable**.

Deploying Unica Insights

You must follow a set of guidelines when you deploy Unica Insights in your web application server. There is a different set of guidelines for deploying Unica Insights on WebLogic and on WebSphere.

When you run the Unica installer, you completed the following actions:

• You created the WAR file of Unica Insights (Insights.war).

We assume that you possess the information on how to work with your web application server. For details, such as navigation in the Administration console, see your web application server documentation.

NOTE

In Unica Insights reports, if you access Unica using a load balancer, a reverse proxy, or a web access management software, configure the reverse proxy URL in the `viewer.properties` file by completing the following steps:

- Open `Insights.war`.
- Navigate to the `WEB-INF` folder and access the `viewer.properties` file.
- Uncomment the `base_url` property by removing "`#`".
- Configure the `base_url` property with Unica Insights' reverse proxy URL.
- Redeploy the `Insights.war` file.

Guidelines for deploying Unica Insights on WebLogic

You must follow a set of guidelines when you deploy Unica Insights on the WebLogic application.

Use the following guidelines when you deploy the Unica Insights products on any supported version of WebLogic:

- Unica products customize the Java virtual machine (JVM) used by WebLogic. If you encounter errors related to JVM, you can create a WebLogic instance that is dedicated to Unica products.
- Open the startWebLogic.cmd file and verify that the SDK that is selected for the WebLogic domain that you are using is the Sun SDK for the `JAVA_VENDOR` variable. The `JAVA_VENDOR` variable must be set to Sun (JAVA_VENDOR=Sun).

  If the `JAVA_VENDOR` variable is set to JAVA_VENDOR, it means that JRockit is selected. You must change the selected SDK as JRockit is not supported. See the BEA WebLogic documentation to change the selected SDK.
- Deploy Unica Insights as a web application.
- If you are configuring WebLogic to use the IIS plug-in, review the BEA WebLogic documentation.
- Complete the following tasks if your installation must support non-ASCII characters, for example for Portuguese or for locales that require multi-byte characters:
  1. Edit the `setDomainEnv` script in the bin directory under your WebLogic domain directory to add `-Dfile.encoding=UTF-8` to JAVA_VENDOR.
  2. In the **WebLogic** console, click the **Domain** link on the home page.
  3. In the **Web Applications**tab, select the **Archived Real Path Enabled** check box.
  4. Restart WebLogic.
  5. Deploy and start the Insights.war file.
- If deploying in a production environment, set the JVM memory heap size parameters to 1024 by adding the following line to the `setDomainEnv` script..

```
Set MEM_ARGS=-Xms1024m -Xmx1024m -XX:MaxPermSize=256m
```

Guidelines for deploying Unica Insights on WebSphere

You must follow a set of guidelines when you deploy Unica Insights on WebSphere. Ensure that the version of WebSphere meets the requirements that are described in the Recommended Software Environments and Minimum System Requirements document, including any necessary fix packs. Use the following guidelines when you deploy Unica Insights on WebSphere:

1. Specify the following custom property in the server:
    - Name: com.ibm.ws.webcontainer.invokefilterscompatibility
    - Value: `true`
2. Set a custom property in WebSphere.
3. Deploy the Insights.war file as an enterprise application. When you deploy the Insights.war file, ensure that the JDK source level of the JSP compiler is set to Java 17 for SDK 1.7 and 18 for SDK 1.8, and that JSP pages are precompiled according to the following information:
    - In the form where you browse to and select the WAR file, select **Show me all installation options and parameters** so the **Select Installation Options** wizard runs.
    - In step 1 of the **Select Installation Options** wizard, select **Precompile JavaServer Pages files**.
    - In step 3 of the **Select Installation Options** wizard, ensure that the **JDK Source Level** is set to `17` for SDK 1.7 and set to `18` for SDK 1.8.
    - In step 8 of the **Select Installation Options** wizard, select **UnicaPlatformDS** as the matching Target Resource.
    - In step 10 of the **Select Installation Options** wizard, the context root must be set to /Insights.
4. In the **Web Container Settings > Web Container > Session Management** section of the server, enable cookies. Specify a different session cookie name for each application that is deployed. Use one of the following procedures to specify a cookie name:
    - Select the **Override session management** check box under **Session Management**. If you deployed separate WAR files for your Unica products, in the WebSphere console, in the **Applications > Enterprise Applications >**

[deployed_application] > Session Management > Enable Cookies > Cookie Name section of the server, specify a unique session cookie name].

- If your installation must support non-ASCII characters, for example for Portuguese or for locales that require multi-byte characters, add the following arguments to **Generic JVM Arguments** at the server level.

```
-Dfile.encoding=UTF-8
```

```
-Dclient.encoding.override=UTF-8
```

5. Navigation tip: Select **Servers > Application Servers > Java and Process Management > Process Definition > Java Virtual Machine > Generic JVM Arguments**. See the WebSphere documentation for additional details.
6. In the **Applications > Enterprise Applications** section of the server, select the WAR file that you deployed, then select **Class loading and update detection** and specify the following properties. If you are deploying a WAR file:
    - For **Class loader order**, select **Classes loaded with local class loader first (parent last)**.
    - For **WAR class loader policy**, select **Single class loader for application**.
7. Start your deployment. If your instance of WebSphere is configured to use a JVM version 1.7 or newer, complete the following steps to work around an issue with the time zone database.
    - Stop WebSphere.
    - Download the Time Zone Update Utility for Java (JTZU).
    - Follow the steps provided by the IBM (JTZU) to update the time zone data in your JVM.
    - Restart WebSphere.
8. In **Websphere Enterprise Applications**, select **Your Application > Manage Modules > Your Application> Class Loader Order > Classes loaded with local class loader first (parent last)**.

- The recommended minimum heap size for the basic functioning of the application is `512` and the recommended maximum heap size is `1024`. Complete the following tasks to specify the heap size:
  - In **WebSphere Enterprise Applications**, select **Servers > WebSphere application servers > server1 > Server Infrastructure > Java and Process Management > Process definition > Java Virtual Machine**.
  - Set the initial heap size to `512`.
  - Set the maximum heap size to `1024`. See the WebSphere documentation for more information about sizing. For DB2, set progressiveStreaming = `2` in WebSphere console at following path: **JDBC >Data sources > UnicaPlatformDS > Custom properties**.

Guidelines for deploying Unica Insights on JBoss

Before deploying Insights.war in JBOSS application server, update the Insights.war file.

You must follow a set of guidelines when you deploy Unica Insights on JBoss. Make sure that the version of JBoss meets the requirements that are described in the Recommended Software Environments and Minimum System Requirements document. Use the following guidelines when you deploy Unica Insights on JBoss:

Note: If you are unable to view Unica Insights reports with JBOSS application server on Unix, ensure that you set all permissions (777) to Platform/Insights/Reports/Insights.war file before Insights.war deployment.

Use the following guidelines when you deploy the Unica Insights products on any supported version of JBoss:

1. Deploy the Insights.war file as an enterprise application. For example: deploy `<Platform_Install>\ Insights.war`.

   See https://docs.jboss.org/jbossweb/3.0.x/deployer-howto.html for instructions on Deploying Web Server Application in JBoss.
2. Complete the following tasks if your installation must support non-ASCII characters, for example for Portuguese or for locales that require multi-byte characters:

- Edit the `standalone.conf` script in the bin directory under your JBOSS /bin directory to add `-Dfile.encoding=UTF-8`.

  `-Dclient.encoding.override=UTF-8`

  `-Djboss.as.management.blocking.timeout=3600`

  to JAVA_VENDOR.
- Restart JBoss server.

Guidelines for deploying Unica Insights on Apache Tomcat

You must follow a set of guidelines when you deploy Unica Insights on Apache Tomcat.

Ensure that the version of Apache Tomcat meets the requirements that are described in the HCL Enterprise Products Recommended Software Environments and Minimum System Requirements document. Use the following guidelines when you deploy Unica Insights on Apache Tomcat:

1. Deploy the Insights.war file as an enterprise application on Tomcat Apache server.
2. Complete the following tasks if your installation must support non-ASCII characters, for example for Portuguese or for locales that require multi-byte characters:
   - Edit the setenv.sh file for the respective product instances script in the bin directory under your tomcat instances directory to add `-Dfile.encoding=UTF-8 - Dclient.encoding.override=UTF-8 to JAVA_VENDOR`.
   - Restart Tomcat.
3. If deploying in a production environment, you can add JVM heap setting for that tomcat instance in app-one/bin/setenv.sh file respectively for all the instances.

Install and configure Unica Insights reports

The Unica Platform application allows you to install Unica Insights reports. For more details, see the Unica Platform Installation Guide.

Here are the configuration properties laid down by the Installer.

Insights | navigation

The Unica suite integrates with Unica Insights to generate reports.

This page displays properties that specify URLs and other parameters that are used by the Unica Insights system.

Seed Name

Description

Used internally by HCL Unica applications. Changes to this value are not recommended.

Default value

Insights

httpPort

Description

This property specifies the port used by the Unica Insights web application server. If your installation of Unica Insights uses a port which is different from the default, you must edit the value of this property.

Default value

7001

httpsPort

Description

If SSL is configured, this property specifies the port used by the Unica Insights web application server for secure connections. If your installation of Unica Insights uses a secure port that is different from the default, you must edit the value of this property.

Default value

7001

serverURL

Description

Specifies the URL of the Unica Insights web application. Use a fully qualified host name, including the domain name (and subdomain, if appropriate) specified in the Domain property. For example: http://MyReportServer.MyCompanyDomain.com:7001/ Insights

Default value

`http://[CHANGE ME]/Insights`

Valid values

A well-formed URL

logoutURL

Description

The logoutURL property is used internally to call the logout handler of the registered application if the user clicks the logout link. Do not change this value.

Default value

`/j_spring_security_logout`

Enabled

Description

Setting the value to `TRUE` ensures that Unica Insights will be used as reporting engine.

Note: If you are upgrading to V12.1 and you have Campaign/Plan/ Interact Reports pack and Unica Platform installed, then you can either see Cognos Reports or Unica Insights reports.

Default value

`False`

Valid values

`FALSE | TRUE`

This table includes the supported databases for Unica Insights reports for the products.

Products	Databases
Unica Campaign	Oracle, SQL Server, DB2, MariaDB, OneDB
Unica Interact	Oracle, SQL Server, DB2, MariaDB, OneDB
Unica Plan	Oracle, SQL Server, DB2, MariaDB, OneDB
Unica Deliver	Oracle, SQL Server, DB2, MariaDB, OneDB
Unica Collaborate	Oracle, SQL Server, DB2

Install Unica Insights reporting components

To install Unica Insights reports for your HCL Unica products, you must install the Unica Insights reporting components.

Reporting components include the following items:

- HCL Unica integration components
- Reporting schemas

Upgrade Unica Insights reports

The Unica Platform application allows you to upgrade Unica Insights reports. For more details, see the Unica Platform Upgrade Guide.

See the following table for Unica Insights upgrade scenarios:

Source version	Target version	Steps
12.0.0.0	12.1.0	See Upgrade from versions 12.0.0.0 or 12.0.0.1 to 12.1.0 *(on page 15)*.
12.0.0.1	12.1.0	See Upgrade from versions 12.0.0.0 or 12.0.0.1 to 12.1.0 *(on page 15)*.
12.1.0	12.1.0.1	See Upgrade from version 12.1.0.0 to 12.1.0.1 *(on page 16)*.
12.1.0	12.1.0.3	See Upgrade from version 12.1.0.0 to version 12.1.0.3 *(on page 17)*.
12.1.0.3	12.1.0.4	See Upgrade from version 12.1.0.3 to version 12.1.0.4 *(on page 18)*.
12.1.0	12.1.0.4	See: 1. Upgrade from version 12.1.0.0 to version 12.1.0.3 *(on page 17)* 2. Upgrade from version 12.1.0.3 to version 12.1.0.4 *(on page 18)*
Any prior release	12.1.1	See Upgrade to version 12.1.3 *(on page 20)*.

Upgrade from versions 12.0.0.0 or 12.0.0.1 to 12.1.0

In version 12.1.0, the node for BIRT reports is replaced with Unica Insights. You must verify the configuration properties after the upgrade is complete. To perform the upgrade, complete the following steps.

1. Perform the following substeps to upgrade from 12.0.x version to 12.1.0 version.
    a. Undeploy hcl-birt.war.
    b. Delete hcl-birt.war from the application server directory.
2. Copy UnicaInsights.war from `<PLATFORM_HOME>/Insights/Insights.war` to application server directory.
3. Deploy UnicaInsights.war.
4. Run templates_sql_load.sql in your Platform system database.
5. Log in to the application and navigate to **Configuration > Report SQL Generator > Select Deliver Reports**.
6. Copy the respective product reports folder for which you want to perform upgrade in Unica Insights directory. Perform the steps mentioned under the following topics.
    - Copy the Campaign and Deliver reports folder in Unica Insights directory *(on page 28)*
    - Copy the Plan report folder in Unica Insights directory *(on page 30)*.
    - Copy the Interact reports folder in Unica Insights directory *(on page 33)*.
7. To perform the upgrade of Deliver reports, perform the steps mentioned under theFor Unica Deliver only: Stored procedures for delta processing *(on page 35)* section

Upgrade from version 12.1.0.0 to 12.1.0.1

Deliver reports must work on version 12.1.0 base environment. You must take a backup of all reporting folders "`Unica_home\Platform\Insights\Reports` ". You must take the backup of Campaign and Platform database as well.

Note: Unica Insights supports four characters for locale. For example, in the `Unica_home\Campaign\reports\tools` directory, if there are two lookup populate SQL files for the french locale:

- `uare_lookup_populate_fr.sql`
- `uare_lookup_populate_fr_FR.sql`

you must execute `uare_lookup_populate_fr_FR.sql` and ignore the other file.

To perform the upgrade of Unica Insights Reports from version 12.1.0.0 to version 12.1.0.1, complete the following steps:

1. Copy and replace report design files from `Unica_home\Campaign\reports` to `Unica_home\Platform\Insights\Reports`, as explained in the Copy the Campaign and Deliver reports folder in Unica Insights directory *(on page 28)* section.
2. Run `Unica_home\Campaign\reports\Deliver-ddl\<DBType>\acer_scripts_<DBType>.sql` and `<DBType>\upgrade\upgrade121to12101.sql` in Campaign system database.
3. Run `Unica_home\Campaign\reports\schema\templates_sql_load.sql` in Platform database.
4. Log in to the application and navigate to **Configuration > Report SQL Generator > Select Deliver Reports**.
5. Select the database type and download the views script, Deliver.sql.
6. Run the Deliver.sql in Campaign database.
7. Configure database job for the following SMS procedures.
    - SP_POPULATE_SMS_CONTACTS
    - SP_POPULATE_SMS_RESPONSES

Upgrade from version 12.1.0.0 to version 12.1.0.3

To upgrade from version 12.1.0.0 to version 12.1.0.3, perform the following steps.

1. Verify that the environment is up and running.
2. Take a backup of Insights.war and Campaign.war.
3. Take a backup of the existing deployment directory.
4. Copy Insights.war on the deployment directory path.
5. Re-deploy new Insights.war file and start the application server.
6. Review `BIRT_RESOURCE_PATH` in web.xml in Insights deployment directory `<Unica_Home>\Platform\Insights\Reports\translated\`.

7. Copy the following properties files to `BIRT_RESOURCE_PATH`.
    - Campaign_Home/reports/translated/Campaign/ *.properties
    - Campaign_Home/reports/translated/Deliver/ *.properties
8. Run InsightsDBUtil for data source upgrade for Campaign and Deliver products.
9. Run `Unica_home\Campaign\reports\schema\templates_sql_load.sql` in Platform database.
10. Log in to the application and navigate to the following locations.
    - For Campaign and Collaborate: **Configuration > Report SQL Generator > Select Campaign Reports**
    - For Deliver: **Configuration > Report SQL Generator > Select Deliver Reports**
    - For Interact: **Configuration > Report SQL Generator > Select Interact Reports**
11. Copy and replace the rpt design files for Campaign, Deliver, Interact, Collaborate, and Plan to the `Platform/Reports/Insights/`. For more configuration details, see the following sections.
    - Copy the Campaign and Deliver reports folder in Unica Insights directory *(on page 28)*
    - Creating and populating reporting tables for Unica Campaign and Unica Deliver *(on page 27)*
    - Copy the Interact reports folder in Unica Insights directory *(on page 33)*
    - Copy the Plan report folder in Unica Insights directory *(on page 30)*
    - Copy the Collaborate report folder in Unica Insights directory *(on page 31)*
12. For Campaign reports, navigate to `Unica_home\Campaign\reports\ddl\<DB Type>\` directory and run `sp_whatifofferperf.sql` in Campaign system database.

    Note: MariaDB is not supported, where the DB types are Oracle/SQLServer/DB2/OneDB.

13. Start the application server.

Upgrade from version 12.1.0.3 to version 12.1.0.4

To upgrade from version 12.1.0.3 to version 12.1.0.4, perform the following steps:

1. To upgrade the Unica Campaign system database, complete the following steps:
    a. Access the location `Campaign_Home\reports\Deliver-ddl\<DBType>\upgrade` and run the command upgrade12103to12104.sql.
    b. Access the location `Campaign_Home\reports\Deliver-ddl\<DBType>` and run the command acer_scripts_<database-name>.sql, where *<database-name>* is the name of the database used. For example, if you are using Oracle, the *<database-name>* is *ora*.
2. To refresh the report files in the installation directory, complete the following steps:
    a. In the `Unica_home\Campaign\reports\Affinium Campaign\Deliver Reports` location, back up existing Deliver report design files.
    b. Move the report design files from `Unica\home\Campaign\reports\Affinium Campaign\Deliver Reports` location to `Unica_home\Platform\Insights\Reports\campaign\partitions\partition1\Affinium Campaign\Deliver Reports` location.
3. Update the datasource in the Report design files. For more information, see Update data source in Unica Insights report design files using Unica Insights utility *(on page 39)*.
4. Set up the database jobs for WhatsApp Contacts, and setup Responses for WhatsApp and Mobile report. For more information, see For Unica Deliver only: How to schedule and run stored procedures *(on page 44)*.
   The procedure names are:
    - `sp_populate_mobile_Responses`
    - `sp_populate_WhtsApp_Contacts`
    - `sp_populate_WhtsApp_Responses`
5. Copy the Unica Deliver translation files from `<Unica_Home>\Campaign\reports\translated\Deliver*.properties` location and paste it to the `<Unica_Home>\Platform\Insights\Reports\translated` location.
6. In the Unica Campaign database, refresh the `uare_lookup_population` script data. Navigate to `Unica_Home\Campaign\reports\tools\` location and execute the `uare_lookup_populate*.sql` script for all the languages.

Upgrade to version 12.1.3

In Unica Insights Reports 12.1.3 release, we have updated the user experience of a few reports for Deliver.

To upgrade to Unica Insights Reports 12.1.3, from any earlier release, complete the following steps:

1. Access the `Unica_Home\Platform\Insights\Reports\` location and back up all Insights reports.
2. Undeploy the existing Insights.war.
3. Deploy the new `Insights.war` from the 12.1.3 release.
4. Copy all reports from the product install directory to the `Unica_Home\Platform\Insights\Reports\` location.
    a. For Campaign reports and Deliver reports, see Copy the Campaign and Deliver reports folder in Unica Insights directory *(on page 28)*.
    b. For Plan reports, see Copy the Plan report folder in Unica Insights directory *(on page 30)*.
    c. For Interact reports, see Copy the Interact reports folder in Unica Insights directory *(on page 33)*.
5. Run the utility `insightsDBUtil` to update the data source in the latest reports. For more information, see Update data source in Unica Insights report design files using Unica Insights utility *(on page 39)*.

Configuring a user with the Reports System role

You must configure a user with the Reports System role. This role is used to configure reporting properties and to generate the SQL script that is used to create the reporting schemas.

