# EX Engage Connector Deployment Guide

EX Engage Connector Current

10/25/2024

# Table of Contents

# EX Engage Connector Deployment Guide

This document describes the necessary steps to deploy Genesys EX Engage Connector (EXEC). Genesys EX Engage Connector allows using Genesys Cloud EX with an existing Genesys Engage contact center.

**Important**

EX Engage Connector is being released to pre-approved customers as part of the Early Adopter Program. This means that both the product and the documentation are still under development. As a result, documentation sections might require revision as the product develops. We advise that you use this documentation with care. Before you make changes that could affect the success of your deployment, verify them with your Genesys representatives.

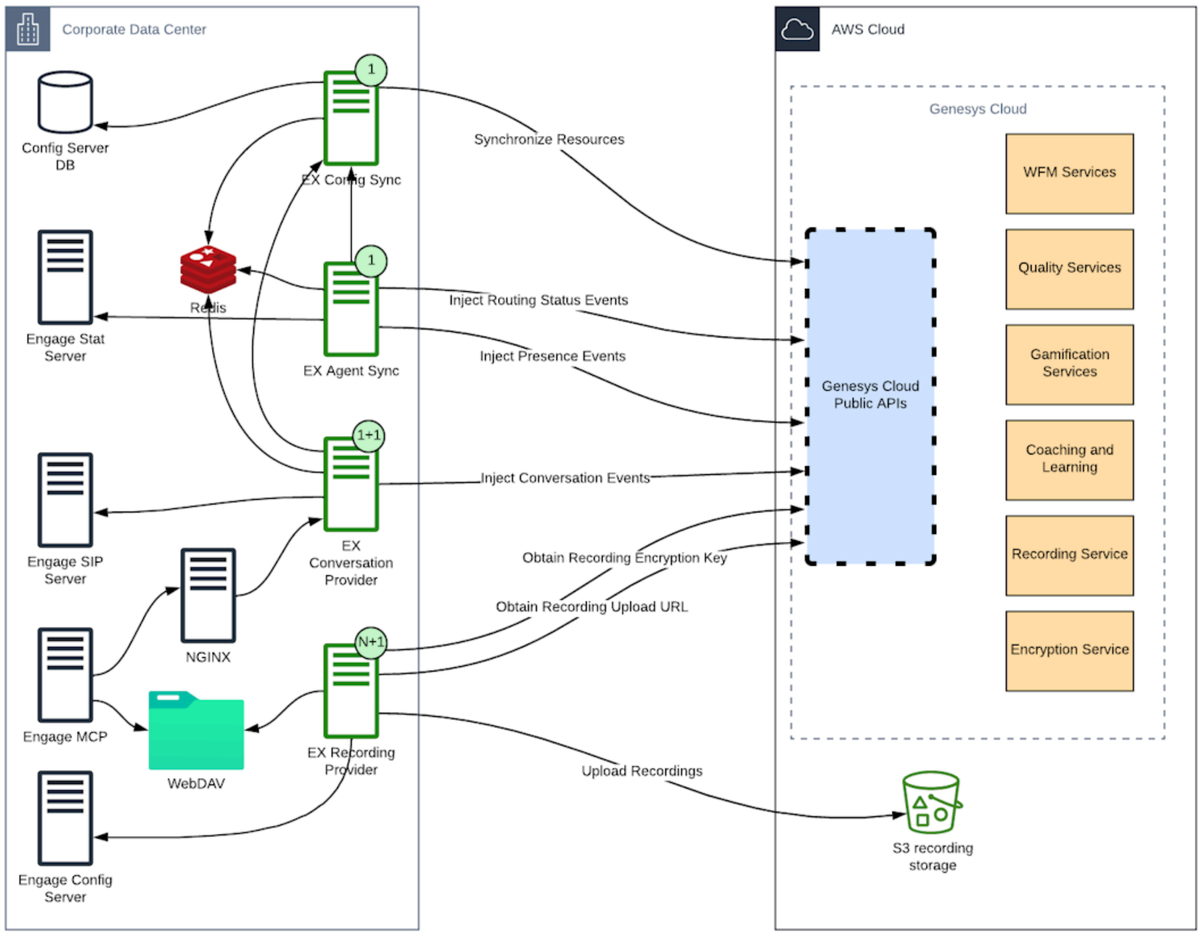
# Overview

Genesys EX Engage Connector allows integrating Genesys Cloud services such as WFM, Gamification, Coaching, Recording Management, Quality Monitoring, Speech and Text Analytics, and others to an existing Genesys Engage contact center through the Genesys Cloud EX Organization.

Engage Connector supports WEM features through REST APIs by the following synchronization options:

- Synchronizing configuration data
- Synchronizing agent statuses and status updates
- Synchronizing interaction data
- Synchronizing recording and metadata

# Architecture



# EX Engage Connector Configuration Sync Service (EXCS)

EXCS is a component responsible for the synchronizing resources with Genesys Cloud EX.

## Configuration Object Mapping Table

Refer to the Resource Synchronization chapter of the *Genesys Cloud EX Integration Guide* for a list of the configuration objects, which should be created in the EX Org to enable event injection. Those configuration objects are listed in the **EX Org Object** column and the **Engage Object** column contains a configuration object of the Engage Contact Center, which is used to create a corresponding object in the EX Org.

| EX Org Object | Engage Object |
|---|---|
| User | Persons |
| Skill | Skill |
| Language | Engage Contact Center doesn't have Language as a configuration object. Languages are assigned to Persons as Skills. |
| Queue | Virtual Queue |
| Wrap-up Code | Business Attribute |
| Source | EXEC creates a new Source object in the EX Org. This data is not mapped. |
| Division | All EX Org Users mapped from Engage are assigned to Home Division |
| Secondary presence status definition | Not Supported |

The Configuration Object Synchronization follows these rules:

- If a new object is created in Engage, a corresponding object is created in the EX Org.
- If a mapped object is deleted in Engage, then the behavior of EXCS depends on the object type:
    - Virtual Queues: The corresponding EX Org object will not be deleted, but all members in that object will be removed from the queue members list in near-real-time.
    - Virtual Queue Members: The corresponding EX Org objects will be removed from the queue members list in near-real-time.
    - Persons: The corresponding EX Org object will not be deleted, but will be moved into an *inactive* state. If the same Person object is later re-created in Engage, then the corresponding EX Org object will be moved back into *active* state (but a different *externalId* value was assigned because the re-created Person object will be assigned a different DBID value in Engage).

  - Skills: The corresponding EX Org object will not be deleted.
  - Wrap-up Codes: The corresponding EX Org object will not be deleted.

- If a mapped object is deleted in EX Org, then this object is not re-created in the EX Org. The objects, deleted in EX Org, may be re-mapped from the Engage configuration only after the clearing of the Redis cache and restarting of the Config-Sync service.

- If a mapped object is modified in Engage, then the changes are applied to the corresponding EX Org object.

- If the mapped parameters of a mapped object (for example, a new skill is added to an agent) are modified in the EX Org, then they are updated in the EX Org to match the Engage configuration.

- If a new parameter not controlled by Engage is added to the EX Org object, then it is not changed by the EXCS

## Deletion of the ACD queues and queue members in the EX Org

If an Engage DN mapped to an ACD queue in the EX Org is deleted from the Engage configuration, then EXCS does not delete it from the EX Org.

If an Engage Person mapped to the EX Org as a member of one or multiple ACD queues is deleted from the Engage configuration, then EXCS deletes the corresponding queue member from all EX ACD queues. EXCS doesn't delete a corresponding User from the EX Org.

## Case-Sensitivity of Object Names

In Engage, the names of Persons, DNs, Skills, and Business Attributes (BA) are case-sensitive. For example, it is possible for two distinct queues with the names *queue1* and *Queue1* to exist. In GC, the names of the Users, Queues, Skills, and Wrap-up Codes (which are derived from the corresponding Engage objects) are not case-sensitive. If a queue with the name queue1 exists, then the attempt to create a new queue with the name Queue1 would fail.

Customers must resolve these naming problems, if they exist, before running EXCS for the first time. This can be done by deleting the conflicting object and re-creating it with a non-conflicting name.

If these naming conflicts are not resolved prior to starting EXCS, or if they are introduced at a later point in time, the result will be that more than one Engage object will be mapped to a single GC object.

## Selecting Engage Objects for Mapping to EX Org using Folder Filtering

Folder filtering is a process of selecting Engage configuration objects to be mapped to the EX Org. It allows Engage customers to enable only part of their contact center to use Genesys Cloud (GC) services. For example, customers can pilot GC services for one of their lines of business (LOB) and add more LOBs later.

Engage configuration folders are used to identify the configuration objects, which should be mapped to the EX Org. Folder filtering is applicable to the following configuration objects:

- Persons
- Skills
- DNs
- Agent Groups

For effective folder filtering, the folder to be synced should be selected for Persons, Skills, DNs, and Agent Groups Engage objects.

Folder filtering cannot be used for the Business Attributes.

To execute this approach two EXCS environment variables should be configured:

- `SYNC_CUSTOMER_OBJECTS_ONLY` = *true*
- `SYNC_CUSTOMER_FOLDER_NAME` = *<FOLDER_NAME_1>*, *<FOLDER_NAME_2>*...

The `SYNC_CUSTOMER_FOLDER_NAME` environment variable contains a comma-separated list of the CME folder names. EXCS searches those folders in all four configuration units listed above (Persons, Skills, DNs, and Agent Groups) and syncs configuration objects found in those folders to GC. If a listed folder has subfolders, then all subfolders are synchronized to the GC too.

Genesys recommends including existing CME folders into the list to avoid moving configuration objects around, which can cause an unnecessary load on Configuration Server. Also, existing CME folders usually represent current business infrastructure, and it can be convenient to use them to mark a certain LOB for syncing to GC.

If customers want to create a pilot agent group, which does not correspond to any existing CME folder, then a new folder with an unique name (for example, *EX_EVOLUTION*) should be created under the configuration unit of each mapped configuration object. For example, to select queues, the *EX_EVOLUTION* folder should be created under the DNs folder. The folder to be filtered can be present in any level of the configuration object and not necessary to be at Root level.

It is advised to use a maintenance window to move the Engage configuration objects to new folders in CME as it can create a substantial load on the Configuration Server (CS) and CS Proxies.

## Synchronization of Wrap-up Codes

GC Wrap-up Codes are derived from the Engage Business Attributes (BA). EXCS selects the BAs to be mapped based on the following requirements:

- **Type** = *Interaction Operational Attributes*, and
- Either of:
  - **Name** = *DispositionCode*, or
  - **Annex** has the following set (case-sensitive): Annex / htcc / contains=dispositions

All values inside all BAs satisfying the above requirements are synchronized to GC. If multiple BAs have Values with the same name, only one Wrap-Up Code will be created for the name. For example, if both BAs, *Tech Support Disposition Codes* and *Billing Disputes Disposition Codes* have a value named 'Resolved', only one Wrap-Up Code called 'Resolved' is created in GC. All mapped Wrap-up codes are assigned to all ACD queues in the EX Org.

Folder Filtering cannot be used for BAs.

## Synchronization of Skills

GC Skills are derived from the Engage Skills, which are selected for mapping based on the Folder Filtering rules.

## Synchronization of Queues

GC ACD queues are derived from the Engage DNs, which are selected for mapping based on the Folder Filtering rules. The following types of Engage DNs are mapped to the GC queues:

- Virtual Queue

GC ACD queues are configured with members. Calls arriving to an ACD queue can be distributed to one of its members. In Engage an association between users and queues can also be configured but most often it is identified at run time. Additional configuration should be applied on the Engage side to explicitly link agents to queues. EXCS uses this configuration to create members of the GC ACD queues.

Customers should identify Engage Virtual Queues as important for the GC WFM scheduling and forecasting. For each selected Engage VQ, the customer must identify an existing or create a Virtual Agent Group (VAG) that will be used for Queue-User mapping. The following configuration should be applied to the Annex of these VAG objects to link them to their corresponding queues:

- `Annex / hybrid / WFM_queue_number = <QUEUE_DN_NAME>`

Changes made to the Engage VAG membership are synchronized to membership of the corresponding GC ACD queues.

## Queues

If an Engage Person is a member of one or several VAGs, which are linked to a VQ using the hybrid/WFM_queue_number parameter, then a GC User derived from the Engage Person is added as a member to the corresponding ACD queue in GC. See more information in the Synchronization of Queues section above.

## Mapping Queue Parameters

EXCS uses the rules below to populate parameters of the ACD queues created in the EX Org.

- **Name**: <ENGAGE_QUEUE_DN_NAME>__<ENGAGE_SWITCH_NAME>__Engage-external
  - VQ DNs created under different Engage switches may have the same name. GC EX ACD queues must have unique names. To guarantee the queue name uniqueness EXCS adds the name of the switch as a suffix to the name of the ACD queue.

    - Example: if an Engage VQ DN called *VQ_Sales_Leads* configured under the switch *VQ-Switch* is mapped to the EX Org, then the name of the corresponding ACD queue is set to `VQ_Sales_Leads__VQ-Switch__Engage-external`.

- **Description**: Engage Reference: *<TENANT_NAME-SWITCH_NAME-DN>; Last modified: <hh:mm:ss MM/DD/YYYY>;*
    - Example: The following Description line is compiled for the DN VQ_Sales_Leads configured under the switch SIP_Switch in tenant tenant001:
        - Engage Reference: `tenant001-SIPSwitch-VQ_Sales_Leads; Last modified: 13:05:25 04/19/2023;`

Important

- Because there is no TimeZone info available for the object last modification time in the Engage Config DB, the *Last modified* date/time in GC Queue Description field is represented in UTC format.
- If the corresponding Engage DN object was not yet modified (there are no related records in the Engage Config DB history table), then the *Last modified* info is not displayed in GC Queue Description field.
- If TENANT_NAME environment variable is not available, value of the environment variable TENANT_ID will be used. If both TENANT_NAME and TENANT_ID is not set, the default value *default_tenant* will be used.

- **Peer Id**: <DN_DBID>

This field contains a DBID of a DN in the Engage Config DB

## Synchronization of Users

GC Users are derived from the Engage Persons, which are selected for mapping based on the Folder Filtering rules.

### Person Details

| EX Org User Parameter | Engage Person Parameter | Comment |
|---|---|---|
| active: Indicates whether the user's administrative status is active. | State Enabled | |
| userName: The user's Genesys Cloud email address. Must be unique. | User Name | GC requires a userName to be a valid email address. EXCS is checking the Persons configuration fields in the following order to find a valid email address, which should be used as a GC User userName: |

| EX Org User Parameter | Engage Person Parameter | Comment |
|---|---|---|
| | | • User Name<br>• Employee ID<br>• E-Mail<br><br>If neither of those fields contains a valid email address, then a Person is not synchronized to the EX Org and corresponding error message is printed in the EXCS log. |
| displayName: The display name of the user. | First Last, or User Name | if both Person's first and last names aren't present, User Name is used |
| emails: Defines a SCIM email address. | E-Mail | |
| externalId: The external ID of the user. Set by the provisioning client. | DBID__username | |

## Roles

When EXCS synchronizes an Engage Person to GC for the first time, EXCS determines whether the Person is a non-agent, an agent, or a supervisor:

- A **non-agent** is a Person who does not have the "Is Agent" flag checked. In a Engage environment, this could be anything including from administrative user to business analyst.
- An **agent** is a Person who has the "Is Agent" flag checked.
- A **supervisor** is a Person who has the "Is Agent" flag checked, and has "Supervisor" as one of the HTCC roles in the annex (i.e., "htcc/roles" has "Supervisor" as one of the roles).

The following roles are assigned to a new GC User created by the EXCS:

- Non-agent Roles: Employee
- Agent Roles: Employee, User
- Supervisor Roles: Employee, User, Supervisor

Default roles can be changed using the following configuration parameters under the "hybrid_integration" transaction:

- Non-agents: nonagent_role_names
- Agents: agent_role_names
- Supervisors: supervisor_role_names

If some of the specified GC Role(s) do not exist, the User will still be created, and all the existing Roles will be assigned.

After the initial creation of the GC User, EXCS does not change any of the assigned GC Role(s) of the User, with the following exceptions:

- If EXCS has detected a Person has transitioned from agent to supervisor, all the GC Roles in supervisor_role_names are added to the User. (None of the existing GC Roles are removed from the User.)
- If EXCS has detected a Person has transitioned from supervisor to agent, all GC Roles in agent_role_names are added to the User. (None of the existing GC Roles are removed from the User.)

### Queues

If an Engage Person is a member of one or several AGs/VAGs, which are linked to an ACD Queue or a VQ using hybrid/WFM_queue_number parameter, then a GC User derived from the Engage Person is added as a member to corresponding ACD queue in GC. See more information in the 'Synchronization of Queues' section above.

### ACD skills

If an Engage Person has one or several skills, and those skills are configured to be mapped to GC, then a GC User derived from this Engage Person has corresponding ACD Skills assigned in GC. See more information in the 'Synchronization of Queues' section above.

Engage Person's skill level range is not limited. Proficiency level range of a GC User is from 0 to 5. EXCS translates the Engage skill level to the GC Proficiency level using the following rules:

1. For a given skill find MIN and MAX values across all configured persons.
2. If MIN or MAX value of a skill level is outside of the range [skill_level_min_value, skill_level_max_value], replace it with the corresponding range boundary value.
3. Proficiency Level = 5 * Skill Level / (MAX-MIN) - the result is rounded to one decimal place.

