HCLSoftware

# Unica Journey V12.1.11 Administrator's Guide

# Contents

# Chapter 1. An Introduction to Unica Journey

Unica Journey is a goal-based orchestration solution to craft, execute, and visualize context-driven, personalized, multi-step omnichannel customer experiences.

Marketers can use Unica Journey to:

- Define goals for customer experience
- Easily adjust journeys in real time to achieve them
- Craft and visualize entire customer journey across channels/touchpoints and events with a sleek and intuitive Journey Canvas

Customer journeys are completely automated and synchronized with every step of your customer's brand engagement. Use the real-time Insights within Journey to understand customer behavior with insights that reflect things as they happen in their Journey.

## Features of Unica Journey

The features of Unica Journey are as follows:

- **Goal driven Experiences**: Define goals for your customer experience and easily adjust your journeys in real time to achieve them.
- **Orchestration Canvas**: Craft and visualize your entire customer journey across channels/touchpoints and events with a sleek and intuitive Journey Canvas.
- **Always on Engagement**: Completely automated execution that is in sync with every step of your customer's brand engagement.
- **Real-time Insights**: Understand your customer behavior with insights that reflect things as they happen in their Journeys.
- **Choice of Touchpoints**: Leverage the out of the box native touchpoints for digital channels or craft a custom touchpoint and seamlessly orchestrate the journey across your eco system.
- **Dynamic Data Framework**: Flexible data definition and entry sources to augment customer journey with contextual data and events from multiple touchpoints and in variety of formats (File, API, etc.)

**Note:** It is recommended that on distributed (Cluster) Journey setup, user should not install Journey Web and Engine on same machine. It is good to install Web and Engine on different machines.

## Benefits of Unica Journey

The benefits of Unica Journey are as follows:

- **Increased Brand Loyalty**: Strengthen your brand following with targeted and automated journeys that acquire, nurture, convert and retain customers.
- **Amplified Omni Channel Engagement**: Deliver a consistent customer experience across channels with native integration for outbound (Unica Campaign) and inbound engagement (Unica Interact, Unica Discover, and Unica Deliver).
- **Shorten your Customer Conversion Cycle**: Be a step ahead and drive your customer to their goals with timely next best actions.
- **React to the Moment**: You will not miss any opportunity to know where your customer is on their journey and delight them with relevant experience.
- **Lower Marketing TCO**: Reduce your marketing TCO with automated flows and plug and play integration to your MarTech ecosystem through an open and flexible framework powered by the Unica Link.

# Chapter 2. New navigation UI

Unica has introduced New UI.

1. Administrators will have access to switch it for any individual user through **User settings > Edit preferences**.
2. Toggle button present at the top of the screen will help user to switch between Classic UI (Old UI) and Neo UI (New UI).

**NPN**

Unica Platform provides a common access point and user interface for Journey. The common interface provides the following features.

- When multiple Unica products are installed, you can navigate between products without launching new windows.
- You can view a listing of the pages that you have recently visited, and navigate back to any of those pages using the **Recent** menu.
- You can set a page as favorite by clicking on **Favorites** icon and page will be listed in **Favorites** list.
- You can access the search function for Folders, Journeys, Entry sources, Data definitions, Template, All folders using the Search field. For example, if you are viewing a list of entry sources within Journey, a search would take place across entry sources.

# Chapter 3. Journey Multi Partition

From V12.1.7 onward, Journey will support multi partition. For users upgrading to 12.1.7, if there is more than one partition in their system then it might lead to cross-partition references amongst the various entities in the Journey application.

Example of cross-partition reference:

Entry source ES001 created by a user of partition1, is being used in a Journey JS0001 created by user of partition2.

An administrator should run following query before the upgrade.

**select * from organisations**

If this query returns only one record and with id column value as 1, then such system can be upgraded directly to 12.1.7. Otherwise, user will have to get in touch with support team to check and fix cross-partition references present in the system (if any)

Prior to 12.1.7, a user could access entities belonged to other partition.

- From 12.1.7 onward, user's view will be restricted to the entities of his own partition. e.g. user of partition1 can see only those entry sources, data definitions, templates, Journeys and folders which belong to their partition.

  Accessing entities of other partition in any way will result in an error 404 (not found). This is to ensure that any malliecious user does not invoke Journey's rest APIs to access cross-partition resources.
- A Journey will not accept the data coming via an entry source which belong to different partition. In such case system will log an error. However, if a given entry source is used in multiple Journeys and some of those belong to the partition of the entry source, then the data will be sent to only those Journeys.
- A Journey will not be able to publish the data to an entry source which belong to different partition via publish action touchpoint. In such case, system will log an error and the audience will move to next touchpoint.
- If a client id /secret pair is used to input the audience data through /api/entry sources/rest/data, then such pair should belong to the partition of an entry source. Otherwise, system will give an error 404 not found. In such case, the administrator will have to generate a new client id/secret pair by logging as the user of an entry source's partition. All the references of existing client id/secret pair in the downstream applications will have to be corrected as well.
- A user cannot copy a Journey with references of entry source or data definition of other partition. Else, system will give 404 not found error to the user.

**Configuration changes in platform configuration.**

On clean installation of 12.1.7 or upgrade to 12.1.7, a new node ***partitions*** will be added in the Journey application's node. Administrator will have to create one or more partitions by clicking on partition template node and specifying the partition name. The number of partitions and their names must match with those present in other installed unica products like campaign, deliver etc.

On adding partition Node, for example partition1, following attributes will appear when user clicks on partition node (e.g. `Affinium|Journey|partitions|partition1`)

- Link_Configured (default No)
- Deliver_Configured (default No)
- Contact_Central_Configured (default No)
- Dedup_End_Audience (default No)
- Goal_Max_Allowed (default 15)
- Mask_Exported_Audience_Data (default No) - This property is specific to engine. If the property is set to **Yes**, then maskable fields in audience details will be masked while publishing the audience details to csv or kafka topic in publish action touchpoint. If the Full_Mask_Exported_Audience_Data property is set to **Yes**, then maskable fields in audience details will be fully masked.
- Full_Mask_Exported_Audience_Data (default No) - This property is applicable for both web and engine. In case of journey web, if the audience details pop-up is visible and clickable in the first place on journey canvas, then the values of maskable fields will be completed masked if this property is set to **Yes** and will be partially masked if the property is set to **No**.

  In case of journey engine, when the audience data is published to csv file or kafka topic on publish action touchpoint and if both the properties Full_Mask_Exported_Audience_Data and Mask_Exported_Audience_Data is set to **Yes**, then the values of maskable fields in that data will be completed masked and will be partially masked if the property is set to **No**.

Following Nodes will be added in each of the partition node

**Journey Configurations**

- Journey_Waittime_Configurations
- Sync_Contact_and_Response_Tables
- Validations_On_Journey_Records
- Validation_Headers - Enable/Disable Audience Details Pop-up depending on the headers and its value configured in Journey Configurations under: `Affinium|Journey|partition|partition1|Journey_Configurations|Validation_Headers`

  If all the mentioned headers found in the http request and each header contains at least one of the supported values mentioned in the configuration, then Audience details pop-up will be shown to the user. Header names and values are not case sensitive i.e. they will match without considering their case (upper/lower). In each of the triplets which consists of header name, supported value(s) and field separator (in case of multiple values), administrator can choose the field separator of their choice. If the value separator field is kept blank, then entire value will be considered as a single supported value.

  - Validation_Header_Template
- Link Configuration
- Deliver Configuration
- Integration

**Note:** The settings in application.properties as well as global settings in platform configurations (i.e. settings outside the partitions) have been retained for customer's reference.

**Journey_Waittime_Configurations**

- Engagement_Split_Waittime - Default wait time for engagement split in minutes
- Decision_Split_Waittime - Default wait time for decision split in minutes
- Max_No_Of_Days_In_Delay_Evaluation - Default value for max days allowed in delay.
- Max_No_Of_Days_In_Delay_Expression - Default value for max days allowed in data definition based delay expression.

**Sync_Contact_and_Response_Tables**

- Sync_Contact_and_Response_Tables - This configuration decides whether to sync CHRH in case of Offer integration or not. If this flag is set to true then only CHRH will ETL to Campaign tables in case of Offer integration. If the flag is set to false, then user can stop posting on to the CH RH topic.

  **Supported Values:** True/False

  **Default Value:** True

**Validations_On_Journey_Records**

When records enter a Journey, users can set validation on the records. When the **Validations_On_Journey_Records** property is set to `Yes`, the validation is performed on Email format, Data type on required fields, and minimum and maximum length of Data Definition for the required fields.

By default, the **Validations_On_Journey_Records** property is set to `Yes`. This property can be handled from Platform side under Platform Configuration. You can navigate under Platform configuration settings: Settings for `Affinium|Journey|Journey_Configurations`.

**Validations_On_Journey_Records**: This flag defines whether to run validation on Journey records.

Possible values: `Yes` / `No`.

**Note:**

- If the **Validations_On_Journey_Records** property is set to deduplication setting, the empty and No, and the user adds some fields in data `NULL` data will not be accepted for data deduplication fields.
- After setting the flag, you must log out from Platform and Journey and log back in to both applications for the changes to reflect.
- For any reason, if the **Validations_On_Journey_Records** property is not accessible in Journey from Platform, you can alternatively set the `journey.journeyrecords.validationrules` property in the `application.properties` file from Journey web. In this case the value True takes the precedence.

For Example - If platform config is set to true and property file has false, then platform config value true will take the precedence. If platform config is set to false and property file has true, then property file value true will take the precedence. If both platform and property file has value as false, only then the validation will not happen.

**Link Configuration**

- Link_URL - [change_me]
- Link_Data_Source_User - [change_me]
- Link_Data_Source_Name - [change_me]
- LinkProjectName - **_app_journey** [Default - Should be kept as is for all partitions]
- Application - **journey**[Default - Should be kept as is for all partitions]

**Note:** From 12.1.8 release Journey supports multi-partition Link support. User can configure same Link instance in multiple Journey partitions.

In each Journey Partition node user needs to configure Link information as below:

1. Navigate to Journey <Partition1> node location -> (Affinium|Journey|partitions|<partition1>| Link_Configurations) and below parameters

   Link_URL: https://Linkhostname.com

   Link_Data_Source_User: <partition1_user>

   Link_Data_Source_Name: <LinkDS>

   LinkProjectName: _app_journey [Default - Should be kept as is for all partitions]

   Application: journey [Default - Should be kept as is for all partitions]

   Now navigate to **Settings > Users > <Partition1_user> > Edit data sources**.

   add <LinkDS> Datasource for Partition1 user

2. Navigate to Journey <Partition2> node location -> (Affinium|Journey|partitions|partition2| Link_Configurations) and below parameters

   Link_URL:https://Linkhostname.com

   Link_Data_Source_User: <partition2_user>

   Link_Data_Source_Name: <LinkDS1>

   LinkProjectName: _app_journey [Default - Should be kept as is for all partitions]

Application: journey [Default - Should be kept as is for all partitions]

Now navigate to **Settings > Users > <Partition2_user> > Edit data sources**.

add <LinkDS1> Datasource for Partition2 user

**Deliver Configuration**

- Deliver_URL - User needs to configure valid Deliver TMS URL [change_me]

  **Note:** The configuration `Affinium|Journey|Deliver_Configurations|Unsubscription_Attribute` is still global i.e. it is not present at partition level.

  The purpose of this node is to handle an un-subscription Event coming from end user who will click on the Unsubscribe landing page provided in any Email Template (created in Deliver). The key-Value pair created under this node in Platform become a mandate requirement to have them as a hidden attribute with same value for the Unsubscribe Landing pages, which needs to be used by Journey for handle the Unsubscription at Journey Level for the deliver template used in Journey.

**Integration**

- preferredCognitiveServiceProvider

**Note:**

From V12.1.7, Journey Engine will also read the partition configuration from platform on startup. Any changes in the configuration inside partition node will require restart of Journey Web and Journey Engine.

Upgrading Journey to 12.1.7 requires platform to be upgraded to 12.1.7.

# Chapter 4. Clean-up Journey Audiences

From V12.1.7 onwards, journey audiences can be removed from live running journeys. There will be a cron job scheduled (customer can configure cron expression on need basis), which will find all audiences with status JOURNEY_COMPLETED and JOURNEY_ENDED and remove those, as well as their related data (like audience responses, milestone, goals, etc data) from respective tables. This removed data will not be recoverable. This will help in improving overall performance of journey engine. To support this, we introduced below properties in engine's application.properties file.

1. cleanup.audience.enable=true/false - Journey data clean-up is enabled/disabled respectively.

   **Note:** This setting is used for deleting audiences of deleted journey and its associated data from Journey system database tables.

   If cleanup.audience.enable=true, then below are the side effects on Journey audience counts:

   a. **Journey Stats count**

      If journey that had audiences older than **(archival.audience.ttl=x)** days ago - the data for these audiences will be deleted and hence the count on journey start node will be decreased accordingly.

      As audience records are deleted, their respective responses (if any) will also get deleted, hence responses (visible on stats) available on touchpoints (Email, SMS, Push, WhatsApp, REST, JDBC, Loop, Salesforce) will also get decreased accordingly.

      **Note:**

      - Publish touchpoint statistics for audiences will remain the same and is not impacted by above deletion.
      - Adtech touchpoint statistics for audiences will remain the same and is not impacted by above deletion.
      - Journey Audience Flow counts across journey canvas branches will not be affected.
      - Audiences that came in journey in less than **(archival.audience.ttl=x)** days will not be affected and will remain as-is on journey statistics and start node.

   b. **Journey Goals and Goal Analysis**

      - Journey that had goals achieved by audiences which are older than **(archival.audience.ttl=x)** days - the goal achieved count will get decreased accordingly after deletion.
      - **For frequency and non-frequency based goals** - As audience response are deleted, achieved target will also get decreased accordingly. Hence, the set target will have to be achieved again.
      - **For non frequency based goals** - If there is journey goal complete criteria marked for aspecific goal, the journey will be completed after achieving the set target.
      - **On Journey Goal Analysis Page** - The graphs for all goals will be displayed as per the new goal achieved count after deletion of journey audiences.

c. **Journey Audience Analysis Tab**

- If journey that had audiences older than **(archival.audience.ttl=x)** days ago, the data for these audiences will be deleted and hence the count on journey audience analysis tab will be decreased, and graphs will be represented accordingly.

d. **Milestones on journey canvas page**

After deletion of audiences, milestones count on journey canvas page will get refreshed only when new audiences comes in Journey milestone and then only user can see reduced count on milestone canvas page.

e. **Milestone Analysis Tab**

If journey that had audiences older than **(archival.audience.ttl=x)** days ago - Count for milestoneanalysis page will get reduces same as **Mileston canvas** page. Hence, **Milestone Achieved By (%)** will be affected.

f. **Journey Performance Tab**

After deletion of audiences , counts on performance tab for Email , SMS and Push Tiles will not have any impact and will remain as-is.

g. **Loop**

If journey that had audiences older than **(archival.audience.ttl=x)** days ago - Response counts shown on loop details link will also be affected accordingly.

