Orchestration Server Deployment Guide

Orchestration Server 8.1.3

1/5/2022

Table of Contents

Orchestration Server 8.1.3 Deployment Guide

Welcome to the Orchestration Server 8.1.3 Deployment Guide. This guide explains Genesys Orchestration Server (ORS) features, functions, and architecture; provides deployment-planning guidance; and describes how to configure, install, uninstall, start, and stop the Orchestration Server.

About Orchestration Server

This section includes information on:

General Configuration & Deployment

This section includes information on:

ORS Features

This section includes information on:

Options

This section includes information on:

ORS and SIP Server

This section includes information on:

Application Server Deployment and Debugging

This section includes information on:

About Orchestration Server

Orchestration Server (ORS) takes the Genesys core capability of routing and extends it, generalizes it, and integrates it more tightly with other Genesys products. Orchestration provides dynamic management of interactions through the use of business rule tools, dynamic data, and applications that are based on open standards.

Orchestration provides dynamic management of interactions through the use of business rule tools, dynamic data, and applications that are based on open standards (for example, SCXML-based routing applications created in Composer).

Managing Customer Conversations

1. A bank customer calls an 800 number to inquire about mortgage preapproval. An Interactive Voice Response (IVR) prompts him to enter his account number, then transfers him to an agent.
2. The agent fills in an application form for the customer and asks him to fax some supporting documents.
3. After the customer faxes the documents, he receives an automated SMS message, which thanks him and informs him that he will receive a response within 48 hours.
4. Within the next day or two, the customer receives an e-mail congratulating him on the approval of his application.

This example, involving voice, IVR, fax, SMS, and e-mail channels, shows how ORS is able to treat all the various transactions as a single service managing customer conversations over time.

Orchestrating Customer Service

Orchestration Server gets its name from its ability to "orchestrate" (direct and control) customer services. ORS routing/customer service applications can process:

- Across multiple interactions with a customer.
- Across multiple channels for interacting with a customer.
- Applications that are integrated and consistent with an organization's business processes.

ORS can work with multiple application types, such as routing strategies, session logic, service logic, and Service State logic of the interaction.

SCXML-Based

Adding ORS to Universal Routing allows an open approach to routing strategy creation:

- Previous to ORS, URS executed routing strategies that were created in Interaction Routing Designer using the proprietary Genesys Interaction Routing Language (IRL).

- In contrast, ORS can execute routing strategies and applications that are created in Composer written in a non-proprietary, open language: SCXML.

Open Standards-Session Based Platform

ORS offers an open standards-session based platform with a State Chart Extensible Markup Language (SCXML) engine, which enables intelligent distribution of interactions throughout the enterprise. In conjunction with Universal Routing Server (URS), ORS can direct interactions from a wide variety of platforms, such as toll-free carrier switches, premise PBXs or CDs, IVRs, IP PBXs, e-mail servers, web servers, and workflow servers.

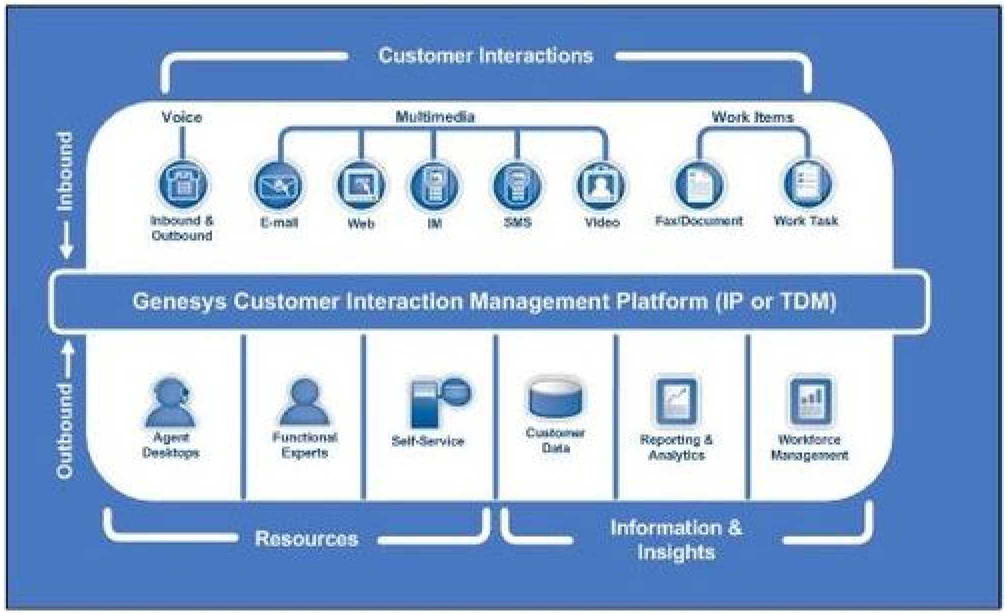
ORS can handle pure-voice, non-voice, and multimedia environments, enabling routing of each media type based on appropriate criteria. Routing strategies and business processes automate interaction routing to the most appropriate agent/resource based on factors such as the type of inquiry, the business value of the customer interaction, context and customer profile, and the media channel.

ORS and CIM Platform

ORS and URS are a part of the Genesys Customer Interaction Management (CIM) Platform, which provides the core interaction management functionality. The Platform is the collection of core servers that enable the rest of your Genesys environment to process the thousands of interactions that represent the needs of your customers. The CIM Platform consists of the following Genesys products:

- Management Framework including Genesys Administrator
- Universal Routing and ORS
- Interaction Management, which in turn consists of:
  - eServices
  - Interaction Workflow
  - Knowledge Management
  - Content Analysis
  - Universal Contact History
  - Composer
  - Reporting & Analytics]]

The figure below depicts the CIM platform.



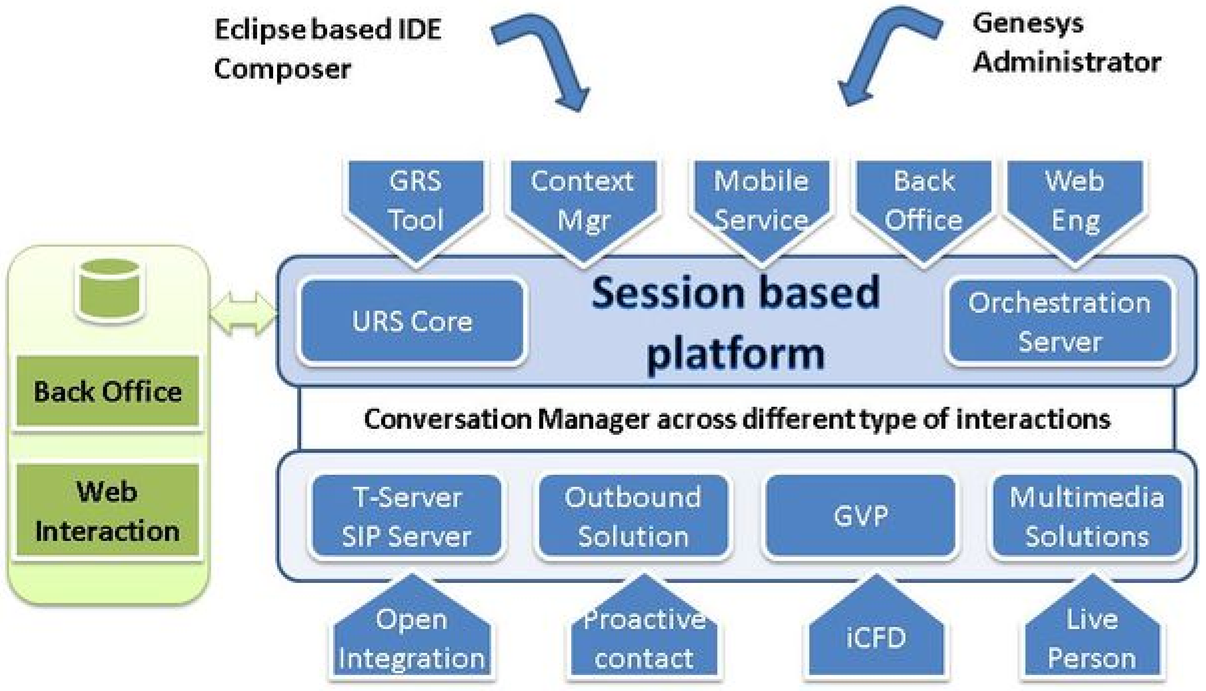
ORS and Multimedia

Orchestration Server in conjunction with URS provides a platform for different Genesys solutions to work together managing interactions regardless of media type. By adding ORS to URS, you now have the possibility to coordinate processing of multiple interactions of different media types that are involved in a single service.

As shown below, this multimedia capability includes some parts of the CIM Platform plus media channels that run on top of the Platform as follows:

- From the CIM Platform, ORS provides centralized handling of interactions regardless of media type.
- From the eServices media channels, at least one of the following:
  - Genesys Email
  - Genesys Chat
  - Genesys Open Media-The ability to add customized support for other media (fax, for example)
  - Optionally, Web Collaboration - The ability for agents and customers to co-browse (simultaneously navigate) shared web pages. This is an option that you can add to either Genesys Chat or Inbound Voice.

An Orchestration solution can consist of many interactive and integrated components. The figure below shows an example.



Supported Application Servers

See the *Composer 8.1.3 Deployment Guide*, Application Server Requirements.

MIME Types

MIME (Multipurpose Internet Mail Extensions) refers to a common method for transmitting non-text files via Internet e-mail. By default the SCXML MIME type is already configured in the Tomcat server bundled with Composer. If you are using the Internet Information Services (IIS) Application Server to deploy SCXML-based applications, add the following file extensions with MIME type through the IIS Manager of your webserver:

- `.json text/json`
- `.scxml text/plain`
- `.xml text/xml`

Security

Genesys uses the Transport Layer Security (TLS) protocol that is based on the Secure Sockets Layer (SSL) 3.0 protocol. TLS uses digital certificates to authenticate the user as well as to authenticate the network (in a wireless network, the user could be logging on to a rogue access point).