### Languages

EXCS doesn't configure a Language for an EX User. If a Language is required for an EX User to enable some EX functionality such as WFM grouping, then it can be configured for the Users mapped from Engage using GC UI. EXCS ignores User's properties, which are not included in its synchronization scope. Hence, EXCS won't change or remove the User's Language.

## Extension DNs

EXCS doesn't map the Extension DNs to the Genesys Cloud but it reads those DNs and stores them in Redis. This data is required by the EX Conversation Provider (EXCP) to interpret the Engage events properly while injecting them into the GC. There can be a large number of Extension DNs configured in Engage contact centers. It is recommended to minimize the number of Extension DNs read by the EXCS. For example, if only a subset of contact center agents are selected for the EX integration, then Folder Filtering can be used to filter only those Extension DNs, which are used by agents selected for the EX integration.

# EX Engage Connector Agent State Sync Service (EXAS)

EX Engage Connector Agent State Sync Service (EXAS) syncs the Engage contact center agents states to the Genesys Cloud (GC) EX Org in near-real-time. EXAS opens a connection to the Engage Stat Server, which monitors all agents synchronized to the EX Org by the EXEC Config Sync Service (EXCS). EXAS opens two statistics for each agent included into the EX-sync scope: CurrentState and CurrentStateReason (Object=Agent, Subject=DNStatus and DNAction correspondingly). EXAS receives real-time updates from the Stat Server, translates them into the Genesys Cloud format, and injects them into the EX Org

Important

- EXAS may fail to sync Engage agent state events to the GC in some failure scenarios. It may happen if EXAS is temporarily down or it cannot connect to the Engage Stat Server.

## Mapping Agent States

A Genesys Cloud User state contains two components:

- Presence
- Routing Status

**Presence** describes user's status when a user is not involved in the contact center activities. As soon as a user starts a work shift and joins an ACD queue, Presence is set to **On Queue**. **Routing Status** is used to indicate the agent state changes during the work shift when agent works on contact center interaction.

The following information obtained from the Engage contact center is used to define the Genesys Cloud User state:

- Agent State
- Call Type (in case if agent is processing a call)
- Reason Code

The table below explains mapping of Engage Agent states into the states of EX Users.

| Engage Agent State | Call Type | EX User Presence | EX User Routing Status |
|---|---|---|---|
| Logged Off | No Call | Offline | Off Queue |
| Not Ready (No reason or | No Call | Busy | Off Queue |

| Engage Agent State | Call Type | EX User Presence | EX User Routing Status |
|---|---|---|---|
| not-mapped reason) | | | |
| Not Ready (Reason: Break) | No Call | Break | Off Queue |
| Not Ready (Reason: Meal) | No Call | Meal | Off Queue |
| Not Ready (Reason: Meeting) | No Call | Meeting | Off Queue |
| Not Ready (Reason: Training) | No Call | Training | Off Queue |
| Not Ready (No reason or Any reason) | Internal | Busy | Communicating |
| Not Ready (No reason or Any reason) | Any type except Internal | Busy | Interacting |
| Ready | No Call | On Queue | Idle |
| Ready | Internal | On Queue | Communicating |
| Ready | Any type except Internal | On Queue | Interacting |
| Not Ready - ACW | No Call | Same as original state ( * ) | Interacting |
| * ACW request doesn't change the value of User Presence. | | | |

Presence of EX Users mapped from Engage is never set to **Available**. If a logged in Engage agent stays in a *Not Ready* state and doesn't participate in contact center activities, the state of a corresponding EX User is set to *Busy/Off Queue*. Also, a Routing Status of the mapped EX Users is set to *Communicating* only if a corresponding Engage agent is on an internal call, which is a call between a contact center extensions not-related to any interaction with a customer. All other call types trigger the *Interacting Routing Status* for the EX Org User.

If Engage customers want to use standard GC values of User Presence such as *Break*, *Meal*, *Meeting*, and *Training*, then those Presence values should be configured as Reasons on the Engage side. In this case the Engage reasons with those names are mapped to GC as explained in the table above. Any other Engage reason are not mapped to GC even if custom presence values with similar names are configured in GC. Custom Engage reason code is mapped to a *Busy GC User* presence.

# EX Engage Connector Conversation Provider Service (EXCP)

EXCP is a component responsible for the conversation injection into the GC EX Org (See Conversation Injection section of the *Genesys Cloud EX Integration Guide* for more information.

## EXCP General Operating Principles

EXCP connects to the default port of an Engage SIP Server and subscribes for a subset of TLib and call monitoring events. Those events are used to obtain information about interactions in the Engage Contact Center. EXCP translates Engage interaction events into the GC events and injects those events into the EX Org using EX REST API. EXCP uses the REST API of the EXCS to obtain the mapping information to translate the Engage object IDs into the GC equivalents.

Two instances of the EXCP should connect to an HA pair of the Engage SIP Servers to support high availability. Both instances connected to the same SIP Server HA pair are subscribed for the same set of events. If one EXCP instance is temporarily unavailable, then the other one continues to listen to the events generated by a SIP Server. Redis is used to deduplicate the events storing only one unique event instance for further processing.

EXCP processes three types of Engage events to build a call representation compatible with the GC call model:

1. Virtual Queue (VQ) DN Events:
    - VQ DNs represent reporting and routing queues in the Engage solution.
    - T-Library events consumed from the VQ DNs: *EventQueued* and *EventDiverted*
2. Extension DN Events
    - Extensions DNs is where agents log in to receive or make calls
    - T-Library events consumed from the Extension DNs: *EventUserEvent* - This event contains the current list of call participants
3. Call Monitoring Events:
    - Those events are not related to any DN and are generated by a SIP Server to provide call progress notifications.
    - Consumed Events: *EventCallCreated*, *EventCallDataChanged*, *EventCallDeleted*, *EventCallPartyAdded*, *EventCallPartyState*, and *EventCallPartyDeleted*

EXCP creates a dedicated SIP Server client to consume each type of events. Those clients can be pointed to one or multiple SIP Servers. Refer to the description of the environment variables VOICE_SERVER_VQ, VOICE_SERVER_AGENT, and VOICE_SERVER_CALL for configuration details.

## EXCP Deployment Topologies

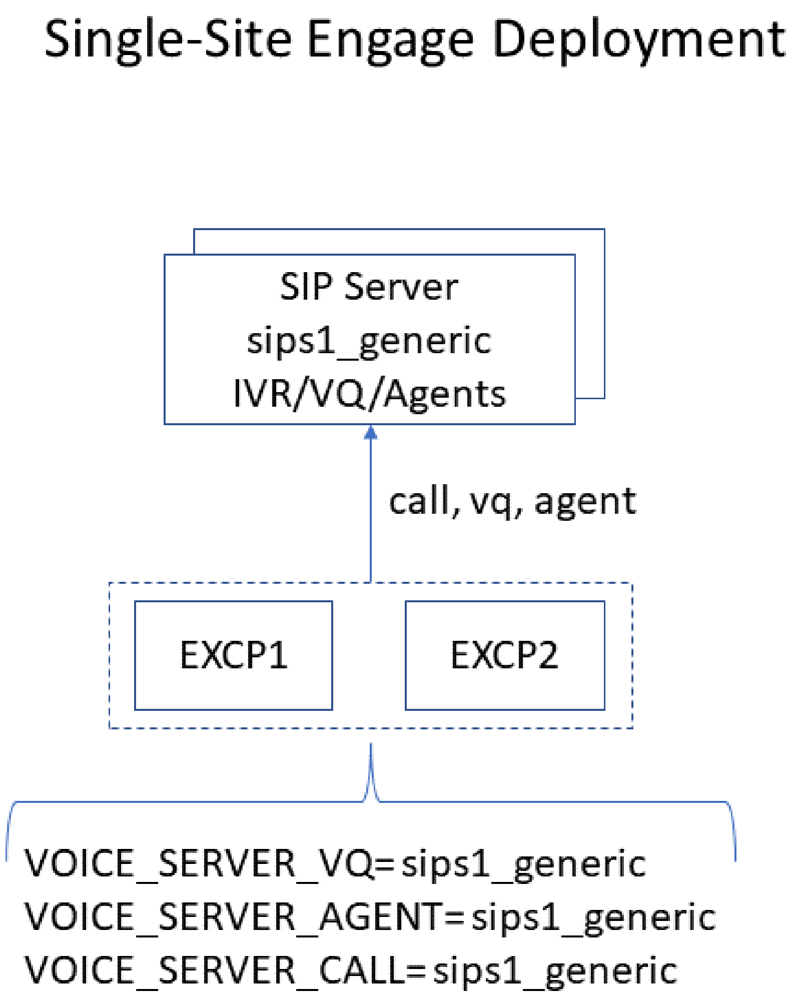
### SIP Server Roles in Engage Contact Center

SIP Servers can be used in different roles in the Engage Contact Centers. The SIP Server role defines the types of DNs, which are configured in a Switch used by this SIP Server. The SIP Server roles to DN types associated with this role are:

1. Generic:
   - Description: Usually, SIP Server is configured in smaller contact centers where one SIP Server HA pair can handle the load of the whole contact center.
   - DN Types: `Any`
2. IVR:
   - Description: IVR applications are hosted on RP DNs. SIP Servers are used in this role in larger environments to implement a 2-layer IVR-agent architecture where IVR processing is handled by the IVR SIP Server. After completing the self-service stage, the call is either terminated or routed to an agent located on a different SIP Server.
   - DN Types: `Routing Point (RP)`
3. Agent:
   - Description: Agent SIP Server is used to host agents and contains the DNs of type Extension. Usually, it is not responsible for any routing logic.
   - DN Types: `Extension`
4. VQ:
   - Description: VQ SIP Servers is a centralized place for VQ DNs that are used for reporting and routing for all calls on all sites of a contact center.
   - DN Types: `Virtual Queue (VQ)`

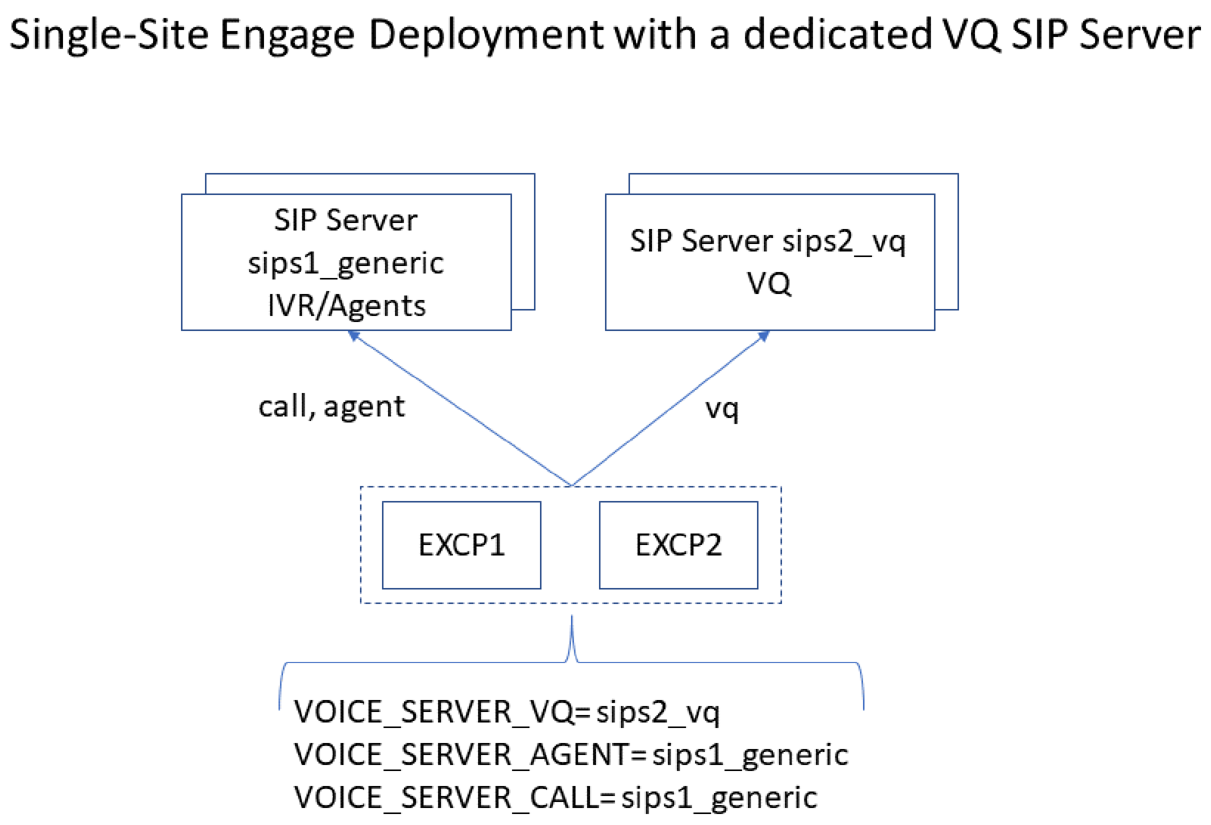
### Single-Site Deployment

In smaller environments, one SIP Server HA pair is used to handle the load of the whole contact center. To support this configuration one pair of the EXCP instances is deployed. Both instances are configured to collect all required Engage interaction events from the only SIP Server available in the environment.


Single-Site Engage Deployment
SIP Server
sips1_generic
IVR/VQ/Agents
call, vq, agent
EXCP1
EXCP2
VOICE_SERVER_VQ=sips1_generic
VOICE_SERVER_AGENT=sips1_generic
VOICE_SERVER_CALL=sips1_generic

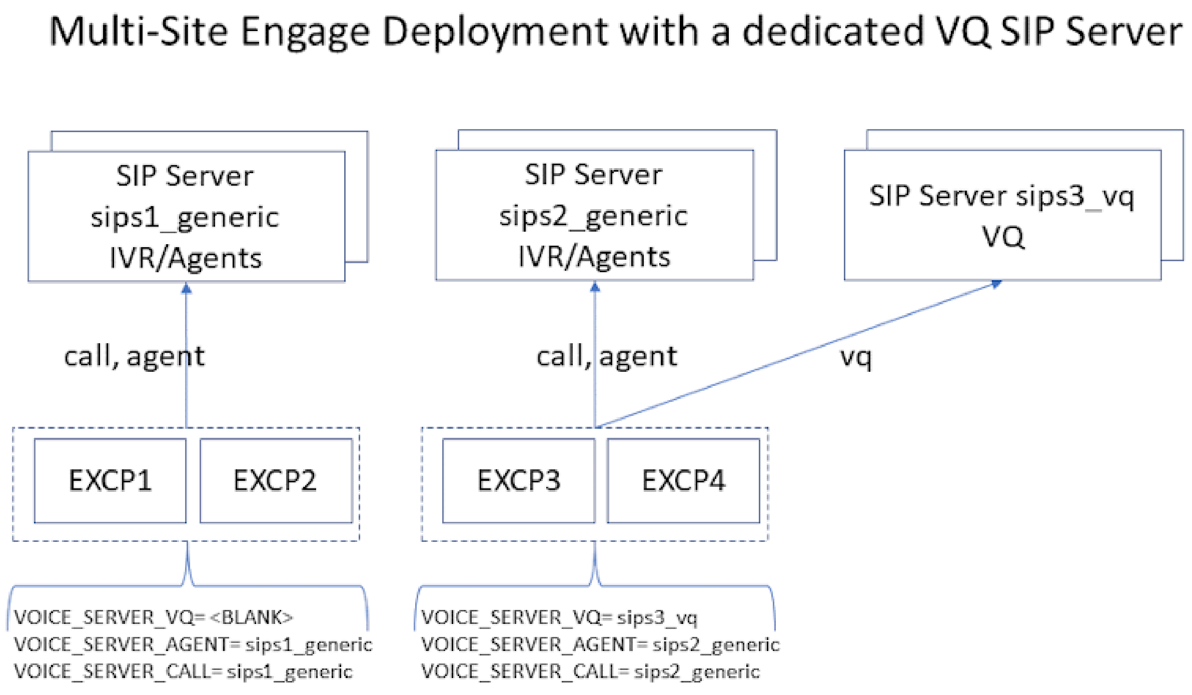
## Single-Site Deployment with a Dedicated VQ SIP Server

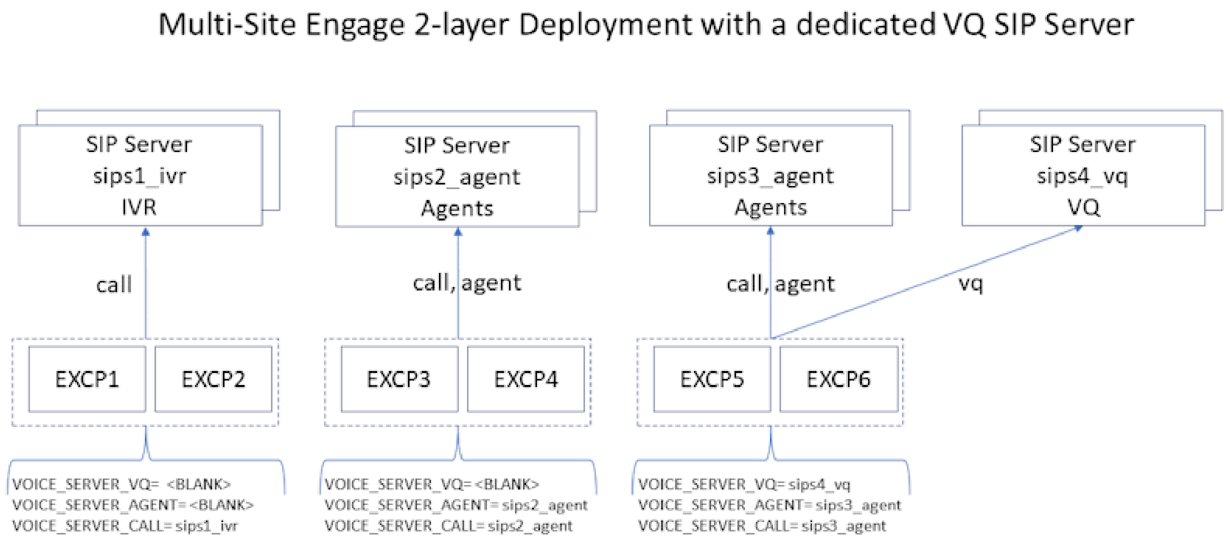
Engage deployment is considered a single-site environment even if two SIP Servers are present but one of them is a VQ SIP Server, which contains only VQ DNs configured on its switch. VQ SIP Server never handles a physical call and is used only for distributing events on behalf of the VQ DNs. In this configuration a single pair of EXCP instances is connected to both SIP Server HA pairs. Call monitoring events and events on agent extensions are received from a generic SIP Server. VQ events are consumed from the VQ SIP Server.


