Orchestration Server Deployment Guide

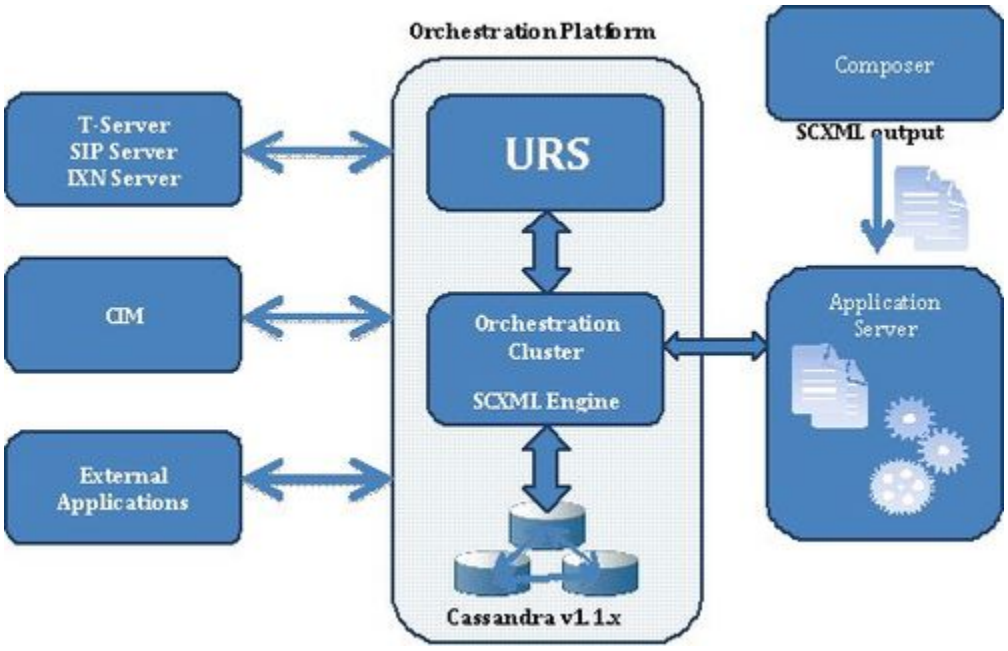
Architecture

12/16/2025

Contents

Architecture

The Orchestration Platform consists of ORS and Universal Routing Server (URS). The platform works with T-Servers, SIP Server, or Interaction Server quite similarly to the way URS previously worked with these components. In this architecture, requests from these services go to ORS and utilize URS services for routing. A high level architectural diagram is shown below.



Composer

Within the Orchestration Platform, Composer serves as the Integrated Development Environment to create and test SCXML-based routing applications executed by ORS. An application server is utilized to provision SCXML routing applications to ORS. See Composer in figure above.