A user with the Reports System role can access the Configuration and Report SQL Generator pages. You must configure a user with access to the HCL Unica Settings > Configuration and Settings > Report SQL Generator pages. Then, you can log in as this user to configure the reporting properties and generate the SQL script that is used to create the reporting schemas.

To configure a user with the Reports System role, complete the following steps.

1. Create a user.

   Note: You can also use the platform_admin user.

2. Go to **User Roles and Permissions > Report > PartitionN** and assign the Reports System role to that user.
3. Verify that the user has access to the **Settings > Configuration** and **Settings > Report SQL Generator** pages.
4. Grant the roles ReportsSystem (Unica Platform Report), ReportsUser (Unica Platform Report) to user.

Creating JDBC data sources

You must configure a JDBC data source for every HCL Unica application for which you want to enable reporting.

The HCL Unica Reports SQL Generator tool must be able to connect to the HCL Unica application databases to generate SQL scripts that create reporting tables. The Reports SQL Generator can generate SQL scripts that create views or materialized views without access to the application databases.

However, the SQL generator cannot validate the SQL code without a data source connection.

If you need more help with this task, see the Product installation documentation. To create the JDBC data source, complete the following steps.

To configure JDBC data sources, use the default JNDI name that is listed in the following table.

Note: If you do not use the default JNDI names, make a note of the names that you use. You must specify the correct name of the data source when you run the SQL Generator tool.

Table 1. Default JNDI names

Applications	Default JNDI name
Unica Campaign and Unica Deliver	campaignPartition1DS If there are multiple partitions, create a data source for each partition.
Unica Interact	For the design-time database: `campaignPartition1DS` For the runtime database: `InteractRTDS` For the learning tables: `InteractLearningDS`
Unica Collaborate	campaignPartition1DS

Loading templates for the Reports SQL Generator

The HCL Unica Insights reports use reporting schemas containing SQL scripts. These SQL scripts load template SQL select statements into the `uar_common_sql` table. The Reports SQL Generator uses the templates when it generates SQL scripts to create reporting views and tables.

To run the script that loads the templates, complete the following steps.

1. Browse to the schema directory under your report pack installation and locate the `templates_sql_load.sql` script.
2. Run the `templates_sql_load.sql` script in the Platform database.

Generating view creation scripts

When you generate reports, you extract reportable data from the reporting views or tables. You can create reporting views by using the view creation scripts. Use the Reports SQL Generator to create view creation scripts.

To create view creation scripts, complete the following steps.

1. Log in to HCL Unica as the user who has the ReportsSystem role.
2. If you have created the default JNDI names for JDBC data sources, continue to step 3. If you did not create the default JNDI names for JDBC data sources, complete the following substeps.
    a. Select **Settings > Configuration > Reports > Schemas > ProductName**.
    b. Change the default values of the JNDI property to match the JNDI names that you used for the JDBC connections
3. Select **Settings > Reports SQL Generator**.

Note: If the JNDI data source names are incorrect or are not configured, the SQL Generator cannot validate the SQL scripts that create tables.

4. In the **Product** field, select the appropriate HCL Unica application.
5. In the **Schema** field, select one or more reporting schemas.
6. Select the Database Type.

Note: In the Database type drop down list, you can see Oracle, SQLSever, DB2, OneDB, and MariaDB. MariaDB and OneDB are not supported for Collaborate reports OneDB is not supported for Plan reports. See the Install and configure Unica Insights reports *(on page 11)* for the supported databases for each product.

7. Under the **Generate Type** list, select the appropriate option for your database type. Table and XML options are not recommended.

Database	Allowed Options
SQL Server	Views
Oracle	Views, Materialized views
IBM DB2	Views, Materialized views
MariaDB	Views
OneDB	Views

For Oracle and IBM® DB2®, it is recommended to use materialized views. The Administrator must schedule them to refresh nightly or hourly as per the requirement.

8. Ensure that **Generate Drop Statement** is set to **No**.
9. If you want to examine the SQL script that is generated, click **Generate**. The SQL Generator creates the script and displays it in the browser window.
10. Click Download.
    The SQL Generator creates the script and prompts for a location in which to save the file. If you selected a single reporting schema, the script name matches the name of schema, for example Deliver_Mailing_Performance.sql. If you selected more than one reporting schema, the script name uses the product name, for example Campaign.sql.

    Note:

    `SQL20059W The materialized query table-name may not be used to optimize the processing of queries.`

    However, the materialized view is successfully created.

11. Specify the location where you want to save the script and click **Save**. If you change the name of the file, ensure that you use a name that clearly indicates the schemas that you selected.
12. Repeat steps 5 through 12 for each script that you want to generate.

Note: The Interact reporting schemas reference multiple data sources. Generate a separate SQL script for each data source.

SQL scripts by data source

Use separate SQL scripts to create views or materialized views for each data source.

The following table provides information about the scripts that you must generate for each data source, the resulting script name, and the scripts that must be run against the HCL Unica application database for creating views or materialized views:

Note:

- The table lists the default names for the data sources and generated scripts. Your names may be different.
- The product reporting schemas reference more than one data source. Generate a separate SQL script for each data source.

Installer placed report design files possess database connection tokens. You must update them for your system database. You must run `insightsdbutil.sh/bat` utility to update the same. You may have one or more data sources configurations for the report. Refer the following table for the same.

Reports	Configurations
Campaign Reports	CampaignDS
Interact Reports	InteractDTDS InteractETLDS InteractLearningDS InteractRTDS
Plan Reports	PlanDS
Deliver Reports	DeliverDS
Collaborate Reports	CampaignDS CollaborateDS CustomerDS

The following table provides information about the scripts that you must generate for each data source, the resulting script name, and the scripts that must be run against the Unica application database for creating views or materialized views.

Note: The table lists the default names for the data sources and generated scripts. Your names may be different.

The Unica Interact reporting schemas reference more than one data source. Generate a separate SQL script for each data source.

This three-columned table provides information about the reporting schema in one column, data source (default names) in the second column, and the script name (default names).

Reporting schema	Data source and default name	Default script name
All Unica Campaign reporting schemas	Unica Campaign system tables `campaignPartition1DS`	`Campaign.sql`, unless you generated separate scripts for each reporting schema. If you did, each script is named after the individual schema.
Unica Deliver Mailing Performance	Unica Deliver tracking tables, which are with the Unica Campaign system tables `campaignPartition1DS`	`Deliver_Mailing_ Performance.sql`
Unica Interact Deployment History, Unica Interact Performance, and Unica Interact Views	Unica Interact design time database `campaignPartition1DS`	`Interact.sql`
Unica Interact Learning	Unica Interact Learning tables `InteractLearningDS`	`Interact_Learning.sql`

Reporting schema	Data source and default name	Default script name
Unica Interact Run Time	Unica Interact run time database `InteractRTDS`	`Interact_Runtime.sql`

Creating and populating reporting tables for Unica Campaign and Unica Deliver

You can use SQL scripts to create and populate reporting tables for Unica Campaign. The reports application uses reporting tables to extract reportable data.

To create and populate reporting tables for Unica Campaign and Unica Deliver, complete the following steps.

1. Connect to Campaign system database.
2. Locate the SQL scripts that you generated and saved previously.
3. Use your database administration tools to run the appropriate script against the appropriate application databases for the report package that you are configuring.
4. For Campaign with a DB2 database, increase the DB2 heap size to at least 10240. The default heap size is 2048.
5. Use the following command to increase the heap size:

   ```
   db2 update db cfg for
           databasename using stmtheap 10240
   ```

   where databasename is the name of the Campaign database.
6. Use your database administration tools to populate the new tables with the appropriate data from the production system database.
7. Complete the following substeps. This step is not required for MariaDB datasource.
   a. Navigate to `<CAMPAIGN_HOME>/reports/ddl/<DBtype>` installation directory.
   b. Locate and execute `sp_whatifofferperf.sql`.

Note: For more than one partition, you must run the script for each partition in Campaign database.

c. For DB2, set `DB2_COMPATIBILITY_VECTOR` using following command. You must stop and start db2 server post parameter set:

```
db2set
        DB2_COMPATIBILITY_VECTOR=ORA
```

Continue with Setting up data synchronization (on page 35).

.

Copy the Campaign and Deliver reports folder in Unica Insights directory

Campaign installer places report design folders or files under Campaign installation directory.

Complete the following steps.

1. Create a folder `campaign/partitions/partitionN` under `<PLATFORM_HOME>/Insights/Reports`.
2. For Campaign reports, create a folder `Unica Dashboard/Campaign/partitions/partitionN` under `<PLATFORM_HOME>/Insights/Reports`.
3. Copy the Affinium Campaign and Affinium Campaign - Object specific Reports folders from `Campaign_Home/reports` and place it in `<PLATFORM_HOME>/Insights/Reports/campaign/partitions/partitionN`, where `N` is your partition number.
4. For Campaign dashboards, copy rpt design files from `Campaign_Home/reports/Unica Dashboards/Campaign` folder into `<PLATFORM_HOME>/Insights/Reports/Unica Dashboard/Campaign/partitions/partitionN`.
5. For Deliver reports, create a folder `Unica Dashboard/Deliver/partitions/partitionN` under `<PLATFORM_HOME>/Insights/Reports`.

6. For Deliver dashboards, copy `rpt` design files from `Campaign_Home/reports/Unica Dashboards/Deliver` folder into `<PLATFORM_HOME>/Insights/Reports/Unica Dashboard/Deliver/partitions/partitionN`.
7. For Campaign reports and dashboards, Deliver reports and dashboards, run `insightsdbutil.sh/bat` to update the datasource for the report design files from `<PLATFORM_HOME>/Insights/tools/bin`. For information on Unica Insights DB utility, see the Update data source in Unica Insights report design files using Unica Insights utility *(on page 39)* section.
    - You must configure the `DeliverDS` for Campaign system database.
8. Copy the following properties files (the source directory is mentioned below) to `<Unica_Home>\Platform\Insights\Reports\Resources\`.
    - `Campaign_Home/reports/Resources/Campaign/ *.properties`
    - `Campaign_Home/reports/Resources/Deliver/ *.properties`

Continue with Setting up data synchronization (on page 35).

Note: Even if the Campaign install host and Unica Insights application server host is same, it is recommended to copy report design files from install directory under `Platform_Home/Insights/Reports` and the folder structure must be `campaign/partitions/partitionN`.

Creating views or materialized views for Unica Campaign and Unica Deliver only:

You can use SQL scripts to create views or materialized views for Unica Campaign and Unica Deliver. The reports application uses views or materialized views to extract reportable data.

Note: For Oracle and DB2®, Unica Deliver requires materialized views. For SQL Server, Unica Deliver requires views.

To create views or materialized views for Unica Campaign or Unica Deliver, complete the following steps.

1. Locate the SQL scripts that you generated and saved previously.
2. Use the database administration tools to run the appropriate script against the appropriate application database(s) for the report package that you are configuring.

   Note: When you run a script that creates materialized views on a DB2® database, you may see the following error:

   ```
   SQL20059W The materialized query table-name may not be used to
   optimize the processing of queries.
   ```

   However, the materialized view is successfully created.
3. For Unica Campaign with a DB2® database, increase the DB2® heap size to at least 10240. The default heap size is 2048. Use the following command to increase the heap size:

   db2 update db cfg for databasename using stmtheap 10240

   where databasename is the name of the Unica Campaign database.

   Increasing the heap size ensures that Unica Insights does not display SQL error messages if a user selects all the campaigns when running a report such as the Financial Summary report.

Continue with Setting up data synchronization (on page 35).

Copy the Plan report folder in Unica Insights directory

While deriving Dashboard report design file name, Platform requires the database type. It is required that the following properties are populated with correct database type.

`Affinium|Plan|umoConfiguration|DBType`

Complete the following steps:

1. Create a folder "Plan" under `<PLATFORM_HOME>/Insights/Reports`.
2. Copy the folders - **Affinium Plan** and **Affinium Plan - Object specific Reports** folders from `<PLAN_HOME>/reports/Insights_Reports` and place it in `<PLATFORM_HOME>/Insights/Reports/Plan`.
3. Create Unica Dashboard folder under `<PLATFORM_HOME>/Insights/Reports`, if not already created.

   Note: Ensure that the folder under `<PLATFORM_HOME>/Insights/Reports/Unica Dashboard` is “Plan”.

4. Copy the respective DB rpt design files from the **Unica Dashboards** folder from `<PLAN_HOME>/reports/Insights_Reports` to `<PLATFORM_HOME>/Insights/Reports/Unica Dashboard/Plan`.
5. Ensure that you possess the execute permissions for the rpt design files.
6. Update the navigation URL and port under `Affinium|Insights|navigation`. The DBType should be displayed correctly under `Affinium|Plan|umoConfiguration`.
7. Navigate to `Affinium|Plan|umoConfiguration|reports` and change the following properties. For example:

reportsAnalysisSectionHome	Plan/Affinium Plan
reportsAnalysisTabHome	Plan/Affinium Plan - Object Specific Report

   Note: You must not include a slash (/) in the beginning of the path of these properties.

Copy the Collaborate report folder in Unica Insights directory

To copy the Collaborate report folder in Insights directory, complete the following steps:

1. Create a folder "Collaborate" under `<PLATFORM_HOME>/Insights/Reports`.
2. Copy the **Affinium Collaborate** folder from `<Collaborate_HOME>/Insights_Reports` and place it in `PLATFORM_HOME/Insights/Reports/campaign/partitions/<value of defaultCampaignPartition parameter>`. You will find the value of "`defaultCampaignPartition` configuration parameter in Platform configuration under `Affinium|Collaborate|UDM Configuration Settings|Campaign Integration`.
3. For Collaborate reports, run the `insightsdbutil.sh/bat` to update the datasource for the report design files from `<PLATFORM_HOME>/Insights/tools/bin`. For information on Unica Insights DB utility, see the Update data source in Unica Insights report design files using Unica Insights utility *(on page 39)* section.
4. Ensure that you possess the execute permissions for the rpt design files.
5. Update the navigation URL and port under `Affinium|Insights|navigation`.

Creating views or materialized views for Unica Interact only

You can use SQL scripts to create views or materialized views for Interact. The reports application uses views or materialized views to extract reportable data.

Before you create views or materialized views for Interact, verify that the language setting for the computer from where you run the `lookup_create` SQL script is enabled for UTF-8 encoding.

To create views or materialized views for Interact, complete the following steps.

1. Locate the SQL scripts that you generated and saved previously.
2. Use the database administration tools to run the appropriate script against the appropriate application database(s) for the report package that you are configuring.

Note: When you run a script that creates materialized views on a DB2 database, you may see the following error:

```
SQL20059W The materialized query
        table-name may not be used to optimize the processing of queries.
```

However, the materialized view is successfully created.

1. Under the Interact installation directory, in the tools subdirectory of report folder, find the `uari_lookup_create_<db_type>.sql` script for your database. For example, for SQL Server, script is available at `<INTERACT_HOME>/Interact/reports/tools/uari_lookup_create_MSSQL.sql`.
2. Run the `uari_lookup_create` script on the Interact design time database.
3. Locate the `uari_lookup_populate.sql` script under `<INTERACT_HOME>/Interact/reports/tools` path and run it on the Interact design time database.

   Note: Ensure that the database tool that you have used commits the changes. For example, you may require to set the auto-commit option of the database to `true`.

Continue with Setting up data synchronization (on page 35).

Copy the Interact reports folder in Unica Insights directory

The Interact installer places report design folders/files under Platform installation directory. You must copy reporting folders for each product reports manually to the server where Insights.war is deployed. Even if the Interact install host and Unica Insights application server host is same, it is recommended to copy report design files from install directory to a new directory of application server host. Unica Interact reports are placed under partition_home directory.

Complete the following steps.

1. Create a folder `campaign/partitions/partitionN` under `<PLATFORM_HOME>/Insights/Reports`.
2. Copy the Unica Interact and Unica Interact - Object specific Reports folders from `Interact_Home/reports` and place it in `<PLATFORM_HOME>/Insights/Reports/campaign/partitions/partitionN`, where `N` is your partition number.
3. Create `Unica Dashboard/Interact/partitions/partitionN` folder under `<PLATFORM_HOME>/Insights/Reports`.
4. Copy rpt design files from `Interact_Home/reports/Unica Dashboards/interact` folder into `<PLATFORM_HOME>/Insights/Reports/Unica Dashboard/Interact/partitions/partitionN`.

Continue with Setting up data synchronization (on page 35).

Note: Even if the Campaign install host and Unica Insights application server host is same, it is recommended to copy report design files from install directory under `Platform_Home/Insights/Reports` and the folder structure must be `campaign\partitions\partitionN`.

Creating and populating reporting tables

You must import following views in design time database and runtime database. You must use your own tools for this step. The SQL Generator does not generate the SQL for you.

- Execute the views on Campaign database. Interact installer lays down database scripts under the Interact installation location which contains these views. Scripts are available at `<INTERACT_HOME>/reports/ddl/<dbtype>/InteractDT.sql`.
- Execute the views on Interact Runtime database. Interact installer lay down database script under the Interact installation location which contains these views. Scripts are available at `<INTERACT_HOME>/reports/ddl/<dbtype>/InteractRT.sql.`

Note: In case if you face any issue while running script through CLI, then you must use IBM Data Studio Client or before running the script you may require to remove leading or trailing spaces from the SQL statement given in the file and close all statements with semicolon.

Setting up data synchronization

Ensure that you use the database administration tools to schedule regular data synchronization between the production databases of the HCL Unica application and the materialized views.

To set up data synchronization, use the following guidelines depending on your application and database type.

- For Unica Campaign, use the scheduled Extraction, Transformation, and Load (ETL) method, or any custom method to schedule regular data synchronization between the production databases and the new reporting tables.
- For Unica Interact on Oracle or DB2 databases, use the scheduled Extraction, Transformation, and Load (ETL) method or any custom method to schedule regular data synchronization between the production databases and the new reporting tables.
- For Unica Interact on a SQL server, use the scheduled Extraction, Transformation, and Load (ETL) method, or any custom method to schedule regular data synchronization between the production databases and the new reporting tables.

For Unica Deliver only: Stored procedures for delta processing

Unica Deliver reports require staging tables that are associated with the Unica Deliver system tables. The system tables are part of the Unica Campaign schema. You must periodically run stored procedures to process message response data for use in Unica Deliver reports.

For more information about schema changes, see Unica Deliver System Tables and Data Dictionary.

The initial setup for the Unica Deliver stored procedures relies on the following database scripts:

- `acer_indexes_dbname.sql`
- `acer_tables_dbname.sql`
- `acer_scripts_dbname.sql`

Note: If you observe any issues while executing the mentioned sql files, use the script terminator based on the database client. If your database client shows errors for `acer_scripts_dbname.sql`, create the procedures one after the other.

The database scripts are in the `Campaign\reports\Deliver-ddl` directory for the Oracle, IBM® DB2®, and Microsoft™ SQL Server databases.

The scripts set up indexes, tables, views, and stored procedures. The stored procedures refresh message data to populate the staging tables. The batch procedures must be run regularly to populate the staging tables. Running the stored procedures is referred to as delta processing.

The initial runs of the Unica Deliver stored procedures can take a long time to complete, depending on the amount of data that is contained in your tables. Subsequent delta processing also can take a long time to complete. You can significantly reduce the processing time by limiting the number of mailing instances (containers) that are processed by the stored procedures.

By default, data is processed for the past 90 days. However, you can change the default value before or after you run the SQL scripts for Unica Deliver.

Example for Oracle

The following examples for an Oracle database illustrate the changes that you can make to the acer_tables script to limit processing to the previous 30 days:

Note: The changes include modifying the `UARE_MAILING_MASTER` view.