2. archival.audience.enable=true/false - Journey audiences archival from **journeyaudiences** to **endjourneyaudience** table is enabled/disabled respectively.

3. archival.audience.ttl=X - If cleanup.audience.enable=true, Journey audiences older than X-days in Journey, then the Journey audiences and their associated data will get deleted at set frequency **archival.audience.cron**, Journey audiences and their associated data less than X-days older in Journey will remain available in Journey.

**Note:** Default value of **archival.audience.ttl** is 365 days. User can set minimum 2 days, if user adds any value less than 2 days, then its equivalent to 2 days for this property.

4. archival.audience.cron=0 0/5 * * * ? -> Journey audience clean up will be executed according to set frequency of this cron job.

   Example:

   cleanup.audience.enable=true

   archival.audience.enable=true

archival.audience.ttl=3

archival.audience.cron=0 0/5 * * * ?

In above case, as archival.audience.enable=true, audiences with status Journey_completed/Journey_ended will be moved from **journeyaudiences** table to **endjourneyaudience** table and as user set cleanup.audience.enable=true and archival.audience.ttl=3, Journey audiences with Journey_completed or Journey_ended and older than set 3-days will get cleaned up along with their data available in the database tables at set frequency "archival.audience.cron=0 0/5 * * * ?"

5. ssl.restclient.http.header.attributes - When the property is set in the engine application.properties file the pipe ( | ) separated values are always added to the headers. When in the REST process box additional headers are define, just append them.

   If engine's application.properties contain a key say ssl.restclient.http.header.attributes=<list of key value pairs>, then these headers will implicit during the runtime while making rest call.

   **Note:** These headers are not displayed on UI. If any of such header is also configured on UI, then its value defined on UI will take priority over the one defined in application.properties.

If journey engine is deployed in cluster mode:

1. To run both archival and cleanup process on one node, make both flags true on that one node.
2. To disable both archival and cleanup processes, make them false on all nodes.
3. To run archival and cleanup processes on different engine nodes, make these properties individually true on any two engine server nodes.

   Example: Let's consider, in cluster we have four engine nodes, if user want to run clean-up process on two different engine node then user needs to do the below settings:

   cleanup.audience.enable=true server 1

   archival.audience.enable=false server 1

   cleanup.audience.enable=false server 2

   archival.audience.enable=true server 2

   cleanup.audience.enable=false server 3

   archival.audience.enable=false server 3

   cleanup.audience.enable=false server 4

   archival.audience.enable=false server 4

**Note:**

While deleting millions of Journey audiences and their related data, QueryTimeOut exception may occur . To avoid this, we are recommending to create below indexes immediately after installing or upgrading Journey to 12.1.7 or before starting Journey engine:

- create index MY_UJR_IDX_LAF on LoopAudienceFlow(audienceId);
- create index MY_UJR_IDX_BIDADM on BatchIDAudiencedataMap(audienceId);
- create index MY_UJR_IDX_JAG on JourneyAudienceGoal(audienceId);
- create index MY_UJR_IDX_ARHD on AudienceResponseHTTPDetail(audienceResponseId);
- create index MY_UJR_IDX_ARE on AudienceResponseExtended(audienceResponseId);
- create index MY_UJR_IDX_ARMD on AudienceResponseMetaData(audienceResponseId);
- create index MY_UJR_IDX_ARI on AudienceResponseInteraction(audienceResponseId);
- create index MY_UJR_IDX_AR on AudienceResponse(audienceId);
- create index MY_UJR_IDX_JAF on JourneyAudienceFlow(audienceId);
- create index MY_UJR_IDX_AWS on AudienceWaitState(audienceId);
- create index MY_UJR_IDX_JDRM on JourneyDeliverResponseMaster(audienceId);
- create index MY_UJR_IDX_JAM on JourneyAudienceMilestone(audienceId);

# Chapter 5. Unica Journey integrations

**Unica Journey execution engines for email**

Unica Journey supports Unica Deliver and Unica Link for email delivery. You can use either for integration with Journey.

**Unica Journey integration with Unica Link**

Unica Link provides capabilities to send communications across email, SMS, CRM, ADTECH and JDBC channels. Unica Link provides the following reference connectors to deliver communications to email, SMS, CRM, ADTECH and JDBC channels.

Install the following reference connectors as per your preference:

- **MailChimp** – for email
- **Mandrill** – for email
- **Twilio** – for SMS
- **Salesforce** – for CRM

Integration with Unica Link allows Journey to integrate with any third-party vendors for email, SMS, CRM, ADTECH and JDBC executions only.

**Table 1. Installation and Configuration of Unica Link**

| Task | Documentation |
|---|---|
| Installation and configuration of Unica Link | See *Unica Link V12.1 Installation Guide*. |
| Installing Unica Link connector app for Journey | See *Unica Link V12.1 Installation Guide*. |
| Installing Unica Link connector – MailChimp | See *Unica Link Mailchimp Connector User Guide*. |
| Installing Unica Link connector – Mandrill | See *Unica Link Mandrill Connector User Guide*. |
| Installing Unica Link connector – Twilio | See *Unica Link Twilio Connector User Guide*. |
| Installing Unica Link connector – Salesforce | See *Unica Link Salesforce Connector User Guide*. |

**Note:** HCL does not provide the account or access to these delivery channel vendors. Based on your preference you can get the entitlements or accounts from these vendors.

**Unica Journey integration with Unica Deliver**

Unica Journey utilizes the capabilities of Unica Deliver for sending email communications. This also helps in capturing the email responses in real-time and process audience Journey. For more details on enabling Unica Deliver integration with Unica Journey, see *Unica Journey Installation Guide*

**Note:** Unica Journey supports Unica Deliver Email, SMS, Whatsapp and Push channels for sending required communications.

**Unica Journey integration with Unica Campaign and Unica Interact**

Unica Journey seamlessly integrates with Unica Campaign and Unica Interact. Unica Campaign and Unica Interact sends audience data to Unica Journey on a specific Kafka topic. The audience data is sent through a Kafka entry source and pushed across all journeys consuming data from these entry sources.

For more information on Unica Campaign and Unica Interact integration with Unica Journey, see the guides mentioned in the following documentation map.

**Journey supports data from multiple partitions of Campaign**

Journey support data from multiple partitions of campaign.

1. Journey application does not support multiple partitions.
2. Only data from multiple partitions of Campaign/Interact/Deliver can be processed in Journeys. For this journey, will run on single partition.

You need to make changes in configuration platform and user roles and permission:

- Campaign flowchart details displayed under the entry sources come from multiple partitions.
- Based on the partition the Deliver templates are displayed in email / SMS / WhatsApp touchpoints.

**Table 2. Integration of Unica Campaign with other HCL products**

| Task | Documentation |
|---|---|
| Integration of Unica Campaign and Unica Journey | See *Unica Campaign Administration Guide* and *Unica Campaign User Guide*. |
| Integration of Unica Campaign and Unica Interact | See *Unica Interact Administration Guide*. |

**Unica Journey integration with Unica Discover**

Unica Journey seamlessly integrates with Unica Discover. Unica Discover sends audience struggle data to Unica Journey. The audience data is sent through REST entry source and pushed across all journeys consuming data from these entry sources. Four scripts will be provided, after installing Journey you need to immediately run the scripts, this will create two entry sources and two data definition called Discover Entry source for CART and Discover Entry source for Form. The scripts are:

- Discover-MariaDB.sql
- Discover-MS-SQL.sql
- Discover-OneDB.sql
- Discover-Oracle.sql

| | |
|---|---|
| DD Name | Cart |
| Description | When Customer abandons any kind of cart or set of selected offerings this event can be triggered. |

**Table 3. Attributes to be sent across**

| Name | Type | Length | Note |
|---|---|---|---|
| Email* | TEXT | 200 | It is a mandatory field. |
| Name | TEXT | 200 | |
| DiscoverSessionId | TEXT | 50 | Discover Session id required to link it back. |
| CartId | TEXT | 50 | Unique id to identify cart. |
| CartValue | NUMBER | | |
| EventDateTime | TIMESTAMP | | Date and time of the event in UTC Longitude |
| EventType | TEXT | | Event Type can be CART_ABANDONED |
| CookieID | TEXT | 1024 | |
| TLT_BROWSER | TEXT | 50 | Browser details |
| TLT_MODEL | TEXT | 50 | Device Details |
| HTTP_ACCEPT_LANGUAGE | TEXT | 50 | Language |

| | |
|---|---|
| DD Name | Form |
| Description | When Customer fills any webform this event can be published. |

**Table 4. Attributes to be sent across**

| Name | Type | Length | Note |
|---|---|---|---|
| Email* | TEXT | 200 | It is a mandatory field. |
| Name | TEXT | 200 | |
| DiscoverSessionId | TEXT | 50 | Discover Session id required to link it back. |
| FormId | TEXT | 50 | Unique id to identify form |
| FormName | TEXT | 100 | |

**Table 4. Attributes to be sent across (continued)**

| Name | Type | Length | Note |
|---|---|---|---|
| EventDateTime | TIMESTAMP | | Date and time of the event in UTC Longitude |
| CookieID | TEXT | 1024 | |
| TLT_BROWSER | TEXT | 50 | Browser details |
| TLT_MODEL | TEXT | 50 | Device Details |
| HTTP_ACCEPT_LANGUAGE | TEXT | 50 | Language |
| EventType | TEXT | | Event Type can be FORM_SUBMITTED, FORM_ABANDONED |

**Note:** From Fixpack 3 onwards Unica Journey integration with Unica Discover feature is available.

# An Introduction to Unica Deliver

Unica Deliver is a web-based, enterprise scale marketing message solution that you can use to conduct outbound bulk messaging and transactional messaging campaigns. Deliver integrates with Unica Campaign and with secure message composition, transmission, and tracking resources that are hosted by Unica.

You can use Deliver to create, send, and track personalized email communication. As Deliver installs and operates with Campaign, you can use Campaign flowcharts to precisely select and segment recipient information to customize each message.

### Select your audience

Use Campaign to select message recipients and data about each person that you can use to personalize each message.

With Deliver, you can reach large numbers of email recipients quickly and personally. However, you can also configure a mailing to automatically send a single email message in response to a transaction.

### Create a message

The Deliver Document Composer provides editing tools that you can use to design, preview, and publish personalized message content. You can create messages with content that you upload to the Document Composer or link to external content when Deliver builds and transmits messages. Deliver provides several ways to design messages that display content conditionally, based on personal data for each recipient.

### Send the message and track responses

Depending on your goals, you can schedule a messaging campaign to run as soon as possible or schedule it to run later. Deliver monitors message delivery and tracks recipient responses. The system returns contact and response data to the Deliver system tables that are installed as part of the Campaign database schema.

### How to get started

To get started, you must install Campaign and have a hosted messaging account.

System administrators must request a hosted messaging account and work with Unica to configure secure access to the remote messaging and tracking systems. Some messaging features are available only upon request to Unica. For more information about establishing a hosted messaging account and configuring access to Unica hosted messaging, see the *Unica Deliver Startup and Administrator's Guide*.

## Unica Deliver integration

To integrate Unica Deliver with Unica Journey, you must make the following configurations inUnica Platform.

1. In Unica Platform, navigate to **Settings > Configuration**.
   **Result**
   The **Configuration categories** page appears.
2. Select **Journey**.
   **Result**
   The **Settings for 'Journey'** page appears.
3. Select **Edit settings**.
   **Result**
   The **(Journey)** page appears.
4. Perform the following steps:
   a. For the **Deliver_Configured** field, select **Yes**.
   b. Click **Save changes**.
5. In the expanded Journey node, select **Deliver_Configurations**.
   **Result**
   The **Settings for 'Deliver_Configurations'** page appears.
6. Select **Edit settings**.
   **Result**
   The **(Deliver_Configurations)** page appears.
7. Perform the following steps:

   a. Provide values for the following fields:
      - **Deliver_URL**: The URL cofigured for Deliver.

      **Note:**

i. User needs to update the TMS URL in Journey configuration. Update the TMS url as SOAP TMS url is not supported, only REST url is supported. Navigate to **Platform > Settings > Configuration > Journey > Deliver_URL**.

ii. From V12.1.6 onwards, users need to update old **Deliver API URL** with new **Deliver REST API URL**. After upgrade, ensure to republish all Journeys with deliver touchpoint.

Location: `(Affinium|Journey|Deliver_Configurations)`

New Deliver URL e.g. http://<host-name>/delivertms/api/deliver/rest/v2/tms

- **Deliver_Partition**: The partition in which the credentials to access the **Deliver_URL** is stored.
- If Journey having any deliver touchpoint configured, User needs to pause and republish the Journey, then only audiences start processing

b. Click **Save changes**.

# Kafka integration

You must configure Kafka in Unica Platform for the Journey node.

### Accessing Kafka_Configurations in Unica Platform

To access Kafka_Configurations, complete the following steps:

1. On Unica Platform, navigate to **Settings > Configuration**.
2. Expand the **Journey** node.
3. Select **Kafka_Configurations**.
4. Select **Edit settings**.

### Mandatory configurations based on CommunicationMechanism value

In the **(Kafka_Configurations)** page, you can select one of the following values for the CommunicationMechanism field:

- NO_SASLPLAINTEXT_SSL
- SASL_PLAINTEXT
- SSL
- SASL_PLAINTEXT_SSL

Based on your selection, the following fields become mandatory:

| **Field name** | **NO_SASLPLAIN TEXT_SSL** | **SASL_PLAIN TEXT** | **SSL** | **SASL_PLAIN TEXT_SSL** |
|---|---|---|---|---|
| KafkaBrokerURL | Yes | Yes | Yes | Yes |

| Field name | NO_SASLPLAIN TEXT_SSL | SASL_PLAIN TEXT | SSL | SASL_PLAIN TEXT_SSL |
|---|---|---|---|---|
| TopicName | Yes | Yes | Yes | Yes |
| sasl.mechanism | | Yes | | Yes |
| UserForKafkaData | | Yes | Yes | Yes |
| sasl.jaas.config.data Source | | Yes | | Yes |
| truststore.location | | | Yes | Yes |
| truststore.password.data Source | | | Yes | Yes |
| keystore.location | | | Yes | Yes |
| keystore.password.data Source | | | Yes | Yes |
| key.password.dataSource | | | Optional | Optional |
| ssl.endpoint.identification. algorithm | | | Yes | Yes |

Make the necessary configurations and click **Save changes**.

**Note:** Running out of disk storage and abruptly shut down the kafka server due to the large size of kafka logs file.

**Note:** Once an audience type is used for configuration, then it cannot be used again for other configurations. While configuring CIF if one audience type is used, then same audience type cannot be used again in the same entry source.