Single-Site Engage Deployment with a dedicated VQ SIP Server
SIP Server sips1_generic IVR/Agents
SIP Server sips2_vq VQ
call, agent
vq
EXCP1
EXCP2
VOICE_SERVER_VQ=sips2_vq
VOICE_SERVER_AGENT=sips1_generic
VOICE_SERVER_CALL=sips1_generic

## Multi-Site Deployment

For deploying EXCP in a multi-site environment, Genesys recommends using one EXCP pair per SIP Server HA pair Each EXCP in this pair connects to both primary and backup SIP Server instances in an HA pair.


Multi-Site Engage Deployment with a dedicated VQ SIP Server
SIP Server sips1_generic IVR/Agents
SIP Server sips2_generic IVR/Agents
SIP Server sips3_vq VQ
call, agent
call, agent
vq
EXCP1
EXCP2
EXCP3
EXCP4
VOICE_SERVER_VQ=<BLANK>
VOICE_SERVER_AGENT=sips1_generic
VOICE_SERVER_CALL=sips1_generic
VOICE_SERVER_VQ=sips3_vq
VOICE_SERVER_AGENT=sips2_generic
VOICE_SERVER_CALL=sips2_generic


Multi-Site Engage 2-layer Deployment with a dedicated VQ SIP Server
SIP Server sips1_ivr IVR
SIP Server sips2_agent Agents
SIP Server sips3_agent Agents
SIP Server sips4_vq VQ
call
call, agent
call, agent
vq
EXCP1
EXCP2
EXCP3
EXCP4
EXCP5
EXCP6
VOICE_SERVER_VQ= <BLANK>
VOICE_SERVER_AGENT=<BLANK>
VOICE_SERVER_CALL=sips1_ivr
VOICE_SERVER_VQ=<BLANK>
VOICE_SERVER_AGENT=sips2_agent
VOICE_SERVER_CALL=sips2_agent
VOICE_SERVER_VQ=sips4_vq
VOICE_SERVER_AGENT=sips3_agent
VOICE_SERVER_CALL=sips3_agent

All EXCP instances use the same centralized Redis cache.

## Engage Call Model Transformation for the EX Conversation

## Injection

### Engage Call Model Transformation Principles

GC EX conversation model supports a limited number of scenarios listed in the Sample Conversation Injection Scenarios section of the *Genesys Cloud EX Integration Guide*. Calls are allowed to be injected into the EX Org only if they match one of the supported scenarios.

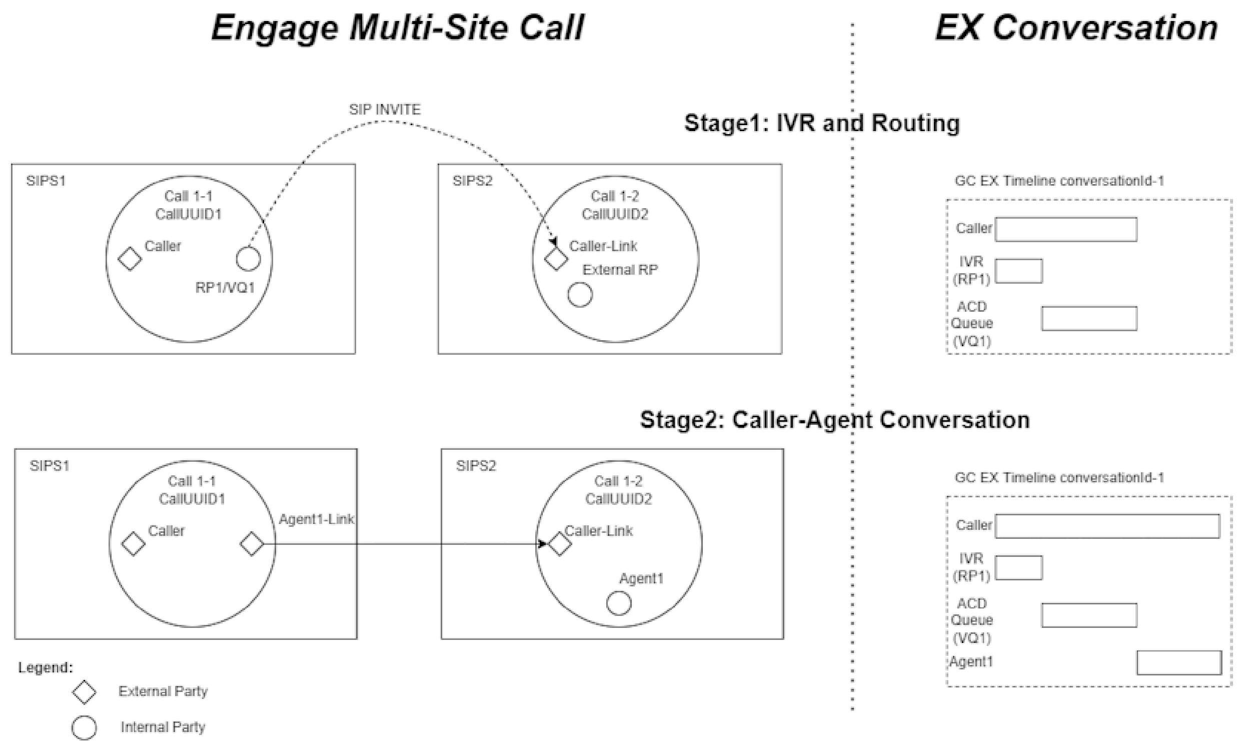
Basic Engage contact center scenarios such as inbound calls to agents can be mapped to one of the EX supported scenarios with simple translation of the Engage call events to the GC conversation events. More complex scenarios such as transfers can also be mapped but they require some changes to be made to the call event flow. There is also a group of more complex Engage call flows such as conferences, which do not have any matching scenario patterns on the EX side. To avoid injecting unsupported scenarios to the EX Org EXCP monitors compatibility of an Engage call with the EX requirements while the call progresses. As soon as the Engage call goes out of compliance with the supported EX conversation scenarios, EXCP stops injecting interaction events for this call to the EX Org by transitioning this call to a Monitoring Suspension Mode, which is described later in this chapter. Such Engage calls are not fully represented in the EX Org.

### Multi-Site Call Conversion to an EX Conversation

A simple multi-site call is depicted:

1. Inbound call arrives on SIP Server SIPS1 where it goes through the IVR application loaded on a Routing Point DN RP1.
2. The call is reported to URS for routing. URS places the call on Virtual Queue DN VQ1 while looking for an available agent.
3. Agent1 selected for the call is located on a different switch. Call is routed to Agent1.
4. Agent1 answers the call and talks to the caller.

The left side of the following diagram represents TLib topology of the Engage call. EXCP reduces a complex Engage multi-site call to one GC EX conversation. The right side shows a mockup of the GC EX Conversation timeline, which users can see in the GC UI. The bar on the right side of the party name in the GC EX conversation timeline represents the time interval when the corresponding party was active in this conversation.



## Call Scenarios Not Supported by the GC EX Org

GC EX integration doesn't support the following scenarios, which can be encountered in the Engage contact centers:

1. Conferences
2. Call Supervision
3. Outbound campaign calls of fully automated (agent-less) campaigns and ASM dialing mode with merging two independent calls made to agent and to customer.
4. Calls to the Extension DNs without logged-in agents.
   - Such calls can be encountered in two scenarios:
     1. Engage contact centers can use remote agents/consultants who never log in to an agent desktop
     2. If an Engage agent is not selected for the EX-integration (through folder filtering) and is not mapped to GC, then an Engage call to such agent cannot be injected into the GC EX Org.
5. Multiple independent calls on an agent DN
   - GC EX agent is allowed to have only one main and one consult call at a time. EXCP ignores all other concurrent calls on agent DN.
6. Nested consult calls

- GC EX call model allows only one consult call made out of the main call. Consult calls made out of other consult calls are not allowed.

## Monitoring Suspension Mode

EXCP moves a GC EX conversation into the monitoring suspension mode if one of the GC EX conversation model rules is violated. In this mode EXCP generates only the *communicationEnded* events for the suspended GC EX conversation to indicate that the corresponding party has left the conversation. EXCP indicates a party removal from the GC EX conversation when corresponding party leaves the Engage call.

A sample activation of the Monitoring Suspension Mode for an Engage conference call process:

1. Inbound call lands on the RP DN and goes through the IVR treatments.
    - EXCP reports to GC that a new conversation is created and caller talks to the IVR application .
2. URS places a call on a VQ DN and starts searching an agent to route the call to:
    - EXCP reports to GC that the IVR party left the call and now call is placed on the ACD queue.
3. URS routes the call to an agent and agent answers the call
    - EXCP reports to GC that a successful user transfer took place and now the caller talks to an agent.
4. Agent initiates a single step conference inviting second agent to the call.
    - EXCP detects an EX call model violation as Conference scenarios are not supported and activates Monitoring Suspension Mode on this call.
    - EXCP doesn't report to GC that the second agent joined the call.
5. First agent leaves the call while caller continues talking to the second agent.
    - EXCP reports to GC that the first agent left the call.
    - GC conversation has only one party, a caller, left at this point because the second agent is not reported to the GC. So, to avoid leaving a call with a single party in GC EXCP reports that the remaining party, a caller, leaves the call.

This example demonstrates how EXCP handles Engage call flows, which do not match any of EX supported scenarios. In this scenario, GC received the information about the call being transferred from ACD queue to the first agent. This information is important for the WFM and Gamification services. However, the information about the presence of the second agent in the call is missing. Also, the duration of the GC conversation is shorter than the one of the corresponding Engage call.

## Handling of Unsupported call flows in EXCP

GC EX supports 17 types of call flows. In Engage, when a call is received which does not match any of the supported scenarios, the calls are matched to the nearest possible scenario. If the call could not be matched, then it's marked as ignored.

For these ignored calls,

1. No new parties are added.

2. The call terminates when there is only one active party in GC.

For example, in SSC in Engage, when a third party is added, this is not supported by GC. EXCP ignores adding additional parties and only the first two parties are shown in GC. When either of the first two parties terminates the call, the call is terminated in GC EX.

# EX Engage Connector Recording Provider (EXRP)

EX Engage Connector (EXEC) enables Genesys Cloud (GC) recording management, Quality Monitoring (QM) services, and Speech and Text Analytics (STA) functionality for interactions happening in the Engage Contact Centers. Recording injection and STA-enablement in EXEC is implemented by two components:

1. The EX Conversation Provider (EXCP) component gathers the recording and STA metadata for passing it to the EX Recording Provider (EXRP) when a voice interaction ends.
2. The EX Recording Provider (EXRP) component is responsible for the injection of recorded voice calls along with its metadata (from the Engage contact center into the GC EX Organization) using the public EX API as outlined in EX Recording Injection. The EXRP supports an *N+1* redundancy model for enhanced reliability.

**Important**

All GC services relying on the injected voice recordings to work such as QM and STA should be properly configured in the GC EX Org. Please refer the GC documentation for details.

## Recording Injection Use Cases for the Engage Contact Centers

### Customers with an existing recording solution

EXEC recording injection can be activated in the Engage Contact centers where active recording is used. An active recording solution includes the following components:

- **Recording management**: Engage GIR solution or a third-party recorder
- **Speech Analytics**: Engage Genesys Interaction Analytics (GIA)
- **Media management**: Media Control Platform (MCP)
  - If GIR is used, MCP records voice conversation and passes it to GIR for storing and processing.
  - When used with a third-party recorder, MCP forks the voice stream to the third-party recorder.
- **Call recording control**: SIP Server controls recording through start/stop/pause/resume commands to the MCP.

### Customers without a recording solution

For customers without a recording solution, EXEC can enable GC recording management, QM, and

STA services if your Engage Contact Center uses SIP Server and GVP (Genesys Voice Platform).

- To inject recordings into GC using EXEC, you will need to:
    - Expand your MCP farm with additional recording MCPs.
    - Configure SIP Server-based active recording for your agents.

**Important**

Recording injection should also be configured and enabled to use STA functionality.

### GIA Replacement

GC provides state-of-the-art AI-based STA services which delivers capabilities better than GIA. Customers can continue using GIR for compliance recording and replace their existing GIA-based analytics solution with GC STA.

## Recording Transcription for the Engage Contact Centers

EXEC deployed in the recording injection mode can request the injected recordings to be processed by the GC STA (Speech and Text Analytics) services. As a result, injected recordings marked to be processed by the GC STA are transcribed, their transcripts are analyzed for the presence of certain topics, emotion trends are identified, and other STA services may also be applied. EXEC inserts the STA Program ID and the recording language into the injected recording metadata to trigger the STA processing on the GC side.

Customers may want to maintain distinct STA programs for different LOBs. Tech Support conversations should be analyzed using one set of topics, while the conversations of the Wealth Advisors may require a completely different set. Also, each LOB can have queues to serve calls in different languages.

EXEC is designed to identify a queue used to deliver a call to an agent (Origination Queue) and add a program ID and a language associated with this queue as metadata properties of a recording injected to GC. If one call is handled by agents from different LOBs who received this call from their corresponding queues, then recording segments produced for those agents will have Program ID and Language properties collected from the corresponding queues, which may not be the same. If a call is delivered to an agent directly without using a queue, then default org program ID and language are used.

### Transcription

EXEC reads the following GC configuration to identify the general settings of transcription for GC Organization and its' queues:

1. Transcription enabled for GC Org - See the *Enable voice transcription* section in Speech and text analytics.

   - EXEC requests STA for this injected recording only if transcription is enabled in the EX org. If transcription for the org is disabled, STA is not requested for any of the injected recordings.
2. Transcription enabled on Queue - See *Configure voice channels* subsection under the *Set behavior and thresholds for all interaction types* in Create and configure queues
   - EXEC requests STA for the injected recording only if transcription is enabled on a queue.

Transcription configuration is pulled from GC by EXEC every 15 minutes.

## Program ID

EXEC reads the following GC configuration to identify the program ID associated with the origination queue:

1. Program assigned to Queue - See the *Edit a program* section in Work with a program.
   - EXEC checks if any program is assigned to a queue. Only one program can be assigned to one queue.
2. Default program - See the *Set a default program* section in Work with a program.
   - If no program is assigned to a queue, EXEC uses a default org program
3. If no program is identified in any of the previous steps or transcription is disabled for the org or queue, EXEC doesn’t request STA to be performed on the injected recording.

Program configuration is pulled from GC by EXEC every 15 minutes

## Language

There are two possible options to configure language, used for transcription:

1. Using Routing Strategies.
2. Setting the default language.

Both options are configured on Engage side by setting corresponding hybrid_integration transaction configuration options.

## Routing Strategies

Complex call centers can have multiple departments, using different languages. In order to use specific language for some department, Routing Strategy which is used to route a call to this department, has to be modified. Sending a ***RequestUpdateUserData*** should be added to the routing strategy. This request should add a key to call ***userData*** to specify the language of the call.

For example:

```
{"language_code":"de-DE"}
```

The key, used for language code (“language_code” in the example above), has to be specified as a hybrid_integration transaction parameter *sta_userdata_language_key*.

Value is a string, containing a language code of the desired language.

A list of supported languages and their codes can be found at Genesys Cloud supported languages under the *Native voice transcription* section.

Language can be updated several times during the call. There must be only one key specifying the language in the call user data.

### Default language

Default language represents the main language, used in the contact center. This language is used if EXEC was not able to get the language configuration from other sources. Default language has to be specified as a hybrid_integration transacton parameter *sta_default_language.*

EXEC defines language for every recording segment made on behalf of an Agent. The language is defined, when Agent answers the call in the following way:

1. Language is taken from call user data if it's set from the routing strategy
2. Otherwise, the default language is used from the *sta_default_language* hybrid_integration parameter.