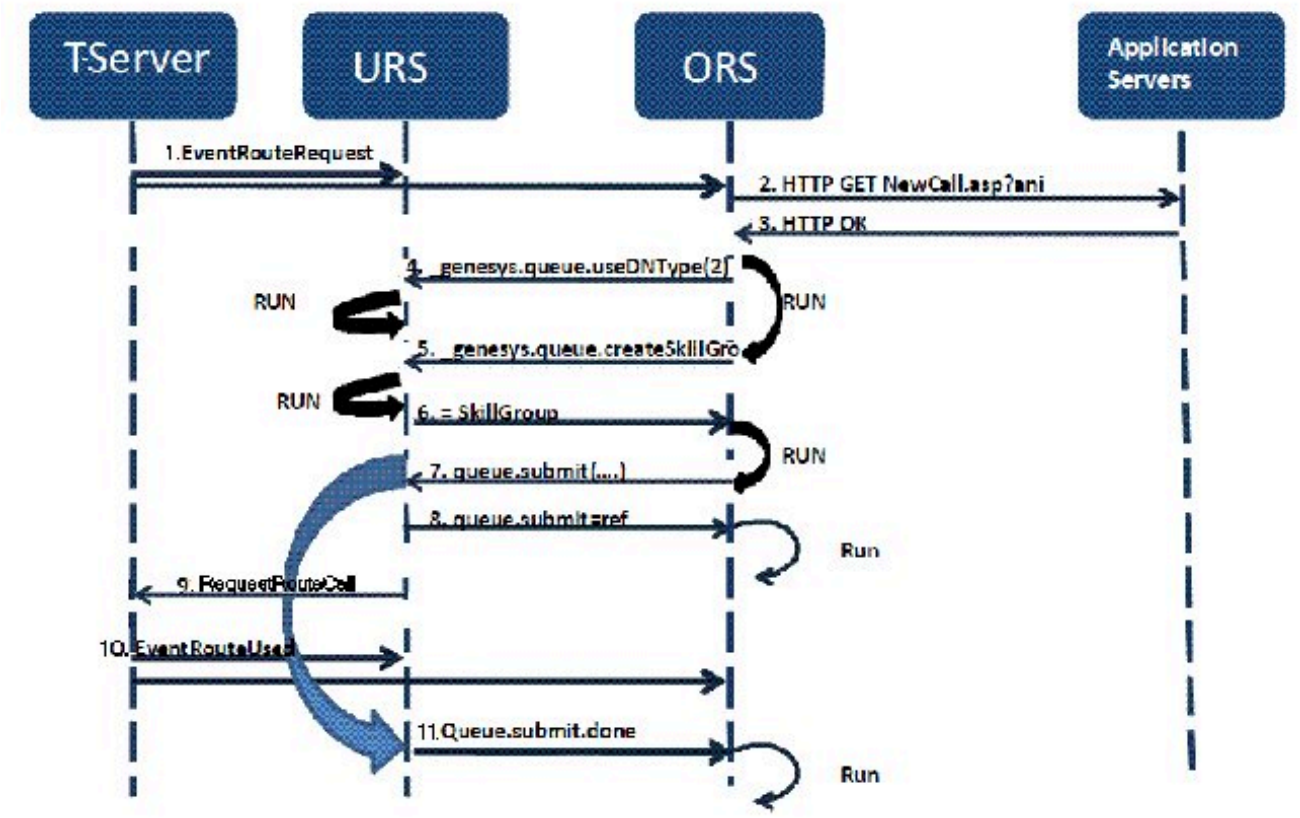
Cassandra

The Orchestration Platform provides persistence of sessions by utilizing Cassandra NoSQL DB, which is packaged and built-in with ORS (see Cassandra in figure above). Cassandra stores information regarding active sessions, as well as scheduled activities.

- For information on Cassandra, see the Cassandra Installation/Configuration Guide.

Voice Interaction Flow

The figure below shows a sample voice interaction flow that is based on the above architecture diagram. For information on the events shown, see the Genesys Events and Models Reference Manual.



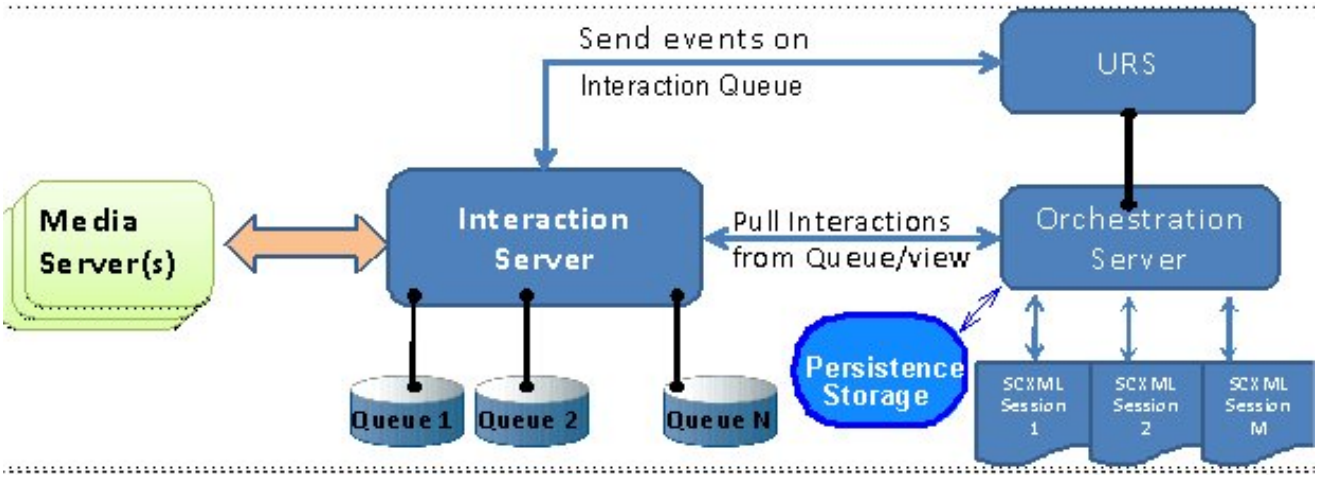
The following explanation describes the sample voice interaction flow.

1. A new interaction arrives for T-Server (EventRouteRequest). T-Server notifies URS and ORS.
2. The configuration is enabled to use an ORS application, so ORS asks the Application Server for an SCXML application session.
3. The Application Server provides an SCXML application to ORS. ORS starts execution of the SCXML application (HTTP OK).
4. At times during the SCXML application execution, ORS asks URS for assistance with tasks, such as routing.
5. It invokes Functional Modules via SCXML action events, shown here as RUN.
6. URS performs the required action(s), shown here as RUN, and reports the results to ORS.
7. If needed, URS sends a request to T-Server. In this example, URS sends a request to T-Server that corresponds with the routing request that ORS sent to URS in Item 7 (queue.submit).