You can secure all communications (SSL/TLS) between Genesys components, using authentication and authorization (certification validation). This functionality is configurable so that you can secure all connections, a selected set of connections, or none of the connections.

Summary information on ORS 8.1 security is presented in the following subsection. For detailed information on how to implement security within Genesys, see the Genesys 8.1 Security Deployment Guide. For information about how to deploy a third-party authentication system in order to control access to Genesys applications, see the Framework 8.1 External Authentication Reference Manual.

Simple TLS and Mutual TLS

ORS supports simple TLS and mutual TLS.

In simple TLS, only the Server has a security certificate. It sends this certificate to the Client, which checks the certificate against its own Certificate Authority (CA). In effect, this authenticates the identity of only the Server.

In mutual TLS, both the Server and the Client have security certificates. They exchange their certificates, then each checks the other's certificate against its own CA. This authenticates the identities of both the Server and the Client.

For detailed information on TLS configuration, see corresponding TLS sections in the Genesys 8.1 Security Deployment Guide.

Client-Side Port Definition

The client-side port definition feature of Genesys security enables a client application (of server type) to define its connection parameters before it connects to the server application. This enables the server application to control the number of client connections. In addition, if the client application is located behind a firewall, the server application is able to accept the client connection by verifying its predefined connection parameters. Table 3 indicates where client-side port configuration is supported for other servers.

CLIENT	CONFIG SERVER/ PROXY	T-SERVER	UNIVERSAL ROUTING SERVER
ORS	YES	YES	YES

Note: Client-side port configuration is also supported for Interaction Server and Message Server.

For detailed information on client-side port configuration, see the "Client-Side Port Definition" chapter of the Genesys 8.1 Security Deployment Guide.

Security for HTTP Requests

Orchestration Server supports SSL for HTTP requests. To configure:

1. Create or modify an existing `Host` object to use a Certificate. The `Host` name should be the fully qualified domain name that was used for generating the Certificate. Instructions for assigning a Certificate to a Host can be found in the Genesys 8.1 Security Deployment Guide, Chapter 18.
2. Create or modify an existing Orchestration `Application` object to use the `Host` from step 1.

3. Create or modify the `http` port for secure connection by selecting `Secure` as the Listening Mode.
4. Save the configuration.

After applying the configuration steps above, new connections to that ORS `http` will be encrypted. No ORS restart is necessary. All requests made to that port should begin with `https://` instead of `http://`.

New in This Release

This section contains a brief description of the new features in Genesys Orchestration Server (ORS) 8.1.3 release. For information on the new features in the Genesys Orchestration Server 8.1.2 release , see the Orchestration Server 8.1.2 Deployment Guide.

Release 8.1.300.25

- ORS provides an enhanced high-availability (HA) environment architecture. The enhancement to a high-availability (HA) architecture implies the existence of redundant ORS Applications: a primary and a backup (i.e., nodes). If the primary application fails, the backup can take over its operations without significant loss of data or impact to business operations.
- Multiple Data Center architecture. ORS now supports a Data Center architecture where each Data Center is served by a cluster of ORS nodes. ORS supports Data Center failover.
- A single Cassandra instance across multiple Data Centers. The Apache Cassandra open source solution packaged with Genesys Orchestration Solution now supports a single Cassandra instance across multiple Data Centers.
- Support of new version of Cassandra. ORS 8.1.3 supports Apache Cassandra 1.1.x, beginning with Version 1.1.12.
- Enhanced voice interaction action. The <createcall> action with type “predictive” was extended by providing Call Progress Detection (CPD) results for voice interactions in Interaction interface events. The CPD result shows the outcome, either positive (live voice), semi-positive (answering machine, fax or silence), or negative (busy, no answer, etc.) of an attempt to reach an intended party. This allows ORS to determine the next actions for this interaction, such as redialing at the later time for negative call results or connecting an established outbound call to pre-recorded message instead of an agent for ‘answering machine’ call result. Refer to the Orchestration Server Developer's Guide for more information.
- Infinite loop prevention. A change has been implemented in ORS behavior called infinite loop prevention. This modification prevents ORS from infinite loops between states or from an endless number of times a state is entered as a direct result of a transition element.
- Enhanced Multi-Site support. Simplification of processing multi-site interactions by separating interactions objects from different sites between different SCXML strategies.

Subsequent Releases

- Starting with Release 8.1.300.47, ORS now populates the values of `AttributeErrorCode` and `AttributeErrorMessage` from `EventError` into the error and description attributes of the `error.interaction.redirect` event when T-Server fails to redirect a voice interaction.

- Starting with Release 8.1.300.45, ORS detects overload conditions by the measuring the time consumed by the SCXML engine to create sessions and the number of pending session-creation requests. If a growing session creation time becomes higher than the configured `max-session-create-time` threshold and the number of pending requests to create sessions is higher than 0, ORS stops creating new sessions and will remain in that state until all pending session creation requests are completed.

- Starting with Release 8.1.300.36, Orchestration Server now allows you to retrieve information directly from the Configuration Layer without sending requests to Universal Routing Server when using the following `_genesys.session` functions: `isSpecialDay`, `timeInZone`, `dateInZone`, `dayInZone`, `listLookupValue`, and `getListItemValue`. To enable this functionality, use the new option `functions-by-urs`.

- The following attributes have been added: Starting with Release 8.1.300.41, to ensure proper session persistence during High Availability recovery, a new attribute, `_statePersistDefault` is added to the top-level `<scxml>` element. Starting with Release 8.1.300.32, a new `_transitionStyle` attribute is added to the `<scxml>` element, which allows defining the order in which the `<transition>` content will be executed. For details on these attributes, see the *SCXML Langage Reference* in the Orchestration Server Developer's Guide.

- Starting with Release 8.1.300.39, ORS reports Standard log level messages upon connecting to or disconnecting from Universal Routing Server:

- Starting with Release 8.1.300.35, ORS supports SSL for HTTP requests. To use this feature, SSL/TLS must be enabled on the `http` port of the ORS `Application` object. All new connections to that port will be encrypted. For detailed information on how to configure security, refer to the Genesys 8.1 Security Deployment Guide. See the Orchestration Server 8.1.x Release Note for the recommended log configuation.

- Starting with Release 8.1.300.32, adjustments were made `Debug` log message level assignments, which allow configuring ORS logging so that log output will contain enough information for initial troubleshooting but, at the same time, produce log output that is significantly more compact than a full `Debug` log.

- Starting with Release 8.1.300.29, ORS reports `Standard` log level messages related to connection to Cassandra. See the Orchestration Server 8.1.x Release Note for details.

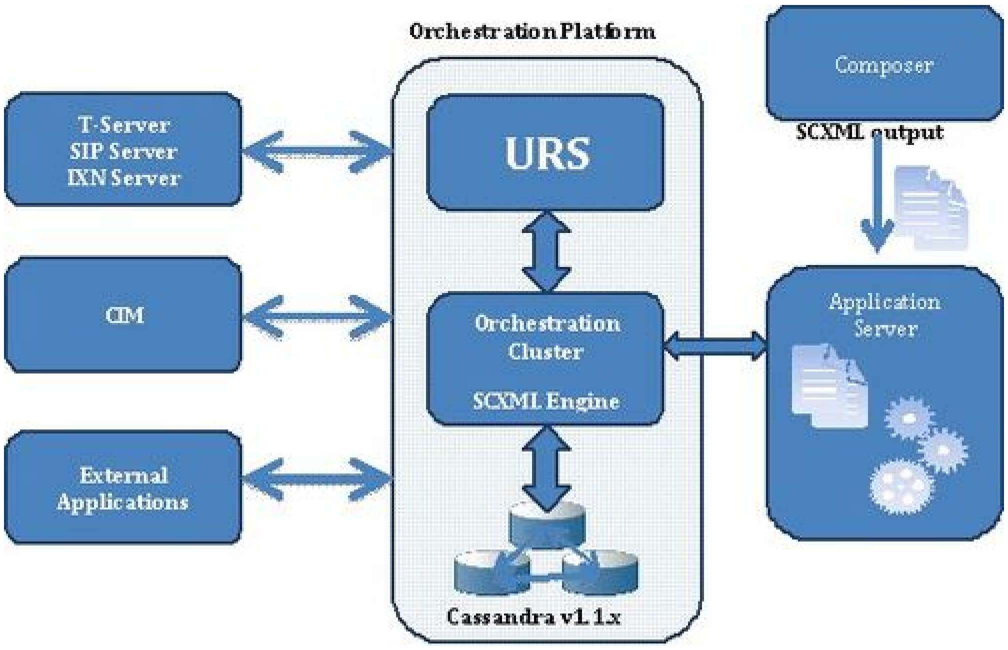
- Starting with Release 8.1.300.29, ORS reports the elapsed time between the time a request to fetch a document is issued and the time the requested document is retrieved. This information is provided in the duration attribute in the `<doc_retrieved>` METRIC.

- Starting with Release 8.1.300.27, ORS successfully distributes `RequestPrivateService` to SIP Server when the `resource` attribute is not specified in the `<ixn:privateservice>` action element when an inbound call diverted from a Routing Point to an external destination is answered. When integrated with SIP Server, the `extensions` parameter of the `<ixn:privateservice>` action element should contain key-value pair `AOC-Destination-DN` with the value referring to the party in established state where the `AoC` (Advice of Charge) notification should be delivered. Note: This feature is available when integrated with SIP Server beginning with version 8.1.100.93.

- Starting with Release 8.1.300.27, ORS allows substitution of parameters, provided in the URL of an SCXML application, with predefined values. See the Orchestration Server 8.1.x Release Note for details.

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.