For recordings made on behalf of a trunk, default language is always used.

Valid language value is a language code of a language, supported by GC. If language value, received from call userData is invalid, default language value is used. If default language value is invalid, STA is not requested for the recording segment.

## EXEC Deployment Options for Recording Injection

This section explains how EXEC can be integrated with two different types of the Engage recording solution:

1. Third-party recorder or GIR without GIA
2. GIR with GIA

Those two categories are defined based on the configuration required to integrate the EXEC. In the first case, customers do not use the Engage speech analytics solution. This architecture allows to connect GC STA as a new analytics solution without changing the existing recording processing flow in the contact center. Solutions in the second category use GIA. As a result, GC STA should be brought as a replacement of the existing analytics solution. In most cases, customers can use only one analytics solution at a time (either GIA or GC STA). At the same time, some Engage contact center configurations may allow to use GC STA only for a subset of call recordings when the rest of the calls are still processed in GIA.

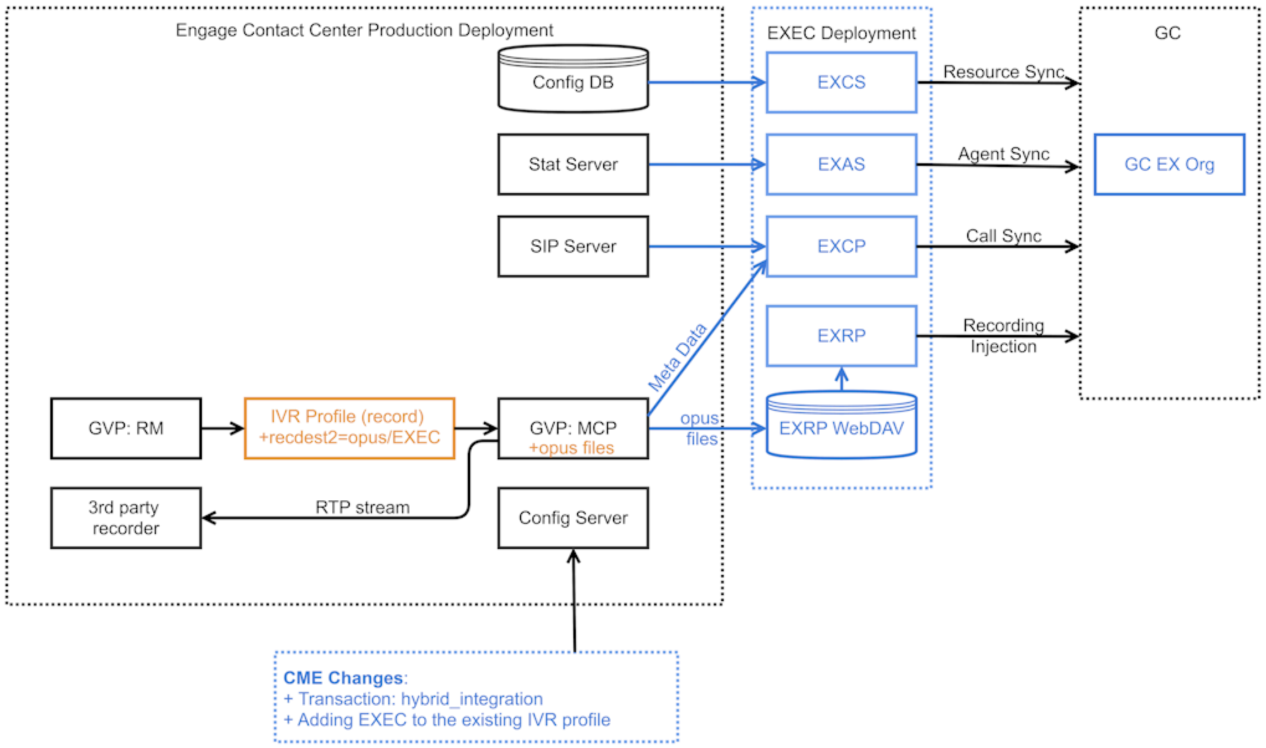
**Important**

Diagrams provided in this section are color-coded: blue indicating a new component or connection and orange for components that require configuration changes. Diagrams show only components that are essential for a particular use case.

## Third-party Recorder or GIR without GIA

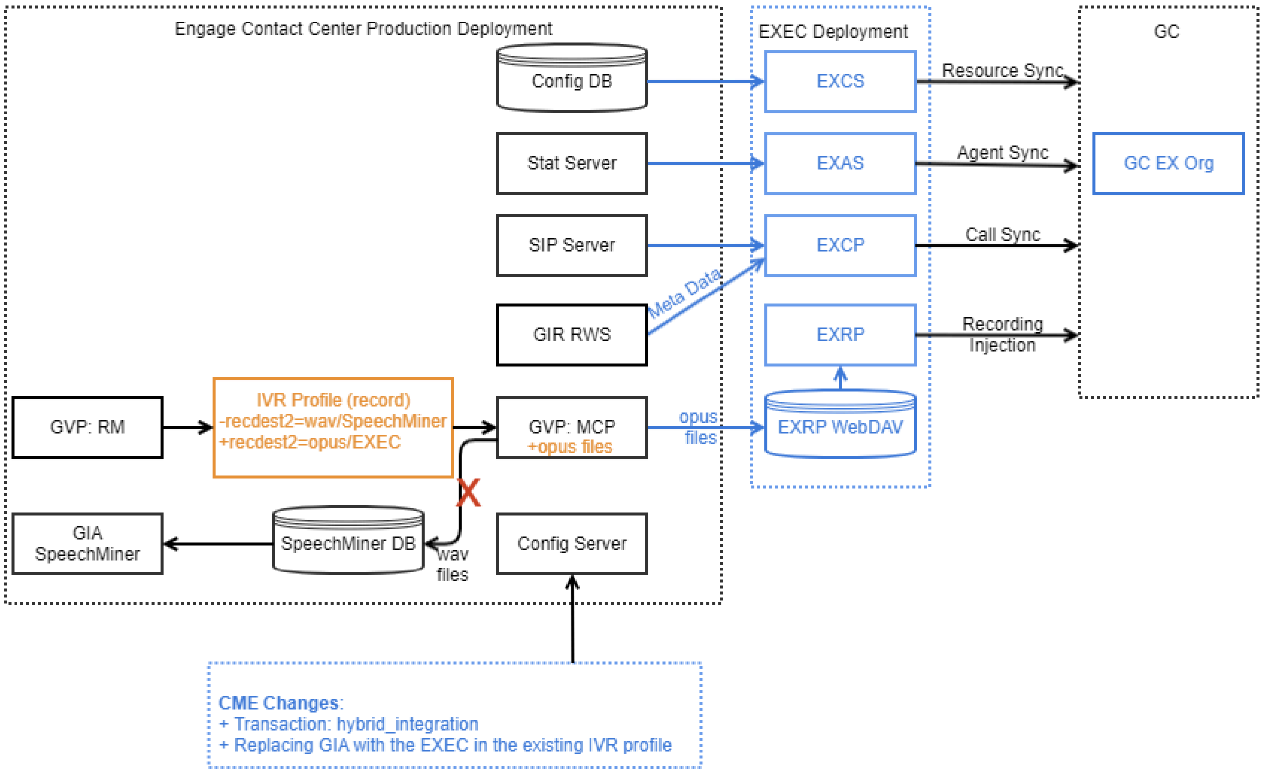
The following diagram depicts an integration with a third-party recorder. Even though the EXEC integration with the GIR-based solution is slightly different, the required Engage configuration changes are the same in both cases.

In this configuration the existing IVR profile ‘record’ is modified. Based on those changes MCP starts producing recording files in opus format to push those files to EXEC. In case of the third-party recorder, IVR profile is also configured to instruct MCP to push the recording metadata to the EXCP. If GIR-based solution is used, EXCP pulls metadata from the GIR RWS component.



## GIR with GIA

If Engage customers use GIA for speech analytics, then GIA should be replaced with the EXEC in the existing IVR Profile 'record' as shown on the diagram below. In this case, MCP stops delivering recordings to the GIA as soon as IVR Profile configuration is changed. Speech analytics can be done only in the GC STA while new configuration is active. Configuration can be rolled back instantly without restart of any components to restore regular GIA-based operations.



## Defining Recording Synchronization Scope (Sync-Scope)

Sync-scope refers to a designated portion of your Engage contact center environment configured for injection into the Genesys Cloud EX organization. You define the sync-scope by adding configurations to the Engage CME. If no sync-scope is specified, EXEC syncs your entire Engage environment. Defining a sync-scope is highly recommended, with a smaller scope being ideal for pilot deployments or gradual migrations.

EXEC components, EXAS and EXCP, inject agent and call-related activity only for those belonging to the sync-scope. EXCP then sends a recording injection request to EXRP for all injected calls. Consequently, EXRP also adheres to the defined sync-scope.

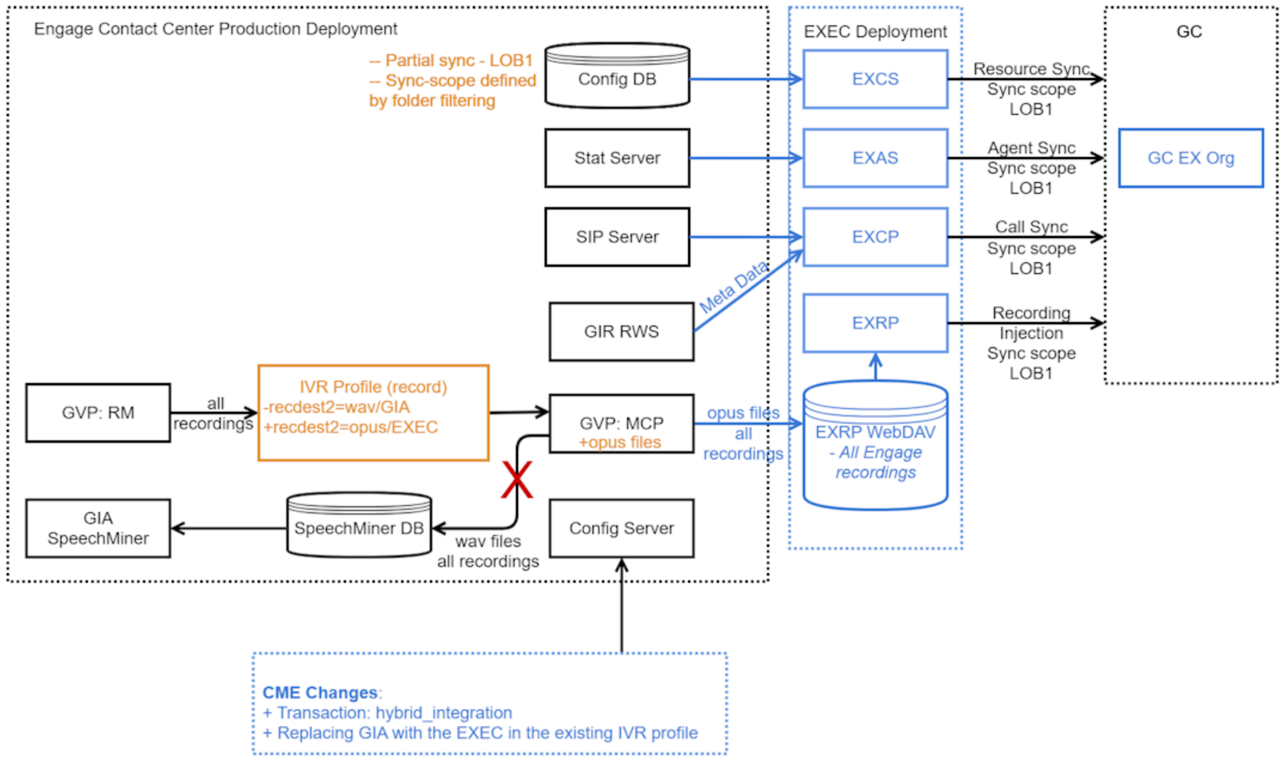
Voice recordings to be injected to GC are collected in the EXEC WebDAV transient storage first. Engage solution can be configured to push either all recordings or only the recordings related to the sync-scope calls into the WebDAV storage. In both cases EXRP injects only the sync-scope recordings into GC. So, those two approaches are identical from the end-user perspective. The difference is in the required size of the WebDAV storage. More details about the pros and cons of those configurations are provided in the following sections.

## Storing All Engage Recordings in WebDAV

The following diagram depicts an EXEC integration with the Engage contact center where GIA is used for speech analytics (mentioned earlier). In the example, customers select one of their lines of business LOB1 to be synced to GC. As a result, all EXEC modules only inject the data related to the selected LOB1. GIA stops receiving any recordings. So, call center agent and supervisors should log in to GC to use the STA services there instead.

The main advantage of this approach is a simple configuration, which can be used in any Engage deployment. Customers need to replace GIA in their 'record' IVR Profile configuration with the EXEC to redirect the recording files accordingly.

This approach requires EXRP WebDAV transient recording storage to hold all Engage contact center recordings produced in the last 24 hours. In large contact centers, it may require a significant amount of storage and processing power.



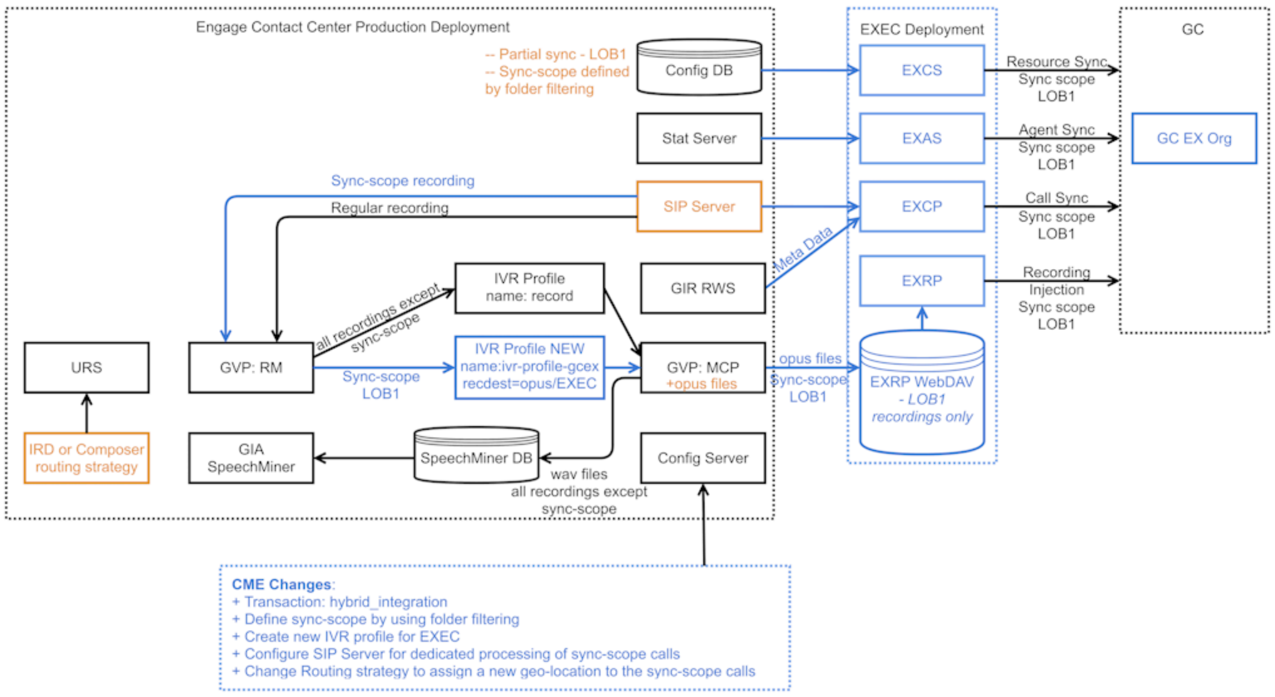
## Storing Only Engage Sync-Scope Recordings in WebDAV

In certain Engage configurations, you can create a dedicated processing flow for sync-scope calls at the SIP Server level. This allows GC STA and GIA to be used in parallel, with GC STA working with sync-scope calls while all other calls are processed by GIA. Specific Engage configurations allow creating a dedicated processing flow for sync-scope calls at the SIP Server level. This can be implemented in environments using SIP Server geo-locations with the MSML strict matching policy, or potentially other environments.