8. URS performs the required action(s), shown here as RUN, and reports the results to ORS.
9. If needed, URS sends a request to T-Server (RequestRouteCall).
10. Then EventRouteUsed occurs.

eServices Interaction Flow

This section provides information on how the Orchestration Platform supports eServices (multimedia) interactions. It also describes key use cases and key functional areas. The figure below shows an eServices-specific architecture diagram for an ORS deployment.



Design Principles

To support processing eServices interactions, the ORS design is based upon the following principles:

- ORS connects to Interaction Server. ORS registers as a Routing client in order to use a subset of requests and events suitable for the Routing client.

Note: Depending on the Interaction Server application type, ORS application may require a connection to an additional Interaction Server application. Interaction Server application can be configured based either on an Interaction Server template or on a T-Server template. If the Interaction Server application is configured based on an Interaction Server template, ORS application needs to be connected as a client to one or more Interaction Server application(s) that was (were) configured based on a T-Server template. This is because communication between ORS and Interaction Server uses T-Server protocol. Both types of Interaction Servers must share the same configuration port.

- ORS processes interactions by "pulling" them from the Interaction Server. The request `RequestPull` is used to retrieve a subset of interactions from the specified queue/view combination. The ORS log shows `EventTakenFromQueue` as evidence that interactions were pulled from the queue/view.
- ORS processes interactions only when interactions are "pulled" from interaction Server. Interactions

may be created and placed into the queue by the Interaction Server, but ORS will only process an interaction after ORS has pulled it from this queue. This allows the following:

  - Interactions are processed in the order in which they arrive, and at the proper rate.
  - The "startup" case is addressed: when ORS starts, if any interactions are queued, ORS begins pulling and processing them.
  - Specific ordering and sequencing functionalities are applied to interactions, as provided by Interaction Server's Queues and Views mechanisms.

- Pulling of interactions should be done by a designated node. See the configuration option `mcr-pull-by-this-node`, which is used to specify that the pulling of eServices interactions is allowed to be performed by a node.
- No other Interaction Server clients of type Routing client may process interactions from queues that are associated with ORS. Media Server(s), as part of open media, may process interactions in these queues. The desktop client may also process interactions.
- To achieve load sharing, multiple ORS instances can pull and process interactions from the same queue.
- ORS utilizes URS to select the target for the eServices interaction when routing is necessary. ORS uses the connection with URS to inform URS that a new eServices interaction is going to be processed. ORS then calls functions (if specified in SCXML) and queue:submit action is invoked to select the target. URS responds with the selected target, and ORS routes interactions to this target using `RequestDeliver`.
- It is acceptable for the SCXML application to redirect (or place) eServices interactions into another queue in the Interaction Server. In this case, processing of this interaction is continued when ORS pulls it again, this time from another queue. When it is pulled, ORS has information about which SCXML session is associated with this interaction, and ORS sends the corresponding event to this SCXML session.
- The queue where the interaction is placed must be associated with ORS so that ORS knows to pull interactions from it. A queue is associated with ORS by creating the `Orchestration` section in the queue's Annex tab. ORS only monitors these associated queues.