Definition of the current view

```
CREATE VIEW UARE_MAILING_MASTER AS
(
(SELECT UCC_CONTAINER.CAMPAIGNID,UCC_CONTAINER.CONTAINERID,
substr(UCC_CONTAINERATTR.STRINGVALUE,1,100) AS CAMPAIGN_NAME,
UCC_CONTAINER.CONTAINERNAME AS MAILING_INST,
UCC_CONTAINER.CREATED AS MAILING_CREATED,
UCC_CONTAINER.CONTAINERTYPEID CONTAINERTYPEID,
UCC_CONTAINER.CONTCHANNELTYPEID CONTCHANNELTYPEID
FROM
UCC_CONTAINER,UCC_CONTAINERATTR
WHERE
UCC_CONTAINERATTR.CONTAINERID=UCC_CONTAINER.CONTAINERID AND
UCC_CONTAINERATTR.ATTRIBUTENAME='CampaignName' AND
UCC_CONTAINER.CREATED >= sysdate - 91
)
```

Definition of the modified view

```
CREATE VIEW UARE_MAILING_MASTER AS
(
SELECT UCC_CONTAINER.CAMPAIGNID, UCC_CONTAINER.CONTAINERID,
substr(UCC_CONTAINERATTR.STRINGVALUE,1,100) AS CAMPAIGN_NAME,
UCC_CONTAINER.CONTAINERNAME AS MAILING_INST, UCC_CONTAINER.CREATED AS
MAILING_CREATED FROM UCC_CONTAINER,UCC_CONTAINERATTR WHERE
UCC_CONTAINERATTR.CONTAINERID=UCC_CONTAINER.CONTAINERID AND
UCC_CONTAINERATTR.ATTRIBUTENAME='CampaignName'
AND
UCC_CONTAINER.CREATED >= sysdate - 30
)
```

To view all available report data, modify the `UARE_MAILING_MASTER` view to remove the date filter from the view. Then, refresh all Oracle or DB2® materialized views. For example, in the sample view creation that is shown above, remove the following line:

```
UCC_CONTAINER.CREATED >= sysdate - 30
```

For Unica Deliver only: Creating stored procedures, staging tables, and indexes

After you install or upgrade reporting templates, you must run specific SQL scripts before you generate Unica Deliver reports. The SQL scripts create stored procedures and staging tables.

The `Campaign/reports/Deliver-ddl` directory is on the server. This directory contains the following database scripts for Oracle, IBM® DB2®, and Microsoft™ SQL Server:

- `acer_indexes_dbname.sql`
- `acer_tables_dbname.sql`
- `acer_scripts_dbname.sql`

Note: If you observe any issues when executing the mentioned sql files, use the script terminator based on the database client. If your database client shows errors for `acer_scripts_dbname.sql`, create the procedures one after the other.

Run the following scripts against the Unica Campaign database in the order listed.

1. `acer_indexes_dbname.sql`

   Make sure that you allow sufficient time for the script to complete. The time depends on the volume of data that is stored in the Unica Deliver system tables.
2. `acer_tables_dbname.sql`

   This script creates the delta processing staging tables in the Unica Deliver system schema.
3. `acer_scripts_dbname.sql`

Important: For DB2® databases, change the termination character from ; (semicolon) to ! (exclamation point).

This script creates the stored procedures that you must configure after you install reports for Unica Deliver.

4. Navigate to `Campaign\reports\tools` directory, under your Campaign installation and locate the following scripts.
   - `uare_lookup_create_DB_type.sql`
   - `uare_lookup_populate*.sql`: Run the script against your Unica Campaign system tables database for all languages.

     Note: Unica Insights supports four characters for locale. For example, in the `Unica_home\Campaign\reports\tools` directory, if there are two lookup populate SQL files for the french locale:
     - `uare_lookup_populate_fr.sql`
     - `uare_lookup_populate_fr_FR.sql`

     you must execute `uare_lookup_populate_fr_FR.sql` and ignore the other file.

Note: You must configure the stored procedures to run on a regular basis to populate the staging tables. You cannot see data in the reports until you run the stored procedures for Unica Deliver reports.

For more information about running and scheduling the stored procedures, see For Unica Deliver only: How to schedule and run stored procedures (on page 44).

Update data source in Unica Insights report design files using Unica Insights utility

Once you copy db specific files as per the details mentioned above, you must update data sources which are required to run the reports using insightsdbutil.sh/bat located at `<PLATFORM_HOME>/Insights/tools/bin`.

The parameters required to run this utility are:

- `ds` for product datasource
- `bPath` for Unica Insights report design file path
- `DBType` for Unica Insights design file database type
- `URL` for JDBC url for database
- `user` for database user
- `pwd` for database user password

Sample command to update the parameters:

```
insightsDBUtil -ds=<datasource name>
        -bPath=<Report folder path> -DBType=<databasetype> [-URL=JDBC
 connection URL>] [-user=<database
        user>] [-pwd=<database password>] [-locale=<Locale>]
```

The following command allows you to view the utility usage.

`insightsDBUtil -h`

The following command allows you to view the usage for localize.

`insightsDBUtil -h -locale=<Locale>`

Instructions:

For the first time, all parameters are mandatory.

For UNIX:

- Grant read, write, and execute permissions to `-bPath=<Report folder path>` for the installation user.
- Grant execute permission to the installation user using the following command.

  `chmod 755 insightsDBUtil.sh`

Users are required to provide all parameters and JDBC URL in expected format as mentioned in help for specific database.

- Update password : -ds, -DBType, -URL -bPath and -pwd is mandatory
- Update user name : -ds, -DBType, -URL -bPath and -user is mandatory

Note: If username and password string contain any special characters, then please enclose them with double quotes(“).

Parameters help:

- `bPath` : Unica Insights report design file path :mandatory
- `ds` : This parameter is for product data source : mandatory
- `ds`: Available options (case sensitive)
- `ds` : Product : Campaign -> CampaignDS
- `ds` : Product : Deliver -> DeliverDS
- `ds` : Product : Plan -> PlanDS
- `ds` : Product : Interact -> InteractDTDS
- `ds` : Product : Interact -> InteractRTDS
- `ds` : Product : Interact -> InteractLearningDS
- `ds` : Product : Interact -> InteractETLDS
- `ds` : Product : Collaborate -> CampaignDS
- `ds` : Product : Collaborate -> CollaborateDS
- `ds` : Product : Collaborate -> CustomerDS
- `DBType` : This parameter is for Database type : mandatory

DBType available options

- DBType : Database : Sql Server -> `sqlserver`
- DBType : Database : DB2 -> `db2`
- DBType : Database : Oracle -> `oracle`
- DBType : Database : MariaDB -> `mariadb`
- DBType : Database : OneDB -> `onedb`

URL

- URL : This parameter is required for JDBC connection

URL is mandatory for the first time and when any parameter of JDBC URL changes.

URL Available options

- URL : `Database : Sql Sever -> "jdbc:sqlserver://<HOST>:<PORT>;instance=<INSTANCE/OPTIONAL>;databaseName=<DB NAME>"`

  Note: In case of Microsoft SQL Server, the URL must be enclosed within quotation marks (`" "`), as shown in the earlier examples. If you do not enclose the URL within quotation marks, you will see an error.

- URL : `Database : DB2 -> jdbc:db2://<HOST>:<PORT>/<sid>`
- URL : `Database : Oracle -> jdbc:oracle:thin:@<HOST>:<PORT>:<sid>`
- URL : `Database : MariaDB -> JDBC url:jdbc:<HOST>:<PORT>/<DB Name>`
- URL : `Database : OneDB -> jdbc:informix-sqli://<HOST>:<PORT>/<DB Name>:DELIMIDENT=Y;DB_LOCALE=en_us.utf8`
- user : `Database user name`
- pwd :`Database password`

Note: In case of DB2, you must use the following URL if you are updating Interact report design files.

URL:

```
Database : DB2 -> jdbc:db2://<HOST>:<PORT>/<sid>:
      useJDBC4ColumnNameAndLabelSemantics=false;
```

Note: In case of MariaDB, you must use the following URL if you are updating Campaign report design files.

URL:

```
JDBC url:jdbc:mariadb://10.115.145.106:3306/up121x
```

Granting permissions for stored procedures for DB2

Before you configure stored procedures for DB2, you must grant permissions.

To grant permissions, complete the following steps.

1. Enable the registry by completing the following steps:
    - Set the `DB2_ATS_ENABLE registry` variable to one of the following values:
        - `YES`
        - `TRUE`
        - `1`
        - `ON`
    - Restart the DB2 database after you set the variable.
2. Create the SYSTOOLSPACE table space.

    Users who belong to the SYSADM or SYSCTRL group can create this space. Use the following query to verify that the space exists:

    ```
    SELECT TBSPACE FROM
            SYSCAT.TABLESPACES WHERE TBSPACE = 'SYSTOOLSPACE'
    ```
3. Grant permissions. In the following examples, substitute the values that are appropriate for your environment.
    - DELIVER: Database that contains the Deliver system tables
    - USER1: Owner of the Deliver database
    - DB2ADMIN: DB2 administrative user
    - Administrator: Super user
4. Connect to DB2 as an administrative user and run the following grant commands:
    - Connect to DB2 as an administrative user and run the following grant commands:
    - db2 GRANT DBADM ON DATABASE TO USER DB2ADMIN
    - db2 GRANT DBADM ON DATABASE TO USER USER1

- db2 grant all on table SYSTOOLS.ADMINTASKS to USER1
- db2 grant all on table SYSTOOLS.ADMINTASKS to DB2ADMIN

5. If the SYSPROC.ADMIN_TASK_ADD table exists, run the following grant commands:
   - ```
     db2 grant execute on
             procedure SYSPROC.ADMIN_TASK_ADD to USER1
     ```
   - `db2 grant execute on procedure SYSPROC.ADMIN_TASK_ADD to DB2ADMIN`

Guidelines for configuring stored procedures

- The database must be DB2 version 9.7.8 or higher.
- Create new jobs in DB2 Administrative Task Scheduler (ATS).
- Schedule the jobs to run at least daily. You must `schedule sp_runid` to run at least 10 minutes before the other scripts.

For Unica Deliver only: How to schedule and run stored procedures

Unica Deliver reports use the data that is contained in staging tables, which are populated by stored procedures. The stored procedures perform a delta refresh operation. Run the stored procedures at least once per day. If you run the procedures more frequently, the delta refresh method prevents multiple concurrent runs.

The following table provides information about the stored procedures and the tasks that they complete:

Table 2. Stored procedures for Deliver

This two-columned table lists the stored procedures in the first column and explains the tasks that the procedures complete in the second column.

Stored procedure	Task
sp_runid	Creates a unique run identifier. The list of the run IDs is stored in the UARE_Runid table.
     ```

Table 2. Stored procedures for Deliver

This two-columned table lists the stored procedures in the first column and explains the tasks that the procedures complete in the second column.

(continued)

Stored procedure	Task
sp_update_ucc_tables_stats	Updates statistics for the ucc_* tables. You can run this script before the sp_populate_* scripts.
sp_populate_mailing_contacts	Processes the mailing contact data that is received since the previous run of stored procedures.
sp_populate_mailing_responses	Processes the mailing response data that is received since the previous run of stored procedures.
sp_populate_sms_contacts	If the SMS feature is enabled: Processes the SMS contact data that is received since the previous run of stored procedures.
sp_populate_sms_responses	If the SMS feature is enabled: Processes the SMS response data that is received since the previous run of stored procedures.
sp_populate_WhtsApp_contacts	If the WhatsApp feature is enabled: Processes the WhatsApp contact data that is received since the previous run of stored procedures.
sp_populate_WhtsApp_responses	If the WhatsApp feature is enabled: Processes the WhatsApp response data that is received since the previous run of stored procedures.
sp_get_delta_mailing_contacts	Called internally by sp_populate_mailing_contacts procedure. Responsible for retrieving the mailing

Table 2. Stored procedures for Deliver

This two-columned table lists the stored procedures in the first column and explains the tasks that the procedures complete in the second column.

(continued)

Stored procedure	Task
	contacts that were sent since the previous run of the stored procedures.
sp_generate_mailing_contacts	Called internally by the sp_populate_mailing_contacts procedure. Responsible for retrieving the mailing and link level counts on contacted customers for the mailings that were run since the previous run of the stored procedures.
sp_get_delta_mailing_responses	Called internally by sp_populate_mailing_responses procedure. Responsible for retrieving the responses that were received since the previous run of the stored procedures.
sp_generate_mailing_responses	Called internally by sp_populate_mailing_responses procedure. Responsible for retrieving mailing and link level responses since the previous run of the stored procedures.
sp_get_delta_sms_contacts	Called internally by sp_populate_sms_contacts procedure. Responsible for retrieving SMS since the previous run of the stored procedures.
sp_generate_sms_contacts	Called internally by sp_populate_sms_contacts procedure. Responsible for retrieving the mailing and link level counts on contacted customers since the previous run of the stored procedures.
sp_get_delta_sms_responses	Called internally by sp_populate_sms_responses procedure. Responsible for retrieving SMS re-

Table 2. Stored procedures for Deliver

This two-columned table lists the stored procedures in the first column and explains the tasks that the procedures complete in the second column.

(continued)

Stored procedure	Task
	sponses since the previous run of the stored procedures.
sp_generate_sms_responses	Called internally by sp_populate_sms_responses procedure. Responsible for retrieving the mailing and link level SMS responses since the previous run of the stored procedures.
sp_get_delta_WhtsApp_contacts	Called internally by sp_populate_WhtsApp_contacts procedure. Responsible for retrieving WhatsApp messages since the previous run of the stored procedures.
sp_generate_WhtsApp_contacts	Called internally by sp_populate_WhtsApp_contacts procedure. Responsible for retrieving the mailing and link level counts on contacted customers since the previous run of the stored procedures.
sp_get_delta_WhtsApp_responses	Called internally by sp_populate_WhtsApp_responses procedure. Responsible for retrieving WhatsApp responses since the previous run of the stored procedures.
sp_generate_WhtsApp_responses	Called internally by sp_populate_WhtsApp_responses procedure. Responsible for retrieving the mailing and link level WhatsApp responses since the previous run of the stored procedures.

Table 2. Stored procedures for Deliver

This two-columned table lists the stored procedures in the first column and explains the tasks that the procedures complete in the second column.

(continued)

Stored procedure	Task
sp_populate_mobile_responses	Processes the mobile response data that was received since the previous run of stored procedures.
sp_get_delta_mobile_responses	Called internally by sp_populate_mobile_responses procedure. Responsible for retrieving the responses that were received since the previous run of the stored procedures.
sp_generate_mobile_responses	Called internally by sp_populate_mobile_responses procedure. Responsible for retrieving mobile responses since the previous run of the stored procedures.

Guidelines for running stored procedures

Use the following guidelines when you run the stored procedures:

- You must create the stored procedures for your database by using the scripts that are provided with the installation files.
- Consider the size of the tables and indexes in your installation. Larger tables require more time to update. Allow sufficient time to process the contact and response data. The initial runs are likely to require more time to complete than subsequent runs.
- Because the stored procedures can run for an extended amount of time, consider running the procedures at times of reduced system activity, such as overnight.
- You can reduce the amount of the time that is required to refresh the reports data by limiting the scope of the reports data processed.
- You must schedule the following procedures to run at least 10 minutes after scheduling sp_runid:

- sp_populate_mailing_contacts
- sp_populate_mailing_responses
- sp_populate_sms_contacts
- sp_populate_sms_responses
- sp_populate_WhtsApp_contacts
- sp_populate_WhtsApp_responses
- sp_populate_mobile_responses

When the scripts have run successfully, they display a final return code of 0.

Sample configuration of stored procedures for Oracle

Use the following guidelines when you configure stored procedures for the Oracle database.

Guidelines for configuring stored procedures

- recommends using Oracle Automatic Memory Management (AMM). For more information, go to http://docs.oracle.com/cd/B28359_01/server.111/b28310/memory003.htm.
- Create stored procedures by using a database utility, such as SQL Plus.
- Schedule the sp_runid procedure to run at least 10 minutes before the other scripts.

Example for creating a run identifier

The following example illustrates how to create a job and generate a run identifier. The example also illustrates the job ID when the job completes.

The example shows how to get a job number every day at 21:00 hours without an end date. The jobs start on November 29, 2014.

```
declare
jobno number;

BEGIN
DBMS_JOB.submit (job =>:jobno,
what => 'sp_runid;',
```

```
next_date => to_date('29-Nov-2014 21:00','DD-MON-YYYY HH24:MI' ),
interval => 'sysdate+1');
commit;
END;
/
```

Example for processing email contact data

The following example shows how to schedule a batch job to process contact data. The job runs at 21:10 hours every day.

```
declare
jobno number;

BEGIN
DBMS_JOB.submit (job =>:jobno,
what => 'sp_populate_mailing_contacts;',
next_date => to_date('29-Nov-2014 21:10','DD-MON-YYYY HH24:MI' ),
interval => 'sysdate+1');
commit;
END;
/
```

Example for processing email response data

The following example shows how to schedule a batch job to process response data. The job runs at 21:10 hours every day.

```
declare
jobno number;

BEGIN
DBMS_JOB.submit (job =>:jobno,
what => 'sp_populate_mailing_responses;',
next_date => to_date('29-Nov-2014 21:10','DD-MON-YYYY HH24:MI' ),
```

```
interval => 'sysdate+1');
commit;
END;
/
```

Example for processing SMS contact data

Important: The SMS feature is not a part of the default Reports offering, and you must buy a license for the feature separately. However, the delta placement takes place regardless of whether you bought the SMS feature.

The following example shows how to get a job number every day at 21:00 hours without an end date. The jobs start on November 29, 2014.

```
BEGIN
DBMS_JOB.submit (job =>:jobno,
what => 'sp_populate_SMS_contacts;',
next_date => to_date('29-Nov-2014 21:10','DD-MON-YYYY HH24:MI' ),
interval => 'sysdate+1');
commit;
END;
/
```

Example for processing SMS response data

The following example shows how to get a job number every day at 21:00 hours without an end date. The jobs start on November 29, 2014.

```
BEGIN
DBMS_JOB.submit (job =>:jobno,
what => 'sp_populate_SMS_responses;',
next_date => to_date('29-Nov-2014 21:10','DD-MON-YYYY HH24:MI' ),
interval => 'sysdate+1');
commit;
```

```
END;
/
```

Example for processing WhatsApp contact data

Important: The WhatsApp feature is not a part of the default Reports offering, and you must buy a license for the feature separately. However, the delta placement takes place regardless of whether you bought the WhatsApp feature.

The following example shows how to get a job number every day at 21:00 hours without an end date. The jobs start on November 29, 2014.

```
BEGIN
DBMS_JOB.submit (job =>:jobno,
what => 'sp_populate_WhtsApp_Contacts;',
next_date => to_date('29-Nov-2014 21:10','DD-MON-YYYY HH24:MI' ),
interval => 'sysdate+1');
commit;
END;
/
```

Example for processing WhatsApp response data

The following example shows how to get a job number every day at 21:00 hours without an end date. The jobs start on November 29, 2014.

```
BEGIN
DBMS_JOB.submit (job =>:jobno,
what => 'sp_populate_WhtsApp_Responses;',
next_date => to_date('29-Nov-2014 21:10','DD-MON-YYYY HH24:MI' ),
interval => 'sysdate+1');
commit;
END;
/
```

Example for processing Mobile response data

The following example shows how to get a job number every day at 21:00 hours without an end date. The jobs start on November 29, 2014.