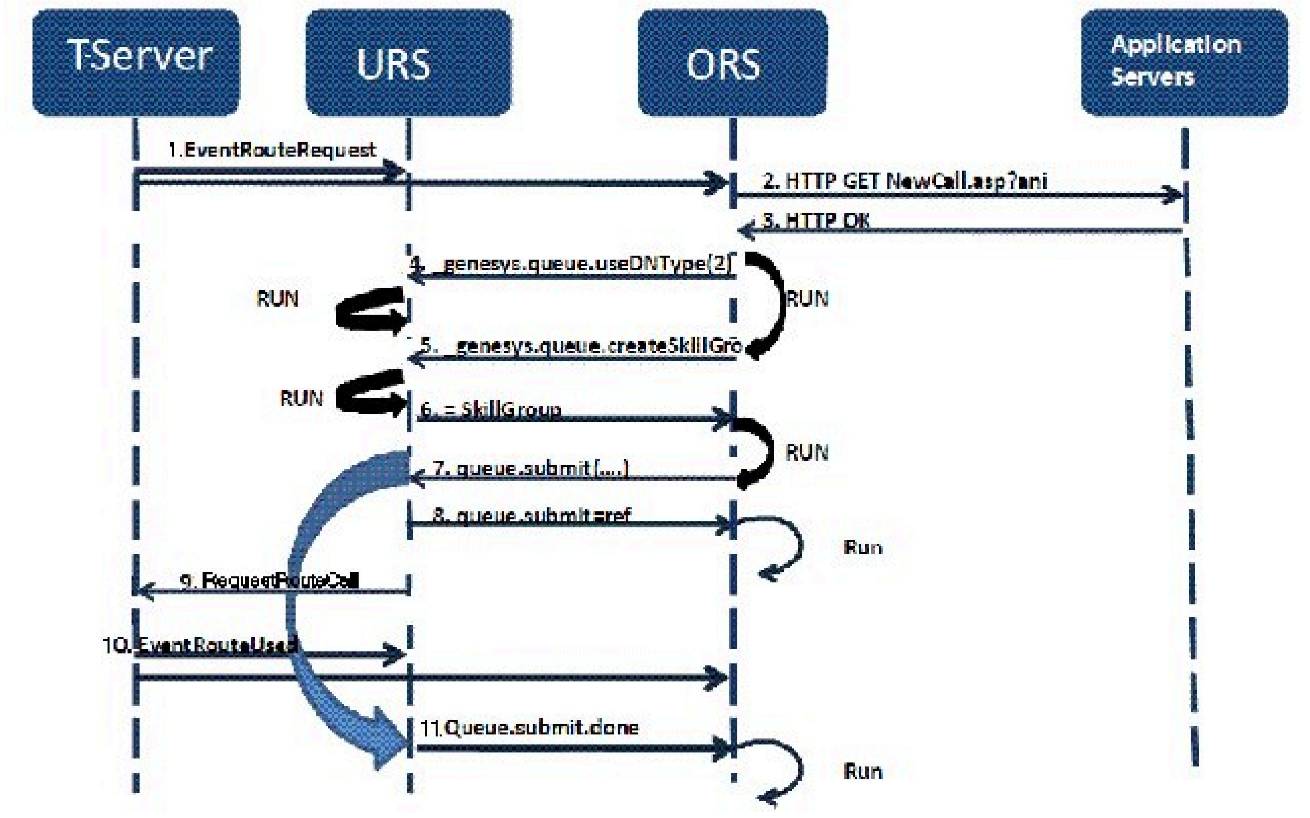
Cassandra

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.



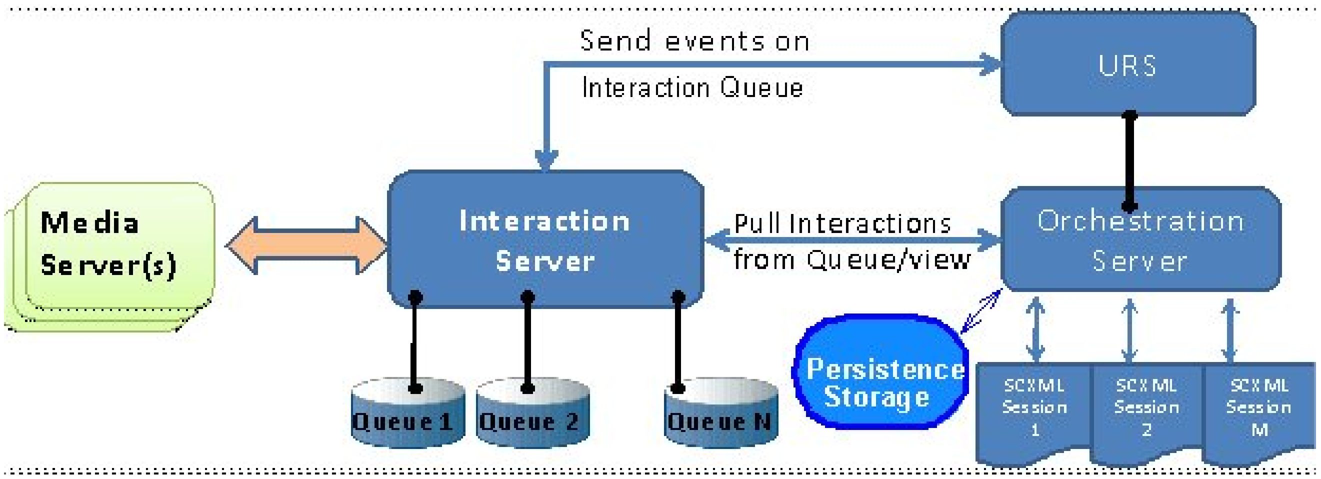
The following explanation describes the sample voice interaction flow.

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.

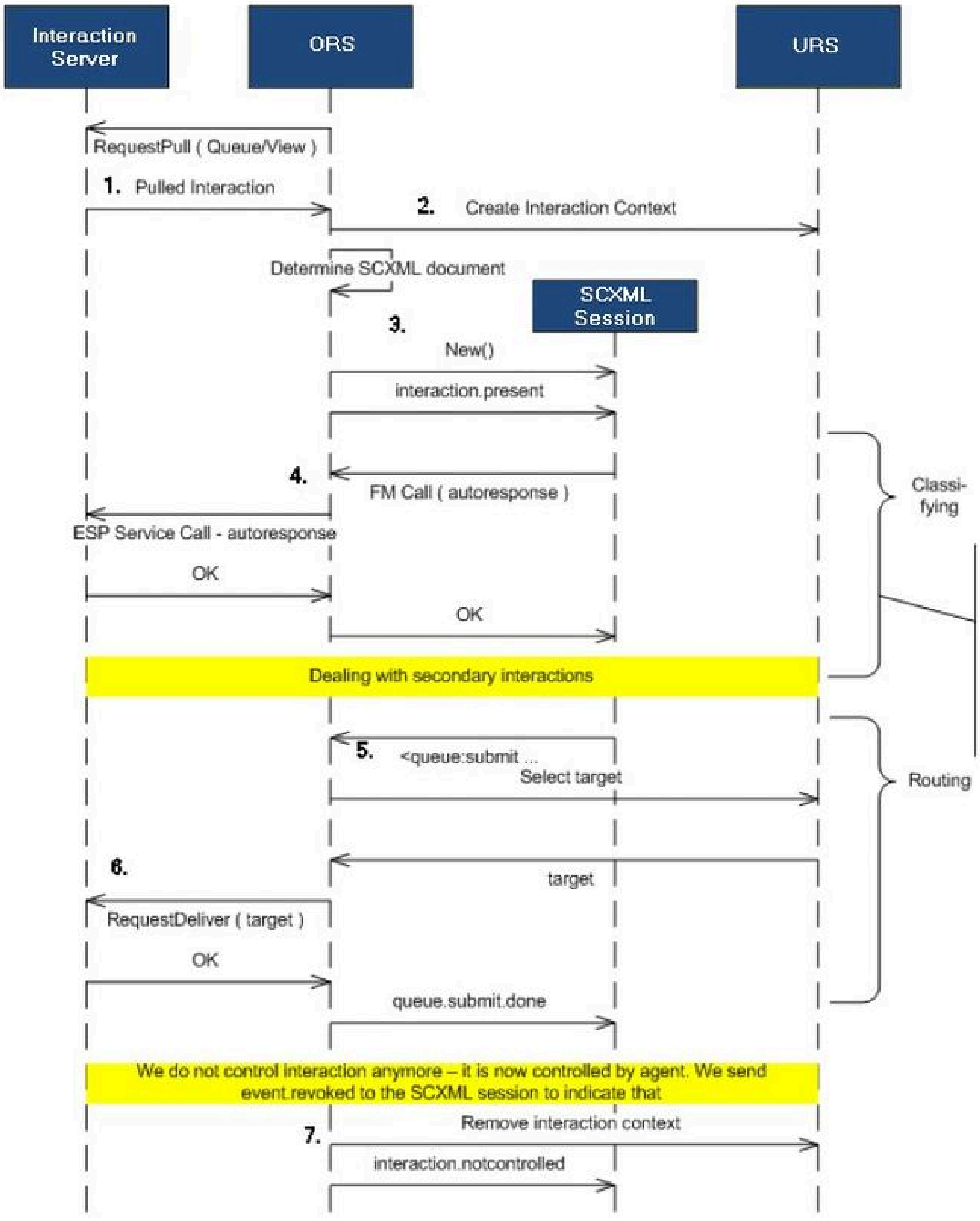
Note: All ORS requests with their attributes, including `RequestPull` can be seen in the Interaction Server log.