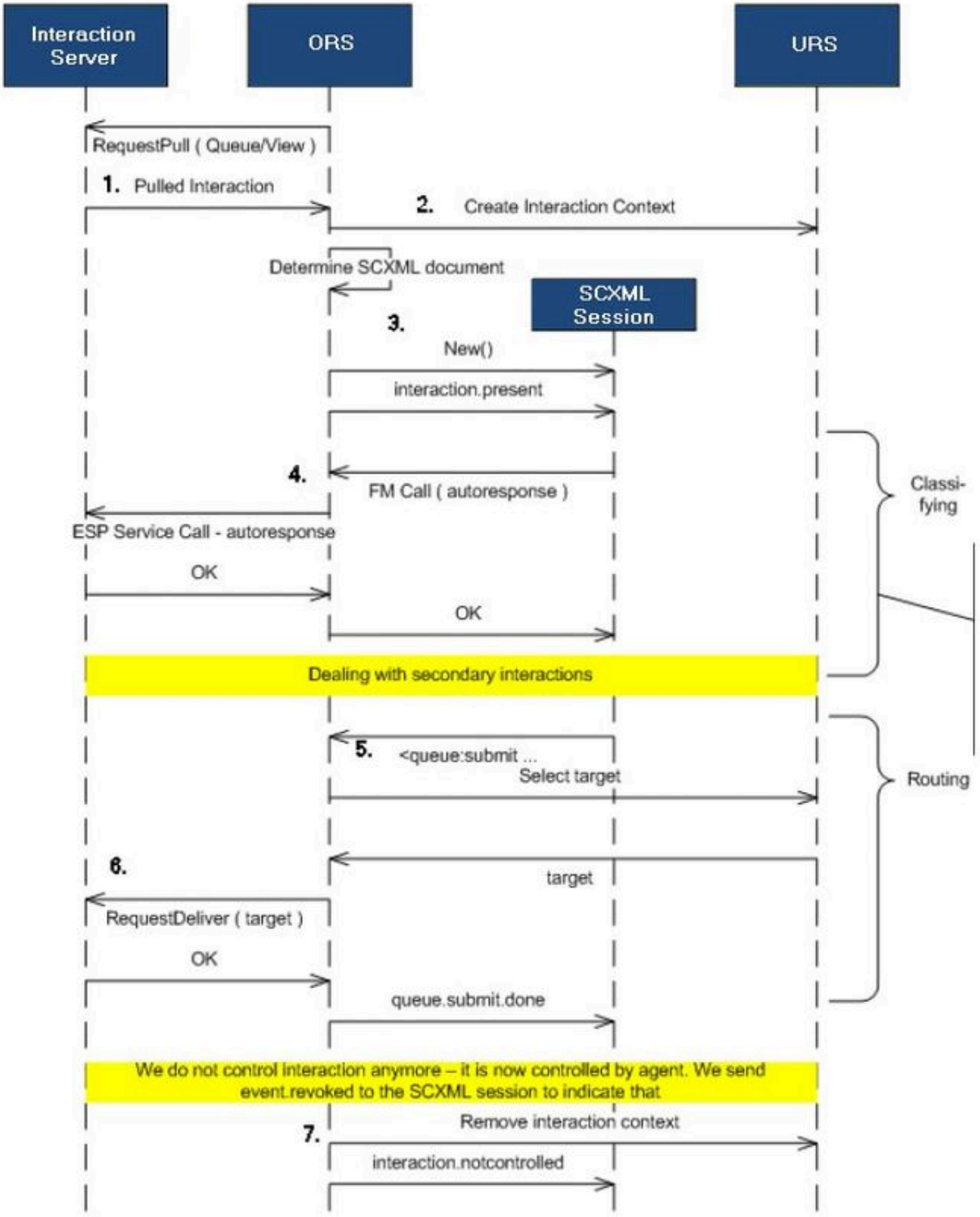
Note: All ORS requests with their attributes, including `RequestPull` can be seen in the Interaction Server log.

- Persistence Storage is used to store SCXML sessions and documents, as well as scheduled activities (such as start and stop).

Note: Previous versions of ORS required a connection to Persistence Storage (Apache Cassandra), however, ORS 8.1.3 and later do not require this connection in order to operate.

Multimedia Interaction Routing

The following sequence diagram illustrates a scenario for basic multimedia (eServices) interaction routing.


Interaction Server
ORS
URS
RequestPull (Queue/View)
1. Pulled Interaction
2. Create Interaction Context
Determine SCXML document
SCXML Session
3.
New()
interaction.present
4.
FM Call (autoresponse)
ESP Service Call - autoresponse
OK
OK
Classi-fying
Dealing with secondary interactions
5.
<queue:submit ...
Select target
Routing
6.
target
RequestDeliver (target)
OK
queue.submit.done
We do not control interaction anymore – it is now controlled by agent. We send event.revoked to the SCXML session to indicate that
7.
Remove interaction context
interaction.notcontrolled

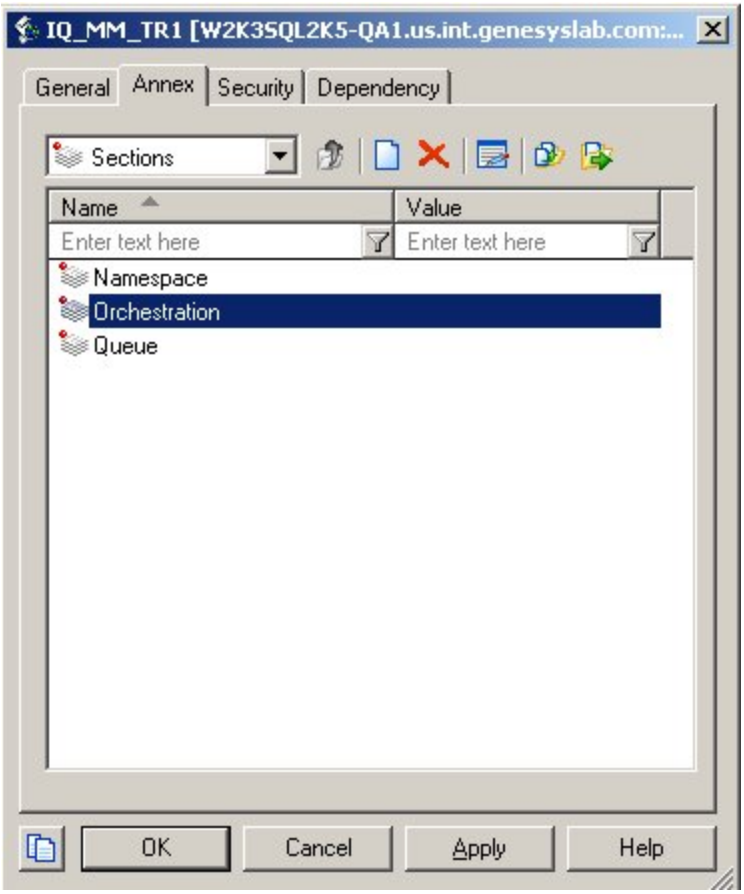
The numbered steps in the preceding figure are identified as follows:

1. ORS pulls the next multimedia interaction from a queue.
2. When the interaction is successfully pulled: Application Server receives a strategy request; URS is notified about the new interaction (and prepares to begin processing).
3. A new session starts, and an interaction.present event is fired into the session, which allows the interaction to be processed.
4. The SCXML application submits a request to Interaction Server to auto respond to the interaction.
5. After this, a queue:submit action is invoked which locates the appropriate resource to process the interaction.
6. When an available resource is found, the interaction is delivered to this resource.
7. When the interaction is redirected to the resource: interaction.notcontrolled is fired into the session; URS also is notified that the interaction is not controlled.

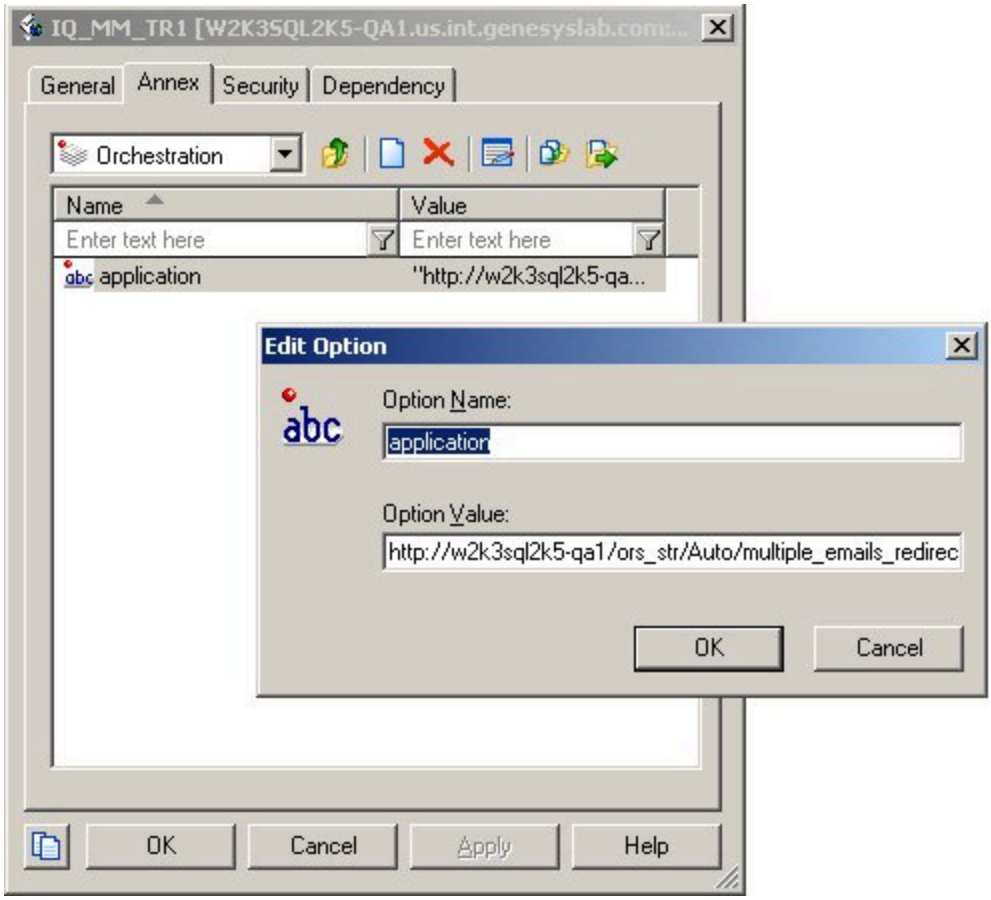
Processing eServices Interactions

The routing strategies associated with multimedia (eServices) interactions that are operating with ORS are not loaded to the Virtual Routing Points that are configured in the multimedia switch (unlike those eServices interactions associated with URS). Instead, all eServices interactions are loaded directly to the Interaction Queues, which are located in the Scripts folder of Configuration Manager.