```
BEGIN
DBMS_JOB.submit (job =>:jobno,
what => 'sp_populate_mobile_Responses;',
next_date => to_date('29-Nov-2014 21:10','DD-MON-YYYY HH24:MI' ),
interval => 'sysdate+1');
commit;
END;
/
```

Sample configuration of stored procedures for Microsoft™ SQL Server

Use the following guidelines when you configure stored procedures for the Microsoft™ SQL Server database.

Guidelines for configuring stored procedures

- Use the SQL Server Agent to create new jobs for each stored procedure.
- Schedule the jobs to run at least daily. You must schedule sp_runid to run at least 10 minutes before the other scripts.
- For each job in the SQL Server Agent interface, you must specify the step type as Transact-SQL script (T-SQL) and select the Unica Campaign database.

Example for creating a run identifier

The following example shows how to create a run identifier.

```
DECLARE @return_value int
EXEC @return_value = [dbo].[SP_RUNID]
```

```
SELECT 'Return Value' = @return_value
GO
```

Example for processing email contact data

The following example shows how to process email contact data. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
DECLARE @return_value int
EXEC @return_value = [dbo].[SP_POPULATE_MAILING_CONTACTS]
SELECT 'Return Value' = @return_value
GO
```

Example for processing email response data

The following example shows how to process email response data. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
DECLARE @return_value int
EXEC @return_value = [dbo].[SP_POPULATE_MAILING_RESPONSES]
SELECT 'Return Value' = @return_value
GO
```

Example for processing SMS contact data

The following example shows how to process SMS contact data.

```
DECLARE @return_value int
EXEC @return_value = [dbo].[SP_POPULATE_SMS_CONTACTS]
SELECT 'Return Value' = @return_value
GO
```

Example for processing SMS response data

The following example shows how to process SMS response data.

```
DECLARE @return_value int
EXEC @return_value = [dbo].[SP_POPULATE_SMS_RESPONSES]
SELECT 'Return Value' = @return_value
GO
```

Example for processing WhatsApp contact data

The following example shows how to process WhatsApp contact data.

```
DECLARE @return_value int
EXEC @return_value = [dbo].[SP_POPULATE_WHTSAPP_CONTACTS]
SELECT 'Return Value' = @return_value
GO
```

Example for processing WhatsApp response data

The following example shows how to process WhatsApp response data.

```
DECLARE @return_value int
EXEC @return_value = [dbo].[SP_POPULATE_WHTSAPP_RESPONSES]
SELECT 'Return Value' = @return_value
GO
```

Example for processing Mobile response data

The following example shows how to process Mobile response data.

```
DECLARE @return_value int
EXEC @return_value = [dbo].[SP_POPULATE_MOBILE_RESPONSES]
SELECT 'Return Value' = @return_value
GO
```

Sample configuration of stored procedures for DB2

Use the following guidelines when you configure stored procedures for the DB2® database.

Guidelines for configuring stored procedures

- The database must be DB2® version 9.7.8 or higher.
- Create new jobs in DB2® Administrative Task Scheduler (ATS)
- Schedule the jobs to run at least daily. You must schedule sp_runid to run at least 10 minutes before the other scripts.

Example for creating a run identifier

The following example shows how to get a job number every day at 20:50 hours without an end date.

```
call SYSPROC.ADMIN_TASK_ADD('RunID_Job',null,null,
null,'50 20 * * *','USER1','SP_RUNID',null,null,null)
```

Example for processing email contact data

The following example shows how to schedule a batch job to process contact data. In this example, the job runs at 21:00 hours every day. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
call SYSPROC.ADMIN_TASK_ADD('Email_Contact_Job',null,null,null,'00 21 * *
 *',
'USER1','SP_POPULATE_MAILING_CONTACTS',null,null,null)
```

Example for processing email response data

The following example shows how to schedule a batch job to process response data. In this example, the job runs at 21:00 hours every day. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
call SYSPROC.ADMIN_TASK_ADD('Email_Response_Job',null,null,
null,'00 21 * * *','USER1','SP_POPULATE_MAILING_RESPONSES',null,
null,null)
```

Example for processing SMS contact data

The following example shows how to schedule a batch job to process contact data. In this example, the job runs at 21:00 hours every day. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
call SYSPROC.ADMIN_TASK_ADD('SMS_Contact_Job',null,null,null,'00 21 * * *',
'USER1','SP_POPULATE_SMS_CONTACTS',null,null,null)
```

Example for processing SMS response data

The following example shows how to schedule a batch job to process response data. In this example, the job runs at 21:00 hours every day.

```
call SYSPROC.ADMIN_TASK_ADD('SMS_Response_Job',null,null,
null,'00 21 * * *','USER1','SP_POPULATE_SMS_RESPONSES',null,
null,null)
```

Example for processing WhatsApp contact data

The following example shows how to schedule a batch job to process contact data. In this example, the job runs at 21:00 hours every day. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
call SYSPROC.ADMIN_TASK_ADD('WHTSAPP_Contact_Job',null,null,null,'00 21 * *
 *',
'USER1','SP_POPULATE_WHTSAPP_CONTACTS',null,null,null)
```

Example for processing WhatsApp response data

The following example shows how to schedule a batch job to process response data. In this example, the job runs at 21:00 hours every day.

```
call SYSPROC.ADMIN_TASK_ADD('WhtsApp_Response_Job',null,null,
null,'00 21 * * *','USER1','SP_POPULATE_WHTSAPP_RESPONSES',null,
null,null)
```

Example for processing Mobile response data

The following example shows how to schedule a batch job to process response data. In this example, the job runs at 21:00 hours every day.

```
call SYSPROC.ADMIN_TASK_ADD('Mobile_Response_Job',null,null,
null,'00 21 * * *','USER1','SP_POPULATE_MOBILE_RESPONSES',null,
null,null)
```

Granting permissions for stored procedures for DB2

Before you configure stored procedures for DB2®, you must grant permissions.

To grant permissions, complete the following steps.

1. Enable the registry by completing the following steps:
    a. Set the **DB2_ATS_ENABLE** registry variable to one of the following values:
        - **YES**
        - **TRUE**
        - **1**
        - **ON**
    b. Restart the DB2® database after you set the variable.
2. Create the `SYSTOOLSPACE` table space.

    Users who belong to the SYSADM or SYSCTRL group can create this space. Use the following query to verify that the space exists:

    ```
    SELECT TBSPACE FROM SYSCAT.TABLESPACES WHERE TBSPACE = 'SYSTOOLSPACE'
    ```
3. Grant permissions. In the following examples, substitute the values that are appropriate for your environment.
    - Deliver: Database that contains the Unica Deliver system tables
    - USER1: Owner of the Deliver database
    - DB2ADMIN: DB2® administrative user
    - Administrator: Super user

4. Connect to DB2® as an administrative user and run the following grant commands:
    - db2 GRANT DBADM ON DATABASE TO USER DB2ADMIN
    - db2 GRANT DBADM ON DATABASE TO USER USER1
    - db2 grant all on table SYSTOOLS.ADMINTASKS to USER1
    - db2 grant all on table SYSTOOLS.ADMINTASKS to DB2ADMIN
5. If the `SYSPROC.ADMIN_TASK_ADD` table exists, run the following grant commands:
    - db2 grant execute on procedure SYSPROC.ADMIN_TASK_ADD to USER1
    - db2 grant execute on procedure SYSPROC.ADMIN_TASK_ADD to DB2ADMIN

Sample configuration of stored procedures for MariaDB

Use the following guidelines when you configure stored procedures for the MariaDB database.

Guidelines for configuring stored procedures

Use the MariaDB events to create new jobs for each stored procedure.

- Schedule the jobs to run at least daily. You must schedule `sp_runid` to run atleast 10 minutes before the other scripts.
- Create events for Unica Campaign database.

Example for creating a run identifier

The following example shows how to create a run identifier.

```
CREATE EVENT SP_RUNS
 ON SCHEDULE EVERY 1 DAY
    STARTS  '2020-11-20 20:30:00'
    DO
      CALL SP_RUNID();
```

Example for processing email contact data

The following example shows how to process email contact data. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
CREATE EVENT EMAIL_CONTACT
 ON SCHEDULE EVERY 1 DAY
    STARTS  '2020-11-20 20:40:00'
    DO
      CALL SP_POPULATE_MAILING_CONTACTS();
```

Example for email response data

The following example shows how to process email response data. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
CREATE EVENT EMAIL_RESPONSE
 ON SCHEDULE EVERY 1 DAY
    STARTS  '2020-11-20 20:40:00'
    DO
      CALL SP_POPULATE_MAILING_RESPONSES();
```

Example for processing SMS contact data

The following example shows how to process SMS contact data. Schedule the job to runat least 10 minutes after the job that generates the run identifier.

```
CREATE EVENT SMS_CONTACT
 ON SCHEDULE EVERY 1 DAY
    STARTS  '2020-11-20 20:40:00'
    DO
      CALL SP_POPULATE_SMS_CONTACTS();
```

Example for processing SMS response data

The following example shows how to process SMS response data. Schedule the job to runat least 10 minutes after the job that generates the run identifier.

```
CREATE EVENT SMS_RESPONSE
 ON SCHEDULE EVERY 1 DAY
    STARTS  '2020-11-20 20:40:00'
    DO
      CALL SP_POPULATE_SMS_RESPONSES();
```

Example for processing WhatsApp contact data

The following example shows how to process WhatsApp contact data. Schedule the job to runat least 10 minutes after the job that generates the run identifier.

```
CREATE EVENT WHTSAPP_CONTACT
 ON SCHEDULE EVERY 1 DAY
    STARTS  '2020-11-20 20:40:00'
    DO
      CALL SP_POPULATE_WHTSAPP_CONTACTS();
```

Example for processing WhatsApp response data

The following example shows how to process WhatsApp response data. Schedule the job to runat least 10 minutes after the job that generates the run identifier.

```
CREATE EVENT WHTSAPP_RESPONSE
 ON SCHEDULE EVERY 1 DAY
    STARTS  '2020-11-20 20:40:00'
    DO
      CALL SP_POPULATE_WHTSAPP_RESPONSES();
```

Example for processing Mobile response data

The following example shows how to process Mobile response data. Schedule the job to runat least 10 minutes after the job that generates the run identifier.

```
CREATE EVENT MOBILE_RESPONSE
 ON SCHEDULE EVERY 1 DAY
```

```
    STARTS  '2020-11-20 20:40:00'
    DO
      CALL SP_POPULATE_MOBILE_RESPONSES();
```

Sample configuration of stored procedures for OneDB

Use the following guidelines when you configure stored procedures for the OneDB database.

Guidelines for configuring stored procedures

Use the OneDB tasks to create new jobs for each stored procedure. These tasks are created using sysadmin database.

- Run following command before initializing the Deliver database.

  `Set environment the DB_LOCALE and GL_USEGLS = 1`
- Schedule the jobs to run atleast daily. You must schedule sp_runid to run atleast 10 minutes before the other scripts.
- Create tasks in Sysadmin database.

Example for creating a run identifier

The following example shows how to create a task to generate a run identifier.

```
INSERT INTO ph_task
(
tk_name,tk_description,tk_type,tk_execute,tk_start_time,tk_frequency,
 tk_attributes
)
VALUES
(
'SP_RUNS',
'This task is to invoke procedure to generate runids for the deliver delta
 refresh runs.',
'TASK',
```

```
'EXECUTE PROCEDURE < Deliver_Database name>@<DB Server
 Instance>:SP_RUNID()',
'20:30:00','1 0:00:00', 0
);
```

Example for processing email contact data

The following example shows how to process email contact data. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
INSERT INTO ph_task
(
tk_name,tk_description,tk_type,tk_execute,tk_start_time,tk_frequency,
 tk_attributes
)
VALUES
(
'MAILING_CONTACTS',
'This task is to invoke procedure to populate mailing contacts.',
'TASK',
'EXECUTE PROCEDURE <
 Deliver_Database_name>@<DBServer_Instance>:SP_POPULATE_MAILING_CONTACTS()'
,
'20:40:00','1 0:00:00', 0
);
```

Example for email response data

The following example shows how to process email response data. Schedule the job to run atleast 10 minutes after the job that generates the run identifier.

```
INSERT INTO ph_task
(
```

```
tk_name,tk_description,tk_type,tk_execute,tk_start_time,tk_frequency,
 tk_attributes
)
VALUES
(
'MAILING_RESPONSES',
'This task is to invoke procedure to populate mailing responses.',
'TASK',
'EXECUTE PROCEDURE <
 Deliver_Database_name>@<DBServer_Instance>:SP_POPULATE_MAILING_RESPONSES()
',
'20:40:00','1 0:00:00', 0
);
```

Example for processing SMS contact data

The following example shows how to process SMS contact data. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
INSERT INTO ph_task
(
tk_name,tk_description,tk_type,tk_execute,tk_start_time,tk_frequency,
 tk_attributes
)
VALUES
(
'SMS_CONTACTS',
'This task is to invoke procedure to populate SMS contacts.',
'TASK',
'EXECUTE PROCEDURE <
 Deliver_Database_name>@<DBServer_Instance>:SP_POPULATE_SMS_CONTACTS()',
'20:40:00','1 0:00:00', 0
```

```
);
```

Example for processing SMS response data

The following example shows how to process SMS response data. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
INSERT INTO ph_task
(
tk_name,tk_description,tk_type,tk_execute,tk_start_time,tk_frequency,
 tk_attributes
)
VALUES
(
'SMS_RESPONSES',
'This task is to invoke procedure to populate SMS responses.',
'TASK',
'EXECUTE PROCEDURE
 <Deliver_Database_name>@<DBServer_Instance>:SP_POPULATE_SMS_RESPONSES()',
'20:40:00','1 0:00:00', 0
);
```

To check scheduled tasks creation in sysadmin database, run the following query.

```
SELECT * from ph_task;
```

To check run status, check in sysadmin database and run the following query.

```
select * from ph_run;
```

You can also check in each channel process data in Deliver database using following query.

```
Select * from uare_delta_refresh_log order by runid desc;
```

Example for processing WhatsApp contact data

The following example shows how to process WhatsApp contact data. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
INSERT INTO ph_task
(
tk_name,tk_description,tk_type,tk_execute,tk_start_time,tk_frequency,
 tk_attributes
)
VALUES
(
'WHTSAPP_CONTACTS',
'This task is to invoke procedure to populate WhatsApp contacts.',
'TASK',
'EXECUTE PROCEDURE <
 Deliver_Database_name>@<DBServer_Instance>:SP_POPULATE_WHTSAPP_CONTACTS()'
,
'20:40:00','1 0:00:00', 0
);
```

Example for processing WhatsApp response data

The following example shows how to process WhatsApp response data. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
INSERT INTO ph_task
(
tk_name,tk_description,tk_type,tk_execute,tk_start_time,tk_frequency,
 tk_attributes
)
VALUES
(
```

```
'WHTSAPP_RESPONSES',
'This task is to invoke procedure to populate WhatsApp responses.',
'TASK',
'EXECUTE PROCEDURE
 <Deliver_Database_name>@<DBServer_Instance>:SP_POPULATE_WHTSAPP_RESPONSES(
)',
'20:40:00','1 0:00:00', 0
);
```

To check scheduled tasks creation in sysadmin database, run the following query.

```
SELECT * from ph_task;
```

To check run status, check in sysadmin database and run the following query.

```
select * from ph_run;
```

You can also check in each channel process data in Deliver database using following query.

```
Select * from uare_delta_refresh_log order by runid desc;
```

Example for processing Mobile response data

The following example shows how to process Mobile response data. Schedule the job to run at least 10 minutes after the job that generates the run identifier.

```
INSERT INTO ph_task
(
tk_name,tk_description,tk_type,tk_execute,tk_start_time,tk_frequency,
 tk_attributes
)
VALUES
(
'MOBILE_RESPONSES',
'This task is to invoke procedure to populate Mobile responses.',
'TASK',
```

```
'EXECUTE PROCEDURE
 <Deliver_Database_name>@<DBServer_Instance>:SP_POPULATE_MOBILE_RESPONSES()
',
'20:40:00','1 0:00:00', 0
);
```

To check scheduled tasks creation in sysadmin database, run the following query.

```
SELECT * from ph_task;
```

To check run status, check in sysadmin database and run the following query.

```
select * from ph_run;
```

You can also check in each channel process data in Deliver database using following query.

```
Select * from uare_delta_refresh_log order by runid desc;
```

How to configure Unica Insights to use HCL Unica authentication

User authorization for Unica Insights folders and reports

Unica Insights reporting functionality is authorized to users who possess **ReportsSystem** and **ReportsUser** roles.

Reporting schemas

You must create reporting views to implement reporting for Unica Campaign, Unica Interact, and Unica Deliver. Reports can extract reportable data in the reporting views. The report packages for Unica Campaign, Unica Interact, and Unica Deliver include reporting schemas that the Reporting SQL Generator uses to produce SQL scripts that create reporting views.

For Unica Campaign and Unica Interact, you must customize the schema templates so that the data that you want to include in your reports is represented. You can run the Reporting SQL Generator after you customize the schema templates. You can run the SQL scripts that the SQL Generator generated and run the scripts on your application databases.

You cannot customize the Unica Deliver reporting schemas. However, you must generate the SQL script that builds the reporting views or tables and then run the scripts on the Deliver databases.

The Report SQL Generator

The Report SQL Generator uses the reporting schemas to determine the analytic logic that is necessary to extract data from the database for the Unica application. The Report SQL Generator then generates the SQL script that creates views or reporting tables that implement the logic and that enable business intelligence tools to extract reportable data.

During installation and configuration, the system implementers configured data source properties that identify the Unica application databases. The Report SQL Generator connects to the application databases to complete the following tasks:

- Validate scripts that create views or materialized views
- Determine the correct data types to use in scripts that create reporting tables

If the JNDI data source names are incorrect or missing, the Report SQL Generator cannot validate the scripts that create the reporting tables.

Also in the case of Unica Platform, Unica Campaign and Unica Interact deployed on different Application server instances/profiles you are required to add Unica Campaign, Unica Interact Runtime and Unica Interact Learning Data Source in the Unica Platform Application instance/profile.

Reporting deployment options

You can choose a deployment option when you run the Report SQL Generator tool.

When you run the Report SQL Generator tool, you specify whether you want the script to create views, materialized views. The deployment option you use depends on the amount of data that is contained in your system. See the Install and configure Unica Insights reports (on page 11) and Loading templates for the Reports SQL Generator (on page 22) sections to get the details on the supported databases for different products.

- For smaller implementations, reporting views that directly query the production data might perform sufficiently for your needs. If they do not, try materialized views.
- For medium-sized implementations, use either materialized views on the production system database or set up reporting tables in a separate database.
- For large implementations, configure a separate reporting database.

Materialized views and Microsoft™ SQL Server

The reports application does not support materialized views for Microsoft™ SQL Server.

In SQL Server, materialized views are called "indexed views." However, the definition that creates an index in a view on SQL Server cannot use certain aggregations, functions, and options that the reporting views include. Therefore, if you are using a SQL server database, use views or reporting tables.

Note: For Unica Deliver, you must use views.

Unica Deliver and Oracle

If your installation includes Unica Deliver and your database is Oracle, you must use materialized views or reporting tables.

Unica Deliver and DB2®

If your installation includes Unica Deliver and your database is DB2®, you must use materialized views or reporting tables.

Data synchronization

When you deploy with materialized views or reporting tables, determine how frequently you want to synchronize the data with the production system data. Then, use your database administration tools to schedule data synchronization processes to refresh the reporting data regularly.

For Unica Deliver, the materialized views are automatically refreshed when you run the Unica Deliver delta refresh stored procedures. For more information, see For Unica Deliver only: How to schedule and run stored procedures (on page 44).

Campaign reports and reporting schemas

You can customize the reporting schemas in the Campaign Report Package by adding contact or response metrics, attributes, or response types.

The reporting schemas in the Campaign Report Package can be customized in the following ways.

- Add contact or response metrics.
- Add custom campaign, offer, or cell attributes.
- Add response types.
- Configure the audience level for performance reports.
- Create reporting schemas for additional audience levels.

The following table maps the individual Unica Insights BI reports provided in the Unica Campaign Reports Package to the reporting schemas that support them.