# Kafka Topics

Below Kafka topics get created when Journey engine starts.

| Kafka Topics | Description |
|---|---|
| DATA_CLEAN | Used by Data Clean-up Service to apply the clen-up rules for the filtering of the incoming data. |

| Kafka Topics | Description |
|---|---|
| JOURNEY_EVENT | Used by Orchestration Service to orchestrate the Journey Canvas flow. |
| Kafkalog | Used for async logging. |
| JOURNEY_PARSER | Used by Journey parser Service to parse the Journey Canvas design. |
| DATA_FETCH | Not being used after 12.1.3 |
| DELAYV2 | Consumed by Delay Service to apply delay on the published audience. |
| END | Used by Journey when the audience's journey ends. |
| ASYNC_DATABASE_STREAM_DATALOG | |
| ASYNC_DATABASE_AUDIENCE_RESPONSE | Used by Engine to insert the audience response data to audienceresponse table. |
| ASYNC_DATABASE_DISCARD_DATA | Used by journey to process discarded audience data. |
| ASYNC_DATABASE_AUDIENCE_FLOW | Used by Engine to insert stats records in journeyAudienceFlow Table. |
| ASYNC_DATABASE_DATA_ERRORS | Used by Journey for processing errors. |
| ASYNC_DATABASE_AUDIENCE_BULK_REPONSE | Used by Engine to insert the audience response data to audience bulk response table for Adtech. |
| ASYNC_DATABASE_UNSUBSCRIBE_EMAIL | Used by Engine to insert the registration of subscription of email from Journey. |
| SMS_SEND | Used by SMS Service to send the SMS via configured channel to the published audience. |
| SALESFORCE_SEND | Used for sending the data to salesforce via Link. |
| JOIN | This topic is used by engine, to simply pass on the audiences from one node(predecessor) to another node(successor) by connecting them through a join control. |
| LOOP | This topic is used by engine, to receive the audiences from any of loop start node's predecessor node. Repeats the audiences as per loop limit configuration. |
| ENGAGEMENT_SPLIT | Used by engagement split service to consume audiences. |
| JOURNEY_GOAL | Used by the Goal service to consume data. |
| INCOMING_RESPONSES | Published Responses by Link, consumed by Engine. |

| Kafka Topics | Description |
|---|---|
| PUBLISH_ACTION_POINT | Used by engine publishactionpointservice to get details of jouney , nodes and Audiences |
| DELIVER_RESPONSES | published Responses by Deliver RCT module, consumed by Engine. |
| ADTECH_SEND | Consumed by Adtech Service to process the audience for Adtech. |
| ADD_MILESTONE | |
| PROCESS_MILESTONE | |
| WHATSAPP_SEND | Used by WhatsApp Service to send the message via configured channel to the published audience. |
| JDBC_SEND | Used by Engine to take action of the published audience ids for JDBC touch point. |
| REST_INIT | NOT IN USE |
| REST_API | Used By engine RestAPIService after configuring rest touchpoint in journey. |
| CHRH | Not in use after 12.1.5 |
| DELAY_UPDATE | Used by Engine to update the audience data which is in delay |
| CALLABLE | |
| NOTIFICATION | |
| RESTREQUEST | Used by journey to send the rest request. |
| RESTEXECUTE | Used by engine RestExecutionservice to call secure and non-secure 3rd party API. |
| RESTRESPONSE | Used by journey to consume the response received from the rest service. |
| RESTUPDATE | Used by journey to process the response received from the rest service & update the journey audience detail. |
| RESTAUDIENCERESPONSE | Used in engine RestAudienceResponseService to save Audience response/data. |
| CIFINTEGRATION | USED by journey to consume data sent by CIF from external system. |

| Kafka Topics | Description |
|---|---|
| EMAILWAIT | Used in engines EmailCapacityWaitService In noncomunication hour if request is received it will be in EMAILWAIT (Refer 12.1.7 user Guide page 31) |
| SMSWAIT | Used in enginesSmsCapacityWaitServiceIn noncomunication hour if request is received it will be in SMSWAIT (Refer 12.1.7 user Guide page 31) |
| WHATSAPPWAIT | Used in engines WhatsAppCapacityWaitService In noncomunication hour if request is received it will be in WHATSAPPWAIT (Refer 12.1.7 user Guide page 31) |
| NOTIFICATIONWAIT | Used in engines PushNotificationCapacityWaitService In noncomunication hour if request is received it will be in NOTIFICATIONWAIT (Refer 12.1.7 user Guide page 31) |
| DELIVER_READER | Used by Deliver Helper Service to send the notification via Deliver to the published audience. |
| DEDUP | Used by Journey to consume the deduped data. |
| ESEVENT | |
| RESPONSE_ACTION_EVENT | |
| PAUSE_AUDIENCE_CALC_COUNT | The topic is used by Journey web to request the journey engine to calculate the count of audiences to be paused for a given pause rule. |
| PAUSE_AUDIENCE_EXEC | The topic is used by Journey web to request journey engine to start the pause execution process for a given rule. |
| PAUSE_AUDIENCE_PROC | The topic is used internally between the two components of pause process. The data extraction service in pause process reads the audience info to be paused and sends this information in the form of batches to the writer service through this topic. |
| JOURNEY_CONTROL | |
| DELIVER_RESPONSES_STAGE | Used by Engine Service to initiate the CH/RH responses. |

**Note:** When a kafka instance file is created at Web and engine and configured on Publish touchpoint. On Publish journey, when data is sent, it is observed that Kafka topic is not getting created.

**Workaround** - After adding kafka instance file, user needs to restart engine then it allows to create kafka topic and process data on kafka topic.

# Multiple Kafka

From V12.1.9 onwards, Journey supports multiple kafka under Publish touchpoint.

**How to Configure Multiple Kafka in Web and Engine**

Using multiple Kafka support Journey allow to write on any other KAFKA apart from one used internally. Publish touchpoint will have ability to pick KAFKA instance and then add kafka topic name. User needs to create folder of partition name e.g. **partition1**, **partition2** under Web and Engine location.

`<Journey_Web_Installation_Location>/conf/external_kafka/partitions/partition/partition1`

`<Journey_Engine_Installation_Location>/conf/external_kafka/partitions/partition/partition1`

`sample.producer.properties` file for reference is available under Web and Engine locations. User needs to modify `sample.producer.properties` as per requirement like file name or Kafka host name. User needs to place same file under below location, user needs to make sure instance file name and its contents are same under web and Engine locations. Partition wise instances will be available under instance drop down on Publish TP under Kafka section.

Location: `<Journey_Web_Installation_Location>/conf/external_kafka/partitions/partition/partition1`

`<Journey_Engine_Installation_Location>/conf/external_kafka/partitions/partition/partition1`

When a Kafka instance file is created at Web and Engine configured on Publish touch point user needs to restart Engine else data will not be processed by External Kafka topic.

Once working Kafka instance file is created with any host name or instance name - user is not allow to modify working Kafka instance file name and its content. Instead of modifying existing instance and its contents user needs to create a new Kafka instance file and contents.

**Note:** Before using an external Kafka instance, ensure it is accessible from the machine where the Journey Engine is installed. This can be verified using a Java test program. User needs to create its own java program to test, If issues persist, contact Unica Support team for assistance and to get Java test program.

# Kafka Replication to Partition

### BATCH_IMPORT

All Entry Source that publishes batch data – via a flat file, either CSV, TSC etc, shall publish messages to this topic. Each message shall correspond to one single batch of records.

**Message Attributes**

- Entry Source ID
- Journey ID
- Data
    - Timestamp – time stamp when the message was added to the topic
    - Entry Source Type
    - FQN

**Object Type**

- entrySourceID:
- journeyID:
- entrySourceType: enum
- data: FQN of the file containing the data
- timestamp:

## STREAMING_IMPORT

All Entry Source that publishes streaming data – via a series of individual records, shall publish messages to this topic. Each message shall correspond to one single record.

**Message Attributes**

- Entry Source ID (if available)
- Data (in JSON format)
- Metadata
    - Timestamp – time stamp when the message was added to the topic
    - Entry Source Type
    - Other Metadata

**Object Type**

- entrySourceID: (this is optional)
- entrySourceType: enum
- data: (JSON of single record)
- timestamp:
- metaData: (JSON of other data)

## DATA_MAP

This topic will have messages once the incoming data is cleaned and needs to be mapped to a Data Definition.

**Message Attributes**

- Entry Source ID
- Journey ID
- Data
    - Timestamp – time stamp when the message was entered in the topic
    - Data, in JSON
    - Number of records, if in batch

**Object Type**

- entrySourceID:
- journeyID
- entrySourceType: enum
- timeStamp
- data: <JSON string>
- numRecords:

## RAW_DATA_FETCH

This topic will have messages to fetch unmapped data, from the DATA ARCHIVE table.

**Message Attributes**

- Entry Source ID
- Timestamp – time stamp when the message was entered in the topic
- State – state of data that need to be fetched, RAW or CLEANED
- Number of records – Number of records to be fetched. If not specified, then all records

**Object Type**

- entrySourceID:
- timeStamp
- state: enum
- numberRecords:

## END_JOURNEY

This topic shall hold all messages for the various running Journeys for which some of the audiences have reached a logical end of the defined Journey.

## JOURNEY_EVENTS

This topic shall hold all the messages for the various events during the lifetime of a running Journey instance.

**Message Attributes**

- ID – this shall be a combination of the Journey ID and an optional Node ID
- Data
    - State – the state of the Journey
    - Timestamp
    - Metadata

**Object Type**

- ID: journeyID
- state: enum:
- timeStamp
- data:

**PUBLISHED**

- ID – Journey ID
- Data
    - State – PUBLISHED
    - Timestamp - time stamp when the message was entered in the topic
    - Metadata – FQN for JSON file having the complete Journey structure

**Object Type**

- ID: journeyID
- state: enum:PUBLISHED
- timeStamp
- data: JSON for Journey structure (this can be the serialized form of the JSON)

**PAUSE**

- ID – Journey ID
- Data
    - Timestamp - time stamp when the message was entered in the topic
    - State – PAUSE

**Object Type**

- ID: journeyID
- state: enum:PAUSE
- timestamp
- data: <empty>

**JOURNEY_PAUSED**

- ID – Journey ID
- Data
    - Timestamp - time stamp when the message was entered in the topic
    - State – PAUSED

**Object Type**

- ID: journeyID
- state: enum:PAUSED
- timestamp
- data: <empty>

**PARSED**

- ID – Journey ID
- Data
    - Timestamp - time stamp when the message was entered in the topic
    - State – PARSED

**Object Type**

- ID: journeyID
- state: enum:PARSED
- timestamp
- data: <empty>

**MAPPED**

- ID – Journey ID
- Data
- Timestamp - time stamp when the message was entered in the topic
- State – PARSED

**Object Type**

- ID: journeyID
- state: enum:MAPPED
- timestamp
- data: <empty>

**FETCHED**

- ID – Journey ID
- Data

    - Timestamp - time stamp when the message was entered in the topic
    - List of record IDs

**Object Type**

- ID: journeyID#nodeID
- state: enum:FETCHED
- timestamp
- numberRecords:
- listRecords:

**DELAY_EXECUTED**

- ID – this ID shall be the Journey ID and the node ID for the following node
- Data
    - State – DELAY_EXECUTED
    - Timestamp - time stamp when the message was entered in the topic
    - List of record IDs

**Object Type**

- ID: journeyID#nodeID
- state: enum:DELAY_EXECUTED
- timestamp
- numberRecords:
- listRecords:

**EMAIL_SENT**

- ID – this ID shall be the Journey ID and the node ID for the following node
- Data
    - State – EMAIL_SENT
    - Timestamp - time stamp when the message was entered in the topic
    - List of record IDs

**Object Type**

- ID: journeyID#nodeID
- state: enum:EMAIL_SENT
- timestamp
- numberRecords:
- listRecords:

## JOURNEY_PARSE

This topic shall hold all messages for journeys to be validated and parsed.

**Message Attributes**

- ID – Journey ID
- Data
    - Timestamp - time stamp when the message was entered in the topic
    - State – PARSE
    - Data

**Object Type**

- ID: journeyID
- state: enum:PARSE
- timestamp
- data: <empty>

## JOURNEY_CACHE

This topic shall hold all messages for Journey Data that needs to be added to the cache.

**Message Attributes**

- Journey ID
- Entry Source ID
- Data
    - Timestamp - time stamp when the message was entered in the topic
    - `Data

**Object Type**

- ID: journeyID
- state: enum:PARSE
- timestamp
- data: <empty>

## ASYNC_DATABASE_INSERTS

This topic shall hold all messages that are to be inserted into some database table asynchronously.

**Message Attributes**

- Entity
- Name

- Message
- Timestamp

### JOURNEY_MAP

This topic shall hold all messages for the Journey Map Service. This data shall consist of all the details of each Entry Source associated with the Journey.

**Message Attributes**

- Journey ID
- Data
    - List of Entry Sources for the Journey, which includes ID, type and FQN

### JOURNEY_DATA

This topic shall hold all messages that are to be pushed to database tables for log or state information. These messages do not impact execution of the Journey.

**Message Attributes**

- Service ID
- Data
    - Query
    - Timestamp

### EMAIL_SEND

This topic shall hold all the messages from the running instances of journeys that need emails to be sent out.

**Message Attributes**

- ID – this shall be a combination of the Journey ID and Node ID
- Audience IDs – the list of audience IDs that are to be sent the email

### OUTGOING_EMAIL

This topic shall hold all the messages from the Journey application that need to be sent out as emails. These will either be acted upon by a suitable Adaptor Service or the Universal Connector.

**Message Attributes**

- sourceAppId: name of application - Journey
- sourceInstanceId – Touch Point ID
- Audiences – List of audience data

### EMAIL_RESPONSES

This topic shall hold all the messages received from the delivery engine via configured connector as audience response events to emails sent.

**Message Attributes**

- Touch Point ID
- Audience ID
- Event ID
- Timestamp

### OUTGOING_SMS

This topic shall hold all the messages from the Journey application that need to be sent out as SMS's. These will either be acted upon by a suitable Adaptor Service or the Universal Connector.

**Message Attributes**

- sourceAppId: name of application - Journey
- sourceInstanceId – Touch Point ID
- Audiences – List of audience data

### SMS_RESPONSES

This topic shall hold all the messages received from the delivery engine via configured connector as audience response events to SMS's sent.

**Message Attributes**

- Touch Point ID
- Audience ID
- Event ID
- Timestamp

### DELAY

This topic shall hold all the messages from the running instances of journeys that need to add a delay before the execution of the following touch point.

**Message Attributes**

- ID – this shall be a combination of the Journey ID and Node ID
- Configuration – this shall be the configuration data for the Delay Touch Point
- Structure – this shall be the JSON structure extracted from the Journey JSON

### DECISION_SPLIT

This topic shall hold all the messages from the running instances of journeys that need to split incoming data based on some filter.