This approach allows:

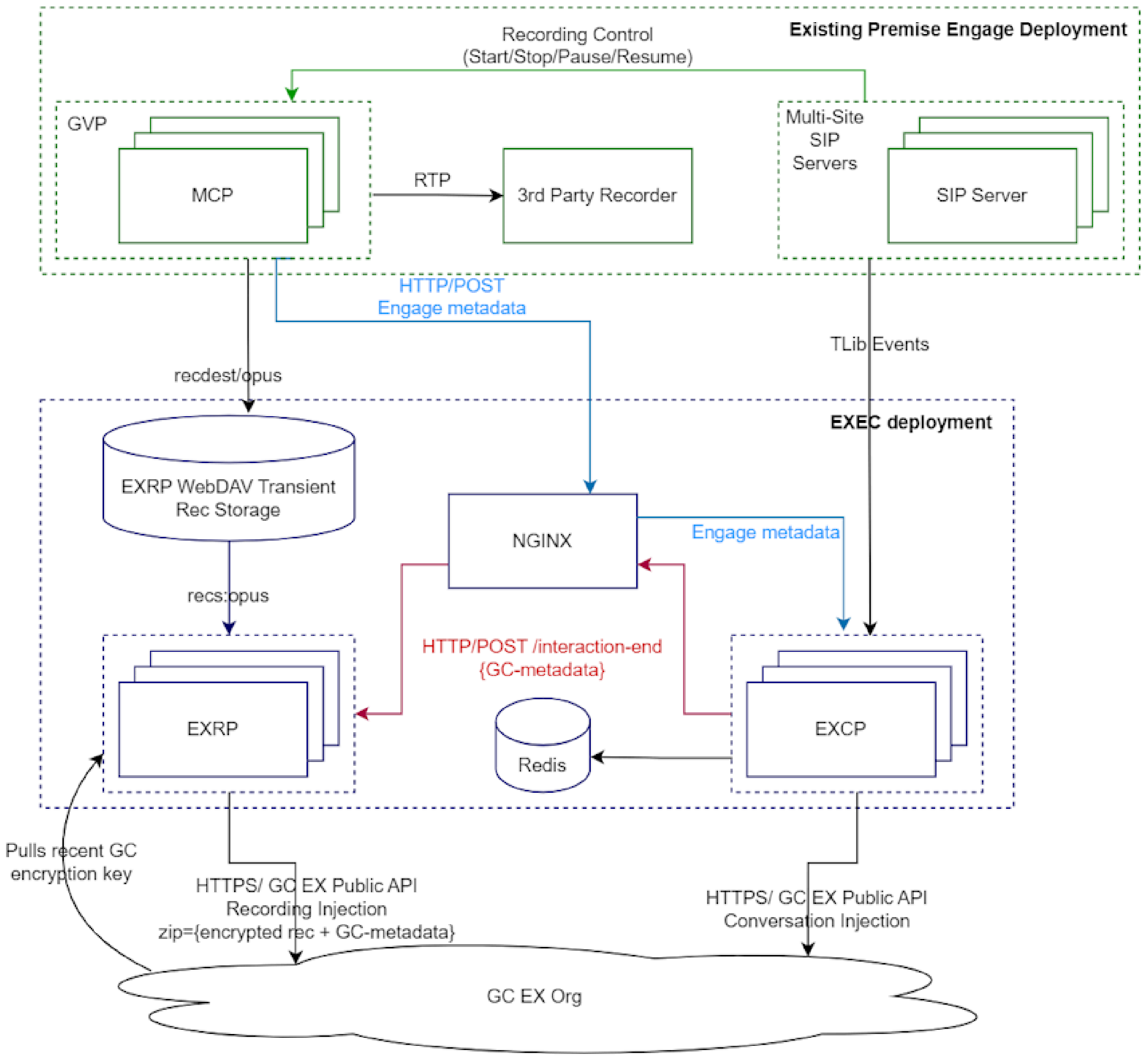
- Process sync-scope calls separately from other types of calls. GC STA can analyze sync-scope calls, while GIA handles all other calls.
- WebDAV storage only needs to accommodate recordings from sync-scope calls.

These advantages stem from SIP Server being configured to identify and report sync-scope calls to the GVP (Genesys Voice Platform), a core component of the Engage recording solution. GVP applies a newly created IVR profile to the sync-scope calls and MCP generates Opus recording files only for the sync-scope calls and pushes them to GC EXEC. Processing of all other recordings remains unchanged.


Engage Contact Center Production Deployment
-- Partial sync - LOB1
-- Sync-scope defined by folder filtering
Config DB
Stat Server
Sync-scope recording
Regular recording
SIP Server
IVR Profile name: record
GIR RWS
Meta Data
all recordings except sync-scope
URS
GVP: RM
Sync-scope LOB1
IVR Profile NEW name:ivr-profile-gcex recdest=opus/EXEC
GVP: MCP +opus files
opus files Sync-scope LOB1
IRD or Composer routing strategy
GIA SpeechMiner
SpeechMiner DB
wav files all recordings except sync-scope
Config Server
EXEC Deployment
EXCS
EXAS
EXCP
EXRP
EXRP WebDAV - LOB1 recordings only
Resource Sync Sync scope LOB1
Agent Sync Sync scope LOB1
Call Sync Sync scope LOB1
Recording Injection Sync scope LOB1
GC
GC EX Org
CME Changes:
+ Transaction: hybrid_integration
+ Define sync-scope by using folder filtering
+ Create new IVR profile for EXEC
+ Configure SIP Server for dedicated processing of sync-scope calls
+ Change Routing strategy to assign a new geo-location to the sync-scope calls

## How recording injection works in EX Engage Connector

This section explains the recording injection processing flow in EXEC. The following diagram depicts EXEC integration with the third-party recorder only. Processing flow is mostly the same for the third-party recorder and GIR/GIA integrations. The difference between the two is how EXCP obtains the recording metadata. In the first case, MCP pushes metadata to the EXCP as soon as the recording segment is complete. In the GIR/GIA integration, EXCP pulls metadata from the GIR RWS component when the interaction ends.


Existing Premise Engage Deployment
Recording Control (Start/Stop/Pause/Resume)
GVP
MCP
RTP
3rd Party Recorder
Multi-Site SIP Servers
SIP Server
HTTP/POST Engage metadata
TLib Events
recdest/opus
EXEC deployment
EXRP WebDAV Transient Rec Storage
NGINX
Engage metadata
recs:opus
HTTP/POST /interaction-end {GC-metadata}
EXRP
Redis
EXCP
Pulls recent GC encryption key
HTTPS/ GC EX Public API Recording Injection zip={encrypted rec + GC-metadata}
HTTPS/ GC EX Public API Conversation Injection
GC EX Org

[Insert diagram illustrating the processing flow]

Recordings are processed as follows:

1. MCP records an audio file in the opus format covering one segment of a contact center voice call.
2. MCP stores the recorded file to the EXRP WebDAV Transient Recording Storage using a link configured in the **recdest2** configuration option.
3. MCP posts the recording metadata to the EXCP via HTTP Load balancer which injects the interaction events into GC.
4. EXCP stores the received metadata to Redis.
5. EXCP detects the end of an interaction and calls the EXRP REST API via HTTP Load Balancer. This interaction-end API is called once for each recorded call segment providing GC-metadata (recording and STA metadata) and the name of the recorded audio file for this segment.

6. EXRP pulls the audio recording file from the WebDAV storage.
7. EXRP encrypts the recording using GC public key which it pulls periodically using GC EX Public API.
8. EXRP creates a ZIP archive, which contains one encrypted audio file and a JSON file with the corresponding GC-metadata.
9. EXRP obtains an S3-pre-signed URL to upload the created archive by calling GC public API.
10. EXRP uploads the ZIP archive to an S3 bucket using the pre-signed upload URL obtained on the previous step.

# Limitations

Currently, the following limitations are present for EX Engage Connector:

- EX Engage Connector can connect to the Engage contact center where configuration is stored in Oracle, PostgreSQL or Microsoft SQL Server databases.
- EX Engage Connector can work only with a single Engage tenant. If customers use Engage multi-tenant Config DB and have multiple tenants provisioned, then they should select one of the tenants for the EX integration and provision this tenant accordingly.
- EX Engage Connector requires a call routed to an agent to go through a VQ DN, which is configured to be mapped to the GC EX Org as an ACD Queue. Call distributed to agents without involving a VQ DN (for example, through an Engage ACD queue) may not be injected into the GC EX Org correctly.
- EX integration supports injection of voice interactions only. To optimize efficiency of the WFM and Gamification services it is recommended to avoid adding digital or blended (voice/digital) agents to the EX integration scope.
- The internal call between agents in same site are not reported to GC EX and these calls are ignored.
- The call overflow (COF) feature is not yet supported.
- If MCP exhausts its retry attempts and fails to upload a voice recording to the WebDAV storage due to some failure, then the recording is not injected to GC.
- The following Engage call scenarios are not injected to GC: ASM outbound campaign calls, internal calls, and calls to DNs with no agent logged in. If those calls are recorded, then those recordings are not injected to GC.
- If an Engage call enters a Monitoring Suspension Mode in EXCP and recordings are produced for the Engage parties, which are not injected into the GC, then those recordings are still injected to GC and linked to the first party in this call.
- Recordings can be fetched from multiple Engage data centers, but they are all stored in one AWS region where EX Org is provisioned.
- The Engage recording file name is restricted to 80 characters as Genesys Cloud API supports only up to 80 characters for recording file names.
- If an Engage recording is made on behalf of a trunk and STA is enabled for a GC Organization, then EXEC requests STA for the whole recording, including Queue and IVR segments. Recording and Transcription suppression settings are not taken into account.

# Deploying EX Engage Connector

There are two options to deploy EX Engage Connector:

1. No-media deployment
2. Deployment for recording injection

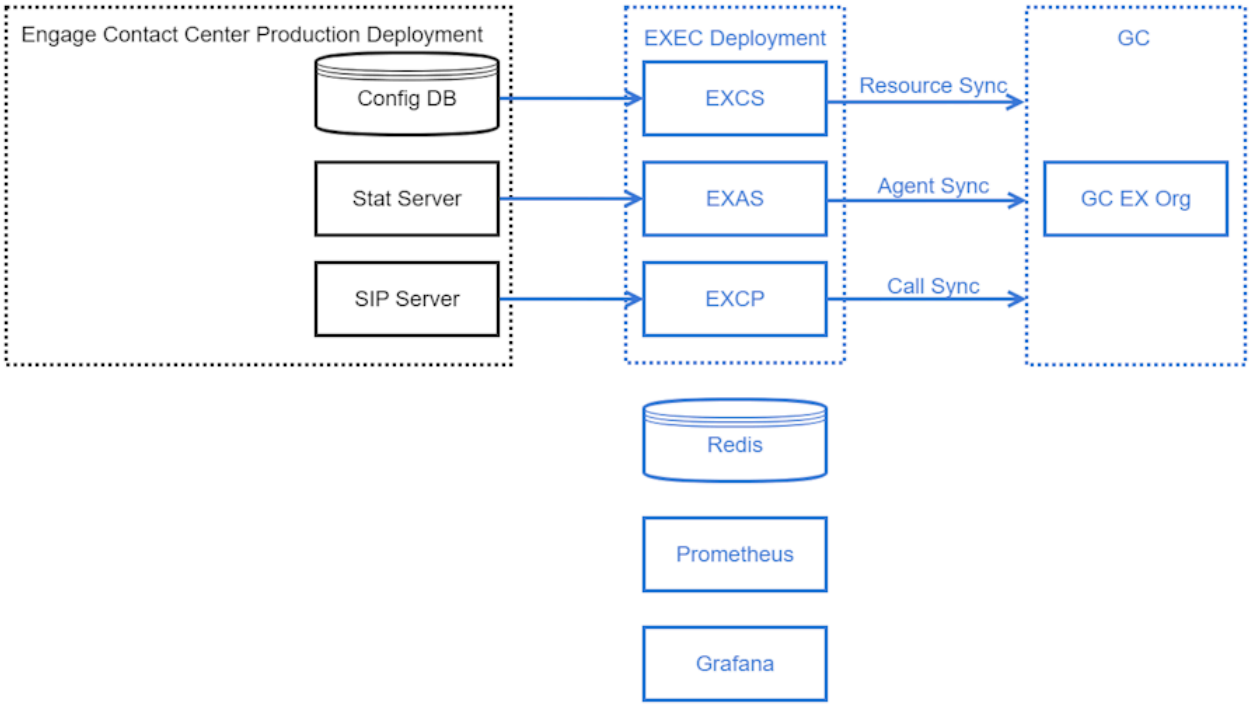
The No-media deployment is always required. Recording injection can be added on top of a working no-media EXEC deployment.

## No-media Deployment

No-media deployment is used to enable such GC services as WFM, coaching, and gamification. Components and connections given in a blue color on the diagram below represent the changes required to integrate existing Engage contact center with GC EX Org. Dashed box ‘EXEC Deployment’ show components provided by Genesys. Blue components not included into this box such as Redis, Prometheus, and Grafana must be supplied by customers.

Some key integration points:

- EXCS pulls configuration from the Engage configuration DB and injects it to the EX Org using GC Public API.
- EXAS subscribes to Stat Server to receive agent state change notifications and injects agent routing and presence states to the EX Org using GC Public API.
- EXCP subscribes to SIP Server for Engage call events and injects GC conversation events to the EX Org using GC Public API.

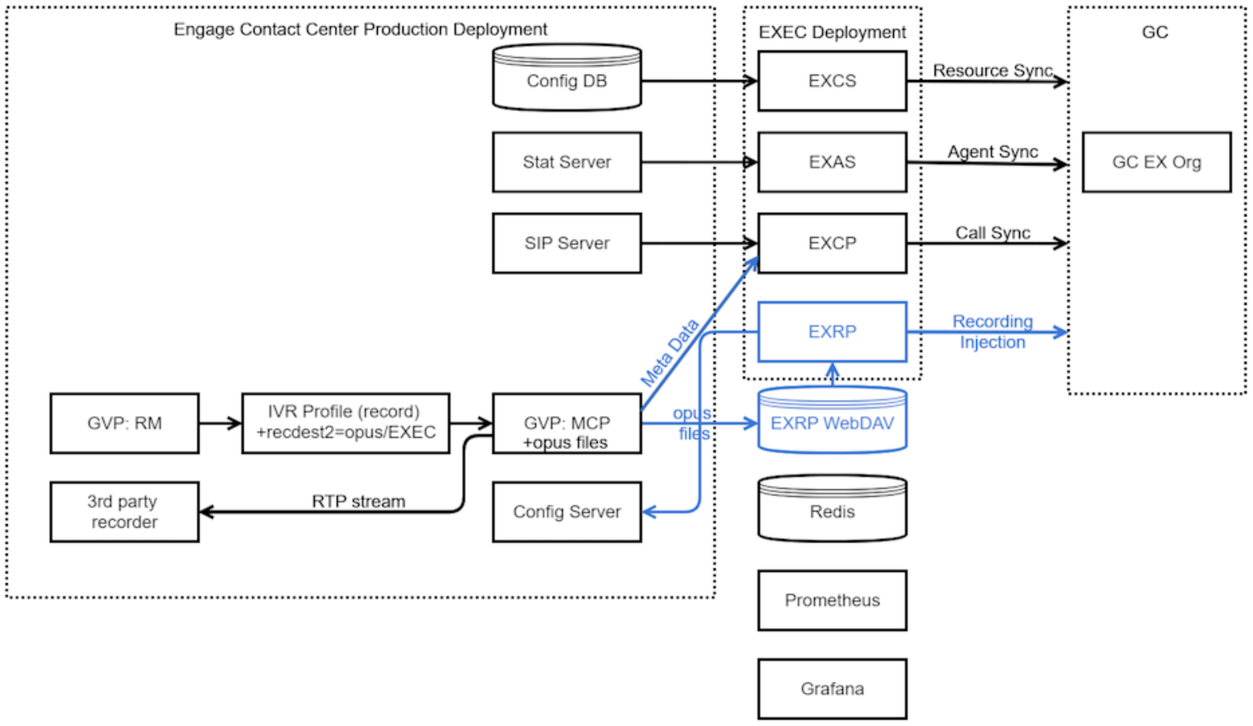


## Deployment for Recording Injection

This deployment is used when GC services such as recording management and Speech-and-Text Analytics (STA) are used for the Engage contact center recording processing. Deployment for recording injection is built on top of working EXEC no-media deployment. Blue components and connections on the diagrams in this section represent the changes required in the EXEC no-media deployment for adding recording injection functionality to it.