Table Mapping of Unica Insights reports to the reporting schema

	Campaign Views schema	**Campaign-Custom Attributes schema**	**Campaign Performance schema**	**Offer Performance schema**	**CampaignOffer Response Breakout**	**Offer Contact Status Breakout**
What If Offer Financial Summary report	X	X		X		
Campaign Detailed Offer Response Breakout	X		X		X	
Offer Response Breakout,	X		X		X	

Dashboard version						
Campaign Financial Summary by Offer (Actual)	X	X	X			
Campaign Return on Investment Comparison	X	X	X			
Campaign Offer Performance by Month	X		X			
Campaign Performance Comparison	X		X			
Campaign Response Rate Comparison	X		X			

	Campaign Views schema	**Campaign-Custom Attributes schema**	**Campaign Performance schema**	**Offer Performance schema**	**CampaignOffer Response Breakout**	**Offer Contact Status Breakout**
Campaign Performance Comparison with Revenue	X		X			

Campaign Performance Comparison by Initiative	X		X			
Campaign Performance Summary by Cell	X		X			
Campaign Performance Summary by Cell with Revenue	X		X			
Campaign Performance Summary by Cell and Initiative	X		X			
Campaign Performance Summary by Offer	X		X			
Campaign Performance Summary by Offer with Revenue	X		X			
Campaign Revenue Comparison by Offer	X		X			
Campaign Summary	X					
Offer Campaign Listings	X					

Offer Performance Metrics	X			X		
Offer Performance by Day	X			X		
Offer Responses for Last 7 Days	X			X		

	Campaign Views schema	**Campaign-Custom Attributes schema**	**Campaign Performance schema**	**Offer Performance schema**	**CampaignOffer Response Breakout**	**Offer Contact Status Breakout**
Offer Performance Comparison	X			X		
Offer Response Rate Comparison	X			X		
Offer Performance Summary by Campaign	X		X	X		

The following reports rely on the standard set of custom contact and response metric attributes that are provided in Unica Campaign:

- What If Offer Financial Summary
- Campaign Detailed Offer Response Breakout
- Campaign Financial Summary by Offer (Actual)
- Campaign Performance Comparison with Revenue

- Campaign Performance Summary by Cell with Revenue
- Campaign Performance Summary by Offer with Revenue

Unica Deliver Reports and Reporting Schemas

Several reports, such as Message Overview report, Detailed Link report, Deliver Reports Processing Overview, and SMS Message Summary Report are available in the Unica Deliver Reports Package.

Table 3. Unica Deliver reports and reporting schemas

Report name	Mailing performance schema
Message Overview report	X
Detailed Link report	X
Detailed Link by Cell report	X
Detailed Bounce report	X
A/B Testing Performance Report	X
Deliver Reports Processing Overview	X

Interact reports and reporting schemas

The Interact Report Package reports are supported by HCL reporting schemas. You can customize the schemas to specify time periods, configure audience levels, and create extra performance reporting schemas.

You can customize the reporting schemas in the Interact Report Package in the following ways:

- Specify calendar time periods for performance reports.
- Configure the audience level for performance reports.
- Create extra performance reporting schemas for extra audience levels.

The following table maps the individual Unica Insights BI reports provided in the Interact Reports Package to the reporting schemas that support them.

	Interactive View schema	**Interact Performance View schema**	**Interactive Channel / Campaign Deployment History**	**Interact Runtime View schema**	**Interact Learning View schema**
A/B Test Performance Analysis Report		X			
Campaign - Interactive Channel Deployment History	X		X		
Campaign - Interactive Cell Performance Over Time	X	X		X	
Campaign - Interactive Cell Performance by Offer	X	X		X	
Campaign - Interactive Offer Performance Over Time	X	X		X	
Campaign - Interactive Offer Performance by Cell	X	X		X	
Campaign - Interactive Offer Learning Details	X				X
Interactive Cell Lift Analysis	X	X		X	X

Interactive Channel - Channel Deployment History	X		X		
Interactive Channel - Channel Event Activity Summary report	X			X	
Interactive Channel - Channel Interaction Point Performance Summary	X	X		X	
Interactive Channel - Channel Treatment Rule Inventory	X				
Interaction Point Performance	X	X		X	

Stored procedures for the Interact Event Pattern report

The Interact Event Pattern report uses the data that is contained in staging tables, which are populated by stored procedures. The stored procedures perform a delta refresh operation.

Interact Event Pattern report data is processed in two steps:

1. The Interact ETL process transforms the audience blob data into ETL database tables.
2. The reports aggregator aggregates the data incrementally for each pattern type in preconfigured parallel execution. This is specific Interact reports pack.

Both processes are integrated with the database trigger on the UACI_ETLPATTERNSTATERUN table. This trigger is fired on successful ETL execution and submits database jobs to aggregate the reports data.

The following tables provide information about the stored procedures and the tasks that they complete.

Stored procedures for the Interact Event Pattern report

Stored procedure	Task
SP_GENERATE_PATTERN_MATCHALL	Called internally by the **SP_POPULATE_PATTERN_MATCHALL** procedure. Responsible for retrieving the data for Match All patterns that were executed since the previous run of the stored procedures.
SP_GENERATE_PATTERN_COUNTER	Called internally by the **SP_POPULATE_PATTERN_COUNTER**procedure. Responsible for retrieving the data for Counter patterns that were executed since the previous run of the stored procedures.
SP_GENERATE_PATTERN_WC	Called internally by the SP_POPULATE_PATTERN_WC procedure. Responsible for retrieving the data for Weighted Counter patterns that were executed since the previous run of the stored procedures.
SP_POPULATE_PATTERN_MATCHALL	Processes the Match All Pattern type data that was received since the previous run of stored procedures.
SP_POPULATE_PATTERN_COUNTER	Processes the Counter Pattern type data that was received since the previous run of stored procedures.
SP_POPULATE_PATTERN_WC	Processes the Weighted Counter Pattern type data that was received since the previous run of stored procedures.

Stored procedure	Task

SP_UPDATE_UACI_TABLES_STATS	Called by the trigger to update the database statistics and the database jobs are submitted for reports data aggregation. Updates the statistics for the following ETL tables: • UACI_ETLPATTERNSTATE • UACI_ETLPATTERNSTATEITEM • UACI_ETLPATTERNEVENTINFO
SP_POPULATE_PATTERN_LOCK (p_parallel_degree)	Updates the UARI_PATTERN_LOCKtable with the degree of parallel execution configured. p_parallel_degree is the degree at which the aggregation processes run in parallel. For MARIA DB, the jobs are not supported and the stored procedures are run sequentially p_parallel_degree is always 1 for MARIA DB
SP_AGGR_RUN_STATUS	Called by the Interact ETL process before the start of the aggregation process to check the lock status of the running stored procedures. Run against the **UARI_PATTERN_LOCK**table.
SP_REFRESH_PATTERNINFO	For Oracle and DB2 only Refreshes the UARI_PATTERNSTATE_INFO table to get the state and audience level information for the ICs and Categories. The call to this procedure is given by a trigger before the aggregation procedures start. As Mviews are not supported for SQL Server, this procedure is not applicable for SQL Server.
SP_UARI_REBIND_PACKAGES	For DB2 only Rebinds the packages that are associated with the aggregation trigger and procedures. Called from the trigger after the SP_UPDATE_UACI_TABLES_STATS procedure call.

Stored procedure	Task
SQ_UARI_RUN	Creates a unique run identifier. The list of the run IDs is stored in the UARI_RUNS table.

For SQL Server, RunID is generated by using the IDENTITY property on the RunId column, which generates new IDs on each run.

Database trigger

Stored procedure	Task
TR_AGGREGATE_DELTA_PATTERNS	After the UACI_ETLPATTERNSTATERUN table is updated with the value 3, the trigger is invoked by submitting the jobs that call the stored procedures for data aggregation. For OneDB: create database trigger TR_AGGREGATE_DELTA_PATTERNS along with store procedure AGGREGATE_DELTA_PATTERNS.
For OneDB only, create stored procedure AGGREGATE_DELTA_PATTERNS	It updates UARI_PROCESSED_PATTERNS table, and create three jobs to invoke the following sub store procedures: SP_POPULATE_PATTERN_MATCHALL, SP_POPULATE_PATTERN_COUNTER, EXECUTE PROCEDURE SP_POPULATE_PATTERN_WC. Two ph_task are used for each invoke store procedure call, in order to block and tracking the job states. One ph_task before invoke call and one ph_task after completing the invoke call.

The ETL process

On the first run, ETL does not insert any values against the respective PatternID in the UARI_DELTA_PATTERNS table because all patterns are new or delta. The reports

aggregation process collects all PatternID from the ETL tables and inserts them into the `UARI_DELTA_PATTERNS` table.

The ETL process calls the `SP_AGGR_RUN_STATUS` procedure. The `SP_AGGR_RUN_STATUS` procedure checks the `UARI_PATTERN_LOCK` table for running jobs based on the JobID.

JobID value	**Reason**
Y	The job is running. Scenarios are running or failed.
N	Failed job.

The ETL process always checks the status of the reports aggregation by checking the status of the submitted jobs. If the ETL finds reports aggregation running, the ETL does not start its run. The ETL starts again according to the schedule.

The ETL process checks the `UARI_PATTERN_LOCK` table for the number of JobIDs with value `Y`. The ETL process starts only if no JobIDs have the value `Y`. If any JobIDs have the value `Y`, then the ETL process is skipped and runs at the next scheduled interval. For more information about the ETL process, see the Unica Interact Administrator Guide.

From the second run onwards, the ETL process updates the UARI_DELTA_PATTERNS table with the update flag for the updated PatternID:

- For updated data, the PatternID is marked with U.
- For deleted data, the PatternID is marked with D.
- For newly added data, the PatternID is identified by the reports aggregation code and is marked with P.

The aggregation process is run for only the PatternIDs that are marked with the U or D flag.

Enabling stored procedures for the Interact Event Pattern report

In addition to the steps that you followed to enable reports, you must enable the Interact Event Pattern report. The Interact Event Pattern report uses the delta refresh process for data aggregation so that reports can render faster.

Administrative Task Scheduler (ATS) depends on table space to store historical data and configuration information. To verify if the table space is defined in the database or to create the table space.

To execute the scheduled job from Task Scheduler, the database must be active.

The ADMIN_TASK_STATUS is an administrative view that is created when the `ADMIN_TASK_ADD` procedure is called for the first time. These views must exist in the database. If the views are missing, create the views with the help of your database administrator. You must have access privilege on the ADMIN_TASK_STATUS administrative view.

To enable stored procedures for the Interact Event Pattern report, complete the following steps.

1. Browse the `<Interact_Home>/reports/ddl/interact-ddl/<DB Type>/` folder.
2. For DB2, set the following parameters:
    - `db2set DB2_COMPATIBILITY_VECTOR=ORA`
    - `db2set DB2_ATS_ENABLE=YES`
3. When the instance is restarted, you must activate DB2 by running the following commands in the order listed:
    - db2 force application all Stop the application on this instance.
    - db2stop force Stop DB2.
    - db2start Start the database.
    - db2 activate db <dbname> Explicitly activate the database. You should see this message:DB20000I The ACTIVATE DATABASE command completed successfully.
    - db2 list active databases Verify that the database is activated. You must see a similar output.

```
Active Databases
Database name = <dbname> Applications connected currently = 0
Database path   = /data04/<DB instance
        owner>/NODE0000/SQL00001/
```

4. On the ETL database, run the following scripts in the order listed:

- `acir_tables_<DB Type>.sql`
- `acir_scripts_<DB Type>.sql`

Note: You must run the `acir_tables_<DB Type>.sql` script if it was not run earlier.

Note: If an exception is thrown after you run the `acir_scripts_db2.sql` script on the target database, delete the trigger and create it with the appropriate database user.

For SQL server, run the `acir_jobs_sqlserver.sql` script. The script creates database jobs for degree 2. To change the degree, see “Changing the degree of parallel execution for the Interact Event Pattern report”.

Note: Ensure that the SQL Server Agent service is running.

5. Before the ETL process starts, you must create parallel batch degree records in the UARI_PATTERN_LOCK table. Run one of the following commands on the ETL database to create these records.
   - For Oracle: execute SP_POPULATE_PATTERN_LOCK(2)
   - For DB2: call SP_POPULATE_PATTERN_LOCK(2)
   - For SQL Server: EXEC [dbo].[SP_POPULATE_PATTERN_LOCK]

     `@p_parallel_degree = 2`

     In this example, 2 is the degree at which the aggregation processes run in parallel.

     The UARI_PATTERN_LOCK table is populated with the stored procedures with the degree value. The degree value is configurable. Increase the degree of parallel execution for the Interact Event Pattern report aggregation process to reduce the elapse time. If the degree is set to a higher value, hardware resource requirements increase proportionally. The number of procedures that are run for data aggregation depend on the degree value.
   - For MariaDB:

```
CALL SP_POPULATE_PATTERN_LOCK(1);
```

The parallel degree execution is not configurable for MariaDB and its value is always 1.

- For OneDB: CALL SP_POPULATE_PATTERN_LOCK(2);

6. Optional: While the ETL feature is running, you can disable the trigger so that reports aggregation is not called. To disable the trigger and turn off the reports aggregation process, run one of the following commands depending on your database type.
    - For DB2: You can contact IBM support.
    - For Oracle: alter trigger TR_AGGREGATE_DELTA_PATTERNS disable;
    - For SQL Server: Disable Trigger TR_AGGREGATE_DELTA_PATTERNS on uaci_etlpatternstaterun
    - For MariaDB: By default, the trigger is enabled. It must be dropped using the following command

      ```
      DROP TRIGGER IF EXISTS TR_AGGREGATE_DELTA_PATTERNS
      ```
    - For OneDB: By default, the trigger is enabled. It must be dropped using the following command.

      ```
      DROP TRIGGER if exists TR_AGGREGATE_DELTA_PATTERNS;;
      ```
7. Optional: To enable the trigger and turn on the reports aggregation process, run one of the following commands depending on your database type.
    - For DB2: You can contact IBM support.
    - For Oracle: alter trigger TR_AGGREGATE_DELTA_PATTERNS enable;
    - For SQL Server: Enable Trigger TR_AGGREGATE_DELTA_PATTERNS on uaci_etlpatternstaterun;
    - For MariaDB: By default, the trigger is enabled.

      If it is dropped, then it can be enabled by the trigger creation command.

      See the `acir_scripts_mariadb.sql` for the trigger creation command.
    - For OneDB: By default, the trigger is enabled.

      If it is dropped, then it can be enabled by the trigger creation command.

      See the `acir_scripts_onedb.sql` along with the create command

      `PROCEDURE AGGREGATE_DELTA_PATTERNS`.

Note: When ETL completes successfully, the status in the UACI_ETLPATTERNSTATERUN table is updated as 3, and the trigger TR_AGGREGATE_DELTA_PATTERNS is called. The trigger calls the stored procedure for the set parallel degree. When the system aggregates all data for the first time, the report aggregation process may take a longer time than subsequent aggregations.

Changing the degree of parallel execution for the Interact Event Pattern report

The degree of parallel execution value is configurable. Increase the degree of parallel execution for the Interact Event Pattern report aggregation process to reduce the elapse time. If the degree is set to a higher value, hardware resource requirements also increase proportionally

Configure the degree at which the aggregation process runs so that the Interact Event Pattern report can render faster.

To configure database jobs for a degree value of 3, complete one of the following steps, depending on your database:

- For Oracle: Run the execute SP_POPULATE_PATTERN_LOCK(3) command against the Interact ETL database.
- For IBM DB2: Run the call SP_POPULATE_PATTERN_LOCK(3) command against the Interact ETL database.
- For SQL Server: Run the default acir_jobs_sqlserver.sql script to create database jobs for degree value 1 and 2. The patterns with degree values 1 and 2 are aggregated in the UARI_PROCESSED_PATTERNS table.
- For MariaDB: Run the `CALL SP_POPULATE_PATTERN_LOCK(1);`

  The parallel degree execution is not configurable for MariaDB and its value is always 1.
- For OneDB: Run the CALL SP_POPULATE_PATTERN_LOCK(2);

To modify the degree to 3 for the Match All Pattern, copy the sample code for degree 1 and complete the following steps:

1. Set the value of @job_name to JOB_MA_3.
2. Set the value of @p_parallel_degree to 3.

Run the following command against the Interact ETL database.

```
DECLARE
@jobId BINARY(16),
@status int,
@schedule_name varchar(16), @dbname varchar(100)
set @dbname= (SELECT DB_NAME());

EXEC msdb.dbo.sp_add_job @job_name=N'JOB_MA_3', @job_id = @jobId OUTPUT;

EXEC msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'first',
 @command=N'EXEC [dbo].[SP_POPULATE_PATTERN_MATCHALL]
@p_parallel_degree = 3', @database_name=@dbname;

EXEC msdb.dbo.sp_add_jobserver @job_id=@jobId, @server_name=N'(local)';
GO
```

You can create the degree for the Counter Pattern and the Weighted Counter Pattern and run the commands against the ETL database.

To modify the degree to 3 for the Counter Pattern, copy the sample code for degree 1 and complete the following steps:

1. Set the value of @job_name to JOB_C_3.
2. Set the value of @p_parallel_degree to 3.

To modify the degree to 3 for the Weighted Counter Pattern, copy the sample code for degree 1 and complete the following steps:

1. Set the value of @job_name to JOB_WC_3.
2. Set the value of @p_parallel_degree to 3

Log messages in the UARI_DELTA_REFRESH_LOG table for the Interact Event Pattern report

The UARI_DELTA_REFRESH_LOG table contains logging information for all procedures.

Aggregation process status

To verify the status of the aggregation process, look for the following text: MESSAGE_LINE:

```
<patterntype> patterns delta
        refresh started for parallel degree <degree value>
```

```
<patterntype> patterns delta refresh completed for parallel degree <degree
    value>
```

where:

- `<patterntype>` is Match All, Counter, or Weighted Counter.
- `<degree value>` is the value with which patterns are processed in parallel. For example, when the degree value is 2, the following messages are logged:

```
MatchAll patterns delta refresh started for parallel degree
MatchAll patterns delta refresh completed for parallel degree
MatchAll patterns delta refresh started for parallel degree
MatchAll patterns delta refresh completed for parallel degree 2
```

UARI_PATTERNSTATE_INFO table

To verify if the UARI_PATTERNSTATE_INFO table is refreshed, look for the following text:

MESSAGE_LINE:

```
Pattern State information refresh procedure started
```

```
--The procedure to refresh the data in UARI_PATTERNSTATE_INFO is running.
```

MESSAGE_LINE:

```
Pattern State information refresh procedure completed
```

```
--The procedure to refresh the data in UARI_PATTERNSTATE_INFO is completed.
```

Lock flags reset by the SP_AGGR_RUN_STATUS procedure

To verify if the lock flags are reset by the SP_AGGR_RUN_STATUS procedure, look for the following text:

`MESSAGE_LINE:`

`patterns lock has been reset for parallel degree <degree value>`

The OBJECT column of the UARI_DELTA_REFRESH_LOG table contains the procedure name for which the lock is reset.

where: <degree value> is the value with which patterns are processed in parallel. For example, when the degree value is 1, the following message is logged:

`patterns lock has been reset for parallel degree 1`

For DB2 only: rebinding of packages

For DB2 only: To verify that rebinding of the packages completed, look for the following text:

MESSAGE_LINE:

```
Rebind of packages started
```

```
--Rebinding of the packages started
```

MESSAGE_LINE:

```
Rebinding of packages completed successfully on <datetime>
```

```
--Rebinding of the packages completed successfully on the given date.
```

Statistics updated on ETL tables

To verify if the statistics were updated on the ETL tables, look for the following text:

MESSAGE_LINE:

```
Table statistics update
        started
```

```
--Update statistics on the ETL
        tables is in process
```

MESSAGE_LINE:

```
Statistics on Tables
        UACI_ETLPATTERNSTATE UACI_ETLPATTERNSTATEITEM
```

```
UACI_ETLPATTERNEVENTINFO and
        indexes have been updated successfully on <datetime>
```

```
--Statistics are updated on the
        mentioned ETL tables on the given date.
```

Degree of parallel execution

To verify the degree of parallel execution, look for the following text:

MESSAGE_LINE:

```
Pattern aggregation processing Parallel degree is set to <degree value>
```

```
--Parallel degree with which report aggregation will run is set to <degree
 value>.
```

For example, when the degree value is 2, the following message is logged:

```
Pattern aggregation processing Parallel degree is set to 2.
```

Format of the Unica Insights Reports

Use the styles included with the global report style sheet, GlobalReportStyles.css, to format the report page.