- 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

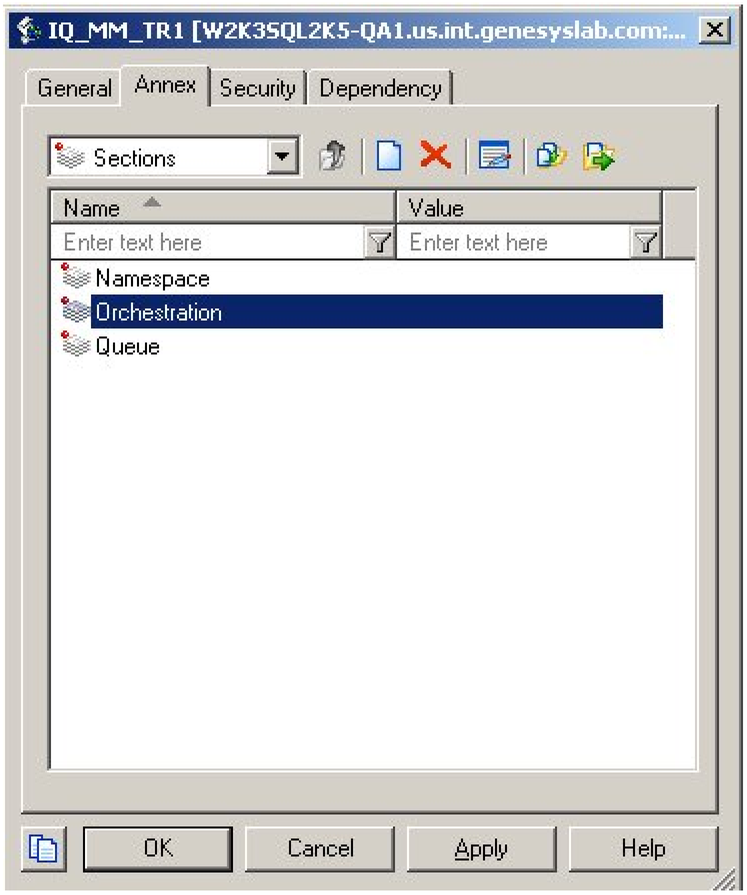
The numbered steps in the preceding figure are identified as follows:

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

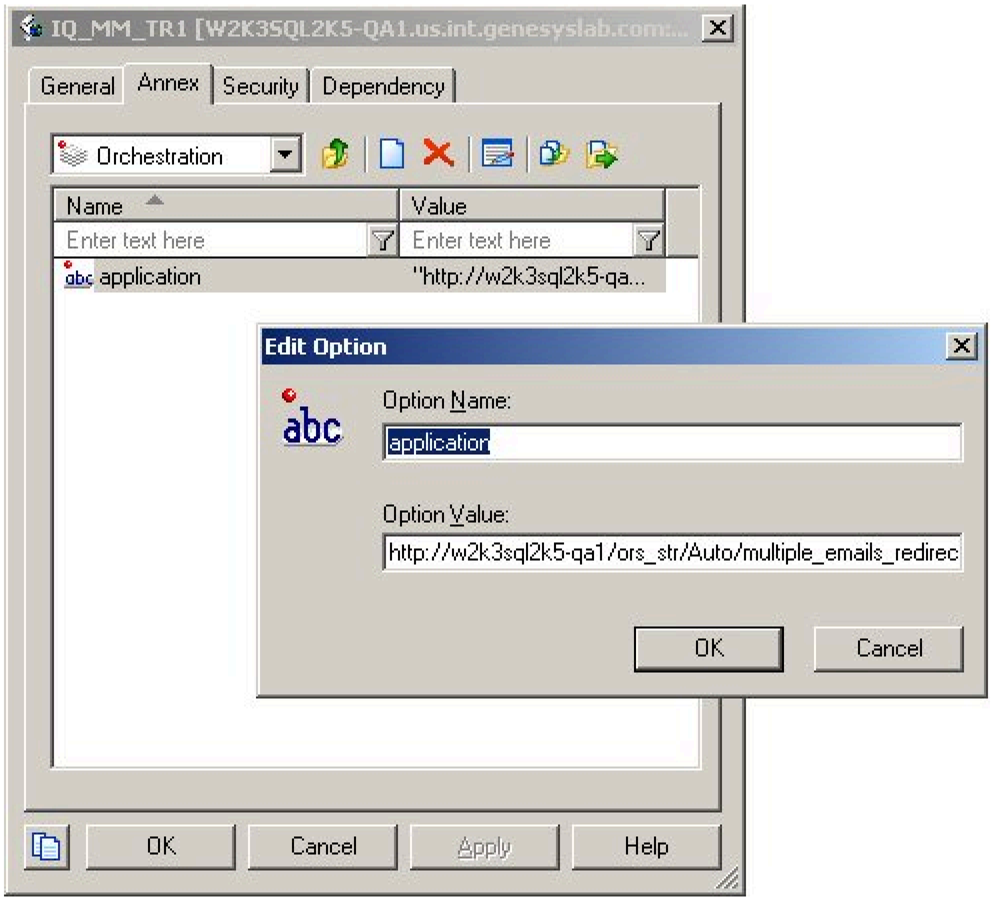
The routing strategies associated with multimedia (eServices) interactions that are operating with ORS are not loaded to the Virtual Routing Points that are configured in the multimedia switch (unlike those eServices interactions associated with URS). Instead, all eServices interactions are loaded directly 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.



Namespace
Orchestration
Queue

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.



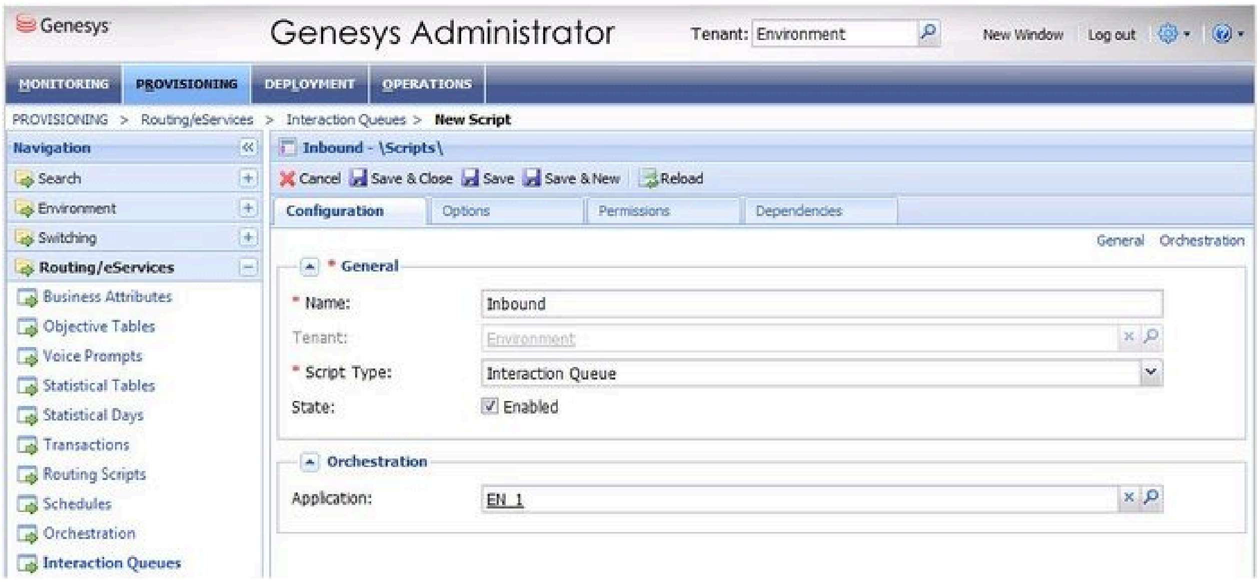
Name	Value
application	"http://w2k3sql2k5-qa...

Option Name: application

Option Value: http://w2k3sql2k5-qa1/ors_str/Auto/multiple_emails_redirec

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.



General

* Name: Inbound

Tenant: Environment

* Script Type: Interaction Queue

State: [✓] Enabled

Orchestration

Application: EN_1

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.

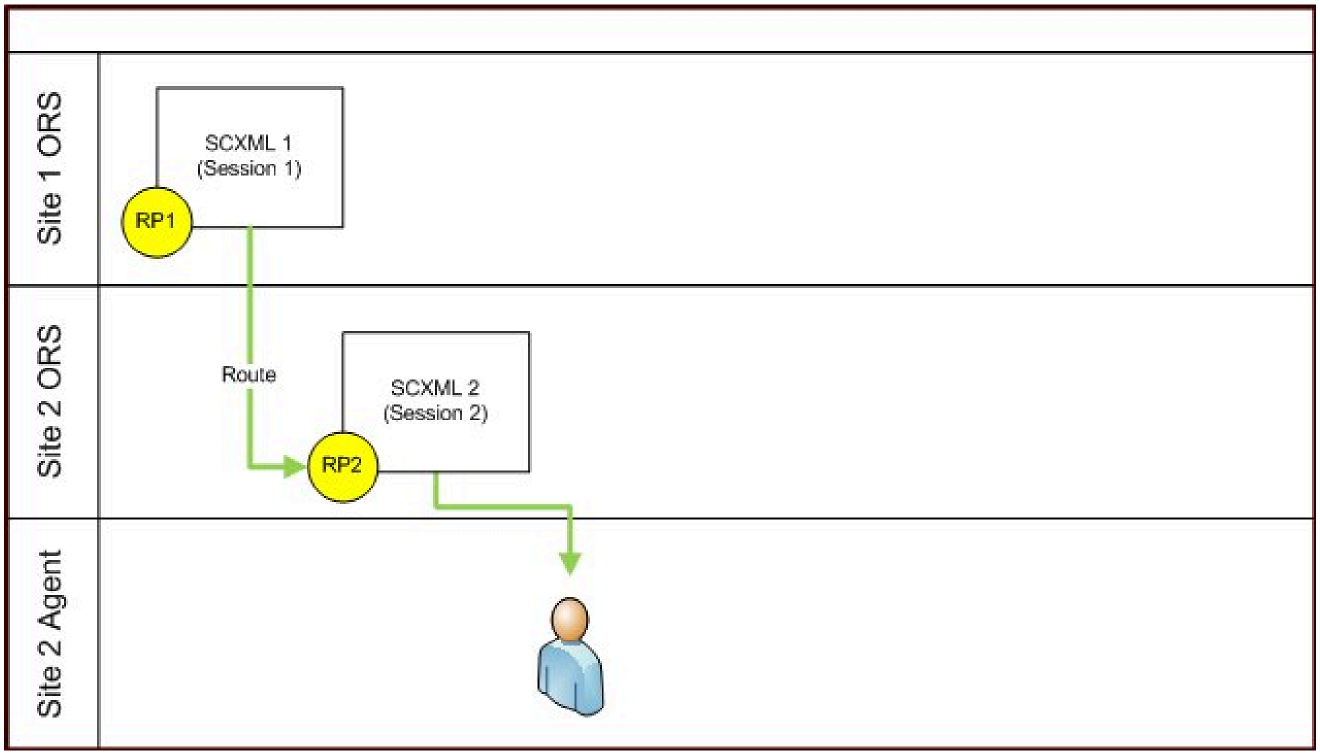
When an interaction is not currently associated with any session, the determination of which SCXML document to execute is based on the existing configuration, and allows for similar structures to be created, as presently done within URS/IRD strategies.