**Message Attributes**

- ID – this shall be a combination of the Journey ID and Node ID
- Configuration – this shall be the configuration data for the Decision Split Touch Point
- Structure – this shall be the JSON structure extracted from the Journey JSON

### JOURNEY_PAUSE

This topic shall hold messages from the Orchestration Service to pause a specific Journey.

**Message Attributes**

- ID – the ID of the Journey which must be Paused

### DATA_PAUSED

This topic shall hold all the messages from the running instances of Journeys that have been Paused.

**Message Attributes**

- Name – name of the Source topic
- ID – the Journey ID
- Message – the original message

### JOURNEY_RESUME

This topic shall hold messages from the Orchestration Service to resume a specific Journey.

**Message Attributes**

- ID – the ID of the Journey which must be Resumed

### DATA_RESUMED

This topic shall hold all the messages that were in the Paused state but now need to be resumed by the individual Services.

**Message Attributes**

- Name – name of the Source topic
- ID – the Journey ID
- Message – the original message

### JOURNEY_ENGINE_ERRORS

This topic shall hold all the messages from the engine which are errors that took place and that could be displayed on the Web on an administrative console when required.

**Message Attributes**

- ID – Some ID that identifies the entity that caused the error – Journey ID, Audience ID
- State – State of the error – MEDIUM / HIGH / CRITICAL
- Category – Name of the Service that detected the error
- Text – The actual error text

### JOURNEY_GOAL_VERIFICATION

This topic shall hold all messages for the Journey Goal Verification Service which needs to verify if the goal for the Journey has been achieved.

**Message Attributes**

- ID – Journey ID
- EVENT_TYPE – EMAIL RESPONSE, SMS_REPOSNE, PUSH_RESPONSE
- Data
- Timestamp

### JOURNEY_GOAL_ACHIEVED

This topic shall hold messages for the Service that handles notification for when a Journey has achieved the set goal.

**Message Attributes**

- ID – Journey ID
- EVENT_TYPE – TIME, COUNT
- Timestamp

### JOURNEY_ENGINE_MONITORING

This topic shall be used by the Engine to send monitoring data across to the web.

**Message Attributes**

- Name – Name of Service is message refers to a Service or then ENGINE indicating that the message is for Journey Engine as a whole
- State – STARTING, GOING DOWN, NOT RESPONDING
- Thread ID
- Timestamp
- IP Address

## States, Types & Status

1. **ENTRY SOURCE TYPES**
    - Flat File
    - REST
2. **DATA PIPELINE STATUS**
    - Imported
    - Cleaned
    - Mapped
3. **LOG TYPES**
    - File
4. **LOG SUB TYPES**
5. **DATA PIPELINE STATES**
6. **JOURNEY STATES**
    - PUBLISHED - when the Journey is pushed into the Published state from the front end
    - PARSED – when the Journey is parsed successfully
    - RUNNING – when the Journey has mapped data and has begun execution
    - PAUSED – when the Journey has been Paused, either explicitly or implicitly. The current Run of the Journey shall be completed
    - STOPPRED – when the Journey has been Stopped, either explicitly or implicitly. The current Run of the Journey shall be terminated
    - COMPLETED – when the Journey has completed its logical execution
    - GOAL ACHIEVED – when the Journey has achieved its defined goal
7. **AUDIENCE TABLE STATES**
    - NEW – these are new records for a given Journey ID, that have not been processed
    - PROCESSED – these are records for a given Journey ID, that have been processed
    - COMPLETED – these audience for a given Journey have completed the Journey – the logical end
    - GOAL ACHEIVED – the Journey associated with these audience has achieved its goal

# Chapter 6. Process of Starting and Stopping Journey

**About this task**

Starting Process

1. Starting Process Web
    a. Configure Kafka and Zookeeper
        i. IP – on which Zookeeper/Kafka is running
        ii. PORT- Kafka(default 9092), Zookeeper default port 2181
        iii. Log path
        iv. auto.create.topic.enable = true, this property should be set to true for Engine Publish service to work.
    b. Start Zookeeper, wait 10 sec
    c. Start kafka
    d. configure Journey.xml --(Refer Doc, Doc2 )
    e. Configure Log4j2.xml under conf folder
    f. Start Webserver (JBOSS/TOMCAT/WebSphere)
    g. Start Journey Web application
2. Starting Engine
    a. Configure application.porperties
        i. Add DB Details
        ii. Add Kafka details(eg: spring.kafka.bootstrap-servers=127.0.0.1:9092, 127.0.0.2:9092)
            - Path for Ignite Storage, spring.ignite.storage.path, User thru which engine is executed should have Read and Write access to the path of Ignite folder
        iii. Configure Log4j2.xml under conf folder
        iv. Configure property spring.ignite.ipFinder.List as below:
            - `spring.ignite.ipFinder.List=127.0.0.1:63501,127.0.0.1:63502, 127.0.0.1:63503,127.0.0.1:63504`
        v. Start Engine (java –jar journeyEngine.jar)

**Stopping Process (Steps for No Data Loss)**

a. Stop Webserver
b. Stop Engine - Use Director or find the Journey engine process ID & run below command
    - **Linux** → kill -SIGINT [engine-process-id] OR kill -2 [engine-process-id]
    - **Windows** → taskkill /PID [engine-process-id]
c. Stop kafka
d. Stop Zookeeper

**Note:**

a. If user see below error while starting Journey Web or Engine or Data migration utility on MS sql server DB please apply given workaround:

**Error:** Caused by: javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested targetThe driver could not establish a secure connection to SQL Server by using Secure Sockets Layer (SSL) encryption. Error: "sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target". ClientConnectionId:7a73f167-03c7-4a8c-880b-a9f63f9028f6]]

**Workaround:**

Please add **;encrypt=false;trustServerCertificate=false;** to jdbc url for journey system database and report database

eg. url="jdbc:sqlserver://<DB_HOST_IP>:<PORT_NO>**;databaseName=<databaseName>;encrypt=false;trustServerCertificate=false;**"/>

b. If user see below error while starting Journey Web on Tomcat, please apply below workaround in tomcat

**Error:** org.apache.catalina.util.LifecycleBase.handleSubClassException Failed to start component [Connector[AJP/1.3-8010]] org.apache.catalina.LifecycleException: Protocol handler start failed

**Workaround:**

Please ensure that secretRequired="false" for ajp connector

Path-> C:\apache-tomcat\instance\conf\server.xml

<Connector port="8010" protocol="AJP/1.3" redirectPort="8444" **secretRequired="false"** />

# Chapter 7. Journey user roles and permissions

Before you begin using Unica Journey, you should assign roles and permissions to users.



**Note:** Any changes in configuration requires a restart of Unica Journey. For more information related to security configurations, see *Unica Platform Administrator's Guide*.

## Assigning permissions to Journey Roles

Before assigning a role to a user, you should assign permissions to the available roles.

**About this task**

Journey offers two user roles:

- **JourneyAdmin**
- **JourneyUser**

To assign permissions to both roles, complete the following steps:

1. From the Unica Platform home page, select **Settings > User roles and permissions**.
   **Result**
   The **User roles and permissions** page appears.
2. In the left panel, expand **Unica Journey > partition1**.
   **Result**
   The **partition1** page appears.
3. Select **Assign Permissions**.
   **Result**
   The **(Properties for administrative roles)** page appears.
4. Click **Save and edit permissions**.
   **Result**
   The **(Permissions for partition1)** page appears.
5. Expand **Application**.
6. Set values for the following fields:

| Operations | JourneyAdmin Default setting | JourneyUser Default setting |
|---|---|---|
| **Create Data Definition** | Yes | No |
| **Edit Data Definition** | Yes | No |

| Operations | JourneyAdmin Default setting | JourneyUser Default setting |
|---|---|---|
| **Delete Data Definition** | Yes | No |
| **Create Entry Sources** | Yes | No |
| **Edit Entry Sources** | Yes | No |
| **Delete Entry Sources** | Yes | No |
| **Create Journey** | Yes | Yes |
| **Edit Journey** | Yes | Yes |
| **Delete Journey** | Yes | No |
| **Publish Journey** | Yes | Yes |
| **Complete Journey** | Yes | Yes |
| **Pause Journey** | Yes | Yes |
| **Goal add/modify/delete** | Yes | No |
| **Goal view** | Yes | Yes |
| **Settings add/modify/delete** | Yes | No |
| **Settings view** | Yes | Yes |

**Note:**

- For the **JourneyAdmin** role, we recommend that you do not reduce the permissions and retain the default permissions. By default, **JourneyAdmin** has all permissions.
- For the **JourneyUser** role, provide permissions that you feel is appropriate. You can give the **JourneyUser** all permissions, but it is not recommended.

7. After providing the permissions, click **Save changes**.

# Assigning JourneyAdmin role to a user

To assign **JourneyAdmin** role to a user, complete the following steps:

1. From the Marketing Platform home page, select **Settings > User roles and permissions**.
   **Result**
   The **User roles and permissions** page appears.
2. In the left panel, expand **Unica Journey**.
3. Select **partition1 > JourneyAdmin**.
   **Result**
   The **JourneyAdmin** page appears.
4. In the **Users** section, select a user. For example, `asm_admin`.

**Result**

The **asm_admin (asm_admin)** user details page appears.

5. Select **Edit roles**.

   **Result**

   The **Edit roles** page appears.
6. From the **Available roles** list, select **JourneyAdmin (Unica Journey)** and click the **>>** button to move the role to the **Selected roles** list.
7. Click **Save changes**.

## Assigning JourneyUser role to a user

To assign **JourneyUser** role to a user, complete the following steps:

1. From the Marketing Platform home page, select **Settings > User roles and permissions**.

   **Result**

   The **User roles and permissions** page appears.
2. In the left panel, expand **Unica Journey**.
3. Select **partition1 > JourneyUser**.

   **Result**

   The **JourneyUser** page appears.
4. In the **Users** section, select a user. For example, `journey_example`.

   **Result**

   The **journey_example (journey_example)** user details page appears.
5. Select **Edit roles**.

   **Result**

   The **Edit roles** page appears.
6. From the **Available roles** list, select **JourneyUser (Unica Journey)** and click the **>>** button to move the role to the **Selected roles** list.
7. Click **Save changes**.

## Assigning ContactCentralAdmin role to a user

Unica Journey administrator must assign the ContactCentralAdmin role to Journey users for them to access Contact Central. For enabling the contact central for Journey the value of Contact_Central_Configured value should be set to 'Yes' from platform. By default, the value is set to No. User can select the desired value Yes/No for Contact_Central_Configured from the path Affinium|Journey in Platform. For more information, see *Unica Contact Central Administration Guide*.

# Chapter 8. Journey interaction logging

Interaction Logging for Journey is executed as a scheduled job. The scheduling parameters are set in `application.properties` file of the Journey Engine. The following is an example of the setting:

```
engine.logging.cron=0 15 3 * * ?
```

The scheduled job exports data into an alternative schema which is defined again in the `application.properties` files of the Journey Engine.

```
journey.report.datasource.url =
journey.report.datasource.username =
journey.report.datasource.password =
journey.report.datasource.driver-class-name=
```

Interaction Logging captures the movement of every contact that enters the Journey application as they move through each Journey, either Published or Completed. Even journeys that are Published but Paused are considered for the Interaction Logging.

All Touchpoints, Email, SMS, or CRM, are considered for Interaction Logging as the audience data is sent using the configured integrations through the respective channels. The responses received, from every contact, is also captured.

### Log4j2

Both Journey Web and Journey Engine uses the standard for logging. The `log4j2.xml` file, for both Journey Web and Journey Engine, is placed within the `conf` folder in the installation location.

Both Journey Web and Journey Engine produce regular application logs as well as performance logs. For Journey Web, the default location of the logs is within the `logs` folder. For Journey Engine, the default location of the logs is within the `performancelogs` folder. For both Journey Web and Journey Engine, the mentioned folders are placed within the installation location.

# Chapter 9. Journey logging

In Journey version 12.1.2 and earlier, the property **JOURNEY_LOGGING**, in the `journey_master_config.properties` file, was only used for tracing the audience.

From version 12.1.3 onwards, the **JOURNEY_LOGGING** feature has the following enhancements:

- Indication of whether the Journey started or not at the default Journey engine log level.
- Easy to customize the logging to display any the following logs: `FATAL`, `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACEAUDIENCE`, or `TRACE`.
- Logs Read-from Kafka and Write-to Kafka at Trace level.
- Creating multiple log files per Journey.

### Configuring the JOURNEY_LOGGING property

The rules for configuring the **JOURNEY_LOGGING** property are as follows:

1. Access the `journey_master_config.properties` file and locate the **JOURNEY_LOGGING** property.
2. Uncomment the property for editing it an providing values.
3. The values must be CSV strings and the format of each string must be *J_ID:LEVEL*, where:
    - *J_ID* is the Journey ID
    - *LEVEL* can one of the following values: `FATAL`, `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACEAUDIENCE`, or `TRACE`. The order of *LEVEL* values from highly specific to less specific are as follows: `FATAL`, `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACEAUDIENCE`, or `TRACE`
4. The default value of *J_ID:LEVEL* is an empty string. To modify the value of *J_ID:LEVEL*, uncomment it and provide the required value.
5. To adjust the default value of the logger `com.hcl.journeyLogger`, access `log4j2.xml` and make the changes. Ensure that the log levels configured must be specific to the one in `log4j2.xml`.
6. After configuring *J_ID:LEVEL*, the system creates a *J_ID.log* file. This log file is specific to the Journey ID. If you want to check the logs of multiple Journeys, configure multiple *J_ID:LEVEL* separated by a comma. The system creates a *J_ID.log* files, each for a Journey ID.
7. If you apply *DEFAULT:<LEVEL>* for all Journeys excluding the ones specified in `JOURNEY_LOGGING`:
    a. If you do not provide value for `DEFAULT`, it is considered as `OFF`.
    b. If *J_ID:LEVEL* is not configured in Journey Logging, the log levels of `log4j2.xml` is considered.
8. To capture the logs of Ignite cache(Delay/ES/DS/Wait4Capacity), provide the values in the following format:

    ```
    JOURNEY_LOGGING=181#361:DEBUG,<J_ID>#<NODE_ID>:DEBUG
    ```

    where:

    <*J_ID*> is the Journey ID and <*NODE_ID*> is the Node ID.

**Note:**

- Changes are automatically reloaded within 60 seconds.
- If you configure log level per package/class level in log4j2, ensure that `JOURNEY_LOGGING` is empty or commented. The combination of journey log level and class specific log config is not supported.