Item	Style
Text	Tahoma font
Report title text	Tahoma 18 point
Page footer text	Tahoma 8 point
Field Set labels	Tahoma 8 point

List report styles

Use the styles included with the global report style sheet, GlobalReportStyles.css, to format list reports. The following table shows formatting from the GlobalStyleSheet.css style sheet for list reports.

Item	Style
Cells	1 px silver line(#c8c8c8) borders (unless otherwise noted)
Column header	Light gray background(F5F5F5); 2px Grey(#c8c8c8) line separates column header from rest of table
Summary header rows (list headers)	Light yellow background
Total row at bottom	Dark gray background; 2px Grey line separates row from rest of table

Item	CSS class name	Style
Page - Header	Ph	font-family: “Tahoma”; font-size: 18pt; font-weight: bold;
Page - Footer	Pf	padding-top:10px; font-size:8pt;

		font-weight:bold;
Table - List column title cell	Lt	text-align:left; border: 1px solid #c8c8c8; background-color: #f5f5f5; background-image: none!important; font-weight:normal; vertical-align: top; padding: 10px 20px; font-family: “Tahoma”; color: #444444; font-size: 14px;
Item	CSS class name	Style
Table - List column body cell interior	lci	border: none; background-color: F5F5F5!important; text-align: right; padding: 3px 5px; vertical-align: middle;
Table - List column body cell	lc	border-top:1px solid #C8C8C8; border-bottom:1px solid #C8C8C8; border-left: 1px solid #C8C8C8; border-right:1px solid #C8C8C8; padding: 3px 5px; text-align: left; vertical-align: middle; font-family: “Tahoma”;

Table - List column body measure cell	lm	vertical-align: top; border:1px solid #c8c8c8; border-right: 0; border-left: 0; padding: 3px 5px; text-align: right;
Crosstab - Totals first row	tr	border-left: 2px solid black; background-color: #f5f5f5 !important; font-weight: bold; padding: 3px 5px;
Complex table total -new class added	ctth	color: #444444; background-color: #f5f5f5; border-bottom:2px solid #c8c8c8; padding: 3px 5px; border-left: 2px solid #c8c8c8;
Table totals row	ttr	color: #444444; font-weight: bold; background-color: #f5f5f5; padding: 3px 5px;
Table totals row	ctr	color: #444444; font-weight: bold; border-left:2px solid #c8c8c8; background-color: #f5f5f5; border-bottom:1px solid #c8c8c8;

Table totals header	cth	color: #444444; border-bottom:2px solid #c8c8c8; border-left:1.5px solid white; border-right:1.5px solid white; font-weight: 100;
List - Inner header cell	ih	border-top:1px solid #c8c8c8; border-bottom:1px solid #c8c8c8; padding: 3px 5px; vertical-align: middle;
List - Outer header cell	oh	font-weight: bold; vertical-align: top; border: 1px solid #c8c8c8; border-right: 0; border-left: 0; padding: 3px 5px; word-break:keep-all; background-color: #f5f5f5;
Item	CSS class name	Style
Outer header cell with top border	ohl	font-weight: bold; vertical-align: top; background-color: #ddd; padding: 3px 5px; word-break:keep-all; border-top:2px solid black; border-left:1.5px solid #c8c8c8; border-right: 5pt solid #c8c8c8;

		border-style:solid; border-bottom:none;
Crosstab	xt	border: 2px solid #C8C8C8; color: #444444; empty-cells: show; font-size: 16px;
Crosstab - Member label cell	ml	font-style: normal !important; color: black; font-weight: 300; height: 30px; border-left: none; border-right: none; border-bottom:1px solid #c8c8c8;
Crosstab - Member label cell	cht	vertical-align: top; background-color:transparent; padding: 3px 5px; text-align: left;
Crosstab - Member value cell	mv	vertical-align: top; white-space: nowrap; border: 1px solid #c8c8c8; padding: 3px 5px; text-align: right; border-left:none; border-right:none; vertical-align: top; white-space: nowrap;

		padding: 3px 5px; text-align: right;
Field set	fs	display: -moz-inline-block; display: inline; text-align: left; f ont-size:8pt; margin-bottom: 15px; color : #444444;
Chart	ch	border:1pt solid #c8c8c8;
Chart - Title	ct	font-weight:bold;
Chart - Axis labels	al	font-size:10pt;
Chart - Axis title	at	font-weight:bold; text-align:center; font-size:10pt; color:#444444;
Item	CSS class name	Style

Chart - Chart Palette	In XML Report Specification	Before the closing chart tag (`</combinationChart>`) in the XML Report Specification, paste the following lines:
		`<chartPalette>`
		`<chartColor value="#6B80BE"/>`
		`<chartColor value="#DDBB4D"/>`
		`<chartColor value="#9CAC61"/>`
		`<chartColor value="#78BF79"/>`

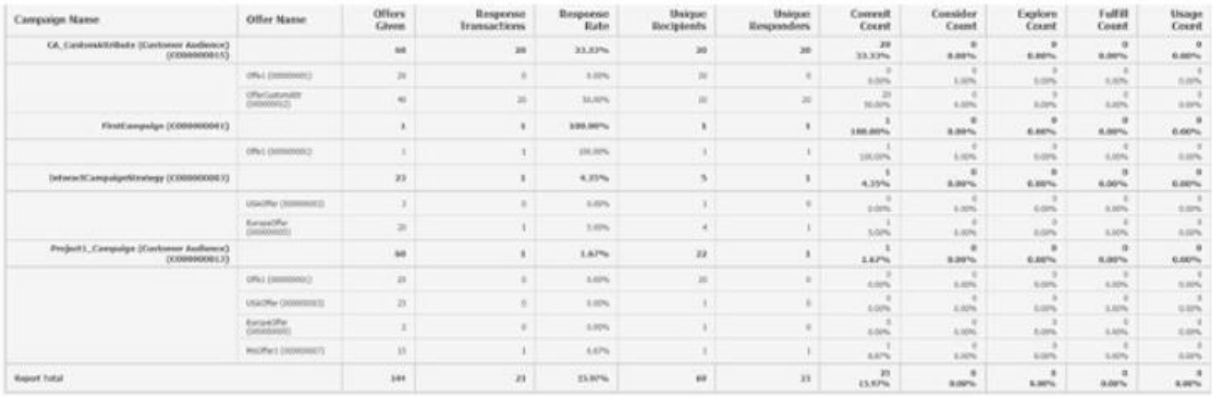
		<chartColor value="#7D5AA6"/> <chartColor value="#efc100"/> <chartColor value="#aeb8b8"/> <chartColor value="#4178be"/> </chartPalette>
Hyperlink	.hy	color: #037bbf; font-size: 14px; font-family: “tahoma”;
Totals first Column	tf	border-left: 2px solid black; background-color: #f5f5f5 !important; font-weight: bold; padding: 3px 5px;
Complex table total	ctt	color: #444; background-color: #f5f5f5 !important; border-left:2px solid black; border-bottom: 1px solid #c8c8c8; padding-left: 5px 5px;
Complex table total row	cttr	color: #444444; background-color: #f5f5f5; font-weight: bold; border-bottom:1px solid #c8c8c8;
List	ls	border: 1px solid #c8c8c8; color: #444444; empty-cells: show; margin-top: 10px; font-size: 14px;

Hover selection class	hoverSelection	background-color: transparent !important; color: #444444 !important;

Additionally, when you create a new list report, use the following guidelines to match existing reports:

- Use List Headers (not List Footers) to display summarization at the object level.
- Manually right-justify any numbers that are displayed in List Headers. Unlike List Footers, List Headers are not separated into the outer component and summary component, which use a right-justified style by default. When you summarize information into a List Header, you must complete the extra step and right-justify the values.
- Optionally, add 2px solid Grey borders to group columns.

The following example shows a list report that does not use the global styles:



Date formats for English versions

If you use a globalized version of the HCL Unica reports package, you see a different date format in your list reports depending on which locale you use. Unica Insights list reports use the date style medium.

The following table shows the date formats for list reports for all available locales.

Unica Insights list reports date formats for globalized versions

Locale	Unica Insights list reports date format example

English	Apr 1, 2020

Crosstab report styles

Use the styles included with the global report style sheet, GlobalReportStyles.css, to format crosstab reports.

The following table shows formatting from the GlobalStyleSheet.css style sheet for crosstab reports:

Item	**Style**
Cells	Light Grey (#f5f5f5) background; 1 px silver line borders
Measure cell (upper left)	2 px Grey (#C8C8C8) line separates the cell from the rest of the crosstable
Outer level totals	Gray/offwhite background

Additionally, when you create a new list report, use the following guidelines to match the existing reports.

- Use 2px grey borders to separate summarization from measures.
- Use 2 px grey borders to group logical column groupings.
- As a general guideline, avoid summarizing both columns and rows in the same report.

The following example shows a crosstab report that uses the global styles and has 1.5 px borders that are applied to show column groupings.

Chart styles

Use the styles included with the global report style sheet, GlobalReportStyles.css, to format charts.

The following table shows formatting from the GlobalStyleSheet.css style sheet for charts:

Charts obtain the following formatting from the GlobalStyleSheet.css.

Item	Style
Charts	1 pt light gray border
Titles and labels	10 point bold font

Additionally, when you create a new chart, use the following guidelines to match the existing chart reports.

- Use the default width, unless there is more than one chart on the report. When you include multiple charts in a single report, set the chart width to 750px.
- To use gradients and color palettes, copy and paste the strings from the table in “Global report styles” into the XML report specification.
- As a general guideline, select the chart type based on the data that you expect to be returned.

    - Use line graphs as the chart type only when you can guarantee the report retrieves continuous data.
    - If there are multiple series, a stacked bar works better than a non-stacked bar.
    - As a best practice, use percentages only when the total percentage equals 100%. Pie charts tend to confuse people when the values do not add up to 100%.
- If there are only two series on a chart and you display both the Y1 and Y2 axes, as a best practice you must match the colors to the first two palette colors for the axis labels.

The following example shows a chart that uses the global styles and has additional formatting applied.

Chart styles

Use the styles included with the global report style sheet, GlobalReportStyles.css, to format charts.

The following table shows formatting from the GlobalStyleSheet.css style sheet for charts:

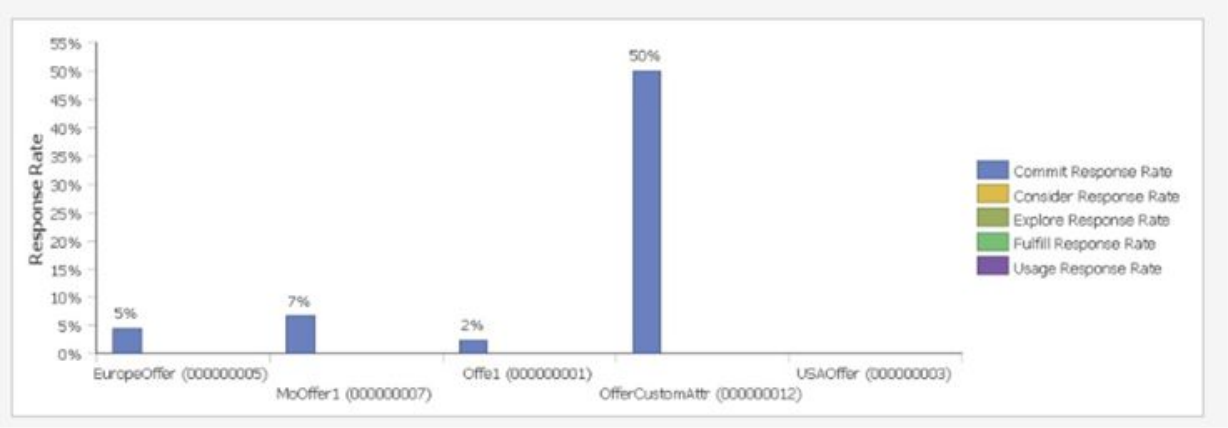
Charts obtain the following formatting from the GlobalStyleSheet.css.

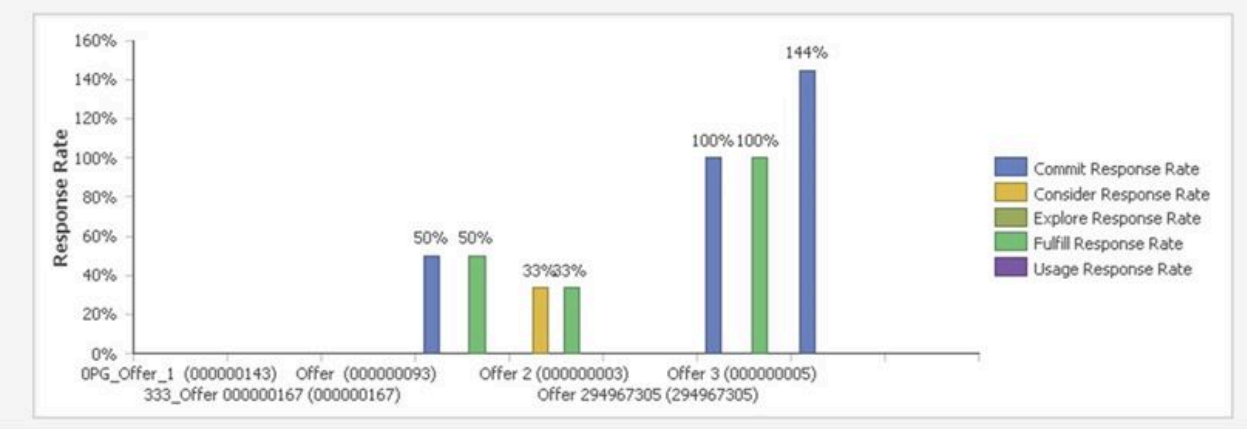
Item	Style
Charts	1 pt light gray border
Titles and labels	10 point bold font

Additionally, when you create a new chart, use the following guidelines to match the existing chart reports.

- Use the default width, unless there is more than one chart on the report. When you include multiple charts in a single report, set the chart width to 750px.
- To use gradients and color palettes, copy and paste the strings from the table in “Global report styles” into the XML report specification.
- As a general guideline, select the chart type based on the data that you expect to be returned.
    - Use line graphs as the chart type only when you can guarantee the report retrieves continuous data.
    - If there are multiple series, a stacked bar works better than a non-stacked bar.
    - As a best practice, use percentages only when the total percentage equals 100%. Pie charts tend to confuse people when the values do not add up to 100%.
- If there are only two series on a chart and you display both the Y1 and Y2 axes, as a best practice you must match the colors to the first two palette colors for the axis labels.

The following example shows a chart that uses the global styles and has additional formatting applied.





Date formats for English version

If you use a globalized version of the HCL Unica reports package, you see a different date format in your chart reports depending on which locale you use. Unica Insights chart reports use the date style short.

The following table shows the date formats for chart reports for English locale.

Unica Insights chart reports date formats for English version

Locale	Unica Insights chart reports date format example
English	04/13/2020

Dashboard report styles

Dashboard reports use the global styles with some manual formatting.

Use the following guidelines to make sure that reports that are displayed in the Dashboard fit properly in Dashboard portlets.

Item	Style
Back-ground color	Keep background color set to gray (hex value F5F5F5).
Size	Specify size by using percentages whenever possible. When sizing with a percentage is not possible, set the size to 323 pixels wide by 175 pixels tall.

Subtitles	Put subtitles on the left side.
Dates	Put dates on the right side.
Legends	Center legends below the chart.
Lines in line charts	Display horizontal lines only. Do not display vertical lines.
Axis line color	Keep axis lines set to black.
Grid line color	Keep grid lines set to gray (hex value c8c8c8).
Lists (tables)	Display a maximum of 10 lines.

Integrate new custom reports in Unica

Unica Insights enables you to integrate newly created custom reports with Unica for Campaign, Deliver, Plan, and Interact products.

Complete the following configuration steps to create new custom reports.

1. Set up your development environment using BIRT designer. For more details, go to https://www.eclipse.org/birt/documentation/tutorial/
2. Create your custom report using the BIRT designer and test the report.
3. Create a custom report folder, for example <My Custom Reports> under the Platform installation directory as shown below. `Unica_home\Platform\Insights\Reports\campaign\partitions\partition1\Affinium Campaign\<My Custom Reports>`.

Note: Ensure that you do not include any sub-folders under the custom report folder. You can provide any name for the custom report folder. You can have mutiple custom report folders.

4. Copy your report design files in the above-mentioned directory.
5. Log in to Unica and navigate to the following configurations templates.
    - For Campaign: navigate to `Affinium|Campaign|partitions|partition1|UnicaInsightsReports|Campaign`.
    - For Interact, navigate to `Affinium|Campaign|partitions|partition1|UnicaInsightsReports|Interact`.
    - For Deliver, navigate to `Affinium|Campaign|partitions|partition1|UnicaInsightsReports|Deliver`.
    - For Plan, navigate to `Affinium|Plan|umoConfiguration|reports`.
6. Select and configure the custom reports templates. Provide the following details.
    - New category name: Provide an appropriate name.
    - reportFolder: Provide the custom report folder name. For example: My Custom Reports.
    - reportName: Provide the report name. For example: Campaign Performance by User
    - reportFileName: Provide report design file name. For example: CampaignPerformancebyUser.rptdesign
    - reportDescription: Provide a description for the report
7. Navigate to the **Analytics** menu and select the appropriate <Product> **Analytics**.
8. Click **Synchronize Folders** to view the **Custom Reports folder**.
9. Click **Custom Reports Folder** to view your custom reports.
10. Click any report and run it.

Localize new Unica custom reports

You can localize your custom reports. Birt uses Resouces.properties files to translate strings. These files are copied to `Unica_Home/Platform/Insights/Reports/Resources` folder.

Resource properties files possess a key for each string. In your custom reports, you must map keys for the report's strings in the report design. You can use Unica properties files for the strings, which are already translated. However, if you have new strings, you can get their translated strings from your language teams. Birt supports ASCII formatted translation.

For all new strings, you must also generate new keys which must follow a custom format. Ensure that you do not use Unica key format or numbers.

The following table provides the resource properties file names for Campaign and Deliver.

Product Name	**Properties file**
Campaign	CAResources_<locale>.properties
Deliver	DLResources_<locale>.properties

Unica provides languages support for Campaign and Deliver reports. It supports to following languages.

Language	**Locale**
English	en_us (default)
German	de_de
Spanish	es_es
French	fr_fr
Italian	it_it
Japanese	ja_jp
Korean	ko_kr
Portuguese	pt_br
Russian	ru_ru
Simplifies Chinese	zh_cn

Traditional Chinese	zh_tw

Navigate to `Unica_Home/Platform/Insights/Reports/Resources` folder to find all *.properties file names.

For more information and to learn how to apply keys or more, go to

https://wiki.eclipse.org/BIRT/FAQ/Internationalization

Unica Insights Reports Customization

You can create new reports, and modify and customize existing Unica Insights reports.

The following videos will help provide a basic understanding, help you set up a development environment, and show you the process to customize reports with examples. You can view the description about each video in the description box.