ORS processes the interactions by pulling them from the Interaction Server. A RequestPull request is used to retrieve a subset of interactions from the specified Queue/View combination (every Queue object must have at least one View associated with it). The Queues from which ORS pulls the interactions must be explicitly associated with that specific ORS Application. ORS checks the Queue section in the Annex tab of the Interaction Queue Application for the Orchestration section and pulls interactions from these Queues only.

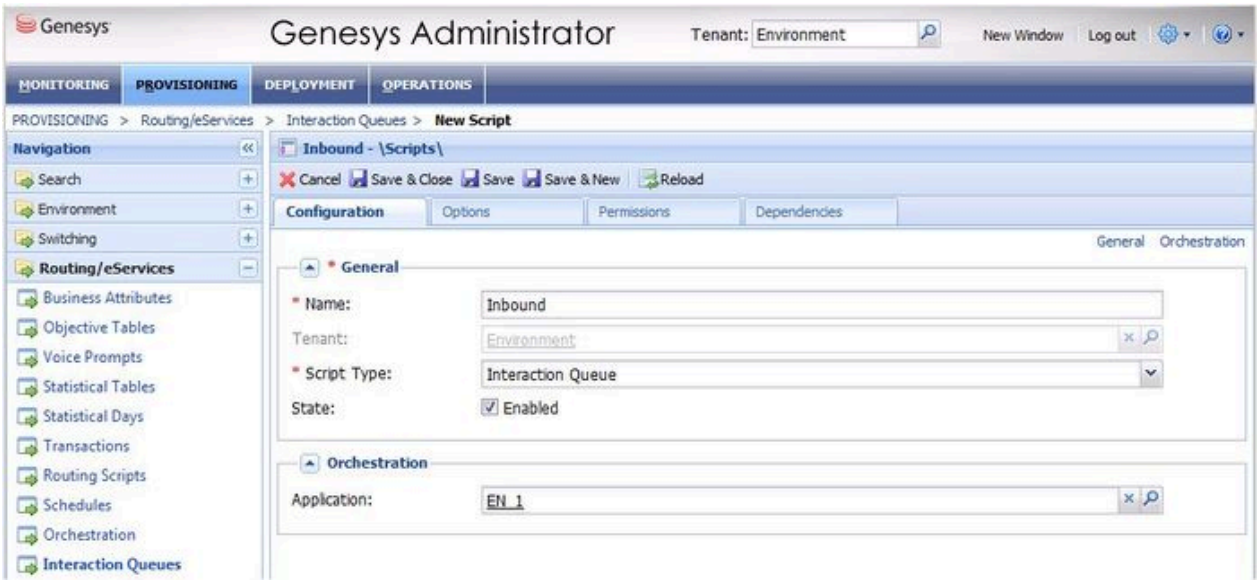


The Orchestration configuration section can contain an application option with a value that is the path to the strategy (SCXML script) that will be executed. Figure 8 shows the manually created Orchestration section and application option on the Annex tab of the Interaction Queue in Configuration Manager.



You can also associate the Interaction Queue with the a specific ORS by assigning the strategy (SCXML script) to that Queue in Genesys Administrator (see Figure 9). For example, in Genesys Administrator:

1. Go to Provisioning > Routing/eServices > Interaction Queues.
2. Navigate to the properties of a particular Interaction Queue.
3. In the Application field of Orchestration section, select the strategy (SCXML script) that will be assigned to that Queue and save the changes.



4. For each Orchestration Server dedicated to processing multimedia interactions, you must create a Script object of type Simple Routing with the same name as the Orchestration Server Application object. When this ORS Application is used by multiple Tenants, create such Script objects for each Tenant configured to work with eServices.

You can also associate an Interaction Queue with ORS by assigning the strategy to a Queue by using Composer. For information about how to assign Queues in Composer, see the Composer 8.1 Routing Applications User's Guide.

All properly associated (managed) combinations of the Interaction Queues/Views are written to the ORS log at startup.

Memory Optimization for Multimedia

When ORS is deployed to process multimedia interactions, there may be periods where there are very few agents available with a large volume of multimedia interactions waiting to be processed.

You can configure the ORS Application object to prevent excessive memory utilization by removing passive multimedia interactions from the memory cache. ORS will place multimedia interactions in the memory cache up to a configured number. Beyond this maximum value, the oldest multimedia interactions are removed from the memory cache. You can also set the number of calls that should be deleted when this maximum value is attained. The applicable options are:

- om-memory-optimization
- om-max-in-memory
- om-delete-from-memory

Multi-Site Support

Multi-site routing within the Orchestration Platform involves the handling of a call across one or more T-Servers (referred to as a site).