**JOURNEY_LOGGING=Archival:WARN** - This parameter is used to set the desired logging level for Journey Engine.

- spring.jpa.properties.hibernate.jdbc.batch_size=900 - This parameter specifies the max number of records in a batch for insert and update using JDBC connection. changing this value may impact the Engine performance.
- spring.jpa.properties.hibernate.order_inserts=true - This parameter specifies if insert statements should be batched or not.
- spring.jpa.properties.hibernate.order_updates=true - This parameter specifies if insert statements should be batched or not.
- journey.kafka.producer.linger.ms=500 - Time in milliseconds Kafka producer waits for gathering 16kb (default) data before making network dispatch
- audience.archival.cron - This is used to mention the archival cron job for the Journey Audience table.

### Log4j

In case of log4j2.xml class or package configurations, access the log4j2.xml file and use the following example by uncommenting it:

```
<Logger name="<<class name>> or <<package name>>" level="<<log level>>" additivity="false">
<appender-ref ref="Routing"/>
<appender-ref ref="console"/>
</Logger>
```

where:

- *<<class name>>* is the fully qualified class name
- *<<package name>>* is the fully qualified package name
- *<<log level>>* is one of the log levels as mentioned in Configuring the JOURNEY_LOGGING property on page 43.

### Interact

```
interact.startSession.enabled=false/true
```

This is a setting in Jounrey Engine's application.properties file to enable debugging the communication between Journey and Interact.

Setting the value to **true** enables the Debug messages

# Chapter 10. Journey Records Validation

When records enter a Journey, users can set validation on the records.

When the **Validations_On_Journey_Records** property is set to `Yes`, the validation is performed on Email format, Data type on required fields, and minimum and maximum length of Data Definition for the required fields.

By default, the **Validations_On_Journey_Records** property is set to `Yes`.

This property can be handled from Platform side under Platform Configuration. You can navigate under Platform configuration settings: Settings for `Affinium|Journey|Journey_Configurations`.

**Validations_On_Journey_Records**: This flag defines whether to run validation on Journey records.

Possible values: `Yes` / `No`.

**Note:**

- If the **Validations_On_Journey_Records** property is set to `No`, and the user adds some fields in data deduplication setting, the empty and `NULL` data will not be accepted for data deduplication fields.
- After setting the flag, you must log out from Platform and Journey and log back in to both applications for the changes to reflect.
- For any reason, if the **Validations_On_Journey_Records** property is not accessible in Journey from Platform, you can alternatively set the `journey.journeyrecords.validationrules` property in the `application.properties` file from Journey web. In this case the value **True** takes the precedence.

  For Example - If platform config is set to true and property file has false, then platform config value true will take the precedence. If platform config is set to false and property file has true, then property file value true will take the precedence. If both platform and property file has value as false, only then the validation will not happen.

# Chapter 11. Configuring OptIn Optout functionality

In 12.1.9 release, Journey supports opt out/in for Email, SMS and Whatsapp channels. Opt out/in is not supported for Push channel.

Journey touchpoint respect the contact central preferences before executing for Opt-in Opt-out specific to the channel and do not send communication if audiences are opted out. This should apply to both Link and Deliver type of touchpoint.

To enable the Opt-Out functionality in Journey, users must have Contact Central installed and enabled in Unica and along with Journey partition wise contact central flag should be enabled.

A scheduler is default configured to fetch audience data every 12 hours from Contact Central to Journey. Users can modify the schedule by updating below property available in the engine's application.properties file. Every time when jobs executed Journey will take only updated and delta audiences from Control Central to Journey.

```
engine.contactcentral.consent.data.cron=0 0 */12 * * ?
```

When above consent jobs executed by Journey engine at set frequency, opted out audiences data is retrieved from Contact Central and stored in the Journey channel wise consent database tables.

Database tables : journeyaudiencewhatsappconsent,journeyaudiencesmsconsent,JourneyAudienceEmailConsent.

Optional: The default page size for storing data in the Journey tables is 1,000 records per iteration. To change this, users can set engine.contactcentral.consent.pagesize=2000 (or another desired value) in the engine's application.properties file.

After making any changes to the application.properties file, users must restart the engine service for the changes to take effect.

# Chapter 12. Journey GDPR

## Accessing Journey GDPR

You can access the GDPR tool from the Journey application folder. The location is as follows:

```
<Journey_Home>\Journey\tools\GDPR\
```

**GDPR supports > MariaDB**, **MS Sql server**, **OneDb data bases** along with **Oracle**

## Executing Journey GDPR

To execute Journey GDPR, complete the following steps:

1. Make changes to the following properties in the `gdpr.properties` file:

| Property Name | Example value | Notes |
|---|---|---|
| `Journey.audience.DBType` | `ORACLE` | Currently, Journey supports only Oracle. |
| `Journey.audience.Db. Schema.Name` | `JourneyUser` | Schema name used in the Journey database. |
| `Journey.audience.Field` | `email/mobileNumber` | Field name in the input CSV file. |
| `Journey.audience.Csv` | `<GDPR_HOME>/sample/JourneyAudiences.csv` | Replace `<GDPR_HOME>` with the current directory path.<br>This is the input `csv` file containing records that you need to opt out from the Journey. |
| `Journey.audience.Output` | `<GDPR_HOME>/JourneyAudiences.sql` | The `JourneyAudiences.sql` is the output file name containing all the SQL queries used to drop all the records from the Journey application. Replace `<GDPR_HOME>` with the current directory path. |
| `Journey.audience.Output. FileSizeLimit` | `10` | Value is in MBs. When the file size exceeds the entered value, it will generate multiple files with the following suffixes: `JourneyAudiences _0`, `JourneyAudiences _1`, and so on. |

2. **Note:** If you see any errors, you can trace it using this log file.

3. To execute the file, perform one of the following steps:
    a. For Windows, locate and execute the `gdpr_purge.bat` file. For example, if the file `gdpr_purge.bat` is in the `D:\workspace\HCL_GDPR\dist\journey\` location, run the `gdpr_purge.bat` file.
    b. For UNIX-based systems, locate and execute the `gdpr_purge.sh` file. For example, if the file `gdpr_purge.sh` is in the `\workspace\HCL_GDPR\dist\journey\` location, run the command ./gdpr_purge.sh.
4. After running `gdpr_purge.bat` (for Windows) or `gdpr_purge.sh` (For Linux), output files *"JourneyAudiences 0"*, *"JourneyAudiences _1"*, *"JourneyAudiences _2"* and so on will be generated at location <GDPR_HOME> specified in above steps. Number of files generated will depend on filesize specified.
5. *"JourneyAudiences_x"* file will have delete queries for records mentioned in JourneyAudiences.csv
6. These queries need to be run manually in "Journey" database as required to have the records deleted from the journeyaudiences table.

GDPR utility removes records from following table: JourneyAudiences, AudienceResponse, AudienceResponseMetaData, AudienceResponseInteraction, JourneyAudienceMilestone and JourneyAudienceGoal. However, it does not delete the data from respective tables, where aggregated counts are stored. For example, tables like journeyFlow, journeyAudienceFlow, JourneyGoalContactTransaction etc. Hence, there will be count mismatch on UI.

With GDPR tool, user would not be able to delete the customer data from either Publish Kafka topic or from the files available on the file system. User need to Delete this data manually as per their requirement.

With GDPR tool user will not able to delete customer data exported by JDBC connector.

# Chapter 13. Kafka authentication using SSL

If you are using your organization's Kafka instance, you can use certificates configured for that Kafka instance. You are not required to generate SSL key and certificates and obtain the client certificates to configure in Journey application properties.

If you do not have the certificates, you can generate self-signed certificate authority (CA), which is simply a public-private key pair and certificate.

You must add the same CA certificate to each Kafka client and broker's trust store.

### Generate SSL key and certificate for each Kafka broker

To generate self-signed certificates for Kafka server, complete the following steps.

**Prerequisites**

- You must have Java keytool and OpenSSL to generate certificates and trust store.
- Optionally, you can use any SSL certificate generation utility instead of OpenSSL.

1. To deploy SSL, generate the key and the certificate for each machine in the cluster. Generate the key into a temporary keystore initially so that you can export and sign it later with CA.

   ```
   keytool -keystore kafka.server.keystore.jks -alias localhost -validity 365 -genkey
   ```

   - keystore: The keystore file that stores the certificate. The keystore file contains the private key of the certificate; therefore, it needs to be kept safely.
   - validity: The valid time of the certificate in days.
2. Create your own CA (certificate authority)

   ```
   openssl req -new -x509 -keyout ca-key -out ca-cert -days 365
   ```

   The generated CA is simply a public-private key pair and certificate, and it is intended to sign other certificates.
3. Add the generated CA to the clients' trust store so that the clients can trust this CA.
   - `keytool -keystore kafka.server.truststore.jks -alias CARoot -import -file ca-cert`
   - `keytool -keystore kafka.client.truststore.jks -alias CARoot -import -file ca-cert`
4. Sign all certificates in the keystore with the CA generated.
   a. Export the certificate from the keystore:

      ```
      keytool -keystore kafka.server.keystore.jks -alias localhost -certreq -file cert-file
      ```
5. Sign it with CA.

   ```
   openssl x509 -req -CA ca-cert -CAkey ca-key -in cert-file -out cert-signed -days 365 -CAcreateserial -passin
   pass:<password>
   ```
6. Import both the certificates of the CA and the signed certificate into the keystore.

   ```
   keytool -keystore kafka.server.keystore.jks -alias CARoot -import -file ca-cert
   ```

```
keytool -keystore kafka.server.keystore.jks -alias localhost -import -file cert-signed
```

7. Create client keystore and import both certificates of the CA and signed certificates to client keystore. These client certificates will be used in application properties.

```
keytool -keystore kafka.client.keystore.jks -alias localhost -validity 365 -genkey

keytool -keystore kafka.client.keystore.jks -alias localhost -certreq -file cert-file

openssl x509 -req -CA ca-cert -CAkey ca-key -in cert-file -out cert-signed -days 365 -CAcreateserial -passin
pass:<password>

keytool -keystore kafka.client.keystore.jks -alias CARoot -import -file ca-cert

keytool -keystore kafka.client.keystore.jks -alias localhost -import -file cert-signed
```

# Configuring Kafka server, Journey and Link components with SSL

The server certificates to be used for Kafka server and client certificates must be used by any application connecting to Kafka server including Journey Web, Journey Engine, Unica Link – Kafka-link or any other tools you require to connect to this Kafka server.

To configure Kafka Server, Journey components, and Link component with SSL authentication, execute the procedures provided in the following sections.

## Configuring Kafka Server with SSL authentication

You must use the following server certificates for Kafka server only. Share these certificates on the required machines and make a note of password.

- `kafka.server.keystore.jks`
- `Kafka.server.truststore.jks`

Update the following server.properties in Kafka server config directory.

```
listeners=SSL://<KAFKA_HOST>:<KAFKA_PORT>
ssl.keystore.location=/PATH/kafka.server.keystore.jks
ssl.keystore.password= password
ssl.key.password= password
ssl.truststore.location= /PATH/kafka.server.truststore.jks
ssl.truststore.password= password
ssl.endpoint.identification.algorithm=
ssl.client.auth=required
security.inter.broker.protocol=SSL
```

## Configure Journey engine with Kafka SSL

Use the following client certificates and share these certificates on the required machines and make a note of password.

- `Kafka.client.keystore.jks`
- `kafka.client.truststore.jks`

1. Update `journey_engine_master.config` from `<JOURNEY_HOME>/Engine/` directory.
2. Update the following property values.

```
kafka.security.enabled=Y
kafka.security.protocols.enabled=SSL
security.protocol=SSL
ssl.truststore.location= /PATH/kafka.client.truststore.jks
ssl.truststore.password=<ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION
TOOL>
ssl.keystore.location= /PATH/kafka.client.keystore.jks
ssl.keystore.password=<ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION TOOL>
ssl.key.password=<ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION TOOL>
ssl.endpoint.identification.algorithm=
```

## Configuring Journey web with Kafka SSL

1. Update Journey Web `application.properties` file from `<JOURNEY_HOME>/Web/properties/` directory.
2. Update the following property values.

```
kafka.security.enabled=Y
kafka.security.protocols.enabled=SSL
ssl.truststore.location= /PATH/kafka.client.truststore.jks
ssl.truststore.password= <ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION
TOOL>
ssl.keystore.location= /PATH/kafka.client.keystore.jks
ssl.keystore.password= <ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION
TOOL>
ssl.key.password= <ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION TOOL>
ssl.endpoint.identification.algorithm=
```

### Configuring Unica Link component with SSL

Update the following property values in Unica Link installations - `kafkalink.properties` file.

```
security.ssl=true
security.protocol=SSL
ssl.truststore.location= /PATH/kafka.client.truststore.jks
ssl.truststore.password=password
security.authentication=username
ssl.keystore.location= /PATH/kafka.client.keystore.jks
ssl.keystore.password=password
ssl.key.password=passwordssl.endpoint.identification.algorithm=
```

# Configuring Kafka server, Journey and Link Components with SASL

To configure Kafka Server, Journey components, and Link component with SASL authentication, execute the procedures provided in the following sections.

## Configuring Kafka Server with SASL authentication

1. Specify JVM parameter in `kafka-run-class.bat/sh`.

   set `JAVA_OPTS=%JAVA_OPTS%`

   `-Djava.security.auth.login.config=/PATH/kafka_server_jaas.conf`

   `set COMMAND=%JAVA% %JAVA_OPTS% %KAFKA_HEAP_OPTS%`

   `%KAFKA_JVM_PERFORMANCE_OPTS% %KAFKA_JMX_OPTS% %KAFKA_LOG4J_OPTS% -cp`

   `"%CLASSPATH%" %KAFKA_OPTS% %*`

   Sample `jaas.config` file:

   ```
   KafkaServer {
      org.apache.kafka.common.security.plain.PlainLoginModule required
      username="admin"
      password="admin-secret"
      user_admin="admin-secret"
      user_alice="alice-secret";
   };
   ```

   ```
   KafkaClient {
     org.apache.kafka.common.security.plain.PlainLoginModule required
     username="alice"
     password="alice-secret";
   };
   ```

2. Update the following Kafka server properties file from `KAFKA_SERVER/config/server.properties`.

   ```
   listeners=SASL_PLAINTEXT:// <KAFKA_HOST>:<KAFKA_PORT>
   security.inter.broker.protocol=SASL_PLAINTEXT
   sasl.mechanism.inter.broker.protocol=PLAIN
   sasl.enabled.mechanisms=PLAIN
   ```

3. In case of WebSphere App server

   Add following configuration in `<Websphere_Location>\profiles<profileName>\properties\wsjaas.conf`

   ```
   KafkaClient

   {org.apache.kafka.common.security.plain.PlainLoginModule required
   username="alice"
   password="alice-secret"; }
   ```

4. Once you've updated the JAAS configuration, ensure that WebSphere knows about it.

   Add this to the JVM arguments:

   - Open WebSphere Admin Console.
   - Go to **Servers > Server Types > WebSphere application servers > Select your server > Java and Process Management > Process Definition > Java Virtual Machine > Custom Properties**.
   - Add a new custom property:

- Name: java.security.auth.login.config
- Value: C:\path\to\your\kafka_client_jaas.conf

## Configure Journey Engine with Kafka SASL

1. Update Journey Engine `log4j2.xml` file from `<JOURNEY_HOME>/Engine/conf/` directory. Uncomment the following lines in `log4j2.xml`.

```
 <!-- Kafka SASL configuration -->
<Property name="security.protocol">${sys:security.protocol}</Property>
<Property name="sasl.mechanism">${sys:sasl.mechanism}</Property>
```

2. Update `journey_engine_master.config` from `<JOURNEY_HOME>/Engine/` directory. Update the following property values.

```
kafka.security.enabled=Y
kafka.security.protocols.enabled=SASL_PLAINTEXT
security.protocol=SASL_PLAINTEXT
sasl.mechanism=PLAIN
java.security.auth.login.config=./kafka_client_jaas.conf
```

## Configuring Journey Web with Kafka SASL

Update Journey Web `application.properties` file from `<JOURNEY_HOME>/Web/properties/` directory.

```
kafka.security.enabled=Y
kafka.security.protocols.enabled=SASL_PLAINTEXT
 java.security.auth.login.config=/PATH/kafka_client_jaas.conf
```

## Configuring Unica Link component with Kafka SASL

Update the following property values in Unica Link installation - `kafkalink.properties` file.

```
security.sasl =true
security.protocol=SASL_PLAINTEXT
security.sasl.auth.login.config =/PATH/kafka_client_jaas.conf
sasl.mechanism=PLAIN
```

# Configuring Kafka server and Journey Components with SASL_SSL configuration

To configure Kafka Server and other Journey components with SASL authentication, complete the procedures provided in the following sections.