- https://youtu.be/l6Fl8ML_rOU
- https://youtu.be/gjoAkY-JYhI
- https://youtu.be/7xP0wz-8Rn4
- https://youtu.be/Kl3Ry-RTZxo
- https://youtu.be/s5rW68Fp4Js

Modify existing out-of-the box Unica Insights reports

Users can modify out-of-the box reports and view them on Unica. For example, if a report requires a new business field, the technical users can update the reports to achieve that. Ensure that the report design file names must not be changed.

Unica Insights report design files can be opened in Unica Insights Eclipse designer. You can download the software and follow the documentation to modify existing reports. https://www.eclipse.org/birt/about/designer.php .

Customization of existing Unica Insights reports for Campaign

The following section includes details on generating customized Unica Campaign Unica Insights reports based on Unica Campaign custom audiences and custom attributes. See the Unica Campaign Administrator Guide for more details on custom audience and attributes.

Unica Campaign Custom Audiences

Unica Campaign is delivered with a single audience level called Customer. You can define any additional audience levels that you require. Audience levels allow the flowchart designers target specific groups, such as households, in marketing campaigns.

Customer' (number) is the default audience, which is available in the system to run marketing campaigns. In cases where business may require to run campaigns on other audience types, for example, a financial organization wants to contact its customers by using its customers' "AccountNO" instead of "Customerid", they must use new audience as 'Account' (text) to run campaigns. In order to show 'Account' audience data in reports, administrator must create the underlying tables or views so that reports can show relevant KPIs correctly.

To support such business requirements, the Unica Campaign administrator must create new audience levels in the system along with an Audience ID in the system. This can be created inside the Campaign settings>Manage audience level. For this, the "Customer" audience's CH/RH tables must be new created inside the Campaign DB and this must be replica of the following existing customer audience CH/RH tables.

- ua_contacthistory
- ua_dtlcontacthist
- ua_responsehistory

Tables

These are the references how the tables are modified or replicated. Once this is done, users must modify the existing Unica Insights report design template in order to view the report in Unica See Generate views for ACCOUNT audience after this section.

- These tables are replica of ‘Customer’ audience tables and are created by replacing ‘CUSTOMERID’ column with the new audience field, example, "ACCOUNTNO". Here is a sample script.

```
create table UA_DTLCONTACTHIST_ACNO
(
  ACCOUNTNO        VARCHAR2(20) not null,
  TREATMENTINSTID   NUMBER(19) not null,
  CONTACTSTATUSID   NUMBER(19),
  CONTACTDATETIME   TIMESTAMP(6),
  UPDATEDATETIME    TIMESTAMP(6),
  USERDEFINEDFIELDS CHAR(18),
  DATEID            NUMBER(19) not null,
  TIMEID            NUMBER(19) not null,
  VALUEBEFORE       NUMBER(19,2),
  USAGEBEFORE       NUMBER(19,2)
);

create table UA_CONTACTHISTORY_ACNO
(
  ACCOUNTNO        VARCHAR2(20) not null,
  CELLID            NUMBER(19) not null,
  PACKAGEID         NUMBER(19) not null,
  CONTACTDATETIME   TIMESTAMP(6),
  UPDATEDATETIME    TIMESTAMP(6),
  CONTACTSTATUSID   NUMBER(19),
  DATEID            NUMBER(19),
  TIMEID            NUMBER(19),
  USERDEFINEDFIELDS CHAR(18),
```

```
  VALUEBEFORE       NUMBER(19,2),
  USAGEBEFORE       NUMBER(19,2)
);

create table UA_RESPONSEHISTORY_ACNO
(
  ACCOUNTNO         VARCHAR2(20) not null,
  TREATMENTINSTID    NUMBER(19) not null,
  RESPONSEPACKID     NUMBER(19) not null,
  RESPONSEDATETIME   TIMESTAMP(6) not null,
  WITHINDATERANGEFLG NUMBER(10),
  ORIGCONTACTEDFLG   NUMBER(10),
  BESTATTRIB         NUMBER(10),
  FRACTIONALATTRIB   FLOAT,
  DIRECTRESPONSE     NUMBER(10),
  CUSTOMATTRIB       FLOAT,
  RESPONSETYPEID     NUMBER(19),
  DATEID             NUMBER(19),
  TIMEID             NUMBER(19),
  USERDEFINEDFIELDS  CHAR(18),
  VALUEAFTER        NUMBER(19,2),
  USAGEAFTER        NUMBER(19,2),
  RESPONSEREVENUE   NUMBER(19,2),
  SALESCOST         NUMBER(19,2),
  RESPONSECHANNEL   VARCHAR2(16)
);
```

Understanding of Reporting Schema

Unica Campaign reports works on pre-aggregated views. These views are created by using Report SQL Generator' functionality and can be found under Platform settings. This feature has reporting schemas and each reporting schema is associated with multiple pre-aggregated views.

The following is the list of all reporting schemas with associated views summary. The template names available in Platform configuration are also provided for each schema.

- Campaign Views – Summary views based on Campaign, Offer, Cell, and Time
    - Campaign custom attributes
    - Campaign performance star schema
    - Offer performance star schema
    - Campaign offer response breakout star schema
    - Campaign offer contact status breakout
- Campaign Custom Attributes – Summary views on Custom attributes, Campaign/Offer/Cell
    - Campaign custom attributes
    - Campaign performance star schema
    - Offer performance star schema
    - Campaign offer response breakout star schema
    - Campaign offer contact status breakout
- Campaign Offer Response Breakout – Summary views on Campaign, offer Response
    - Campaign custom attributes
    - Campaign performance star schema
    - Offer performance star schema
    - Campaign offer response breakout star schema
    - Campaign offer contact status breakout
- Campaign Offer Contact Status Breakout – Summary views on Campaign, offer Contacts
    - Campaign custom attributes
    - Campaign performance star schema
    - Offer performance star schema
    - Campaign offer response breakout star schema
    - Campaign offer contact status breakout
- Campaign Performance – Summary views on campaign performance various analysis
    - Campaign custom attributes
    - Campaign performance star schema
    - Offer performance star schema

    - Campaign offer response breakout star schema
    - Campaign offer contact status breakout
- Offer Performance – Summary views on offer performance various analysis
    - Campaign custom attributes
    - Campaign performance star schema
    - Offer performance star schema
    - Campaign offer response breakout star schema
    - Campaign offer contact status breakout

Category	**SQL Configuration**
Campaign Views	Campaign View
	Offer View
	Cell View
	Campaign to Offer View
	Calendar View
	Time View
Campaign Custom Attributes	Campaign Custom Attribute View
	Offer Custom Attribute View
	Cell Custom Attribute View
Campaign Offer Response Breakout	Campaign Response Breakout
	Campaign Offer Response Breakout
Campaign Offer Contact Status Breakout	Campaign Contact Status Contact History
	Campaign Offer Contact Status Contact History
Campaign Performance	Campaign Contact History
	Campaign Cell Contact History

	Campaign Offer Contact History
	Campaign Offer Cell Contact History
	Campaign Cell Offer Contact History
	Campaign Response History
	Campaign Offer Response History
	Campaign Cell Response History
	Campaign Offer Cell Response History
	Campaign Cell Offer Response History
	Campaign Contact History Summary
	Campaign Cell Contact History Summary
	Campaign Offer Contact History Summary
	Campaign Offer Cell Contact History Summary
	Campaign Cell Offer Contact History Summary
	Campaign Response History Summary
	Campaign Offer Response History Summary
	Campaign Cell Response History Summary
	Campaign Offer Cell Response History Summary
	Campaign Cell Offer Response History Summary
Offer Performance	Offer Contact History
	Offer Response History

	Offer Campaign Contact History
	Offer Campaign Response History
	Offer Campaign Cell Contact History
	Offer Campaign Cell Response History
	Offer Contact History Summary
	Offer Response History Summary
	Offer Campaign Contact History Summary
	Offer Campaign Response History Summary
	Offer Campaign Cell Contact History Summary
	Offer Campaign Cell Response History Summary
	Offer Performance Metrics Summary

Campaign installer registers 'Customer' audience's report views. It also registers report views' templates, which are used to generate report views for new audiences.

Create reporting schema for custom audience

To create reporting schemas for ACCOUNT audience, complete the following steps.

1. Select a tempalate out of the five campaign reporting schema templates, which use CH/RH tables for their SQL definition.
2. Provide New Category Name and respective contact and response history tables and its Audience key (column name) for all relevant categories. Administrators may also add additional time level grouping if required, new views definitions are added for each time level grouping.

   a. Campaign Views and Campaign Custom Attributes category are audience independent so they will be same for any custom audience.
   b. Campaign Offer Response Breakout, Campaign Offer Contact Status Breakout, Campaign Performance and Offer Performance categories have columns. For all custom audience category you need to create exact same columns which are available for the default Customer audience.
3. Repeat the above step for all templates. All new categories will be listed under campaign.
4. Select each ACNO category and configure them for their view names under 'Sql configuration', these view names must be unique to 'Customer' audience's view names. There are two types of view names one that ends with underscore (UARC_OCH_) and without underscore (UARC_CRBO_ACNO). The first one is used to create various time level views like UARC_OCH_ACNO_DY, UARC_OCH_MO, etc.

Select each ACNO category and configure them for their Key point indicator (KPI) using column template under **Columns**'. Administrator must ensure that all KPIs are created by 'Customer' audience.

There are two types of KPI metric templates, Contact and Response. Contact metric is defined from contact history table column, whereas Response metric is defined from response history table column.

Generate views for ACCOUNT audience

To generate views for ACCOUNT audience, complete the following steps.

1. Navigate to **Settings > Reports SQL Generator** option. All schemas are listed under product 'Campaign'.
2. Select all ACNO categories and generate views.
3. Save the scripts.

Merge Customer and ACCOUNT audiences views

To merge Customer and ACCOUNT audiences views, complete the following steps.

1. Perform the “union all" action on each view of both audiences, example of UARC_COCH_MO and UARC_CORH_MO are attached. Unica Campaign has 37 audience dependent views; administrators must follow this procedure for all.

```
CREATE OR REPLACE VIEW UARC_COCH_MO AS
(
(( SELECT DISTINCT
 UA_Treatment.CampaignID AS CAMPAIGNID,
 UA_Treatment.OfferID AS OFFERID,
 UA_Calendar.Month AS MONTH, UA_Calendar.Year AS YEAR,
 count(CASE UA_Treatment.CntrlTreatmtFlag WHEN 0 THEN
 UA_Treatment.TreatmentSize END) as NUM_OF_OFFERS,
 count(CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
 UA_Treatment.TreatmentSize END) as NUM_OF_OFFERS_CG,
 count (distinct (case UA_Treatment.CntrlTreatmtFlag when 0 then
 UA_Treatment.OfferHistoryID END)) as NUM_OFF_VERS,
 count (distinct (case UA_Treatment.CntrlTreatmtFlag when 1 then
 UA_Treatment.OfferHistoryID END)) as NUM_OFF_VERS_CG,
 count(distinct (CASE WHEN UA_Treatment.CntrlTreatmtFlag = 0 and
 UA_ContactStatus.CountsAsContact=1 THEN UA_ContactHistory_ACNO.ACNO
 END)) as UNIQUE_RECIPIENTS,
 count(distinct (CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
 UA_ContactHistory_ACNO.ACNO END)) as UNIQUE_RECIP_CG FROM
 UA_ContactStatus,
 UA_Calendar,
 UA_Treatment
 LEFT OUTER JOIN
 UA_ContactHistory_ACNO
 ON
 UA_Treatment.PackageID = UA_ContactHistory_ACNO.PackageID
 WHERE
 UA_ContactHistory_ACNO.CellID = UA_Treatment.CellID
 AND
```

```
UA_ContactHistory_ACNO.ContactStatusID =
UA_ContactStatus.ContactStatusID
AND
UA_ContactHistory_ACNO.DateID = UA_Calendar.DateID
AND
UA_Treatment.HasDetailHistory = 0 GROUP BY
UA_Treatment.CampaignID,
UA_Treatment.OfferID, UA_Calendar.Month, UA_Calendar.Year ) UNION ALL
( SELECT DISTINCT
UA_Treatment.CampaignID AS CAMPAIGNID,
UA_Treatment.OfferID AS OFFERID,
UA_Calendar.Month AS MONTH, UA_Calendar.Year AS YEAR,
count(CASE UA_Treatment.CntrlTreatmtFlag WHEN 0 THEN
UA_Treatment.TreatmentSize END) as NUM_OF_OFFERS,
count(CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
UA_Treatment.TreatmentSize END) as NUM_OF_OFFERS_CG,
count (distinct (case UA_Treatment.CntrlTreatmtFlag when 0 then
UA_Treatment.OfferHistoryID END)) as NUM_OFF_VERS,
count (distinct (case UA_Treatment.CntrlTreatmtFlag when 1 then
UA_Treatment.OfferHistoryID END)) as NUM_OFF_VERS_CG,
count(distinct (CASE WHEN UA_Treatment.CntrlTreatmtFlag = 0 and
UA_ContactStatus.CountsAsContact=1 THEN UA_DtlContactHist_ACNO.ACNO
END)) as UNIQUE_RECIPIENTS,
count(distinct (CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
UA_DtlContactHist_ACNO.ACNO END)) as UNIQUE_RECIP_CG FROM
UA_ContactStatus,
UA_Calendar,
UA_Treatment
LEFT OUTER JOIN
UA_DtlContactHist_ACNO
ON
UA_Treatment.TreatmentInstID = UA_DtlContactHist_ACNO.TreatmentInstID
```

```
 WHERE
 UA_DtlContactHist_ACNO.ContactStatusID =
 UA_ContactStatus.ContactStatusID
 AND
 UA_DtlContactHist_ACNO.DateID = UA_Calendar.DateID
 AND
 UA_Treatment.HasDetailHistory = 1 GROUP BY
 UA_Treatment.CampaignID,
 UA_Treatment.OfferID, UA_Calendar.Month, UA_Calendar.Year ))
UNION ALL
(( SELECT DISTINCT
 UA_Treatment.CampaignID AS CAMPAIGNID,
 UA_Treatment.OfferID AS OFFERID,
 UA_Calendar.Month AS MONTH, UA_Calendar.Year AS YEAR,
 count(CASE UA_Treatment.CntrlTreatmtFlag WHEN 0 THEN
 UA_Treatment.TreatmentSize END) as NUM_OF_OFFERS,
 count(CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
 UA_Treatment.TreatmentSize END) as NUM_OF_OFFERS_CG,
 count (distinct (case UA_Treatment.CntrlTreatmtFlag when 0 then
 UA_Treatment.OfferHistoryID END)) as NUM_OFF_VERS,
 count (distinct (case UA_Treatment.CntrlTreatmtFlag when 1 then
 UA_Treatment.OfferHistoryID END)) as NUM_OFF_VERS_CG,
 count(distinct (CASE WHEN UA_Treatment.CntrlTreatmtFlag = 0 and
 UA_ContactStatus.CountsAsContact=1 THEN UA_ContactHistory.CustomerID
 END)) as UNIQUE_RECIPIENTS,
 count(distinct (CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
 UA_ContactHistory.CustomerID END)) as UNIQUE_RECIP_CG FROM
 UA_ContactStatus,
 UA_Calendar,
 UA_Treatment
 LEFT OUTER JOIN
 UA_ContactHistory
```

```
ON
UA_Treatment.PackageID = UA_ContactHistory.PackageID
WHERE
UA_ContactHistory.CellID = UA_Treatment.CellID
AND
UA_ContactHistory.ContactStatusID = UA_ContactStatus.ContactStatusID
AND
UA_ContactHistory.DateID = UA_Calendar.DateID
AND
UA_Treatment.HasDetailHistory = 0 GROUP BY
UA_Treatment.CampaignID,
UA_Treatment.OfferID, UA_Calendar.Month, UA_Calendar.Year ) UNION ALL
( SELECT DISTINCT
UA_Treatment.CampaignID AS CAMPAIGNID,
UA_Treatment.OfferID AS OFFERID,
UA_Calendar.Month AS MONTH, UA_Calendar.Year AS YEAR,
count(CASE UA_Treatment.CntrlTreatmtFlag WHEN 0 THEN
UA_Treatment.TreatmentSize END) as NUM_OF_OFFERS,
count(CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
UA_Treatment.TreatmentSize END) as NUM_OF_OFFERS_CG,
count (distinct (case UA_Treatment.CntrlTreatmtFlag when 0 then
UA_Treatment.OfferHistoryID END)) as NUM_OFF_VERS,
count (distinct (case UA_Treatment.CntrlTreatmtFlag when 1 then
UA_Treatment.OfferHistoryID END)) as NUM_OFF_VERS_CG,
count(distinct (CASE WHEN UA_Treatment.CntrlTreatmtFlag = 0 and
UA_ContactStatus.CountsAsContact=1 THEN UA_DtlContactHist.CustomerID
END)) as UNIQUE_RECIPIENTS,
count(distinct (CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
UA_DtlContactHist.CustomerID END)) as UNIQUE_RECIP_CG FROM
UA_ContactStatus,
UA_Calendar,
UA_Treatment
```

```
 LEFT OUTER JOIN
 UA_DtlContactHist
 ON
 UA_Treatment.TreatmentInstID = UA_DtlContactHist.TreatmentInstID
 WHERE
 UA_DtlContactHist.ContactStatusID = UA_ContactStatus.ContactStatusID
 AND
 UA_DtlContactHist.DateID = UA_Calendar.DateID
 AND
 UA_Treatment.HasDetailHistory = 1 GROUP BY
 UA_Treatment.CampaignID,
 UA_Treatment.OfferID, UA_Calendar.Month, UA_Calendar.Year ))
);


CREATE OR REPLACE VIEW UARC_CORH_MO AS
(
(SELECT DISTINCT
 UA_Treatment.CampaignID AS CAMPAIGNID,
 UA_Treatment.OfferID AS OFFERID,
 UA_Calendar.Month AS MONTH, UA_Calendar.Year AS YEAR,
 count (CASE UA_Treatment.CntrlTreatmtFlag WHEN 0 THEN
 UA_ResponseHistory_ACNO.BestAttrib END) as RESP_TRANS,
 count(CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
 UA_ResponseHistory_ACNO.BestAttrib END) as RESP_TRANS_CG,
 count(distinct (CASE WHEN UA_Treatment.CntrlTreatmtFlag = 0  THEN
 UA_ResponseHistory_ACNO.ACNO END)) as UNIQUE_RESPONDERS,
 count(distinct (CASE WHEN UA_Treatment.CntrlTreatmtFlag = 1  THEN
 UA_ResponseHistory_ACNO.ACNO END)) as UNIQUE_RESP_CG,
 count(distinct (CASE WHEN UA_ResponseHistory_ACNO.OrigContactedFlg
 = 0 AND UA_Treatment.CntrlTreatmtFlag= 0 THEN
 UA_ResponseHistory_ACNO.ACNO END)) as NOT_CONT_RESP,
```

```
count (CASE WHEN UA_ResponseHistory_ACNO.WithinDateRangeFlg=0
AND UA_Treatment.CntrlTreatmtFlag=0 THEN
UA_ResponseHistory_ACNO.BestAttrib END) as RESP_AFTER_EXP,
count (CASE WHEN UA_ResponseHistory_ACNO.WithinDateRangeFlg=0
AND UA_Treatment.CntrlTreatmtFlag=1 THEN
UA_ResponseHistory_ACNO.BestAttrib END) as RESP_AFTER_EXP_CG,
AVG(CASE UA_Treatment.CntrlTreatmtFlag WHEN 0 THEN
UA_ResponseHistory_ACNO.ResponseRevenue END) AS
REVENUE_PER_RESP, SUM(CASE UA_Treatment.CntrlTreatmtFlag
WHEN 0 THEN UA_ResponseHistory_ACNO.ResponseRevenue END)
AS GROSS_REVENUE, AVG(CASE UA_Treatment.CntrlTreatmtFlag
WHEN 1 THEN UA_ResponseHistory_ACNO.ResponseRevenue END) AS
REV_PER_RESP_CG, SUM(CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
UA_ResponseHistory_ACNO.ResponseRevenue END) AS GROSS_REVENUE_CG FROM
UA_UsrResponseType,
UA_Calendar,
UA_Treatment
LEFT OUTER JOIN
UA_ResponseHistory_ACNO
ON
UA_Treatment.TreatmentInstID =
UA_ResponseHistory_ACNO.TreatmentInstID
WHERE
UA_ResponseHistory_ACNO.ResponseTypeID =
UA_UsrResponseType.ResponseTypeID
AND
UA_UsrResponseType.CountsAsResponse = 1
AND
UA_ResponseHistory_ACNO.BestAttrib = 1
AND
UA_ResponseHistory_ACNO.DateID = UA_Calendar.DateID GROUP BY
```

```
 UA_Treatment.CampaignID,UA_Treatment.OfferID, UA_Calendar.Month,
 UA_Calendar.Year)
Union All
(SELECT DISTINCT
 UA_Treatment.CampaignID AS CAMPAIGNID,
 UA_Treatment.OfferID AS OFFERID,
 UA_Calendar.Month AS MONTH, UA_Calendar.Year AS YEAR,
 count (CASE UA_Treatment.CntrlTreatmtFlag WHEN 0 THEN
 UA_ResponseHistory.BestAttrib END) as RESP_TRANS,
 count(CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
 UA_ResponseHistory.BestAttrib END) as RESP_TRANS_CG,
 count(distinct (CASE WHEN UA_Treatment.CntrlTreatmtFlag = 0  THEN
 UA_ResponseHistory.CustomerID END)) as UNIQUE_RESPONDERS,
 count(distinct (CASE WHEN UA_Treatment.CntrlTreatmtFlag = 1  THEN
 UA_ResponseHistory.CustomerID END)) as UNIQUE_RESP_CG,
 count(distinct (CASE WHEN UA_ResponseHistory.OrigContactedFlg = 0 AND
 UA_Treatment.CntrlTreatmtFlag= 0 THEN UA_ResponseHistory.CustomerID
 END)) as NOT_CONT_RESP,
 count (CASE WHEN UA_ResponseHistory.WithinDateRangeFlg=0 AND
 UA_Treatment.CntrlTreatmtFlag=0 THEN UA_ResponseHistory.BestAttrib
 END) as RESP_AFTER_EXP,
 count (CASE WHEN UA_ResponseHistory.WithinDateRangeFlg=0 AND
 UA_Treatment.CntrlTreatmtFlag=1 THEN UA_ResponseHistory.BestAttrib
 END) as RESP_AFTER_EXP_CG, AVG(CASE UA_Treatment.CntrlTreatmtFlag
 WHEN 0 THEN UA_ResponseHistory.ResponseRevenue END) AS
 REVENUE_PER_RESP, AVG(CASE UA_Treatment.CntrlTreatmtFlag
 WHEN 1 THEN UA_ResponseHistory.ResponseRevenue END) AS
 REV_PER_RESP_CG, SUM(CASE UA_Treatment.CntrlTreatmtFlag
 WHEN 0 THEN UA_ResponseHistory.ResponseRevenue END) AS
 GROSS_REVENUE, SUM(CASE UA_Treatment.CntrlTreatmtFlag WHEN 1 THEN
 UA_ResponseHistory.ResponseRevenue END) AS GROSS_REVENUE_CG FROM
 UA_UsrResponseType,
```

```
 UA_Calendar,
 UA_Treatment
 LEFT OUTER JOIN
 UA_ResponseHistory
 ON
 UA_Treatment.TreatmentInstID = UA_ResponseHistory.TreatmentInstID
 WHERE
 UA_ResponseHistory.ResponseTypeID = UA_UsrResponseType.ResponseTypeID
 AND
 UA_UsrResponseType.CountsAsResponse = 1
 AND
 UA_ResponseHistory.BestAttrib = 1
 AND
 UA_ResponseHistory.DateID = UA_Calendar.DateID GROUP BY
 UA_Treatment.CampaignID,UA_Treatment.OfferID, UA_Calendar.Month,
 UA_Calendar.Year)
);
```