Interaction Routing Designer strategies typically were provisioned against Routing Points across various T-Servers to support the desired behavior.

This scenario allowed control over the segmentation of routing logic. The call was redirected from strategy to strategy, or from an agent resource back to a Routing Point with an associated strategy, within the various switches.

Within Orchestration, segmentation of the routing logic may be accomplished by combining SCXML documents and controlling the flow programmatically, rather than requiring the movement of interaction to dictate which SCXML document needs to be executed.

Interactions and SCXML Sessions

Orchestration is able to undertake this by associating an interaction with a controlling SCXML session. Such association between an Interaction and an Orchestration session is created when a call enters the site and executes its first Orchestration SCXML Session. The association is maintained until one of the following is true:

- The SCXML session exits.
- The SCXML explicitly transfers the association to another session using the <associate> action.
- The Interaction is no longer valid due to deletion of the interaction from the system, for example, when the customer hangs up.

While this type of associated session is active, all interaction events are directed to this owning or controlling session for handling. This allows for the creation of a single SCXML session, which can control the complete interaction handling, encompassing pre-routing and selection of a resource. It also allows control of post-routing once a resource has redirected it to another Routing Point.

This provides valuable benefits, such as streamlining the amount and type of configuration that is required. It also allows for the creation of more obvious interaction and conversation processing-logic, because the interaction can be controlled during periods that are not currently supported by URS/IRD.

Legacy Customers

The ability to maintain this association might be perceived as providing unexpected behavior for customers who wish to maintain the legacy way of breaking up the routing logic based on Routing Points. Legacy customers may view this as ORS executing SCXML logic that does not result in the creation/execution of a new session because of a pre-existing association with an existing session.

For this reason, in order to maintain equality with existing structures, Orchestration 8.1.2 introduced additional SCXML actions that can be used to help control the association between an interaction and an Orchestration session. This allows customers to recreate the legacy routing logic separated by provisioning, rather than centralized SCXML control logic which results in the handling of the interaction across multiple sessions.

SCXML Actions

The following SCXML actions provide the application developer with increased control of the association.

- <attach> - Allows a session to explicitly request an association with an interaction that is currently not associated with any session.
- <detach> - Allows a session to explicitly inform Orchestration that it no longer requires an association with the indicated interaction.
- <associate> -Allows a session to explicitly associate an interaction it owns to any other session.

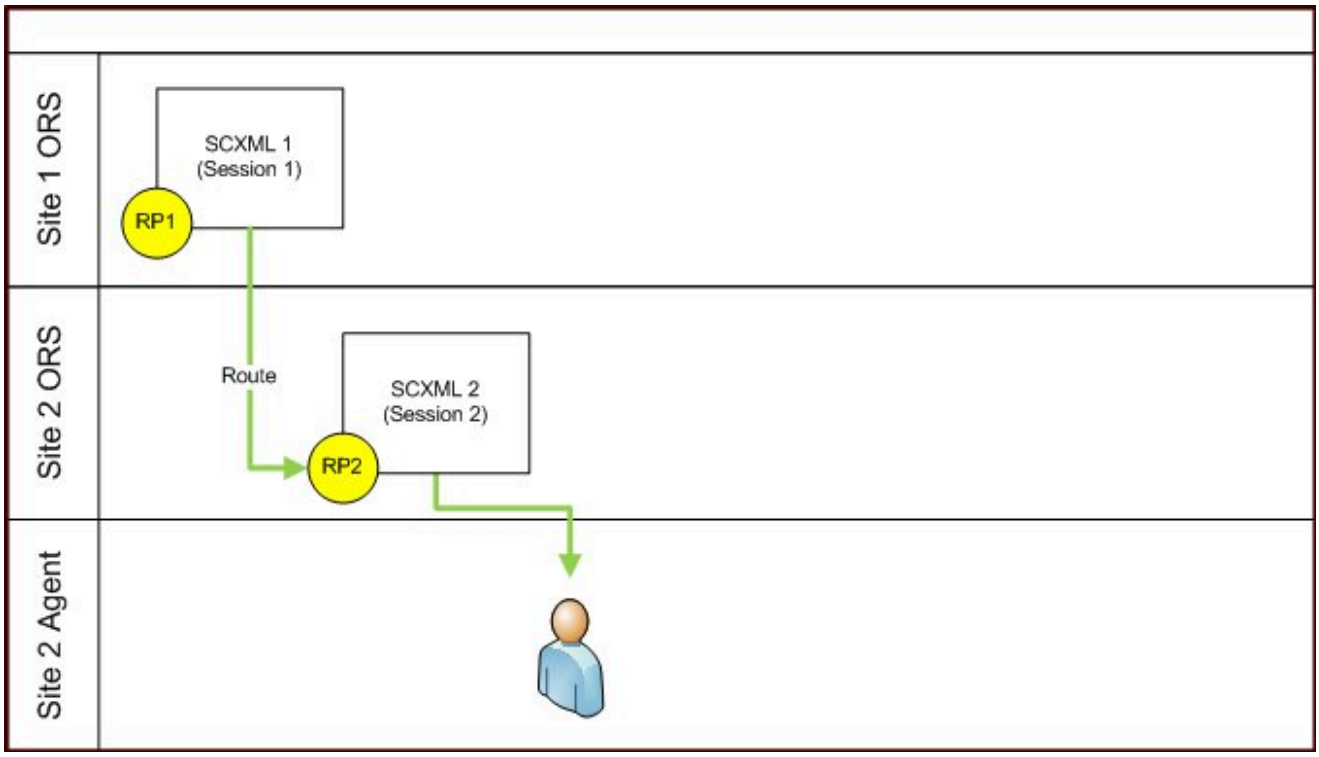
When an interaction is not currently associated with any session, the determination of which SCXML document to execute is based on the existing configuration, and allows for similar structures to be created, as presently done within URS/IRD strategies.