**Note:** Unica Link does not support connecting to Kafka-link using SASL_SSL authentication mechanism. You must either use SASL or SSL authentication mechanism.

## Configuring Kafka Server with Kafka SASL_SSL

Update the following server.properties in Kafka server configuration directory.

```
listeners=SASL_SSL:// <KAFKA_HOST>:<KAFKA_PORT>
security.inter.broker.protocol=SASL_PLAINTEXT
sasl.mechanism.inter.broker.protocol=PLAIN
sasl.enabled.mechanisms=PLAIN
ssl.keystore.location=/PATH/kafka.server.keystore.jks
ssl.keystore.password=password
ssl.key.password= password
ssl.truststore.location=/PATH/kafka.server.truststore.jks
ssl.truststore.password= password
ssl.endpoint.identification.algorithm=
ssl.client.auth=required
security.inter.broker.protocol=SSL
```

1. In case of WebSphere App server

   Add following configuration in `<Websphere_Location>\profiles<profileName>\properties\wsjaas.conf`

   ```
   KafkaClient

   {org.apache.kafka.common.security.plain.PlainLoginModule required
   username="alice"
   password="alice-secret"; }
   ```

2. Once you've updated the JAAS configuration, ensure that WebSphere knows about it.

   Add this to the JVM arguments:
   - Open WebSphere Admin Console.
   - Go to **Servers > Server Types > WebSphere application servers > Select your server > Java and Process Management > Process Definition > Java Virtual Machine > Custom Properties**.
   - Add a new custom property:
     - Name: java.security.auth.login.config
     - Value: C:\path\to\your\kafka_client_jaas.conf

## Configuring Journey Engine with Kafka SASL_SSL

1. Update Journey Engine `log4j2.xml` file from `<JOURNEY_HOME>/Engine/conf/` directory.

   Uncomment the following lines in `log4j2.xml`.

   ```
   <Property name="sasl.mechanism">${sys:sasl.mechanism}</Property>
   <Property name="security.protocol" >${sys:security.protocol}</Property>
   <Property name="ssl.truststore.location" >${sys:ssl.truststore.location}</Property>
   <Property name="ssl.truststore.password">${sys:ssl.truststore.password}</Property>
   <Property name="ssl.keystore.location">${sys:ssl.keystore.location}</Property>
   ```

```
<Property name="ssl.keystore.password">${sys:ssl.keystore.password}</Property>
<Property name="ssl.key.password">${sys:ssl.key.password}</Property>
<Property
 name="ssl.endpoint.identification.algorithm">${sys:ssl.endpoint.identification.algorithm}</Property>
```

2. Update the following `journey_engine_master.config` from `<JOURNEY_HOME>/Engine/` directory.

   Update the following property values.

```
kafka.security.enabled=Y
kafka.security.protocols.enabled=SASL_SSL
ssl.truststore.location=/PATH/kafka.client.truststore.jks
ssl.truststore.password=<ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION TOOL>
ssl.keystore.location=/PATH/kafka.client.keystore.jks
ssl.keystore.password=<ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION TOOL>
ssl.key.password=<ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION TOOL>
ssl.endpoint.identification.algorithm=
java.security.auth.login.config=/PATH/kafka_client_jaas.conf
```

## Configuring Journey Web with Kafka SASL_SSL

Update the following Journey Web `application.properties` file from `<JOURNEY_HOME>/Web/properties/` directory.

```
kafka.security.enabled=Y
kafka.security.protocols.enabled=SASL_SSL
ssl.truststore.location=/PATH/kafka.client.truststore.jks
ssl.truststore.password=<ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION TOOL>
ssl.keystore.location=/PATH/kafka.client.keystore.jks
ssl.keystore.password=<ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION TOOL>
ssl.key.password=<ENCYPTED PASSWORD WITH JOURNEY ENCRYPTION TOOL>
ssl.endpoint.identification.algorithm=
java.security.auth.login.config=/PATH/kafka_client_jaas.conf
```

## Configuring Journey Web with Kafka SASL_SSL and mechanism SCRAM-SHA-512

```
spring.kafka.bootstrap-servers=<kafka brokers list>
kafka.security.protocols.enabled=SASL_SSL
sasl.mechanism=SCRAM-SHA-512
ssl.truststore.enable=NO
java.security.auth.login.config= <Path to file - kafka_jaas.conf>
```

## Configuring Journey Engine with Kafka SASL_SSL and mechanism SCRAM-SHA-512

Update the below changes in application.properties:

```
spring.kafka.bootstrap-servers=<kafka brokers list>
ssl.truststore.enable=NO
```

Changes in journey_master_config.properties

```
kafka.security.protocols.enabled=SASL_SSL
sasl.mechanism=SCRAM-SHA-512
```

```
ssl.truststore.enable=NO
java.security.auth.login.config= <Path to file - kafka_jaas.conf>
```

# Affinium|Journey|Kafka_Configurations

## KafkaBrokerURL

**Description**

Use this property to define IP and port on which Zookeeper or Kafka is running.

**Default value**

No default value defined.

**Valid Values**

Any valid Kafka broker URL.

## CommunicationMechanism

**Description**

Specifies the configuration for Kafka client authentication.

**Default value**

No default value defined.

**Valid Values**

In the Kafka configurations page, you can select one of the following values for the CommunicationMechanism field depending on your organization's Kafka server's stream security.

- NO_SASLPLAINTEXT_SSL
- SASL_PLAINTEXT
- SSL
- SASL_PLAINTEXT_SSL

## sasl.mechanism

**Description**

Specifies the Kafka client authentication.

**Default value**

No default value defined.

**Valid Values**

You can select one of the following values depending on Kafka server's authentication configurations.

- SASL_PLAINTEXT
- SASL_PLAINTEXT_SSL
- SSL

### UserForKafkaDataSource

**Description**

Specifies the HCL Unica user that references the data source that contains the Kafka services access credentials. You configure this value when you create a system user.

**Default value**

No default value defined.

**Valid Values**

Any valid user that reference Kafka datasource

### sasl.jaas.config.dataSource

**Description**

Kafka uses the Java Authentication and Authorization Service (JAAS) for SASL configuration. You must provide JAAS configurations for all SASL authentication mechanisms.

**Default value**

No default value defined.

**Valid Values**

Refer “listener.name.sasl_ssl.plain.sasl.jaas.config” as in kafka_home/server.properties

### truststore.location

**Description**

Specifies the path of “kafka.server.truststore.jks”

**Default value**

No default value defined.

**Valid Values**

Path of “kafka.server.truststore.jks” file as mentioned in kafka_home/server.properties/ssl.truststore.location.

### truststore.password.dataSource

**Description**

Specifies the Platform data source that contains the Kafka truststore login credentials. You configure this value when you create a system user.

**Default value**

No default value defined.

**Valid Values**

Path of “kafka.server.keystore.jks” as mentioned in `kafka_home/server.properties/ssl.keystore.location`.

## keystore.password.dataSource

**Description**

Specifies the Platform data source that contains the Kafka keystore login credentials. You configure this value when you create a system user.

**Default value**

No default value defined.

**Valid Values**

Datasource which contains the kafka keystore login credentials

## key.password.dataSource

**Description**

Specifies the Platform data source that contains the Kafka key login credentials. You configure this value when you create a system user.

**Default value**

No default value defined.

**Valid Values**

Datasource which contains the kafka key login credentials

## ssl.endpoint.identification.algorithm

**Description**

Specifies the endpoint identification algorithm used by clients to validate server host name. Disable server host name verification by setting ssl.endpoint.identification.algorithm to an empty string.

**Default value**

empty

**Valid Values**

Refer ssl.endpoint.identification.algorithm as mentioned in kafka_home/server.properties.

**Note:** On premise Confluent Kafka support - Journey works with on-premise Confluent Kafka with no security mechanism. Google Managed Confluent Kafka - journeys works with google managed confluent kafka with security mechanism as SASL_SSL with PLAIN mechanism and no SSL certificates.

# Chapter 14. Kafka authentication using Kerberos

Clients (producers, consumers, connect workers, etc) will authenticate to the cluster with their own principal (usually with the same name as the user running the client), so obtain or create these principals as needed. Then configure the JAAS configuration property for each client. Different clients within a JVM may run as different users by specifying different principals.

Before starting Kafka kerberos configuration with Journey:

From Kerberos hosted machine copy all the files available at location `/var/kerberos/krb5kdc` to Journey installation location `<Journey-HOME>/Journey/Web/properties`

From Kerberos hosted machine copy **krb5.conf** file available at location /etc to Journey installation location -> `<Journey-HOME>/Journey/Web/properties`

**Note:** Host name of `kafka_Hosted` machine is added on Unica application hosted machine and vice versa and make sure that Kafka communication working good between two machines.

1. **Enable the Kafka Security for Journey Engine and Journey WEB**
   - **For Journey Engine** - Please replace the krb5 file location to the `<JOURNEY_HOME>/Engine/journey_master_config.properties`

     kafka.security.enabled=Y

     kafka.security.protocols.enabled=GSSAPI
   - **For Journey WEB** - Please replace the krb5 file location to the `<JOURNEY_HOME>/WEB/properties/application.properties`

     kafka.security.enabled=Y

     kafka.security.protocols.enabled=GSSAPI
2. Make sure the keytabs configured in the JAAS configuration are readable by the operating system user who is starting kafka client.
   - **For Journey Engine** - Please replace the kafka_server_jaas.conf file location to the `<JOURNEY_HOME>/Engine/journey_master_config.properties`

     `java.security.auth.login.config = <jass LOCATION> example - /etc/kafka_server_jaas.conf`
   - **For Journey WEB** - Please replace the kafka_server_jaas.conf file location to the `<JOURNEY_HOME>/WEB/properties/application.properties`

     `java.security.auth.login.config = <jass LOCATION> example - /etc/kafka_server_jaas.conf`
3. Pass the krb5 file locations as JVM parameters to each client JVM

- **For Journey Engine** - Please replace the krb5 file location to the `<JOURNEY_HOME>/Engine/journey_master_config.properties`

  `java.security.krb5.conf = <krb5 LOCATION> example - /etc/krb5.conf`

- **For Journey WEB** - Please replace the krb5 file location to the `<JOURNEY_HOME>/WEB/properties/application.properties`

  `java.security.krb5.conf = <krb5 LOCATION> example - /etc/krb5.conf`

4. While configuring Kerberos with Kafka add/update below changes in below file:

   Location - `/<Journey-HOME>/Journey/KafkaStandalone/config/server.properties`

```
sasl.enabled.mechanisms=GSSAPI
sasl.mechanism.inter.broker.protocol=GSSAPI
security.inter.broker.protocol=SASL_PLAINTEXT
listeners=SASL_PLAINTEXT://0.0.0.0:9092
advertised.listeners=SASL_PLAINTEXT://<Kafka_Kerberos_principle_host_name>:9092
#E.g.
 advertised.listeners=SASL_PLAINTEXT://
ip-10-10-10-100.ap-south-1.compute.internal.nonprod.hclpnp.com:9092
listener.name.sasl_plaintext.gssapi.sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule
 required useKeyTab=true storeKey=true
keyTab="/var/kerberos/krb5kdc/kfkserver.keytab" principal="kafka/<Kafka_Kerberos_principle_host_name>";
sasl.kerberos.service.name=kafka
#e.g. keyTab="/var/kerberos/krb5kdc/kfkserver.keytab"
 principal="kafka/ip-10-10-10-100.ap-south-1.compute.internal.nonprod.hclpnp.com";
sasl.kerberos.service.name=kafka
```

   **Note:** This value will change as per your kafka principle

   `ip-10-10-10-100.ap-south-1.compute.internal.nonprod.hclpnp.com@HCLPNP.COM`

5. Add Kafka kerberos host name in Journey Engine and Journey WEB application.properties file

```
spring.kafka.bootstrap-servers=<Kafka_Kerbersos_Host_Name>:9092
```

6. Create folders by name -> **krb5.conf.d** and **log** at location `/<Journey-HOME>/Journey/Web/properties`
7. update file `kafka_server_kerberos_jaas.conf` OR `equivalent Kafka Kerberos client conf` available at location `/<Journey-HOME>/Journey/Web/properties` with applicable kerberos details and recent paths as per Journey installation location

```
keyTab="/var/kerberos/krb5kdc/zkserver.keytab"
principal="zookeeper/ip-10-10-10-10.ec2.internal.nonprod.hclpnp.com";
keyTab="/var/kerberos/krb5kdc/kfkserver.keytab"
principal="kafka/ip-10-10-10-10.ec2.internal.nonprod.hclpnp.com"
```

8. update file **krb5.conf** available at location `/<Journey-HOME>/Journey/Web/properties` with applicable kerberos details and recent paths as per Journey installation location:

```
includedir /<Journey-HOME>/Journey/Web/properties/krb5.conf.d/
 [logging]
 default = FILE:/<Journey-HOME>/Journey/Web/properties/log/krb5libs.log
```

```
kdc = FILE:/<Journey-HOME>/Journey/Web/properties/log/krb5kdc.log
admin_server = FILE:/<Journey-HOME>/Journey/Web/properties/log/kadmind.log
```