Note:

- Keep the views names same as defined by the system for Customer audience.
- The above procedure enables the summary views to have the audience data and the marketers can view the out-of-the-box reports using same reports and model.

Custom Attributes

You can customize campaigns by adding custom campaign attributes to store metadata about each campaign.

Before you begin:

Create the custom attribute. See the Campaign Administration Guide for more details.

To configure Unica Campaign reports for custom attributes, complete the following steps.

Text custom attributes

To include text custom attributes in schema and views, complete the following steps:

Note: It is assumed that text attribute belongs to the Campaign attributes.

1. Get AttributeID campaign system database using below query:

```
select AttributeID,Name,DisplayName from UA_AttributeDef where
 DisplayName = <>
```

2. Navigate to **Settings > Configuration > Report > schemas > Campaign > Campaign custom attributes**.
3. Click the template **`(Campaign custom column)`** and provide information for the following entries.
   - **New category name**
   - **Column Name** as the offer custom attribute name.
   - **Attribute ID** from the abomined query.
   - Value type as `NumberValue`.
4. Click **Save Changes**.

Numeric custom attributes

To include numeric custom attributes in schemas and views, complete the following steps.

Note: For this, it is assumed that the numeric attribute belongs to Offer attributes.

1. Get AttributeID campaign system database using below query:

```
select AttributeID,Name,DisplayName from UA_AttributeDef where
 DisplayName = <>
```

2. Navigate to **Settings > Configuration > Report > schemas > Campaign > Campaign custom attributes**.
3. Click the `Offer custom column` template and provide the following information.
    - New category name.
    - Column Name as the offer custom attribute name.
    - Attribute ID from the abomined query.
    - Value type as `NumberValue`.
4. To include Number attribute as a KPI in report views, navigate to the schema. For example:

    Settings > Configuration > Report > schemas > Campaign Performance.
5. Select the column template `Contact metric` if the Number attribute is related to Contact or (Response metric) if the Number attribute is belonged to response. Fill in the following information
    - New category name.
    - Column Name as the offer custom attribute name.
    - Function from the **Count/Count Distinct/Min/Max/Avg** list.
    - Column name
    - Under **Control treatment** flag, the default value is 0 and if the KPI is applicable for control cell the value is 1.
6. Click **Save Changes**.

If a custom attribute is relevant to more reporting schemas for your business reporting requirement, repeat steps 4 and 5 for each schema. It is not required that a KPI must fall under all reporting schemas, to understand more on this, see the **Understanding of Reporting Schema** section of this document.

Generate Custom Audience and Custom Attribute Views using Run SQL Generator

To generate views using above custom attributes, complete the following steps.

1. Navigate to **Settings > Reports SQL Generator > Update or Create new schema names** from the **Schema** list.
2. Select DB type.
3. Download views.

Downloaded SQL script must be executed in the Campaign system database. You may have to drop view(s) and recreate them if they already exist in the system.

Composite audience report views

Report views can also be created for composite audience case, where two fields are combined together for identifying unique target customer. To create report views, one common set of CH/RH tables must be created that have both the audience fields. All columns must be captured in comma separated in Audience Key field of the template configuration.

Customization of existing Unica Insights reports for Interact

The following section includes details on generating customized Unica Interact Unica Insights reports based on Unica Interact custom audiences. See the Unica Interact Administrator Guide for more details on custom audience and attributes.

Unica Interact Custom Audiences

Unica Interact is delivered with a single audience level called Customer. You can define any additional audience levels that you require. Audience levels allows the flowchart designers target specific groups, such as households.

Customer (number) is the default audience, which is available in the system to deliver offer in marketing Interact. In cases, where business require to deliver offers to other audience types, for example, a financial organization wants to contact its customers by using its customers' "AccountNO" instead of "Customerid", they must use new audience as 'Account' (text) to run Interact. In order to show 'Account' audience data in reports, administrator must modify the underlying tables or views so that reports can show related KPIs correctly.

To support such business requirements, the Unica Interact administrator must create new audience in the system. For this, either "Customer" audience's CH/RH tables must be modified or new CH/RH tables must be created which are replica of the following customer audience CH/RH tables.

- ua_contacthistory
- ua_dtlcontacthist
- ua_responsehistory

Tables

The following CH/RH tables are supposed to be created under Campaign system database. For more details, see the Unica Campaign Administrator Guide.

- These tables are replica of ‘Customer’ audience tables are created by replacing ‘CUSTOMERID’ column with the new audience field, example, "ACCOUNTNO". Here is a sample script.

```
CREATE TABLE [dbo].[ACCT_UA_DtlContactHist](
[AccountID] [varchar](512) NOT NULL,
[TreatmentInstID] [bigint] NOT NULL,
[ContactStatusID] [bigint] NULL,
[ContactDateTime] [datetime] NULL,
[UpdateDateTime] [datetime] NULL,
[UserDefinedFields] [nchar](18) NULL,
[DateID] [bigint] NOT NULL,
[TimeID] [bigint] NOT NULL,
[RTSelectionMethod] [int] NULL,
[RTLearningMode] [int] NULL,
[RTLearningModelID] [bigint] NULL ) ON [PRIMARY]
CREATE INDEX ACCT_cDtlContHist_IX1 ON ACCT_UA_DtlContactHist
 (AccountID,TreatmentInstID);
ALTER TABLE [dbo].[ACCT_UA_DtlContactHist]  WITH CHECK ADD  CONSTRAINT
 [ACCT_DCH_FK3] FOREIGN KEY([TimeID]) REFERENCES [dbo].[UA_Time] ([TimeID])
ALTER TABLE [dbo].[ACCT_UA_DtlContactHist]  WITH CHECK ADD
 CONSTRAINT [ACCT_DtlCH_FK1] FOREIGN KEY([ContactStatusID]) REFERENCES
 [dbo].[UA_ContactStatus] ([ContactStatusID])
```

```
ALTER TABLE [dbo].[ACCT_UA_DtlContactHist]  WITH CHECK ADD  CONSTRAINT
 [ACCT_DtlCH_FK2] FOREIGN KEY([DateID]) REFERENCES [dbo].[UA_Calendar]
 ([DateID])
alter table ACCT_UA_DtlContactHist add RTSelectionMethod int;



CREATE TABLE [dbo].[ACCT_UA_ContactHistory](
[AccountID] [varchar](512) NOT NULL,
[CellID] [bigint] NOT NULL,
[PackageID] [bigint] NOT NULL,
[ContactDateTime] [datetime] NULL,
[UpdateDateTime] [datetime] NULL,
[ContactStatusID] [bigint] NULL,
[DateID] [bigint] NULL,
[TimeID] [bigint] NULL,
[UserDefinedFields] [nchar](18) NULL,
 CONSTRAINT [ACCT_CHist_PK] PRIMARY KEY CLUSTERED
([AccountID] ASC,,[CellID] ASC,[PackageID] ASC)
CREATE INDEX ACCT_cContactHist_IX1 ON ACCT_UA_ContactHistory(CellID);
CREATE INDEX ACCT_cContactHist_IX2 ON
 ACCT_UA_ContactHistory(PackageID,CellID);
ALTER TABLE [dbo].[ACCT_UA_ContactHistory]  WITH CHECK ADD  CONSTRAINT
 [ACCT_CHist_FK1] FOREIGN KEY([ContactStatusID])
REFERENCES [dbo].[UA_ContactStatus] ([ContactStatusID])
ALTER TABLE [dbo].[ACCT_UA_ContactHistory]  WITH CHECK ADD  CONSTRAINT
 [ACCT_CHist_FK2] FOREIGN KEY([DateID])
REFERENCES [dbo].[UA_Calendar] ([DateID])
ALTER TABLE [dbo].[ACCT_UA_ContactHistory]  WITH CHECK ADD  CONSTRAINT
 [ACCT_CHist_FK3] FOREIGN KEY([TimeID])
REFERENCES [dbo].[UA_Time] ([TimeID])
```

```
CREATE TABLE [dbo].[ACCT_UA_ResponseHistory](
[AccountID] [varchar](512) NOT NULL,
[TreatmentInstID] [bigint] NOT NULL,
[ResponsePackID] [bigint] NOT NULL,
[ResponseDateTime] [datetime] NOT NULL,
[WithinDateRangeFlg] [int] NULL,
[OrigContactedFlg] [int] NULL,
[BestAttrib] [int] NULL,
[FractionalAttrib] [float] NULL,
[DirectResponse] [int] NULL,
[CustomAttrib] [float] NULL,
[ResponseTypeID] [bigint] NULL,
[DateID] [bigint] NULL,
[TimeID] [bigint] NULL,
[UserDefinedFields] [nchar](18) NULL,
[RTSelectionMethod] [int] NULL,
[RTLearningMode] [int] NULL,
[RTLearningModelID] [bigint] NULL,
 CONSTRAINT [ACCT_RHistory_PK] PRIMARY KEY CLUSTERED
([AccountID] ASC,[TreatmentInstID] ASC,[ResponsePackID] ASC)
ALTER TABLE [dbo].[ACCT_UA_ResponseHistory]  WITH CHECK ADD  CONSTRAINT
 [ACCT_RHistory_FK1] FOREIGN KEY([TreatmentInstID])
REFERENCES [dbo].[UA_Treatment] ([TreatmentInstID])
ALTER TABLE [dbo].[ACCT_UA_ResponseHistory]  WITH CHECK ADD  CONSTRAINT
 [ACCT_RHistory_FK2] FOREIGN KEY([TimeID])
REFERENCES [dbo].[UA_Time] ([TimeID])
ALTER TABLE [dbo].[ACCT_UA_ResponseHistory]  WITH CHECK ADD  CONSTRAINT
 [ACCT_RHistory_FK3] FOREIGN KEY([ResponseTypeID])
REFERENCES [dbo].[UA_UsrResponseType] ([ResponseTypeID])
ALTER TABLE [dbo].[ACCT_UA_ResponseHistory]  WITH CHECK ADD  CONSTRAINT
 [ACCT_RHistory_FK4] FOREIGN KEY([DateID])
```

```
REFERENCES [dbo].[UA_Calendar] ([DateID])
alter table ACCT_UA_ResponseHistory add RTSelectionMethod int;
```

Understanding of Reporting Schema

Unica Interact reports works on pre-aggregated views. These views are created by using Report SQL Generator' functionality. This feature has reporting schemas and each reporting schema is associated with "n" number of pre-aggregated views.

The details of view are described in the "SQL scripts by data source" section.

Interact installer registers 'Customer' audience's report views. It also registers report views' templates, which are used to generate report views for new audiences.

To create reporting schemas for ACCOUNT audience, complete the following steps.

1. Use Interact Performance Star Schema to create Interact Performance Schema. Complete the following substeps to create an Interact Performance schema.
    a. Select **Settings > Configuration** and expand **Reports > Schemas > Interact > Interact Performance Star Schema**.
    b. In the **New category name** field, enter a descriptive name for the reporting schema that indicates the audience level. For example, Interact Performance Household.
    c. In the **Input Tables** section, identify the tables that support the audience level and the audience key.
    d. In the **Schema Settings** section, select all the Over Time Variations options that apply and then click **Save Changes**. A new node appears in the Configuration tree for the schema. You cannot change the name of the node.

The remaining views, i.e. Deployment History, Interact Runtime Views and Interact Learning Views are audience independent so they are same for any custom audience.

Generate views for ACCOUNT audience

To generate views for ACCOUNT audience, complete the following steps.

1. Navigate to **Settings > Reports SQL Generator**. All schemas are listed under product 'Interact'.
2. Select all ACNO categories and generate views.
3. Save the scripts and import in the respective data source.

For composite audience level

Merge Customer and ACCOUNT audiences views

To merge Customer and ACCOUNT audiences views, complete the following steps. Administrator must merge Customer and Account audiences' views, as explained above.

1. Perform the "union all" action on each view of Interact Performance for both the audience levels.
2. Keep the views names same as defined by system for Customer audience.

The above procedure enables the summary views to have both the audience data and the marketers can view all out of the box reports using same reports and model.

Note: To run Interact report "Zone_Performance_By_Offer" with custom audience level, you must possess audience specific views. To accomplish this, perform the following steps.

For example, you want to use Account audience level.

1. Drop the following views from Campaign database:
    - `UARI_ZONEPERF_PRES_REJ`
    - `UARI_ZONEPERF_MASTER`
2. Open the file from the Interact installation location `<INTERACT_HOME>/reports/ddl/<db type folder>/InteractDT.sql`.
3. Change the name of following tables in above-mentioned views to audience specific as per the tables you created above. Save the files and import them in database.
    - UA_ResponseHistory > ACCT_UA_ResponseHistory
    - UA_DtlContactHist > ACCT_UA_DtlContactHist

Note: If you are using Weblogic application server and the reports which contains charts are not loaded on the page, then you may require to clean application server cache and restart.

Known issues

The following is the list of Known Issues in Unica Insights Reports

Table 4. Known Issues in Unica Insights Reports

ID	Issue Description
HMA-330749 (V12.1.1)	When configuring Unica Insights Reports or Unica Open Insights, if you see an error when executing `acer_scripts_sqlserver.sql` (on Microsoft SQL Server, DB2, or Oracle), raise a PMR. HCL Support will provide assistance on the issue.
HMA-330431 (V12.1.1)	Campaign Journey CH/RH integration feature may produce inappropriate data on "Offer performance Metrics" report, when common offers are being used in Unica Campaign and Unica Journey.
HMA-325523 (V12.1.1)	If there are conditional links in an email document, incorrect data is seen in the reports.
HMA-320280	In the Approval and Compliance listing report, all the approval items are not displayed.
HMA-306949 (V12.1.1)	If you export Analytics Reports, the column headers are truncated for the `PDF` format.
HMA-314296 (V12.1.0.4)	When two charts are beside each other in a report, and if you export the report to Microsoft Excel, the second chart shrinks. This is a known issue in Unica Insights.

Table 4. Known Issues in Unica Insights Reports (continued)

ID	Issue Description
	Workaround: Open and edit the excel report to resize the chart.
HMA-320542 (V12.1.0.3)	Update event pattern state ETL to support advanced patterns. • Pattern State ETL process currently works only for Interactive Channel with event patterns MatchAll, Counter and Weighted Counter. • Event Pattern report for all databases must be corrected to display the new advanced event patterns.
HMA-313087	Permissions to execute files are not available for Insights tools. Workaround: Perform the following steps. 1. Navigate to `<INSIGHTS_HOME>/tools/bin`. 2. Change the permission of the files by running the following command `chmod -R 777`.
HMA-313086	Insights Reports with images do not open with required permissions. Workaround: Perform the following steps. 1. Navigate to `<INSIGHTS_HOME>/report`. 2. Change the permission of the **images** folder by running the following command `chmod -R 766 images`. 3. Restart the application server, if it is already running.

Table 4. Known Issues in Unica Insights Reports (continued)

ID	Issue Description
HMA-310915	The What If Offer Financial Summary report is not available in Campaign with MariaDB data source.
HMA-307152	PDF output sometimes shrinks if the number of report columns are many. XLSX format works better in such cases because it allows custom formatting.
HMA-305517	For all Unica Insights object-specific reports, you may see the object name, for example, Campaign/Offer/Plan/Program/Project drop-down filter disabled.
HMA-305352	An error is generated in the Unica Insights report on using the project name with $ character.
HMA-303387	If any exception is generated on the reports configuration, it is displayed as per the Unica Insights engine. It helps in understanding the error.
HMA-302526	Unlike IBM Cognos reports, few Unica Insights reports do not have Campaign, Plan, and Interactive Channel objects' hyperlinks.