Some key integration points:

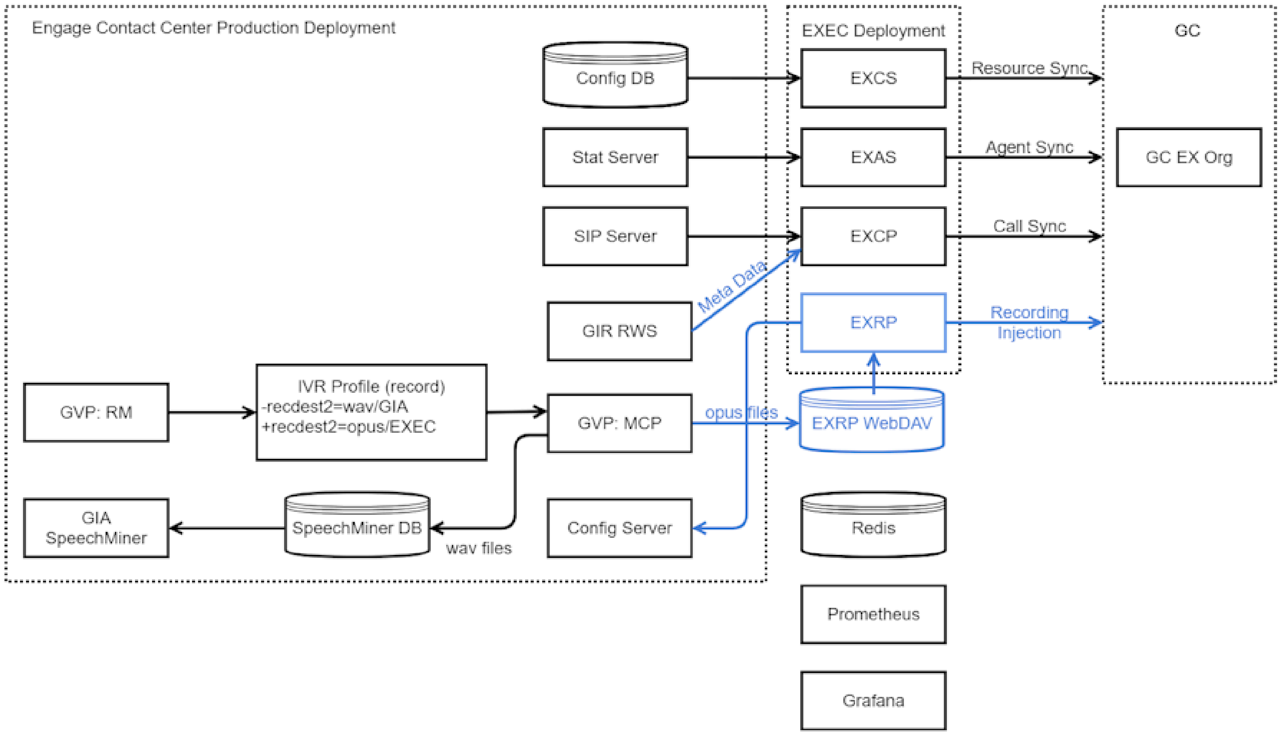
1. MCP
    1. Generates voice call recordings in an opus format
    2. Pushes recording metadata to the EXCP.
    3. Pushes recording files to the EXRP transient WebDAV storage.
2. EXRP
    1. Pulls configuration from the Engage Config Server.
    2. Injects recordings to the EX Org using GC Public API.



The following diagram demonstrates the EXEC integration with the Engage contact center where recording solution is based on Engage native GIR/GIA components.

Some key integration points:

1. EXCP pulls recording metadata from the GIR RWS.
2. MCP
    1. Generate voice call recordings in an opus format
    2. Pushes recording files to the EXRP transient WebDAV storage.
3. EXRP
    1. Pulls configuration from the Engage Config Server.
    2. Injects recordings to the EX Org using GC Public API.


Engage Contact Center Production Deployment
Config DB
Stat Server
SIP Server
GIR RWS
GVP: RM
IVR Profile (record)
-recdest2=wav/GIA
+recdest2=opus/EXEC
GVP: MCP
GIA
SpeechMiner
SpeechMiner DB
wav files
Config Server
Meta Data
opus files
EXEC Deployment
EXCS
EXAS
EXCP
EXRP
EXRP WebDAV
Redis
Prometheus
Grafana
Resource Sync
Agent Sync
Call Sync
Recording
Injection
GC
GC EX Org

# No-media Deployment

## Prerequisites

**Important**

Review the Limitations page before planning.

This section lists the requirements for the environment where EXEC is planned to be deployed. Mandatory and optional requirements are given in the corresponding sections below.

### Mandatory Environment Elements

#### Redis Cache

EX Engage Connector supports Enterprise Redis version 6.x. Redis Cluster mode is not supported. Customers should plan to use 2 GB of Redis memory for contact centers with 1000 agents and a call rate of 5 CAPS.

#### Container Registry

EXEC components are provided as Docker containers. Customers should provide a container registry where EXEC Docker images can be hosted. Docker composer files of the EXEC components will be provisioned to pull Docker images from this registry.

#### Virtual Machines for the EXEC Components

Four VMs have to be created to deploy EXEC components to have each VM to carry one component instance:

- 1 VM for EX Engage Connector Config Sync (EXCS)
- 1 VM for EX Engage Connector Agent State Sync (EXAS)
- 2 VMs for 1 SIPS HA Pair for deploying EX Engage Connector Conversation Provider (EXCP)

VMs allocated for the EXEC components should be provisioned as follows:

- 64-bit Linux OS with kernel version 4.18 or later
- Recommended resource allocation: 2 CPU Cores, 8 GB RAM, and at least 80 GB HDD
- Docker version 20 or later

The EXEC VM must sync the system time at least once every 24 hours as the communication between the EXEC services and Genesys Cloud are very time-sensitive.

All VMs running EX Engage Connector components should belong to the same local network segment and be interconnected so that all components can communicate over the network. EXEC components can use either FQDNs or IP addresses to establish communication with each other. The connectivity table lists both incoming and outgoing connections required by the EXEC components.

| Service | Direction | Protocol | Local Port | Remote Peer | Remote Peer Port | Purpose |
|---|---|---|---|---|---|---|
| EXCS | Outgoing | TCP/TLS | Any | Engage Config DB | 5432 (default RDBMS listening port) | Read Engage configuration. |
| EXAS | Outgoing | TCP | Any | Engage Primary Stat Server | 2060 ( default Primary Stat Server listening port) | Obtain real-time agent state data. |
| EXAS | Outgoing | TCP | Any | Engage Backup Stat Server | Backup Stat Server listening port (2060) default | Obtain real-time agent state data |
| EXCP | Outgoing | TCP | Any | Engage SIP Servers (multiple) | 8000 (default SIP Server default listening port) | To fetch real-time Call, DN, and VQ details from SIP Server. |
| EXCP/EXAS | Outgoing | TCP | Any | EXCS | 3640 (default) | Use EXCS REST API. |
| EXCS | Inbound | TCP | 3640 (default) | EXCP/EXAS/ Prometheus | Any | Use EXCS REST API and scrape metrics. |
| EXAS | Inbound | TCP | 3630 (default) | Prometheus | Any | Scrape metrics. |
| EXCP | Inbound | TCP | 3650 (default) | Prometheus | Any | Scrape metrics. |
| EXCS/EXAS/ EXCP | Outgoing | TCP/TLS | Any | Redis | 6379 (default) | Store config data to provide ID mapping services |
| EXCS/EXAS/ EXCP | Outgoing | HTTPS | Any | Genesys Cloud | 443 | Call GC REST API. |

#### Bootstrap Virtual Machine

Bootstrap VM is required to execute EXEC deployment procedure by running a bootstrap Python script. Bootstrap VM specs include:

- Linux OS
- Docker version 20 or later
- Python 3.9+
- pip3 package manager
- SSH connectivity to EXEC VMs

#### Genesys Cloud EX Organization

The EX Organization should be created in Genesys Cloud to be used as a target for data injection by the EXEC components.

An OAuth client with grant type *Client Credentials* created in the EX Org to connect to GC and the *EX Integration* role should be assigned to the OAuth client. This client's credentials will be used by the EXEC components to connect to Genesys Cloud.

#### Engage Environment

EX Engage Connector components operate with Genesys core services version 8.5+ on the back-end. Ensure the following Engage components (Config DB, Stat Server, and SIP Server) are deployed and running. Also, the EXEC components should be able to communicate with Config DB, Stat Server, and SIP Server.

### Optional Environment Elements

- **Monitoring Solution** - EXEC component provides metrics that can be consumed by Prometheus to enable effective monitoring.
- **Observability Solution** - Prometheus-based observability solution (Grafana) to make monitoring and alerting friendly and efficient.
- **Centralized Logging Solution** - Log collection systems (Filebeat/ElasticSearch, Promtail/Grafana Loki, etc) to collect and index logs in a centralized place.
- **Container Monitoring Solution** - third-party container monitoring tools (Portainer) to simplify management and monitoring of the EXEC containers

## Deployment Plan

1. Execute the procedure described in the Preparing Engage Configuration for the EX integration section.
2. Execute the procedure described in the Configuring OAuth Client in the GC Org section.
3. Execute the procedure described in the Configuring hybrid_integration transaction in CME section.

4. Download all EXEC components from Genesys FTP:
   - EXCS (excs-100.0.100.XX.tgz) - EXCS image
   - EXAS (exas-100.0.100.XX.tgz) - EXAS image
   - EXCP (excp-100.0.100.XX.tgz) - EXCP image
   - EXOP (exop-100.0.100.XX.tgz) - bootstrap package
5. Load EXEC images to the local container registry.
   1. Extract tarballs of the EXEC component image from the *.tgz archives.
   2. Deploy tarballs as images to the local container registry using `docker import` command.
   3. Save the paths to the images in the registry (for example myregistry.mycompany.com/exec) to use them later in the deployment process.
6. Execute the procedure described in the Installing EXEC Bootstrap Script section.
7. Execute the procedure described in the Configuring .env file section.
8. Execute the procedure described in the Deploying EXEC section.

## Deployment Procedure

## Preparing Engage Configuration for the EX integration

1. Review Folder Filtering rules and how they are applied to the configuration objects of different types. Set the value of the EXCS environment variable, `SYNC_CUSTOMER_FOLDER_NAME` to the comma-separated list, which includes the names of all CME folders selected to be synchronized to GC.
2. Identify a routing domain to map to the EX Org
   - In this context Engage Routing Domain refers to a set of configuration objects, which create isolated or semi-isolated routing environment. Line of Business (LOB) can be an example of such environment. One LOB can be responsible for processing inbound calls landing on a set of toll free numbers of a contact center. In the Engage contact center such LOB would be configured with a set of dedicated resources: RP and VQ DNs, skills, agents, agent groups, and business attributes. Those are main configuration object, which define the routing domain of a selected LOB.
   - Usually, it is not possible to find a fully isolated LOB in the contact center. It is recommended to start with a small routing domain, observe mapping results for all three injected data types (configuration, agent states, and interactions), and then gradually increase the routing domain by adding more Engage configuration objects to it with the goal of injecting all Engage contact center configuration and activity into the EX Org.
   - Take current EXEC limitations into account while defining the routing domain. For example, consider selecting voice agents serving inbound calls because currently EXEC doesn't support digital interactions. Also, outbound campaign calls cannot be injected into the EX Org. So, if digital, blended, or voice outbound agents are selected their workload wouldn't be properly represented in the EX Org through the injected data, which may impact the quality of the WFM and Gamification services.

3. Skills:
    - Check that CME folders in the Skills unit selected to be synchronized to GC are added to the EXCS environment variable, `SYNC_CUSTOMER_FOLDER_NAME`.
4. Persons:
    - Check that CME folders in the Persons unit selected to be synchronized to GC are added to the EXCS environment variable, `SYNC_CUSTOMER_FOLDER_NAME`
5. Extension DNs
    1. Extension DNs are not mapped to the EX Org but EXCS reads the DNs of this type and stores them into Redis. Those DNs are used by the EXCP for the call event processing.
    2. All Extension DNs used to log in the Persons selected for the EX integration at the previous step should also become a part of the EX integration context.
    3. Review the EXCP Deployment Topologies section and select a SIP Server, which processes contact center voice calls and is panned to be used as a source for the interaction event injection to the EX Org, and identify a Switch this SIP Server uses.
    4. Check that CME folders in the DNs selected to be synchronized to GC are added to the EXCS environment variable, `SYNC_CUSTOMER_FOLDER_NAME`
6. Queues:
    1. EXCS can convert four types of the Engage DNs into the ACD queues in GC EX Org: ACD Queue, Routing Queue, Virtual Queue (VQ), and Routing Point (RP). However, the only supported call distribution model from a queue to an agent is the one where call lands on RP DN, then it is placed on a VQ DN for an agent selection, and then is distributed to a selected agent. From this perspective it is recommended to map Engage queue DNs of only two types to the EX Org:
        1. RP: select all RP DNs, which belong to a routing domain being mapped
        2. VQ: select only routing VQ DNs, which are used by the URS for target selection, do not map reporting VQ DNs
    2. Select a SIP Server, which processes contact center voice calls and is panned to be used as a source for the interaction event injection to the EX Org, and identify a Switch this SIP Server uses.
    3. Check that CME folders in the DNs configuration unit under the selected switch selected to be synchronized to GC are added to the EXCS environment variable, `SYNC_CUSTOMER_FOLDER_NAME`
    4. Review the EXCP Deployment Topologies section and select a SIP Server, which processes contact center voice calls and is panned to be used as a source for the interaction event injection to the EX Org, and identify a Switch this SIP Server uses.
    5. Check that CME folders in the DNs configuration unit under the selected switch that has RP and VQ DNs selected to be synchronized to GC are added to the EXCS environment variable, `SYNC_CUSTOMER_FOLDER_NAME`
7. Agent Groups
    - Select a folder at the Agent Group configuration unit to be synced to the GC.
    - Identify agent groups (AG) and virtual agent groups (VAG) used for routing or reporting by the Persons selected to be mapped to the EX Org.
    - Link selected AGs and VAGs to their corresponding queue DNs by applying the following configuration to the properties of these groups:
        - Annex / hybrid / WFM_queue_number = <ROUTING_VQ_DN_NAME>

8. Supervisors
    - Engage Supervisors should be marked as shown below to get a Supervisor role assigned in the EX Org:
        - Person has 'Is Agent' checked
        - Person has Annex/htcc/roles list containing a 'Supervisor' as one of its elements
    - If customers use WWE/GWS, then their supervisors should already be configured like this. In other cases this configuration should be added.
9. Review all objects of all types under CME folders selected to be synchronized to GC for the case-sensitivity conflicts. Refer to the Case-Sensitivity of Object Names section. Resolve those conflicts if found.

## Configuring OAuth Client in the GC Org

EXCS requires an OAuth client with grant type *Client Credentials* created in the EX-Org to connect to GC. Create a new OAuth client and assign the *EX Integration* role to the OAuth client. Refer to Create an OAuth client for more information.

## Configuring hybrid_integration transaction in CME

### EX Org parameters required for the hybrid_integration transaction configuration

The following configuration parameters should be fetched from EX Org to provision the hybrid_integration transaction in the Engage CME. Configuration Alias is used to refer to the value of the EX Org parameter.

| Configuration Alias | EX Org Parameter Path | Parameter Description |
|---|---|---|
| CFG_ALIAS_EX_ORG_SHORT_NAME | Organization Setting / Organization Details / Short Name | Short name of the EX Org |
| CFG_ALIAS_EX_ORG_ID | Organization Setting / Organization Details / Advanced / Organization ID | EX Org Organization ID |
| CFG_ALIAS_EX_ORG_OAUTH_CLIENT_ID | Integrations / OAuth / <OAUTH_CLIENT_NAME> / Client Details / Client ID | OAuth client ID |
| CFG_ALIAS_EX_ORG_OAUTH_CLIENT_SECRET | Integrations / OAuth / <OAUTH_CLIENT_NAME> / Client Details / Client Secret | OAuth client secret |

### Transaction Object

Transaction object hybrid_integration contains EXEC configuration. It should be configured as:

- Path to the transaction object:
    - for the single-tenant Engage deployment the transaction object should be created in the

Transactions configuration unit in the Resources structure

- for the multi-tenant Engage deployment there must be a separate transaction object under each of the tenant. Each transaction should point at a dedicated EX Org. EX Orgs cannot be shared by multiple Engage tenants.

- Name: *hybrid_integration*
- Type: *List*

## Transaction Annex Folders

| Folder | Parameter Name | Value | Description |
|---|---|---|---|
| general | organization_sname | CFG_ALIAS_EX_ORG_SHORT_NAME | Short name of the EX Org. |
| | organization_id | CFG_ALIAS_EX_ORG_ID | EX Org Organization ID. |
| | base_service_url | | Base service URL used for framing the REST API calls to the EX Org. |
| | base_auth_url | | Base auth URL used for logging into the EX Org. |
| | region | GENESYS_CLOUD_REGION_URL | Service URL for the particular region. |
| config_sync | client_id | CFG_ALIAS_EX_ORG_OAUTH_CLIENT_ID | OAuth client ID. |
| | client_secret | CFG_ALIAS_EX_ORG_OAUTH_CLIENT_SECRET | OAuth client secret. |
| | skill_level_max_value | ENGAGE_MAX_SKILL_LEVEL | Maximum skill level value set for Engage agents to match the skill proficiency in Genesys Cloud. See ACD Skills for more information. |
| | skill_level_min_value | ENGAGE_MIN_SKILL_LEVEL | Minimum skill level value set for Engage agents to match the skill proficiency in Genesys Cloud. See ACD Skills for more information. |
| agent_state_sync | client_id | CFG_ALIAS_EX_ORG_OAUTH_CLIENT_ID | OAuth client ID. |
| | client_secret | CFG_ALIAS_EX_ORG_OAUTH_CLIENT_SECRET | OAuth client secret. |
| conversation_provider | client_id | CFG_ALIAS_EX_ORG_OAUTH_CLIENT_ID | OAuth client ID. |
| | client_secret | CFG_ALIAS_EX_ORG_OAUTH_CLIENT_SECRET | OAuth client secret. |
| | sta_default_programid | <DEFAULT_STA_PROGRAM_ID> | Program UUID for topic detection |
| | sta_default_language | <DEFAULT_STA_LANGUAGE> | ISO 639-1 two letter language code + '-' + ISO 3166-1 alpha-2 two letter country code |

| Folder | Parameter Name | Value | Description |
|---|---|---|---|
| | | | string. Default language to use for transcription |
| | password | CFG_ALIAS_EX_ORG_OAUTH_CLIENT_SECRET | OAuth client secret. |

## Create a genesys user on Virtual Machines

It is mandatory to create 'genesys' user on all the Virtual Machines including Bootstrap Virtual Machine. This step ensures seamless, password-free deployment of EXEC components onto their corresponding Virtual Machines.