### JBOSS Setting for Kerberos Kafka Authentication

The below changes has been done in Jboss standalone.xml file to before start the Journey web:

1. Go to `<JBOSS_HOME>\standalone\configuration\`
2. Open standalone.xml
   a. **Confirm the servlet-container node** must have all the below values. If not please replace the node with below provided set of lines

```
<servlet-container name="default" disable-caching-for-secured-pages="false">
<jsp-config/>
<websockets/>
</servlet-container>
```

   b. **kafka_server_kerberos_jaas.conf and krb5.conf file path configure in standalone.xml as mention below**

```
<system-properties>
 <property name="java.security.auth.login.config"
 value="<Journey-HOME>/Journey/Web/properties/kafka_server_kerberos_jaas.conf"/>
 <property name="java.security.krb5.conf"
 value="<Journey-HOME>/Journey/Web/properties/krb5.conf"/>
 <property name="java.security.krb5.debug" value="true"/>
 <property name="java.security.disable.secdomain.option" value="true"/>
 </system-properties>
```

   c. **Add the KafkaClient and KafkaServer conf in standalone.xml as shown below**

```
<security-domain name="KafkaClient" cache-type="default">
 <authentication>
 <login-module code="com.sun.security.auth.module.Krb5LoginModule" flag="required">
 <module-option name="storeKey" value="true"/>
 <module-option name="useKeyTab" value="true"/>
 <module-option name="refreshKrb5Config" value="true"/>
 <module-option name="principal" value="<CLIENT-PRINCIPLE>"/> example -
 "kafka/ip-10-10-10-10.ec2.nonprod.hclpnp.com"
 <module-option name="keyTab" value="<Journey-HOME>/Journey/Web/properties/kfkserver.keytab"/>
 <module-option name="doNotPrompt" value="true"/>
 </login-module>
 </authentication>
</security-domain>
<security-domain name="KafkaServer" cache-type="default">
 <authentication>
 <login-module code="com.sun.security.auth.module.Krb5LoginModule" flag="required">
 <module-option name="storeKey" value="true"/>
 <module-option name="useKeyTab" value="true"/>
 <module-option name="refreshKrb5Config" value="true"/>
 <module-option name="principal" value="<Server-PRINCIPLE>"/> example -
 "zookeeper/ip-10-10-10-10.ec2.nonprod.hclpnp.com"
 <module-option name="keyTab" value="<Journey-HOME>/Journey/Web/properties/zkserver.keytab"/>
 <module-option name="doNotPrompt" value="true"/>
 </login-module>
 </authentication>
</security-domain>
```

**Note:** The values of the above mentioned parameter keyTab and principle must be same which has been generated at the time of Kerberos Server setup.

3. After saving the above changes in `Jboss <JBOSS_HOME>\standalone\configurationstandalone.xml`, Journey Web can be started.After configuring Journey with Kafka Kerberos, on Journey while creating Rest Entry source if application gives error like below user needs to verify node configuration of Confirm the **servlet-container** node as below.

   Errors: Unable to fetch rest Api Or Unable to fetch Kafka

   - Confirm the **servlet-container** node must have all the below values. If not please replace the node with below provided set of lines at below location:

     ```
     <JBOSS_HOME>\standalone\configuration\standalone.xml
     <servlet-container name="default" disable-caching-for-secured-pages="false">
     <jsp-config/>
     <websockets/>
     </servlet-container>
     ```

## WAS Setting for Kerberos Kafka Authentication

There are 2 files which needs to set in the environment of WebSphere.

1. Set krb5 in System Env

   **For Krb5 Server > Server Type > Select the server > Java and Process Management > Process Definition > Java Virtual Machine > Generic JVM arguments** append the below mentioned argument in the Argument List

   -Djava.security.krb5.conf=<KRB File Location>\krb5.conf

2. jass entries

   Go to <WebSphere Home>\AppServer\profiles\<Profile>\properties\wsjaas.conf

   Append the following entries into the file

   **KafkaServer**

   ```
   { com.ibm.security.auth.module.Krb5LoginModule required useKeytab="<Zookeper Key Tab Path>
   \zkserver.keytab" storeKey=true principal= <Zookeper Principle> eg- "zookeeper/ip-10-10-10-10.ap-
   south-1.compute.internal.nonprod.hclpnp.com" credsType = both; }
   ```

   **KafkaClient**

   ```
   { com.ibm.security.auth.module.Krb5LoginModule required useKeytab="<Kafka Server Kay
   Tab Path>\kfkserver.keytab" principal= <Kafka Principle> eg - "kafka/ip-10-10-10-10.ap-
   south-1.compute.internal.nonprod.hclpnp.com" credsType = both debug=true; }
   ```

   Restart Application Server.

**Configuring Kafka Kerberose for Deliver Responses:**

Login to Unica application and navigate to below location:

1. **Location > Settings > configuration > HCL Unica > Journey > Kafka > Configurations**

   Set:

   ```
   KafkaBrokerURL:     <Priciple_Hostname>:9092
   CommunicationMechanism:     GSSAPI
   sasl.mechanism: GSSAPI
   sasl.jass.config.location:
    <JOURNEY_HOME>/WEB/properties/application.properties/kafka_server_Kerberose_jaas.conf
   java.security.krb5.conf.location: <JOURNEY_HOME>/WEB/properties/application.properties/krb5.conf
   ```

   .
2. Location: **Settings > configuration > HCL Unica > Deliver > Kafka > RCT**

   Set:

   ```
   Set
   KafkaBrokerURL:     <Priciple_Hostname>:9092
   CommunicationMechanism:     SASL_PLAINTEXT
   sasl.mechanism: GSSAPI
   ```

## Kafka Configuration for Configuring External resources in Journey (AssetPicker and Journey configuration)

Login to Unica application and navigate to below location:

**Location > Settings > configuration > HCL Unica > Journey > Integration > dataSource > <System_Configuration Template Name> > Kafka Configurations**

```
Bootstrap servers (comma separated list of hosts)          <Karberos_Kafka_Principle_Host:>7001
(e.g. ip-10-10-10-100.nonprod.hclpnp.com:7001){}
Security protocol             SASL_PLAINTEXT
SASL mechanism             KERBEROS
Kerberos - Configuration file path (e.g. /etc/krb5.conf, C:/Windows/krb5.ini)
 <JOURNEY_HOME>/WEB/properties/application.properties/krb5.conf
Kerberos - Keytab file path               <JOURNEY_HOME>/WEB/properties/kfkserver.keytab
Kerberos - Principal        kafka/<Karberos_Kafka_Principle_Host>@HCLPNP.COM
e.g. kafka/ip-10-10-10-100.nonprod.hclpnp.com@HCLPNP.COM
Kerberos - Service Name              kafka
```

# Chapter 15. Configure Web Application servers Tomcat for SSL

On every application server on which a Unica application is deployed, configure the web application server to use the certificates you have decided to employ.

See your web application server documentation for details on performing these procedures.

## Ensuring cookie security

Some cookies may not be properly secured in the client browser. Not securing cookies leaves the application vulnerable to man-in-the-middle and session hijacking attacks. To fix this issue, take the following precautions.

- Enforce the use of SSL at all times to reduce the risk of cookies being intercepted on the wire.
- In the web application server, set the `secure` and `httponly` flags on all cookies.
    - The `secure` flag tells the browser to send the cookie only over an HTTPS connection. You must enable SSL on all applications that communicate with each other if you set this flag.
    - The `httponly` flag prevents cookies from being accessed through a client side script.

### Setting the flags for SSL in Tomcat

To set the `secure` and `httponly` flags in Tomcat, perform the following changes in the `.xml` server of Tomcat.

**About this task**

```
<Connector port="7003" protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads="150" SSLEnabled="true" scheme="https" acceptCount="100" clientAuth="false"
disableUploadTimeout="true" enableLookups="false" secure="true" sslProtocol="TLS"
 keystoreFile="/opt/v12.1/v12.1.0.1.1/Campaign/SSL_NEW/PlatformClientIdentity.jks" keystorePass="password" >
  </Connector>
```

## Configure Unica Journey with SSL

To configure Unica Journey to use SSL, you must set some configuration properties. Use the procedures in this section that are appropriate for your installation of Unica Journey and the communications that you want to secure using SSL.

**About this task**

When you access your Unica installation over a secure connection, and when you set navigation properties for applications as described in the following procedures, you must use `https` and the secure port number in the URL. The default SSL port is`8443` for Tomcat.

Follow this procedure to configure Journey with SSL

1. Log in to Unica and click **Settings > Configuration**.
2. Set the value of the `Affinium | Journey | navigation` property to Unica Journey URL.

   For example: `https://host.domain:SSL_port/unica`

where:

- *host* is the name or IP address of the machine on which Unica Journey is installed
- *domain* is your company domain in which your Unica products are installed
- *SSL_Port* is the SSL port in the application server on which Unica Journey is deployed

Note `https` in the URL.

# Chapter 16. Mailchimp Configuration

Configuration of Mailchimp as external source with Journey.

**About this task**

Navigate to **Unica Platform > Settings > Configuration**

From Configuration categories page navigate to **Journey > Integration > dataSources**

**For Adding Journey datasource**

1. Click **systemconfigurationTemplates**
   **Result**
   The **systemConfigurationTemplates** page appears.
2. Provide the below information:
   - **New category name**: <Journey>
   - **systemIdentifier**: Journey
   - **userCredentials**: Default user
   - **defaultUserCredentials** : asm_Admin
   - **dataSourceNameForCredentials** : <JOURNEY_DS_1>
   - **AdditionalParameters** :
   - **event-publisher-service.kafka.topics**: CIFINTEGRATION
   - **event-publisher-service.kafka.topics.CIFINTEGRATION.value.format**: Json

   **Note:** For more information on the fields, see *Unica Content Integration 12.1.9 Installation and Configuration Guide*.

3. Click **Save**.
4. Expand the Journey node and click **httpGateway**
   **Result**
   The **Settings for 'httpGateway'** page appears.
5. Provide below information:

   baseUrl : http:<hostname>:<port>/journey
6. Click on **Kafka configuration** link and add below information

   Bootstrap servers (comma separated list of hosts) : <Kafkahost>:<port>

   e.g. <IP OR Hostname>:9092

## For Adding Mailchimp datasource

1. Click **systemconfigurationTemplates**
   **Result**

The **systemConfigurationTemplates** page appears.

2. Provide the below information:
    - **New category name**: <Mailchimp>
    - **systemIdentifier**: Mailchimp
    - **userCredentials**: Default user
    - **defaultUserCredentials** : asm_admin
    - **dataSourceNameForCredentials** : <Mailchimp_DS>

    **Note:** For more information on the fields, see *Unica Content Integration 12.1.9 Installation and Configuration Guide*.

3. Click **Save**.
4. Expand the Mailchimp node and Click **httpGateway**

    **Result**

    The **Settings for 'httpGateway'** page appears.
5. Provide below information

    baseUrl : https://<MailchimpHostname>/<Version>/

    User needs to add these datasources in Journey. User need to gather below information:

    a. Journey Data Source UserId and password. Navigate to **Journey application > Settings > Rest** either create a new Rest Integration or use existing rest integration. Copy clientid and Client Secret.
    b. Mailchimp data source userid and password. Login to Mailchimp application and Navigate to **Profile > Extra > API keys**. Get the api keys and User column name will be user

    In mailchimp a/c **add Webhook** url as below:

https://<AssetpickerHostname>:<Port>/asset-viewer/api/AssetPicker/webhook/Mailchimp/events/webhook_listener

Navigate to **Platform Settings > Users**

Click on **Required user** e.g. asm_Admin

Click on **Edit dataSources link** and add below data sources

**Mailchimp_DS - user**: <user> and password <API key>

**JOURNEY_DS_1 - user**: <clientid> and password <Client Secret>

**Navigate to Unica Platform > Settings > configuration > Unica Platform > Security > API management > Unica Content Integration**

Click **API configuration template** and add below information

- **New category name** : <Mailchimp>
- **API URI** : /webhook/Mailchimp/events/*
- **Require authentication for API access** - unselected

Click **Save**

After performing these steps Mailchimp will be added as external source in Journey.

**Note:**

- If the user configures an external source in Journey, keep the following properties as `False`. This will allow the user to get the data from external source as per significant field configured in Journey data deduplication settings.
  - `<rule-enabled>false</rule-enabled>\<JourneyEngine>\conf\data-validation-rules.xml`
  - In Platform, navigate to **Settings > Configuration > Journey > Journey Configuration** and set the value of the **Validation_On_Journey_Records** property to `False`.
- For configuring external source user must install Unica Content Integration component while installing or upgrading Unica Platform as a prerequisite.

# Chapter 17. Settings

Use the settings menu to manage the Journey integrations like Email connectors, SMS connectors, CRM connections, and REST integrations.

## Setting a default email connection

If you have multiple connectors to Unica Link for sending an email, you can set the default email connection in the **Settings** menu.

**About this task**

To set a default email connection, complete the following steps:

1. Select  **> Link > Email**.
   **Result**
   The **Email** page appears.
2. From the **Available Connections** list, select a connection.
   The available connection includes Mandril, Mailchimp, etc.
3. Click **Save**.
   You can also deselect an existing connection and click **Save**. This ensures that no default connection is set.

## Setting a default SMS connection

If you have multiple connectors to Unica Link for sending an SMS, you can set the default SMS connection in the **Settings** menu.

**About this task**

To set a default SMS connection, complete the following steps:

1. Select  **> Link > SMS**.
   **Result**
   The **SMS** page appears.
2. From the **Available Connections** list, select a connection.

   **Note:**

   Phone number formats should be mentioned as per the specification of the delivery channel. Journey will send the phone number in the same format to delivery channel. For example, in reference Twilio connection phone number format supported with Journey is as follows:

- *<plus sign><country-code><10-digit phone number>* - `+15403241212`.
- *<plus sign> <country-code <(area-code)> <three-digit number><four-digit number>* - `+1 (540) 324 1212`.
- *<plus sign>-<country-code>-<area-code>-<three-digit number>-<four-digit number>* - `+1-540-324-1212`.
- *<plus sign> <country-code>-<area-code>-<three-digit number>-<four-digit number>* - `+1 540-324-1212`.

Whatever format of phone number you provide, Unica Journey will save the number in the following format: *<plus sign><country-code><10-digit phone number>*. For example, if you provide phone number as `+1 540-324-1212`, Unica Journey stores the phone number as `+15403241212`.

If you select Twilio as the default SMS connection, it will accept phone numbers only in the following format: *<plus sign><country-code><10-digit phone number>*. For example, `+15403241212`.

3. Click **Save**.

# Setting a default CRM connection

If you have multiple CRM connections, you can set the default CRM connection in the **Settings** menu.

**About this task**

To set a default CRM connection, complete the following steps:

1. Select  **> Link > CRM**.
   **Result**
   The **CRM** page appears.
2. From the **Available Connections** list, select a connection.
3. Click **Save**.