Example Use Case

The following is a use case to exemplify the behaviors that <attach> and <detach> may provide. In most cases, to allow a new session to be created, <detach> should be called before the routing operation. Upon failure of the routing operation, <attach> should be called.

ORS Routes to ORS controlled Route Point

The figure below exemplifies an SCXML session (Session 1) that has been executed as a result of a call entering Routing Point 1 (RP1). Upon entry, Session 1 may determine to continue the logic and call handling that it needs to transfer the call to Routing Point 2 (RP2) on Site 2. To accomplish this, a <queue:submit> is performed with route equal to false. Upon success, the interaction is then detached from Session 1 and is then redirected to the destination returned back from the <queue:submit>, upon success, a new session, Session 2 is started and Session 1 exits. On failure to route to RP2, Session 1 will perform an <attach> to reestablish the association with Session 1 and perform some other processing within the same session. The second session will route the call to a local agent.



For more information and detailed SCXML samples please refer to the *Orchestration Developer's Guide* on the Genesys Documentation website.

Implementation Notes

The following should be taken into account when working with these actions to support multiple site and multi-session controlled routing.

<detach> may result in error.interaction.detach

To be able to detach an interaction, and remove the associate between the Orchestration session and the interaction, the interaction must be confirmed to be owned by the Orchestration session for the <detach> action to succeed. This is in part controlled by the addition of User Data that is used to help track such an association, as well as internal state within the ORS.

When an interaction is associated to a session, either directly through the creation of a session from a Routing Point, or indirectly by another session calling <associate>, there is a small window of time required to allow this information to be successfully propagated by the user data update. Should <detach> be called prior to this update succeeding, the <detach> call will return the error.interaction.detach event.

To help safe guard against this, the user should ensure that they call <detach> just prior to the session undertaking its routing of the interaction. The user should also ensure that they correctly handle the error.interaction.detach within a transition, and should ensure that they only undertake their routing operation once it has been confirmed that the interaction has been detached from the session. This can be observed by only commencing the routing operation once the interaction.detach.done event has been received.

By observing this implementation pattern, it will allow the user to correctly build SCXML documents that operate in a multi-session manner and across multiple sites.

Handling Routing errors

After calling <detach> and having confirmation that the interaction is detached, the user should then immediately route the interaction. If the routing fails, to ensure that subsequent interaction related events are provided back to the SCXML session, the interaction should be re-associated with the session if it is desired to provide subsequent processing within the SCXML document.

This can be achieved by calling the <attach> action. To undertake this, it is expected that the user should store the Interaction ID until the SCXML document has completely handled the detach, routing and or error handling and subsequent routing. By detaching an interaction, the session loses the ability to obtain interaction related events that are not related to interaction related actions, therefore the developer should be aware that to ensure the session has all related events for the interaction, it must be associated with the interaction. It is therefore recommended that <detach> is not called too far in advance of the action used to route the call.

<detach> and <queue:submit>

When using <queue:submit>, and there is a need to have multiple sessions running, it is not recommended to call <detach> prior to <queue:submit>. For scenarios that are using <queue:submit>, it is recommended that the resource is targeted first with the route attribute of <queue:submit> set to be false. This will allow the resource selection events (queue.submit.done) to be returned back to the current session. Once a resource has been located successfully you may then proceed to <detach> the interaction from the session and <redirect> the interaction to the selected resource as indicated in the queue.submit.done event.