## Installing EXEC Bootstrap Script

### Important

You should always deploy EX Engage Connector using the bootstrap script.

Perform the following items before deploying EX Engage Connector:

1. Move EXOP bootstrap package downloaded from the FTP (for example exop-100.0.100.XX.tgz) to the Bootstrap VM and place it under **/usr/home/genesys/exec**.
2. Go to the installation directory where the bootstrap script is copied and extract the package in the same folder.
3. Under the installation directory, verify if the following files are available:
   - **.env**
   - **bootstrap.py**
   - Docker Compose files for EXCS, EXCP, and EXAS
   - **alerts.yml**
   - **prometheus.yml**
4. Make sure the file is executable: `chmod +x bootstrap.py`
5. Run `./bootstrap.py --help` and verify if the following screen is displayed:

```
Welcome to EXEC deployment bootstrap script. Following actions are supported

init                                      - Initialize EXEC environment
start                                     - Start EXEC services
stop                                      - Stop EXEC services
restart                                   - Restart EXEC services
update                                    - upgrade/downgrade EXEC services
rollback                                  - rollback EXEC service
```

## Configuring .env file

The **.env** file contains the description of the EXEC Docker environment. Configure all mandatory parameters to describe the environment where EXEC components are planned to be deployed. Bootstrap script converts data from this file into the Docker compose yml configuration files.

You can specify parameters for the deployment by overriding the default values in the .env file. See the Parameters table for a full list of overridable values.

### Common Parameters

| Parameter Name | Description |
|---|---|
| EXEC_DOCKER_REPOSITORY * | Jfrog Artifactory Edge repository for pulling EXEC images. |
| EXEC_INFRA_DOCKER_REPOSITORY | Docker repo for Redis, grafana, and Prometheus containers, if empty - docker hub will be used. |
| EXEC_HOST_LOGS_VOLUME_PATH * | Host path which is mounted inside the container to store logs. |
| EXEC_ORG_ID * | EX Org ID is taken from the following path in the GC UI Admin view:<br>Organization Settings / Organization Details / Advanced / Organization ID |
| EXEC_TENANT_ID * | Unique ID to identify the Engage tenant. DBID of the Tenant configuration object in the Engage Configuration DB can be used. |
| EXEC_REDIS_HOST * | IP Address of the Redis proxy in case of enterprise Redis and IP Address of Redis server in case of Lab environment |
| EXEC_REDIS_PORT * | Port of Redis server |
| EXEC_REDIS_SECURE | If the Redis server is password protected via the requirepass option. |
| EXEC_REDIS_PASSWORD | Do not configure the password in the .env file, provide it as input when you run the bootstrap script |
| A ***** indicates mandatory fields. | |

### EXCS Parameters

| Parameter Name | Description |
|---|---|
| EXCS_VM_HOST * | IP Address of VM machine where CS will be deployed |
| EXCS_PORT * | Config Sync Service port |

| Parameter Name | Description |
|---|---|
| EXCS_TAG * | Config Sync Service Image Version |
| EXCS_CONFIG_DB_HOST * | IP Address of Config Database. |
| EXCS_CONFIG_DB_PORT * | Port Number of Config Database |
| EXCS_CONFIG_DB_USER * | Config Database Username |
| EXCS_CONFIG_DB_SSL* | Config Database SSL connectivity |
| EXCS_CONFIG_DB_PASSWORD * | Do not configure the password in the .env file, provide it as input when you run the bootstrap script |
| EXCS_CONFIG_DB_NAME * | Config Database name |
| EXCS_CONFIG_DB_TYPE * | Config Database Type. Supported Values are `postgresql`, `oracle`, or `mssql`. Default value is `postgresql`. |
| EXCS_SYNC_CUSTOMER_OBJECTS_ONLY | When set to *true*, EXCS only synchronizes objects under customer folder hierarchy to Genesys Cloud. If this environment variable is set to *true*, then environment variable SYNC_CUSTOMER_OBJECTS_ONLY must contain a comma-separated list of folders to be synchronized to the GC. See Folder Filtering section for more information. Default is *false*, i.e. all objects of the supported types are synchronized.<br>Values: true/false (default) |
| EXCS_SYNC_CUSTOMER_FOLDER_NAME | When SYNC_CUSTOMER_OBJECTS_ONLY is set to "true", , set this EXCS environment variable to a comma-separated list of the CME folder names. EXCS searches those folders in four configuration units, Persons, Skills, DNs, and Agent Groups, and syncs configuration objects found in those folders to GC. If a listed folder has subfolders, then all subfolders are synchronized to the GC too. |
| EXCS_LOG_PATH | File system path for logs. If not provided, logs will be directed to stdout. |
| EXCS_LOG_FILENAME * | File name for logs. If LOG_FILENAME is specified, this environment variable further describes the file name.<br>Example: 'excs-<TENANT_ID>.%Y%m%d_%H%M%S_%L.log' |
| EXCS_LOG_LEVEL | Defines the EXCS log level. Values: trace, debug, info, warn, error, fatal |
| EXCS_GC_QUEUE_NAME_SUFFIX * | Set it to *-Engage-external*. This suffix is needed to distinguish Queue objects synced from Engage environment. |
| A ***** indicates mandatory fields. | |

## EXAS Parameters

| Parameter Name | Description |
|---|---|
| EXAS_HOST * | IP Address of VM machine where AS will be deployed |
| EXAS_PORT * | Agent Sync Service port |
| EXAS_TAG * | Agent Sync Service Image Version. |
| EXAS_STAT_SERVER_PRIMARY_HOST * | Primary Stat Server's host IP/FQDN |
| EXAS_STAT_SERVER_PRIMARY_PORT * | Primary Stat Server's port number. |
| EXAS_STAT_SERVER_BACKUP_HOST * | Backup Stat Server's host IP/FQDN |
| EXAS_STAT_SERVER_BACKUP_PORT * | Backup Stat Server's port number. |
| EXAS_LOG_PATH | File system path for logs. If not provided, logs will be directed to stdout |
| EXAS_LOG_FILENAME * | File name for logs. If LOG_FILENAME is specified, this environment variable further describes the file name.<br>Example: 'exas-<TENANT_ID>.%Y%m%d_%H%M%S_%L.log' |
| EXAS_LOG_LEVEL | Defines the EXAS log level. Values: trace, debug, info, warn, error, fatal |
| A ***** indicates mandatory fields. | |

## EXCP Parameters

| Parameter Name | Description |
|---|---|
| EXCP_1_VM_HOST_PAIR * | VM Host IPAddress list where Conversation Provider service will be running |
| EXCP_1_PORT * | Conversation Provider Service port |
| EXCP_1_TAG * | Conversation Provider Container version |
| EXCP_1_VOICE_SERVER_VQ * | A list of SIP Server host:port for receiving Virtual Queue Events. EXCP should connect to the default port. Format: primary1:port/backup1:port |
| EXCP_1_VOICE_SERVER_AGENT * | A list of SIP Server host:port for receiving Extension DN Events<br>. EXCP should connect to the default port. Format: primary1:port/backup1:port |
| EXCP_1_VOICE_SERVER_CALL * | A list of SIP Server host:port for receiving Call Monitoring Events. EXCP should connect to the default port. Format: primary1:port/backup1:port |
| EXCP_1_LOG_PATH | File system path for logs. If not provided, logs will be directed to stdout |
| EXCP_1_LOG_FILE_NAME * | File name for logs. If EXCP_1_LOG_FILE_NAME is specified, this environment variable further |

| Parameter Name | Description |
|---|---|
| | describes the file name.<br>Example: 'excp-.%Y%m%d_%H%M%S_%L_${TENANT_ID}_${HOSTNAME}.log' |
| EXCP_1_LOG_LEVEL | Defines the EXCP log level. Values: trace, debug, info, warn, error, fatal |
| A * indicates mandatory fields. | |

## Deploying EX Engage Connector

To set up the EX Engage Connector services, run the `init` command: `./bootstrap.py --action init`.

The `init` script performs the following actions:

1. Prompt for following passwords:
   - DB password - The password for the Config database.
   - Redis password - The password for the Redis server (if the **redis-secure** parameter is set to *true*).
   - Config Server password - The password for the Config Server.
2. Create a **/tmp/.env** file containing the provisioning information along with password information.
3. Extract the VM address for each service and copy the **docker-compose** file of the respective service to their respective VMs.
4. Copy the **/tmp/.env** file to the list of VMs on which the connector services need to be installed and delete the **/tmp/.env** file.

The docker-compose files and .env files are stored under the **exec** directory in the home path of the Genesys user.

## Run EXEC Services for First Time

Once the EXEC service environment is initialized, the EXEC services should be started in the following order:

1. Start the EXCS Service by running `./bootstrap.py --action start --service excs`.
2. Check the health of EXCS Service and once it shows as healthy then it signifies that EXCS service is running fine.

   ```
   curl -v http://<EXCS_VMHOST_IPADDRESS>:3640/health
    200 OK with dependency status
   ```
3. Wait for all the Engage Objects to be synced to the GC side. Initial sync-up of contact centers with 1000 agents will be around 30 - 40 minutes.
4. To check if all the Engage Objects are synced to GC successfully, run the status command to see if the initialization status is "success".

   ```
   curl http://<EXCS_VMHOST_IPADDRESS>:3640/status
   ```

```
 {"status":{"code":0,"message":"Latest database updates for
orgId:6662cb10-0d4d-4710-b979-db69011e66fd have been successfully processed on
2023-07-11T10:23:08.062Z","corrId":"bd4c0e9a-
ba02-491c-93e5-6fd1e3a7e51e"},"data":{"syncStatus":{"timeStamp":"2023-07-11T10:23:08.062Z","status":"Succes
```

5. Start the EXAS Service by running `./bootstrap.py --action start --service exas`.
6. Check the health of EXAS Service; once it shows as healthy, it signifies that EXAS service is running fine.

```
curl -v http://<EXAS_VMHOST_IPADDRESS>:3630/health
 200 OK with dependency status
```

7. Start the EXCP Service by running `./bootstrap.py --action start --service excp --pair 1`.
8. Check the health of EXCP Service; once it shows as healthy, it signifies that EXCP is running fine.

```
curl -v http://<EXCP_VMHOST_IPADDRESS>:3650/health
 200 OK with dependency status
```

## Starting EXEC Services

The EXEC services can be started/stopped/restarted using the below commands:

- `./bootstrap.py --action start` - starts all the EXEC services in their respective VMs configured.
- `./bootstrap.py --action start --service=exas` - starts the EXAS service in its respective VM
- `./bootstrap.py --action start --service=excp --pair 1` - starts the EXCP service in both the VMs belonging to excp pair 1
- `./bootstrap.py --action start --service=excp --pair 1 --host 1` - starts the EXCP service in the VM host 1 belonging to excp pair 1

## Stopping or Restarting EXEC Services

All services or a specific service or a specific instance of a service can be stopped using the `./bootstrap.py --action stop` command. The `./bootstrap.py --action restart` command can be used to restart all services or a specific service or a specific instance of a service.

# Recording Deployment

## Prerequisites

### No-media Integration Deployment

EXEC deployment for No-media Integration should be up and running before the configuration required for the Recording Injection is added to the environment. See No-media Deployment for more information.

### Mandatory Environment Elements

#### WebDAV

WebDAV is required only for the Recording Injection. EX Engage Connector supports any standard WebDAV server. Customers should plan to use 30 GB storage for WebDAV server for contact centers with 1000 concurrent recording calls.

#### HTTP Load Balancer

HTTP Load Balancer is required only for the Recording Injection. EX Engage Connector supports any standard HTTP load balancer.

**Important**

Genesys does not deploy and operate databases, WebDAV servers, or load balancers in on-premises deployments. It is the responsibility of the customer. In a production deployment, data store components (Redis) and WebDAV must be deployed outside of the Docker network and managed by the customer's DBA team. The customer's DBA team is also responsible for ensuring that the data store components are configured with the appropriate scalability, resiliency, and data protection (backups).

#### Media Control Platform (MCP)

The minimal version required for MCP is 9.0.052.00 for the EXEC Recording Injection feature.

### Virtual Machines for the EXRP Component

- A minimum of 2 VMs (N+1) for EX Engage Connector Recording Provider (EXRP)

The EXEC VM must sync the system time at least once every 24 hours as the communication between the EXEC services and Genesys Cloud are very time-sensitive.

All VMs running EX Engage Connector components should belong to the same local network segment and be interconnected so that all components can communicate over the network. EXEC components can use either FQDNs or IP addresses to establish communication with each other. The connectivity table lists both incoming and outgoing connections required by the EXEC components.

| Service | Direction | Protocol | Local Port | Remote Peer | Remote Peer Port | Purpose |
|---|---|---|---|---|---|---|
| MCP | Outgoing | HTTP | ANY | WebDAV | 80 (default) | Post the recording files to WebDAV Server using HTTP requests. |
| MCP | Outgoing | HTTP | ANY | HTTP Load Balancer | HTTP_LB_EXCP_PORT | Deliver recording metadata using HTTP POST requests. |
| HTTP Load Balancer | Outgoing | HTTP | ANY | EXCP | 3650 (default) | HTTP POST proxied by HTTP Load Balancer for delivering recording metadata. |
| EXCP | Outgoing | HTTP | Any | HTTP Load Balancer | HTTP_LB_EXRP_PORT | HTTP POST to HTTP Load Balancer for interaction-end with recording metadata. |
| HTTP Load Balancer | Outgoing | HTTP | ANY | EXRP | 8080 (default) | HTTP POST proxied by HTTP Load Balancer to EXRP for interaction-end with recording metadata. |
| EXRP | Outgoing | TCP | Any | WebDAV | 80 (default) | HTTP request by EXRP to fetch recording. |
| EXCP | Outgoing | TCP | Any | GIR-RWS | 443 (default) | HTTPS GET Request by |

| Service | Direction | Protocol | Local Port | Remote Peer | Remote Peer Port | Purpose |
|---|---|---|---|---|---|---|
| | | | | | | EXCP to GIR to retrieve recording metadata. (Applicable only for GIR deployments) |

## High-Level Deployment Plan

1. Refer the sample deployment procedure for WebDAV & Load Balancer.
2. Execute the procedure described in the Configuring hybrid_integration transaction in CME section.
3. Execute the procedure described in Configuring record IVR profile inside Voice Platform Profiles under CME section.
4. Download the EXRP image from Genesys FTP:
   - EXRP (exrp-100.0.100.XX.tgz) - EXRP image
5. Load the EXRP image to the local container registry.
   - Extract the tarball of the EXRP component image from the *.tgz archives.
   - Deploy the tarball as an image to the local container registry using docker import command.
   - Save the paths to the images in the registry (for example myregistry.mycompany.com/exec) to use them later in the deployment process.
6. Execute the procedure described in the Configuring .env file section.
7. Execute the procedure described in the Deploying EXRP section.

## Deployment Procedure

### Reference deployment for WebDAV server and load balancer

#### Deploying Apache WebDAV Server for Recording Injection

1. Deploy WebServer that supports the WebDAV module and ensure that the WebDAV module is installed and enabled as well.
2. Create the directory to keep the recording files, and provide the required permission for the webserver user to access those recording files.

3. Create the user and password and configure it for authentication of the recording folder.
4. Open the firewall for the port used by the WebDAV server.

The sample installation and configuration for the Apache2 WebDAV server is described below:

1. Install Apache Web Server. The installation process for WebDAV Server on an Ubuntu machine can be found at How to install Apache2 page.
2. Check if the DAV module is installed and enabled in the machine.
3. Edit the **/etc/apache2/apache2.conf** file, and append the following to the end of the file:

```
Alias /recordings /mnt/recordings
 <Directory /mnt/recordings>
    Options Indexes MultiViews FollowSymLinks
    EnableSendfile off
    AllowOverride None
    Order allow,deny
    allow from all
 </Directory>
 <Location "/recordings">
    DAV On
    AuthType Basic
    AuthName "user"
    AuthUserFile /var/www/htpasswd
    Require valid-user
 </Location>
```

4. Open the firewall. Ensure that the default incoming port 80 is open.
5. Create the directory to keep the recording files, and set the permission to Apache, using the following commands

```
mkdir /mnt/recordings
 chown www-data:www-data /mnt/recordings
```

6. Create an Apache HTTP Server User for httpd, and configure the password. The following example creates a user called *user*:

```
htpasswd -cm /var/www/htpasswd user
```

7. Configure the Apache HTTP server start on boot up (and start it now) using the following commands:

```
sudo systemctl enable apache2
```

8. Test the Apache HTTP Server installation:
   - Upload a sample **hello.world** file to the Apache HTTP Server using the following command:

     ```
     curl -T hello.world -u user:password http://<WEBDAV_URL>/hello.world  //
      WEBDAV_URL: <WEBDAV_VM_ADDRESS>:<WEBDAV_PORT>/recordings
     ```

   - Using a browser, open the http://<WEBDAV_URL>/hello.world URL. The browser will request user credentials.

## Deploying HTTP Load Balancer for Recording Injection

1. Deploy HTTP Load Balancer in a different VM from the EXEC service VMs.
2. Configure the upstream instances for EXCP and EXRP that will be used for load balancing the request.