# Setting a default ADTECH connection

If you have multiple ADTECH connections, you can set the default ADTECH connection in the **Settings** menu.

**About this task**

To set a default ADTECH connection, complete the following steps:

1. Select  **> Link > ADTECH**
   **Result**
   **ADTECH** page appears
2. From the **Available Connections** list, select a connection.
3. Click **Save**.

# Setting a default Database connection

If you have multiple Database connections, you can set the default database connection in the **Settings** menu.

**About this task**

To set a default Database connection, complete the following steps:

1. Select  **> Link > Database**

   **Result**

   **Database** page appears
2. From the **Available Connections** list, select a connection.
3. Click **Save**.

# Setting a default PUSH connection

If you have multiple connectors to Unica Link for sending PUSH, you can set the default email connection in the **Settings** menu.

**About this task**

To set a default PUSH connection, complete the following steps:

1. Select  **> Link > Push**.

   **Result**

   The **Push** page appears.
2. From the **Available Connections** list, select a connection.

   The available connection includes `Batch_Android` and `Batch_iOS`. For more information, see Manage connections on page 72.
3. Click **Save**.

   You can also deselect an existing connection and click **Save**. This ensures that no default connection is set.

# Manage connections

You can manage Unica Link connections from this menu.

**About this task**

You can create a connection with Unica Link connectors like Mailchimp, Mandrill, Salesforce, and Twilio. You can view all existing connections in the **Existing Connections** (*n*) panel, where *n* is the number of connections.

1. To create a Mailchimp connection, complete the following steps:

   a. Select  **> Link > Manage Connections > Create New**.

   **Result**

The **Create New Connection** page appears.

b. Provide values for the following fields:
   - **Name** - Mandatory
   - **Description** - Optional

c. Click **Next**.

d. From the **Choose Connection** panel, select **Mailchimp**.

e. In the **Connection Properties** panel, provide values for the following mandatory fields:

**Note:** To know about the fields and the values to be put, see *Unica Link Mailchimp Connector User Guide*.

   - **Base URL**
   - **User ID**
   - **API Key**
   - **Activity Fetch Frequency**
   - **Activity Fetch Units**

f. Click **Test** to the test the connection. If the provided values are correct, you will see a success message. If the provided values are incorrect, you will see an error message.

g. To save the connection, click **Save**.
**Result**
The new connection is successfully saved and it appears in the **Existing Connections** panel.

2. To create a Mandril connection, complete the following steps:

a. Select  **> Link > Manage Connections > Create New**.
**Result**
The **Create New Connection** page appears.

b. Provide values for the following fields:
   - **Name** - Mandatory
   - **Description** - Optional

c. Click **Next**.

d. From the **Choose Connection** panel, select **Mandrill**.

e. In the **Connection Properties** panel, provide values for the following mandatory fields:

**Note:** To know about the fields and the values to be put, see *Unica Link Mandrill User Guide*.

- **API Key**
- **Activity Fetch Frequency**
- **Activity Fetch Units**

f. Click **Test** to the test the connection. If the provided values are correct, you will see a success message. If the provided values are incorrect, you will see an error message.

g. To save the connection, click **Save**.
**Result**
The new connection is successfully saved and it appears in the **Existing Connections** panel.

3. To create a Salesforce connection, complete the following steps:

a. Select  **> Link > Manage Connections > Create New**.
**Result**
The **Create New Connection** page appears.

b. Provide values for the following fields:
- **Name** - Mandatory
- **Description** - Optional

c. Click **Next**.

d. From the **Choose Connection** panel, select **Salesforce**.

e. In the **Connection Properties** panel, provide values for the following mandatory fields:

**Note:** To know about the fields and the values to be put, see *Unica Link Salesforce User Guide*.

- **Instance URL**
- **Access Token**
- **Version**

f. Click **Test** to the test the connection. If the provided values are correct, you will see a success message. If the provided values are incorrect, you will see an error message.

g. To save the connection, click **Save**.
**Result**
The new connection is successfully saved and it appears in the **Existing Connections** panel.

4. To create a Twilio connection, complete the following steps:

a. Select  **> Link > Manage Connections > Create New**.

**Result**

The **Create New Connection** page appears.

b. Provide values for the following fields:

- **Name** - Mandatory
- **Description** - Optional

c. Click **Next**.

d. From the **Choose Connection** panel, select **Twilio**.

e. In the **Connection Properties** panel, provide values for the following mandatory fields:

**Note:** To know about the fields and the values to be put, see *Unica Link Twilio User Guide*.

- **Base URL**
- **Account SID**
- **Auth Token**
- **From Number**
- **Retry Interval**
- **Retry Attempts**

f. Click **Test** to the test the connection. If the provided values are correct, you will see a success message. If the provided values are incorrect, you will see an error message.

g. To save the connection, click **Save**.

**Result**

The new connection is successfully saved and it appears in the **Existing Connections** panel.

5. To create a Batch connection for Android or iOS, complete the following steps:

a. Select  **> Link > Manage Connections > Create New**.

**Result**

The **Create New Connection** page appears.

b. Provide values for the following fields:

- **Name** - Mandatory. For example, `Batch_Android` / `Batch_iOS`
- **Description** - Optional

c. Click **Next**.

d. From the **Choose Connection** panel, select **Batch**.

e. In the **Connection Properties** panel, provide values for the following mandatory fields:
   - **BAse URL**
   - **Live Key**
   - **Rest Key**
   - **Activity Fetch Frequency**
   - **Activity Fetch Units**

f. Click **Test** to the test the connection. If the provided values are correct, you will see a success message. If the provided values are incorrect, you will see an error message.

g. To save the connection, click **Save**.
   **Result**
   The new connection is successfully saved and it appears in the **Existing Connections** panel.

**Note:** If you want Batch connection for both Android and iOS, you must create a Batch connection for each.

# REST Integration

REST keys are used for third-party login to the application. You can generate key-value pair and using the key value pair, you can login to Journey using third-party applications.

## Creating a new REST integration

To create a new REST integration key pair, complete the following steps:

1. Select  **> REST**.
   **Result**
   The **REST** page appears.
2. Click **+ REST Integration**.
   **Result**
   The **New REST Integration** page appears.
3. Provide values for the following fields:
   - **App Name** - Mandatory.
   - **Description** - Optional.
4. Click **Generate Keys**.
   **Result**
   The system generates a **ClientID** and **ClientSecret**.
5. Use the toggle bar to change the **Status** to `Active` or `Inactive`. By default, the **Status** is `Active`.
6. To save the REST integration, click **Save**.
   To send audience data to Journey, follow the details mentioned on the REST Entry Source used for configuring the REST end point. Use the **ClientID** and **ClientSecret**, which you received when executing Step (4), for configuring the REST end point on Entry Source.

7. Generate the authentication token using the URL example: "http://comp-4946-.nonprod.hclpnp.com:80/journey/api/thirdpartylogin". Use this authentication token and entry source code while sending the data.

   **Note:** Entry source code is mandatory to send data on REST entry source.

8. 8. REST API end point (as available on REST entry source creation page) example - http://comp-4946-1.nonprod.hclpnp.com:80/journey/api/entrysources/rest/data
9. Sample format of JSON that /journey/api/entrysources/rest/data takes as input is as below -

   {

   "entrySourceCode": "ES-00000125",

   "data": [

   { "Email": "pooja_sharma@hcl.com", "FirstName": "Pooja", "LastName": "Sharma", "Age": 30, "Address": 125, "CreatedDate": "15 09 22", "PHONE_NUMBER": "+919623444160", "DeviceID":"ac79649c-ca1b-4c3f-99ce-56d5ad69fbba" }

   ]

   }

   **Note:** The json fields should be changed as per the data definition which is associated with the journey.

10. Journey should be in published state before pushing the data on REST entry source.

## Viewing REST integration list

Unica Journey maintains a list of REST integrations created.

**About this task**

To view a list of REST integrations, complete the following steps:

1. Select  **> REST**.
   **Result**
   The **REST** page appears.
2. Perform any one of the following operations:
   a. To view the list of REST integrations in ascending order or descending order on the Name field, click **Name**.
   b. To view the list of REST integrations in ascending order or descending order on the Description field, click **Description**.

## Modifying an existing REST integration

You can only modify the description and the status of an existing REST integration.

**About this task**

To modify an existing REST integration, complete the following steps:

1. Select  **> REST**.
   **Result**
   The **REST** page appears.
2. To modify a rest integration, you can either:
   **Choose from:**
   - select the required REST integration from the list
   - select  **>** 

   **Result**
   The **Update REST Integration** page appears.
3. You can update only the following fields:
   **Choose from:**
   - **Description**
   - **Status**
4. To save the modifications, click **Save**.

## Deleting REST integrations

You can only delete inactive REST integrations that are no longer used or needed.

**Before you begin**

To change the status of a REST integration entry, see Modifying an existing REST integration on page 77.

**About this task**

To remove existing inactive REST integrations, complete the following steps:

1. Select  **> REST**.
   **Result**
   The **REST** page appears.
2. Perform either of the following steps:
   **Choose from:**
   - To delete a REST integration, select  **>**  succeeding the REST integration in the list.
   - To delete multiple REST integrations, select the checkboxes preceding the REST integrations, in the list, that you want to delete and click **Delete**.
3. A confirmation box appears. To proceed with the deletion, click **Ok**.

# Use of custom key store and trust store

In case of configuring a rest touchpoint for secure Rest API i.e. the API with url that starts with [https://|https:], a need might arise to configure ssl key-certificates depending on the implementation of third party rest API.

Usually, ssl certificates can be configured in java installation folder or can be configured in entirely different folder. In later case, system administrator needs to make following settings in the `application.properties` of journey engine i.e. in the file `<installation location of journey application>\Engine\application.properties`

- **ssl.restclient.custom.store**

  true: If the trust store and keystore of journey application are present at some custom location (or folder) and the certificates of journey and 3rd party system are to be imported in that location instead of in the installation directory of java runtime.

  false: If the certificates are to be imported in the installation directory of java runtime.

- **ssl.restclient.truststore.defaultalgorithm**

  The standard name of the requested trust management algorithm.

  Refer https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html to get more information on the standard algorithm names.

  Specimen value SunX509

- **ssl.restclient.keystore.defaultalgorithm**

  The standard name of the requested algorithm. Refer https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html to get more information on the standard algorithm names.

  Specimen value SunX509

- **ssl.restclient.truststore.type**

  Represents the type of trust store (defaults to JKS)

  A trust store is a repository or storage location for digital certificates, specifically public key certificates, that are used to establish trust in secure communications.

  Trust stores store public key certificates of trusted entities, such as certificate authorities (CAs) and trusted servers.

  They are used to verify the authenticity of remote entities (such as websites or servers) during the SSL/TLS handshake process.

- **ssl.restclient.truststore.location**

  A folder or a directory on the file system which represents the truststore of journey application.

- **ssl.restclient.truststore.password**

Usually, the truststores are password protected. This parameter represents the trust store password.

- **ssl.restclient.keystore.type**

  Represents the type of keystore (defaults to JKS)

  A keystore represents a storage facility for cryptographickeys and certificates. Keystores often store private keys, which are essential for encrypting and decrypting data, creating digital signatures, and establishing secure communications.

- **ssl.restclient.keystore.location**

  A folder or a directory on the file system which represents the keystore of journey application.

- **ssl.restclient.keystore.password**

  Usually, the keystores are password protected. This parameter represents the key store password.

- **ssl.restclient.key.password**

  Usually, the keys inside a keystore are also password protected. This parameter represents the key password.

- **ssl.restclient.protocols**

  comma separated list of supported cryptographic protocols which are designed to provide secure communication over a network

  Recommended values TLSv1.2

- **ssl.restclient.truststore.password.encrypted**

  When set to true or missing, specify the encrypted value of the password in ssl.restclient.truststore.password

  When set to false, specify plain text value of the password in ssl.restclient.truststore.password

  **Note:** If the password is going to be blank, then set this flag to false.

- **ssl.restclient.keystore.password.encrypted**

  When set to true or missing, specify the encrypted value of the password in ssl.restclient.keystore.password

  When set to false, specify plain text value of the password in ssl.restclient.keystore.password

  **Note:** If the password is going to be blank, then set this flag to false.

- **ssl.restclient.key.password.encrypted**

  When set to true or missing, specify the encrypted value of the password in ssl.restclient.key.password

  When set to false, specify plain text value of the password in ssl.restclient.key.password

**Note:** If the password is going to be blank, then set this flag to false.

To encrypt the password, Please execute following utility which is present in the <Journey install location>/tools folder `JourneyEncryptionUtility.bat(or .sh)`

Usage : JourneyEncryptionUtility.bat(or .sh) <plaintext password>

Example

If the plain text password is 'abcd', then JourneyEncryptionUtility.bat(or.sh) abcd

It will output something similar to the one given below

Entered String is : abcd

Encrypted String is : MIuFcm7mkspvIMEx7XywAA==

Set the encrypted value MIuFcm7mkspvIMEx7XywAA== in the application.properties for the parameter in question. If you are using different password values for all the above-mentioned three parameters, then run this utility thrice and use the values accordingly.

## Journey Proxy Integration

Proxy server has been integrated into journey web and Engine Projects, this gives user an upper hand in adding security and keeping application server behind proxy servers. Proxy server will interact with Deliver, Link and Platform servers.

Journey Web – Communicates with Deliver, Link and Platform server for fetching configuration details and while integrating Email/SMS/AdTech Point in Journey.

Journey Engine – Uses proxy to communicate with Deliver/Link Server for submitted Email/SMS/Adtech details to end servers.

Proxy Supported by Journey Web

1. SOCKS
2. HTTP
3. HTTPS

Proxy Supported by Journey Engine

1. HTTP

**Note:** SOCKS and HTTPS proxy is not supported by SOAP (Apache Axis2) used by Engine to communicate with Deliver.

Property to be configured for Engine in Engine application.properties file

- journey.proxy.type=NONE
- spring.proxy.host=[IP]
- spring.proxy.port=[PORT]
- spring.proxy.username=[username]
- spring.proxy.password=[password]

Property to be configured for Web in Web application.properties file

- journey.proxy.type=NONE
- spring.proxy.host=[IP]
- spring.proxy.port=[PORT]
- spring.proxy.username=[USERNAME]
- spring.proxy.password=[PASSWORD]
- server.use-forward-headers=true

**Note:** Default value of property journey.proxy.type is NONE, when set to NONE proxy is disabled.

Journey Engine Connection Pool settings

- journey.datasource.maxpool.size=[MAX_POOL_SIZE] – set size of Db cnnection Pool
- journey.datasource.minIdle.size=[MIN_IDLE_SIZE] – sets size of minimum idle connections

# Developer Tools

Displays the list of developer tools.

## API Documentation

User can find the list of REST APIs for Journey.