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>

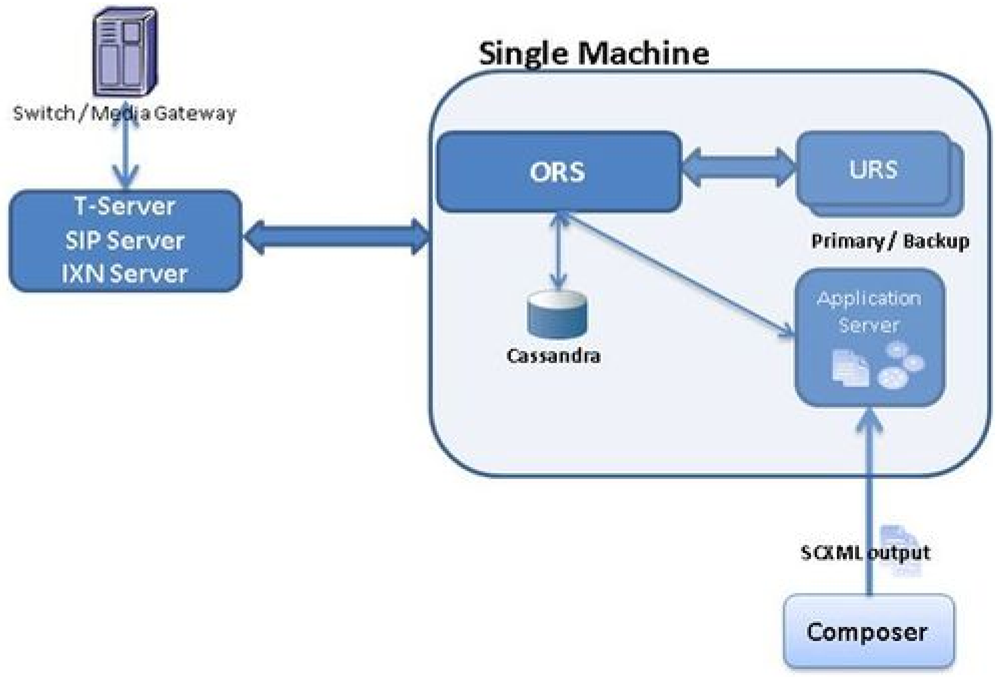
When using <queue:submit>, and there is a need to have multiple sessions running, it is not recommended to call <detach> prior to <queue:submit>. For scenarios that are using <queue:submit>, it is recommended that the resource is targeted first with the route attribute of <queue:submit> set to be false. 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.

Deployment Models

The following section provides examples of various ORS 8.1.3 and later deployment models and provides an overview of behavior, for single site, single site with redundancy, multi-site, and a single Cassandra cluster across sites:

Single-Site Deployment Model

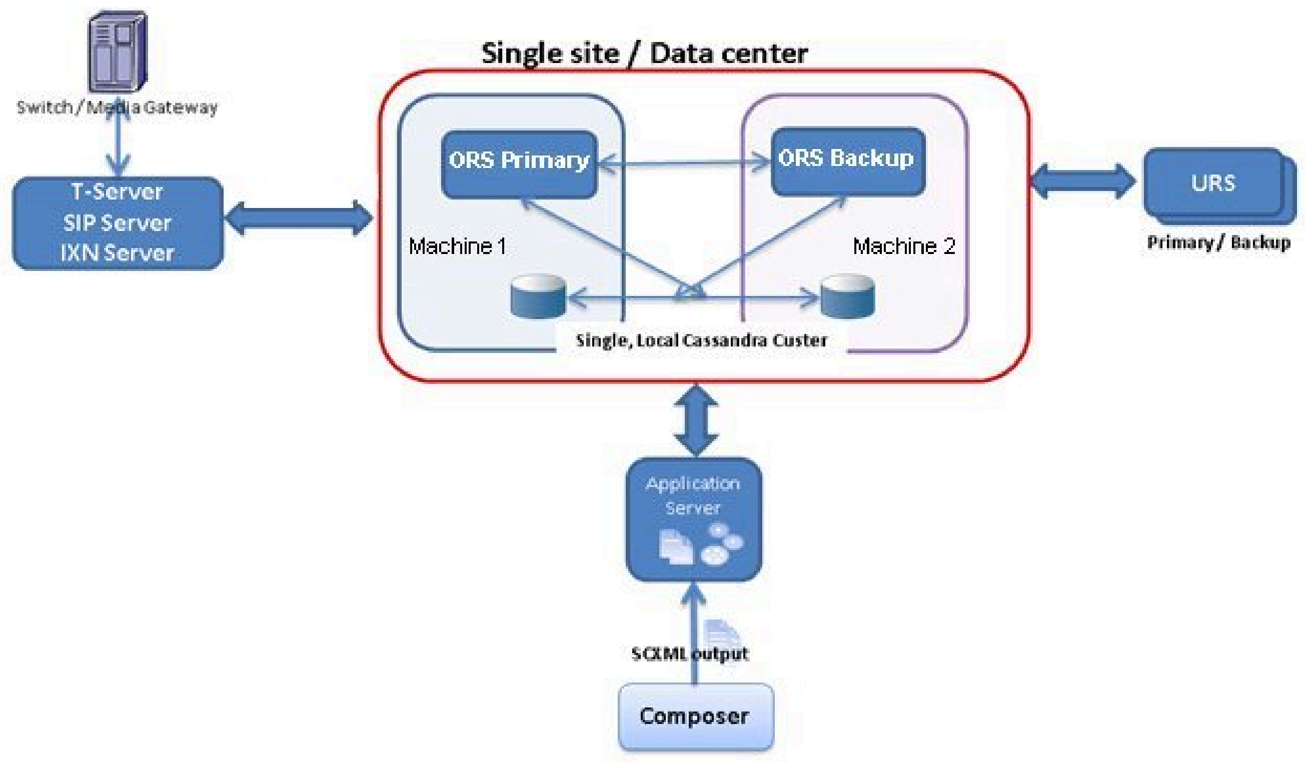
The single-site deployment example assumes that all components are planned to reside within a single box. This deployment does not provide high availability, redundancy, or failure handling. The figure below shows a single site deployment example.



Single-Site with Redundancy Deployment Model

The single site architecture with software redundancy across multiple boxes represents a standard deployment for enterprises which do not have the need or ability to provide geo-redundancy. In this deployment, two or more instances of ORS exist in a single cluster, on the same LAN.

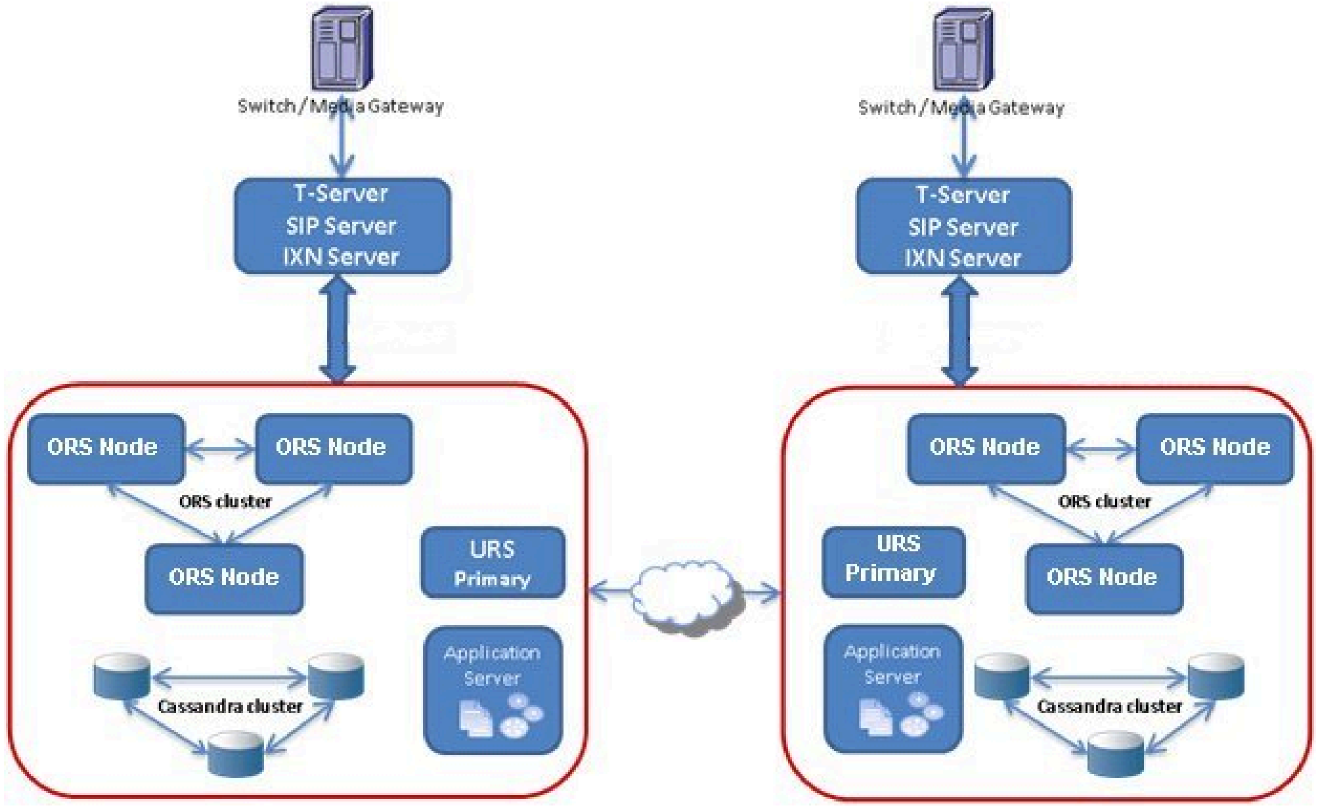
The figure below shows a single site with redundancy deployment example.