3. Configure the port that the HTTP Load Balancer will listen for both EXCP and EXRP requests.

The sample installation and configuration for the NGNIX HTTP Load Balancer is described below:

1. Install NGNIX Proxy in a different VM which is accessible to EXEC VM. See Installing nginx for information on installing nginx..
2. Edit the **nginx-upstreams.conf** under the **/etc/nginx/** directory in NGINX VM.

```
upstream excp {
    server <EXCP_1_VM_HOST_PAIR_IPADDR1>:<EXCP_1_PORT> max_fails=0;
    server <EXCP_1_VM_HOST_PAIR_IPADDR2>:<EXCP_1_PORT> max_fails=0;
}
upstream exrp {
    server <EXRP_VM_HOST_IPADDR1>:<EXRP_PORT> max_fails=0;
    server <EXRP_VM_HOST_IPADDR2>:<EXRP_PORT> max_fails=0;
    server <EXRP_VM_HOST_IPADDR3>:<EXRP_PORT> max_fails=0;
}
```

3. Edit the **nginx.conf** file under **/etc/nginx/** directory in NGINX VM.

```
events {
    worker_connections 1000;
}

http {
    include nginx-upstreams.conf;

    # conversation-provider
    server {
        listen <HTTP_LB_EXCP_PORT>; #eg: listen 3650

        location / {
            proxy_pass http://excp/;
            proxy_next_upstream_tries 3;
        }
    }

    # recording-provider
    server {
        listen <HTTP_LB_EXRP_PORT>; #eg: listen 8080

        location / {
            proxy_pass http://exrp/;
            proxy_next_upstream_tries 3;
        }
    }
}
```

## Configuring hybrid_integration transaction in CME

### Common Configuration

This section contains configuration required for both third-party recorders and GIR.

### Transaction Object

Transaction object hybrid_integration contains EXEC configuration. It should be configured as:

- Path to the transaction object:
  - for the single-tenant Engage deployment the transaction object should be created in the Transactions configuration unit in the Resources structure
  - for the multi-tenant Engage deployment there must be a separate transaction object under each of the tenant. Each transaction should point at a dedicated EX Org. EX Orgs cannot be shared by multiple Engage tenants.
- Name: *hybrid_integration*
- Type: *List*

#### Transaction Annex Folders

| Folder | Parameter Name | Value | Description |
|---|---|---|---|
| conversation_provider | recording_enabled | True | Enable Recording Support. Default Value: False. |
| | sta_default_programid | <DEFAULT_STA_PROGRAM_ID> | Program UUID for topic detection |
| | sta_default_language | <DEFAULT_STA_LANGUAGE> | Represents the main Contact Center language. The ISO 639-1 two letter language code + '-' + ISO 3166-1 alpha-2 two letter country code string. The language should be chosen from the list of languages supported by Genesys Cloud (see *Native voice transcription* section). |
| | sta_userdata_language_key | <STA_USERDATA_LANGUAGE_KEY> | Key name for the `languageId` sent in user data from Routing Point, for example, "language" or "language_key". |
| recording-uploader.exec.transientStorage | type | WebDAV | The value must be WebDAV. |
| | username | WEBDAV_USERNAME | The username used for downloading recording files from WebDAV. |
| | password | WEBDAV_PASSWORD | Password used for downloading recording files from WebDAV. |
| | url | WEBDAV_URL | WebDAV URL from |

| Folder | Parameter Name | Value | Description |
|---|---|---|---|
| | | | where the recording files are downloaded. |
| recording-uploader.exec.genesysCloud | client_id | CFG_ALIAS_EX_ORG_OAUTH_CLIENT_ID | OAuth client ID. |
| | password | CFG_ALIAS_EX_ORG_OAUTH_CLIENT_SECRET | OAuth client secret. |

## Third-party recorder integration

For third-party recorders, no additional configuration is required.

## GIR Recorder integration

For integrating with GIR, add the following parameters to the transaction object.

| Folder | Parameter Name | Value | Description |
|---|---|---|---|
| conversation_provider | gir_orgId | *<Engage_Contact_Center_ID>* | The Engage contact center ID to be used in request to GIR RWS. This value is same as the value of `rp.defaultccid` configured in RWS. |
| | gir_user | user1 | GIR RWS user<br>Important<br>This is Ops user from RWS deployment specified under Ops Account section in Interaction Recording |
| | gir_password | *<RWS_PASSWORD>* | GIR RWS user password |
| | gir_metadata_fetch_delay | 10 | Determines the initial delay (in minutes) for trying out the first GIR request after the call completes. |
| | gir_metadata_fetch_interval | 1380 | Determines the time (in minutes) after which the retry of failed GIR request has to be made. |
| | gir_metadata_fetch_duration | 1440 | Determines how long (in minutes) the GIR request has to be retried. |

## Configuring record IVR profile inside Voice Platform Profiles under CME

### Third-party recorder integration

Engage deployments with third-party recording solutions use recording client *recdest* parameters for connecting to third-party recorders. It is expected that parameter *callrec_dest* is not used for the third-party recorder integration and can be utilized to direct recording metadata to EXEC.

For recording injection functionality, MCP posts the recording file into the WebDAV server and also it posts the recording metadata to the EXCP service. As part of this process, the recording IVR profile inside the Voice Platform Profile must be configured.

| Folder | Parameter | Value | Description |
|---|---|---|---|
| gvp.general | service-type | `record` | IVR profile type: always set to *record*. |
| gvp.service-parameters | recordingclient.callrec_dest | `fixed,http://<HTTP_LB_HOST_IPADDR>:<HTTP_LB_EXCP_PORT>/a v1/recording` | HTTP URL for posting recording metadata; pointed at the NGINX front-ending all EXCP instances. |
| gvp.service-parameters | recordingclient.httpauthorization2 | `fixed,<WEBDAV_USERNAME>:<WEBDAV_PASSWORD>` | Credentials to post recordings to the intermediate WebDAV recording storage. |
| gvp.service-parameters | recordingclient.recdest2 | `fixed,http://<WEBDAV_URL>/` | HTTP URL of the intermediate WebDAV recording storage. |
| gvp.service-parameters | recordingclient.type2 | `fixed,audio/opus` | Recording file format. Currently only the opus format is supported. |

### GIR Recorder integration

#### Storing All Engage Recordings in WebDAV

See Storing All Engage Recordings in WebDAV for the description of this type of EXEC integration.

To configure, follow the following:

1. Identify the recording profile where GIR and GIA destinations are configured.
2. Comment out the following parameters in the **gvp.service-parameters** folder, which are currently pointed at GIA:
    - `service-parameters.httpauthorization2`
    - `service-parameters.recdest2`
    - `service-parameters.type2`

3. Create instances of the three parameters in the recording IVR profile and configure them to point at EXEC services as described in the third-party recorder configuration.

### Storing Only Engage Sync-Scope Recordings in WebDAV

See Storing Only Engage Sync-Scope Recordings in WebDAV for the description of this type of EXEC integration.

> **Important**
>
> For more information on configuration related to sync-scope recordings, please reach out to Genesys Customer Care.

## Configuring .env file

The **.env** file contains the description of the EXEC docker environment. Configure all mandatory parameters to describe the environment where EXEC components are planned to be deployed. Bootstrap script converts data from this file into the docker compose yml configuration files.

You can specify parameters for the deployment by overriding the default values in the .env file. See the Parameters table for a full list of overridable values.

### Common Configuration

#### EXCP Parameters

| Parameter Name | Description |
|---|---|
| EXCP_RECORDING_PROCESSOR_PROXY_HOST* | HTTP Load Balancer Proxy's IP address or FQDN which proxies the request to EXRP instances. |
| EXCP_RECORDING_PROCESSOR_PROXY_PORT* | HTTP Load Balancer Proxy's Port which proxies the request to EXRP instances. |
| A ***** indicates mandatory fields. | |

#### EXRP Parameters

| Parameter Name | Description |
|---|---|
| EXRP_VM_HOST_LIST * | Comma Separated VM IP Address list where |

| Parameter Name | Description |
|---|---|
| | Recording Provider service will be running. |
| EXRP_PORT * | Recording Provider Service Port |
| EXRP_TAG * | Recording Provider container version |
| EXRP_LOG_LEVEL | Defines the EXRP log level. Values: trace, debug, info, warn, error, fatal |
| EXRP_CONFIG_SERVER_USER * | Config Server Username |
| EXRP_CONFIG_SERVER_PASSWORD * | Config Server Password |
| EXRP_CONFIG_SERVER_APPLICATION * | Config Server Application |
| EXRP_CONFIG_SERVER_PRIMARY_HOST * | IP Address of Config Server |
| EXRP_CONFIG_SERVER_PRIMARY_PORT* | IP Port of Config Port |
| A * indicates mandatory fields. | |

For third-party recorders, no additional configuration is required.

For GIR Integration, configure the GIR RWS URL in EXCP using the environment variable **EXCP_GIR_HOST**.

| Parameter Name | Description |
|---|---|
| EXCP_GIR_HOST* | Interaction Recording Web service *FQDN/host:port* |
| A * indicates mandatory fields. | |

## Deploying EXRP service

To deploy the EXRP service, run the procedure described in the Deploying EXEC section.

### Starting EXEC services

1. Start the EXRP Service by running `./bootstrap.py --action start --service exrp`.
2. Check the health of EXRP Service and once it shows as healthy then it signifies that EXRP is running fine. Repeat the same for all the EXRP service instances.

```
curl -v http://<EXRP_VMHOST_IPADDRESS>:8080/health
 200 OK with dependency status
```

3. Re-start the EXCP Service by running `./bootstrap.py --action restart --service excp`.
4. Check if the EXCP Service is healthy and ready to support the recordings by running below command:

```
curl -X POST http://<EXCP_VMHOST_IPADDRESS>:3650/api/v1/recording
 {"status":{"code":40402,"message":"CallUUID not found"}}
```

5. Start the NGINX with the Configurations for EXCP and EXRP.
6. Check whether NGINX properly forwards requests to both EXCP and EXRP properly.

```
curl -X POST http://<NGINX_IPADDRESS>:<EXCP_PORT>/api/v1/recording
 {"status":{"code":40402,"message":"CallUUID not
 found","corrId":"2fdd55fe-8b16-41ab-84c9-2c5cc42d4eee"}}
```

```
curl http://<NGNIX_IPADDRESS>:<EXRP_PORT>/health
HTTP Status Code = 200
```

# Maintenance

Once you have deployed and started the EX Engage Connector service, you can upgrade the EXEC services to a new version, monitor the status of containers, and access logs.

## Upgrade EXEC Services

- Ensure that the new container version of EXEC service is available in the docker registry.
- Update the _TAG parameters in the .env file with the new version of the EXEC service. You can update all or individual EXEC services:

```
EXCS_TAG=100.0.100.0001
EXAS_TAG=100.0.100.0001
EXCP_1_TAG=100.0.100.0001
```

- Run `./bootstrap.py --action update`. This command will download the .env and docker-compose files from the remote EXEC Service VMs.
    - The script compares the new **.env** file parameters and **docker-compose** files with the files present in remote EXEC service VMs.
    - If the files differ, the bootstrap script will generate backups of the **remote.env** and **docker-compose** files in the respective service VMs under the backup directory.
    - The backed-up filenames will be appended with the current date in the format of "_<ddmmyyyy>". For instance, .env_23052023 and docker-compose-EXAS.yml_23052023.
    - Update parameters downloaded remote .env file to add any newly added parameters or remove any deprecated parameters, and reupload the .env file into the respective service VMs.
    - Update parameters downloaded remote docker-compose files to add any newly added parameters or remove any deprecated parameters, and reupload the docker-compose file into the respective service VMs
- Restart the exec services using start/stop or restart commands for the changes to take effect.

You can also update individual services by using the `--service` option as shown below:

- `./bootstrap.py --action update --service=exas` - Update the EXAS service.
- `./bootstrap.py --action update --service=excs` - Update the EXCS service.
- `./bootstrap.py --action update --service=excp --pair 1` - Update the EXCP service for excp pair 1.
- `./bootstrap.py --action update --service=excp --pair 1 --host 1` -Update the EXCp service running on VM host 1 belonging to excp pair 1.

## Rollback EXEC Services

After upgrading the EXEC service to a newer version or updating any parameters and if the service is not running as expected then the admin can roll back the service.

- Run `./bootstrap.py --action rollback`. The bootstrap script will find the latest backup of the EXEC service taken during the update operation.
    - The script will copy the .env and docker-composer files from the backup folder into the **exec** directory.
- Restart the EXEC services using start/stop or restart commands for the changes to take effect.

You can also rollback individual services by using the --service option as shown below:

- `./bootstrap.py --action rollback --service exas` - Roll back .env and docker-compose file to the pre-update state for the exas service.
- `./bootstrap.py --action rollback --service excs` - Roll back .env and docker-compose file to the pre-update state for the excs service.
- `./bootstrap.py --action rollback --service excp --pair 1` - Roll back .env and docker-compose file to the pre-update state for the for excp pair 1
- `./bootstrap.py --action rollback --service excp --pair 1 --host 1` - Roll back .env and docker-compose file to the pre-update state for the host 1 belonging to excp pair 1

### Cleaning up recordings not injected to GC

In some scenarios, recordings may fail to be injected into the GC. Those recordings are accumulated in two places:

- Recording MCP hosts: in the directory configured in the MCP option *msml.record.irrecoverablerecordpostdir*.
- WebDAV directory where MCP uploads the recordings.

It is recommended that the customers automate a cleanup of those storages.

# Observability

EX Engage Connector provides the following observability features:

- Logging - Logs are available as JSON files.
- Monitoring - Container and service monitoring are available.

# Logging

EX Engage Connector Services writes logs in JSON format to the container's STDOUT. Those logs are expected to be consumed by the log monitoring system (for example filebeat) and moved to a log processing system where they can be queried and analyzed (for example Elastic Search).

EXCS can also write logs to a file on a disk. This functionality is enabled by the environment variables <SERVICE>_LOG_PATH and <SERVICE>_LOG_FILE_NAME.

Environment variable <SERVICE>LOG_LEVEL will be used to define the logging level.

| **Name** | **Default** | **Mandatory** | **Description** |
|---|---|---|---|
| Name | Default | Mandatory | Description |
| LOG_CONSOLE_ENABLE | true | NO | Specifies if log to console is enabled (default) |
| LOG_FILE_ENABLE | false | NO | Specifies if log to file is enabled |
| LOG_FILE_LEVEL LOG_CONSOLE_LEVEL | INFO | NO | Log levels<br>• OFF - A special level to use in configuration in order to turn off logging.<br>• FATAL - A critical service failure or complete inability to service requests of any kind.<br>• ERROR - A significant disruption in a request or the inability to service a request.<br>• WARN - A non-critical service error or problem that may not require immediate correction.<br>• INFO - Service lifecycle events or important related very low-frequency information.<br>• DEBUG - Messages |

| Name | Default | Mandatory | Description |
|---|---|---|---|
| | | | that convey extra information regarding lifecycle or non-request-bound events, useful for debugging.<br>• TRACE - Messages that convey extra per-request debugging information that may be very high frequency.<br>• ALL - A special level to use in configuration to turn on logging for all messages, including custom levels. |
| LOG_FILE_MAX_BACKUP_INDEX | 20 | NO | Specifies maximum number of log files to keep on disk before rotating. |
| LOG_FILE_PATH | /rup/logs/ccerp.log | NO | Specifies file to log to. Note that file is rotating after 50Mb and suffix is added to file at rotation. For example rotated file might be named ccerp.log.2020-05-19.1 |
| LOG_FILE_SIZE | 50M | NO | Specifies size of log file after which log files should rotate. |
| LOG_JSON_ENABLE | true | NO | Specifies logging using JSON format |

# Monitoring

EX Engage Connector supports two types of monitoring:

- Container Monitoring
- Service Monitoring

**Important**

Genesys recommends the use of Portainer to monitor containers; however, please note that Portainer is a third-party tool, and Genesys is not responsible for its functionality or support.

## Container Monitoring

Portainer Community Edition (CE) is a powerful, open-source toolset that allows you to easily monitor containers running in remote VMs. Setting up Portainer involves:

- Additional configuration at EXEC VMs to open 2375 port.
- Installing and configuring the Portainer tool.

### Configure EXEC VM

Each EXEC VMs needs to be configured to connect with Portainer for monitoring. In each EXEC VM,

- Create a **daemon.json** file in **/etc/docker** path with the following configuration:

```
{
 "hosts": ["tcp://0.0.0.0:2375", "unix:///var/run/docker.sock"]
}
```

- Add the following configurations in **/etc/systemd/system/docker.service.d/override.conf** file:

```
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd
```

- Run `systemctl daemon-reload` to reload the systemd daemon.
- Run `systemctl restart docker.service` to restart docker.

## Service Monitoring

EXEC Services are designed to be monitored by Prometheus. EXEC Services generates a number of metrics, which can be used to observe different types of EXEC operations: container metrics, availability of the architectural elements EXEC services depends on, and others. Full list of the available EXEC service metrics can be obtained through the call to the EXCP REST API /metrics. Genesys recommend building monitoring dashboards for EXEC services based on the metrics exposed by the services.

Prometheus can also be used to generate alerts based on the EXEC service metrics. bootstrap VM will have the alert files under the `<exec installation directory>/alert_rules/<service>/alert.yaml`. Administrators can either upload this yaml if they want to use Prometheus based alerts or can use it as a reference.