Multiple-Site Deployment Model

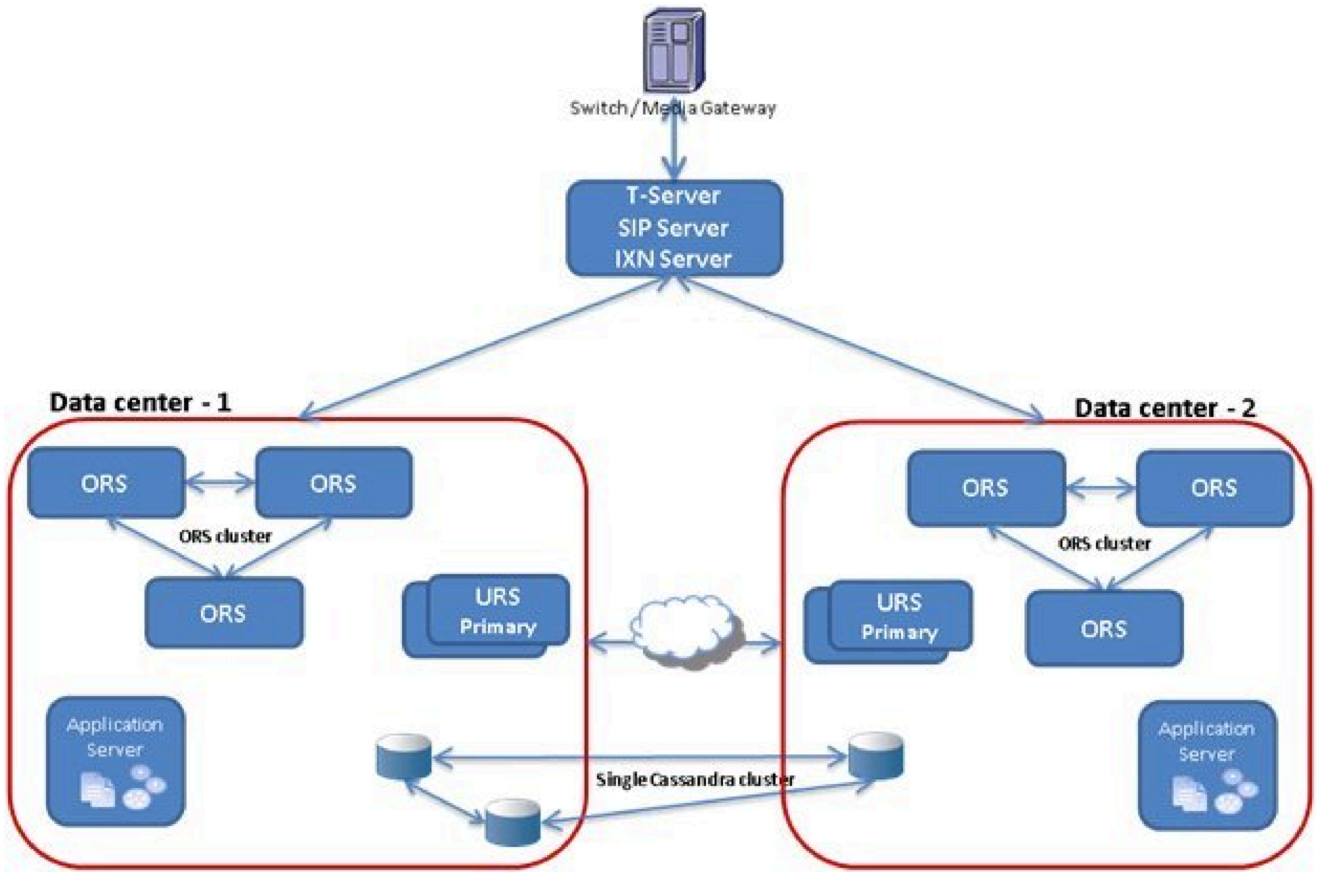
In a logically separated environment such as this with multiple T-Servers, each distinct T-Server is connected to its own ORS cluster. Note that in this environment, each ORS cluster has its own private Cassandra instance.

The figure below shows a multiple-site deployment example.



Single Cassandra Cluster across Multiple ORS Data Centers Deployment Model

The figure below shows a Single Cassandra Cluster across Multiple-site Deployment example.


Switch / Media Gateway
T-Server
SIP Server
IXN Server
Data center - 1
ORS
ORS
ORS cluster
ORS
URS
Primary
Application Server
Data center - 2
ORS
ORS
ORS cluster
URS
Primary
ORS
Application Server
Single Cassandra cluster

SCXML and ORS Extensions

Orchestration applications are created by writing SCXML documents either via your favorite text editor or via Genesys Composer. Orchestration-specific instructions are specified in the executable content of SCXML in the form of SCXML extensions (action elements) and/or ECMAScript extensions (properties of special ECMAScript objects). See SCXML Language Reference in the Orchestration Server Developer's Guide.

ORS Extensions to SCXML

ORS 8.1 extensions to the SCXML executable content are described in Orchestration Extensions available in the Orchestration Server Developer's Guide. The example below uses the Queue module. The namespace definition of the Queue module is the following:

```
xmlns:queue="www.genesyslab.com/modules/queue"
```

The Queue module can be called/addressed within the logic of the application as follows:

```
<queue:submit queue="vq1" priority="5" timeout="100">
```

ORS 8.1 provides the modules listed below.

- **Classification module**. This module element uses the namespace label classification which stands for www.genesyslab.com/modules/classification. It implements the ability to classify non-voice interactions such as e-mail, chat and open media by a given category and screen content of non-voice interaction by a given set of screen rules.
- **Queue module**. This module element uses the namespace label queue which stands for www.genesyslab.com/modules/queue. It implements the target selection functionality of URS (finding resources for interactions and delivering interactions to the resource).
- **Dialog module**. This module element uses the namespace label dialog which stands for www.genesyslab.com/modules/dialog. It implements the call treatment functionality.
- **Web Services module**. This module element uses the namespace label ws, which stands for www.genesyslab.com/modules/ws. This module provides Web Services support in Orchestration, and covers both Web 2.0 RESTful Web Services interface and legacy SOAP Web Services interface.
- **Statistics module**. This module element uses the namespace label statistic, which stands for www.genesyslab.com/modules/statistic. It implements the statistics retrieving functionality of URS.
- **Interaction module**. This module element uses the namespace label ixn, which stands for www.genesyslab.com/modules/interaction. It implements getting and changing interaction related data, such as attached data, and makes calls, transfers, conferences, and performs other related functions.
- **Session module**. This module element uses the namespace label session which stands for www.genesyslab.com/modules/session. The module maintains common objects used by the Orchestration Platform for Orchestration logic reporting and management functionality.

- **Resource module**. This module element uses the namespace label resource which stands for www.genesyslab.com/modules/resource. This module maintains a common entity (called a resource) that is used across functional module interfaces.

Action Elements

Developers specify action items as custom action elements inside SCXML executable content. All modules are prefixed with the corresponding namespace label; for example:

`queue:submit` (in this case, the queue prefix stands for www.genesyslab.com/modules/queue)

`dialog:playandcollect` (in this case, the dialog prefix stands for www.genesyslab.com/modules/dialog)

Executing Modules

When modules are executed, events return back from the platform to the instance of logic that is running the SCXML document that requested the action.

Functions and data are exposed (and used) as properties of ECMAScript objects. ORS provides a built-in ECMA Script object for every module listed above.

General Deployment

This topic contains general information for the deployment of your Orchestration Server (ORS). In addition, you may have to complete additional configuration and installation steps specific to your Orchestration Server and devices.

Note: You must read the Framework 8.1 Deployment Guide before proceeding with this Orchestration Server guide. That document contains information about the Genesys software you must deploy before deploying Orchestration server.

Prerequisites

Orchestration server has a number of prerequisites for deployment. Read through this section before deploying your Orchestration Server.

Framework Components

You can only configure ORS after you have deployed the Configuration Layer of Management Framework as described in the Management Layer User's Guide. This layer contains DB Server, Configuration Server, Configuration Manager, and, at your option, Deployment Wizards. If you intend to monitor or control ORS through the Management Layer, you must also install and configure components of this Framework layer, such as Local Control Agent (LCA), Message Server, Solution Control Server (SCS), and Solution Control Interface (SCI), before deploying ORS. Refer to the Framework 8.1 Deployment Guide for information about, and deployment instructions for, these Framework components.

When deploying ORS 8.1.3 or later, Local Control Agent and Solution Control Server version 8.1.2 or later are required.

Orchestration Server and Local Control Agent

To monitor the status of Orchestration Server through the Management Layer, you must load an instance of Local Control Agent (LCA) on every host running Orchestration Server components. Without LCA, Management Layer cannot monitor the status of any of these components.

Persistent Storage

Determine whether you will use persistent storage (Apache Cassandra). If you chose to do so, then installing Cassandra should be performed as the first step before you deploy ORS. See the Cassandra Installation/Configuration Guide.

Supported Platforms

For the list of operating systems and database systems supported in Genesys releases 8.x. refer to the Genesys System-Level Guides, such as *Supported Operating Environment Reference Guide* and

Interoperability Guide on the Genesys documentation website at docs.genesys.com.

Task Summary: Prerequisites for ORS Deployment

Objective	Related Procedures and Actions
Deploy Configuration Layer and ensure that Configuration Manager is running.	See the Framework 8.1 Deployment Guide for details.
Deploy Network objects (such as `Host` objects).	See the Framework 8.1 Deployment Guide for details.
Deploy the Management Layer.	See the **Framework 8.1 Deployment Guide** for details. Also see the Management Layer User's Guide.
Deploy Local Control Agent on every host where Orchestration Server components to be running.	See the Framework 8.1 Deployment Guide for details. Also see the Management Layer User's Guide.
Deploy persistent storage.	See the Cassandra Installation and Configuration Guide.

About Configuration Options

Configuring Orchestration Server is not a one-time operation. It is something you do at the time of installation and then in an ongoing way to ensure the continued optimal performance of your software. You must enter values for Orchestration Server configuration options on the Options tab of your Orchestration Server Application object. The instructions for configuring and installing Orchestration Server that you see here are only the most rudimentary parts of the process. You must refer extensively to the configuration options section of this document.

Familiarize yourself with the options. You will want to adjust them to accommodate your production environment and the business rules that you want implemented.

Deployment Tasks

You can configure ORS entirely in Configuration Manager or in Genesys Administrator. This chapter describes ORS configuration using Configuration Manager.

The table below presents a high-level summary of ORS deployment tasks.

Task	Related Action or Procedure
In Configuration Manager, import the `Application` Template for ORS. The file name is `OR_Server_813.apd`.	If you need help with this task, consult the Configuration Manager Help available on the Management Framework page. See Configuration Database Objects > Environment > Application Templates.
Create the ORS `Application` object. Configure **Server Info** and **Tenants** tabs.	Configure the **Server Info** tab: • **Host**: Select name of the `Host` where the ORS Application will be running.

	• **Port**: Enter the available Listening Port of ORS. Configure **http port**: • **Port ID**: `http` • **Communication port**: Enter any available port. • **Connection protocol**: select `http`. Configure the **Tenants** tab: Set up the list of `Tenants` that ORS works with. See the section on Creating the ORS Application object.
Create a connection for the ORS `Application` object to the following servers: • T-Server(s) • Universal Routing Server • Interaction Server (if needed for multimedia)	Use the **Connection** tab of the `Application` object to set up connections to other servers. See Creating the ORS Application object.
Configure the ORS Application to work with persistence storage.	Go to the Orchestration Server `Application` object > **Options** tab > `persistence` section: `cassandra-listenport`: Set to the value of Cassandra port. `cassandra-nodes`: Enter semi-colon separated list of host names of Cassandra nodes in cluster. `cassandra-keyspace-name`: Specify the name of Cassandra keyspace. `cassandra-schema-version`: Enter Cassandra schema version. `cassandra-strategy-class`: Set to `SimpleStrategy` if Cassandra is deployed as a single cluster, Set to `NetworkTopologyStrategy` in the case of Data Centers Cassandra cluster deployment. `cassandra-strategy-options`: Set the replication factor for a given keyspace. Examples: If `SimpleStrategy`: `cassandra-strategy-options = replication_factor:2` If `NetworkTopologyStrategy`: `cassandra-strategy-options = DC1:2;DC2:3|`
Configure ORS cluster.	See Clustering.
Manually loading an SCXML application on objects other than ORS.	See the following: Manually Loading an SCXML application on a DN. Manually Loading an SCXML Application on an Enhanced Routing Script object. Configuring the ApplicationParams Section of an Enhanced Routing Script Object.

Configure the ORS Application to work with multimedia interactions (if needed).	Go to the Orchestration Server `Application` object, **Options** tab, `orchestration` section: `mcr-pull-by-this-node`: Set to `true` if the ORS `Application` should work with multimedia interactions. Define the Orchestration default interaction queue as described below. Also see: Configuring eServices Application with Type T-Server.
Set a Redundancy type (if needed).	In **Server Info** tab: **Primary server**: **Redundancy** type field, set to `Warm Standby`. **Backup server**: select the backup server application **Backup server**: **Redundancy type** field, set to `Warm Standby` See High Availability.
Add the ORS Application into the ORS cluster.	See Configuring an ORS Cluster.
Install Orchestration Server.	See Installation.
Configure the Universal Routing Server `Application`.	In the `default` section, configure the following: option: `strategy`: Set to `ORS`.
Test your ORS Deployment.	See Debugging SCXML Applications with Composer.

Note: The above ORS Deployment Tasks assumes you have installed/configured any other Genesys components which interact with Orchestration Server, for example, T-Server/SIP Server, Stat Server, Universal Routing Server, Interaction Server (if needed), Composer (if needed), Genesys Administrator (if needed), Genesys Voice Platform (if needed).

Creating the ORS Application Object

1. In Configuration Manager, select **Environment** > **Applications**.
2. Right-click either the `Applications` folder or the subfolder in which you want to create your `Application` object.
3. From the shortcut menu that opens, select **New** > **Application**.
4. In the **Open** dialog box, locate the template that you just imported, and double-click it to open the ORS `Application` object. For Configuration Server versions before 8.0.3, the **Type** field will display `Genesys Generic Server`.
5. Select the **General** tab and change the `Application` name (if desired).
6. Make sure that the **State Enabled** check box is selected.
7. In a multi-tenant environment, select the **Tenants** tab and set up the list of `Tenants` that use ORS. Note: Order matters. The first `Tenant` added will become the default `Tenant` for Orchestration Server. Please ensure that the list of Tenants is created in the same order for both Orchestration Server and

Universal Routing Server.

8. Click the **Server Info** tab and select the following: **Host**, select the name of the Host on which ORS resides; **Ports**, select the Listening Port. Note that a default port is created for you automatically in the `Ports` section after you select a Host. Select the port and click **Edit Port** to open the **Port Info** dialog box.
9. Enter an unused port number for the **Communication Port**. For information on this dialog box, see the Port Info Tab topic in the *Framework 8.1 Configuration Manager Help*.
10. For Web Service access to this ORS Application, configure the HTTP port:
    - In the **New Port Info** dialog box, in **Port ID**, enter `http`.
    - For **Communication Port**, enter an unused port number.
    - For **Connection Protocol**, select `http` from the drop-down menu list.
    - Click **OK**.
11. Select the **Start Info** tab and specify the following:
    - **Working Directory**, enter the `Application` location (example: `C:/GCTI/or_server`)
    - **Command Line**enter the name of executable file (example: `orchestration.exe`).

    Note: If there is a space in the ORS `Application` name, you must place quotation marks before and after the name of the ORS `Application`.

    - **Command Line Arguments**, enter the list of arguments to start the `Application` (example: -host <name of Configuration Server host> -port <name of Configuration Server port>-app <name of ORS Application>

    Note: If you are using Configuration Server Proxy and do not use Management Layer, enter the name of Configuration Server proxy for host and the port of the Configuration Server Proxy.

    - **Startup time**, enter the time interval the server should wait until restarting if the server fails.
    - **Shutdown time**, enter the time interval the server takes to shut down.
    - **Auto-Restart setting**, selecting this option causes the server to restart automatically if the server fails.
    - **Primary setting**, selecting this option specifies the server as the primary routing server (unavailable).
12. Select the **Connections** tab and specify all the servers to which ORS must connect
    - T-Server
    - Interaction Server
    - Universal Routing Server

Configuring an ORS Cluster

ORS provides a new configuration `Transaction` of the type `List`, called `ORS` in the `Environment` tenant , to determine the ORS cluster configuration. Each section in the List represents a single Orchestration cluster. Each of the key/value pairs in that section links a specific Orchestration application to a Data Center.

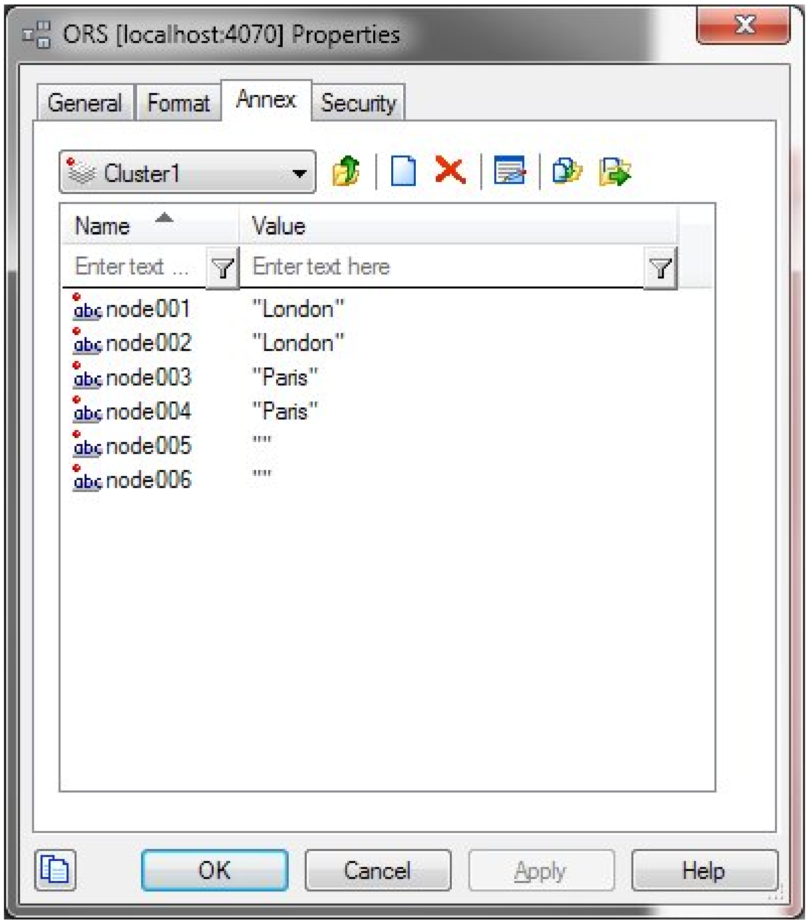
Notes:

- All ORS nodes with the Data Center set to an empty string will belong to one "nameless” Data Center.
- ORS 8.1.3 and later requires creating an ORS `Transaction List` even if the deployment has only one ORS node.

Adding an ORS Application to Cluster and Data Center

Configure each section to represent a single Orchestration cluster, and each of the key/value pairs to link a specific Orchestration Application to a Data Center.

1. In Configuration Manager, select the Tenant `Environment` and navigate to the `Transactions` folder.
2. Right-click inside the `Transactions` window and select **New** > **Transaction** from the shortcut menu.
3. On the **General** tab, enter the following information:
   - **Name:** ORS
   - **Alias:** ORS
   - **Type:** List (pulldown menu)
   - **Recording Period**: 0
   - **State Enabled** should be checked.
4. Click the **Annex** tab to enter the cluster information.Right-click inside the Section window and select **New** from the shortcut menu. Enter the name of your cluster.
5. Double-click on the cluster name.
6. Right-click inside the **Section** window and select **New** from the shortcut menu.
7. In the **Option Name** field, enter the name of an Orchestration application configured as Primary.
8. In the **Option Value** field, enter the name of the Data Center associated with the Orchestration Node.
9. Click **OK** to save.
10. Repeat Steps 7 - 10 for all Orchestration Nodes that belong to this cluster.
11. Click on **Up One Level**.
12. Repeat Steps 5 - 12 for all clusters.
13. Click **OK** to save and exit. An example is shown below.



Name	Value
node001	"London"
node002	"London"
node003	"Paris"
node004	"Paris"
node005	""
node006	""

In the above example, `Cluster1` consists of six nodes presented by Primary instances of Orchestration Servers:

- `node001` and `node002`, which are linked to Data Center London.
- `node003` and `node004`, which are linked to Data Center Paris.
- `node005` and `node006`, which are linked to a "nameless” Data Center.

When a Data Center value is left empty, the nodes default to a "nameless" Data Center.

Notes:

- In ORS 8.1.3 and later, work allocation happens automatically, based on the configuration of the cluster described above.
- ORS 8.1.3 and later requires creating an ORS `Transaction List` even the deployment has only one ORS node.

Manually Loading an SCXML Application on a DN

This section describes manually loading an SCXML application on a `DN`. The following types of DNs can be configured: `Extension`, `ACD Position`, `Routing Point`. See DN Level Options.

1. In Configuration Manager, select the appropriate `Tenant` folder, `Switch` name, and `DN` folder.
2. Open the appropriate `DN` object.
3. Select the **Annex** tab.
4. Select or add the `Orchestration` section.
5. Right-click inside the **Options** window and select **New** from the shortcut menu.
6. In the resulting **Edit Option** dialog box, in the **Option Name** field, type `application`.
7. In the **Option Value** field, type the URL of the SCXML document to load.
8. Refer to the `application` option description for a full description of this configuration option and its valid values.
9. Click **OK** to save

Manually Loading an SCXML Application on an Enhanced Routing Script

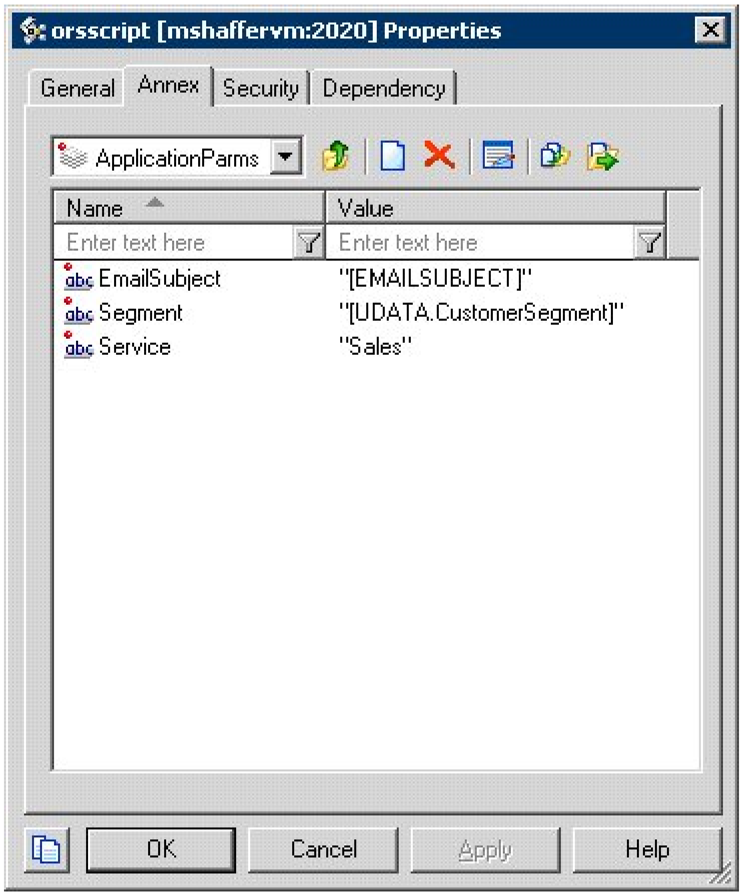
See Enhanced Routing Script Options.

1. In Configuration Manager, select the appropriate `Tenant` and navigate to the `Scripts` folder.
2. Open the appropriate `Script` object of type `Enhanced Routing Script` (`CfgEnhancedRouting`).
3. Select the **Annex** tab.
4. Select or add the `Application` section.
5. Right-click inside the options window and select **New** from the shortcut menu.
6. In the **Option Value** field, create the `url` option.
7. Refer to the `url` option description in the `Application` section for a full description of this configuration option and its valid values.
8. Click OK to save

In addition, an option can be used to specify a string that represents a parameter value that is to be passed to the Application. The `ApplicationParms` section contains the values for data elements that can be referred to within the SCXML application. The `Enhanced Routing Script` object is named as such to identify SCXML applications and Routing applications. Existing IRD-based IRL applications are provisioned as `Script` objects.

Configuring the ApplicationParms Section of an Enhanced Routing Script Object

1. In Configuration Manager, select the appropriate `Tenant` and navigate to the `Scripts` folder.
2. Open the appropriate `Script` object of type `Enhanced Routing Script` (`CfgEnhancedRouting`).
3. Select the **Annex** tab.
4. Select or add the `ApplicationParms` section.
5. Right-click inside the options window and select **New** from the shortcut menu.
6. In the resulting **Edit Option** dialog box, in the **Option Name** field, type a name for the `parameter` option.
7. In the **Option Value** field, type the value for the option. **Note:** Refer to the option description for `{Parameter Name}` for a full description of this configuration option its valid values. The table *Parameter Elements for ApplicationParms* provides useful information about parameters that can be added. The figure below shows an example of the use of the ApplicationParms section.



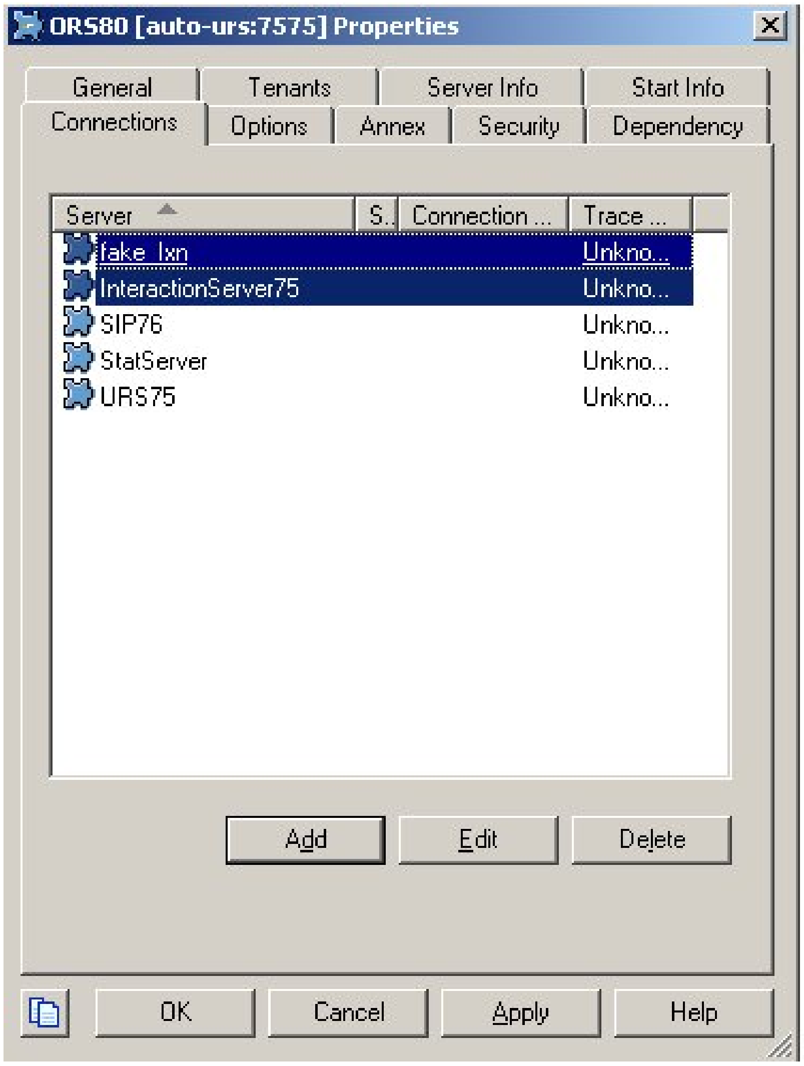
Name	Value
EmailSubject	"[EMAILSUBJECT]"
Segment	"[UDATA.CustomerSegment]"
Service	"Sales"

8. Click **OK** to save.

9. Repeat from Step 5 to add another option in this section.

Interacting with eServices

In your Configuration Environment, the ORS `Application` as a client of Interaction Server, must have two separate connections to each Interaction Server `Application`. As shown below, Interaction Server is represented by two `Application` objects in the ORS **Connections** tab.



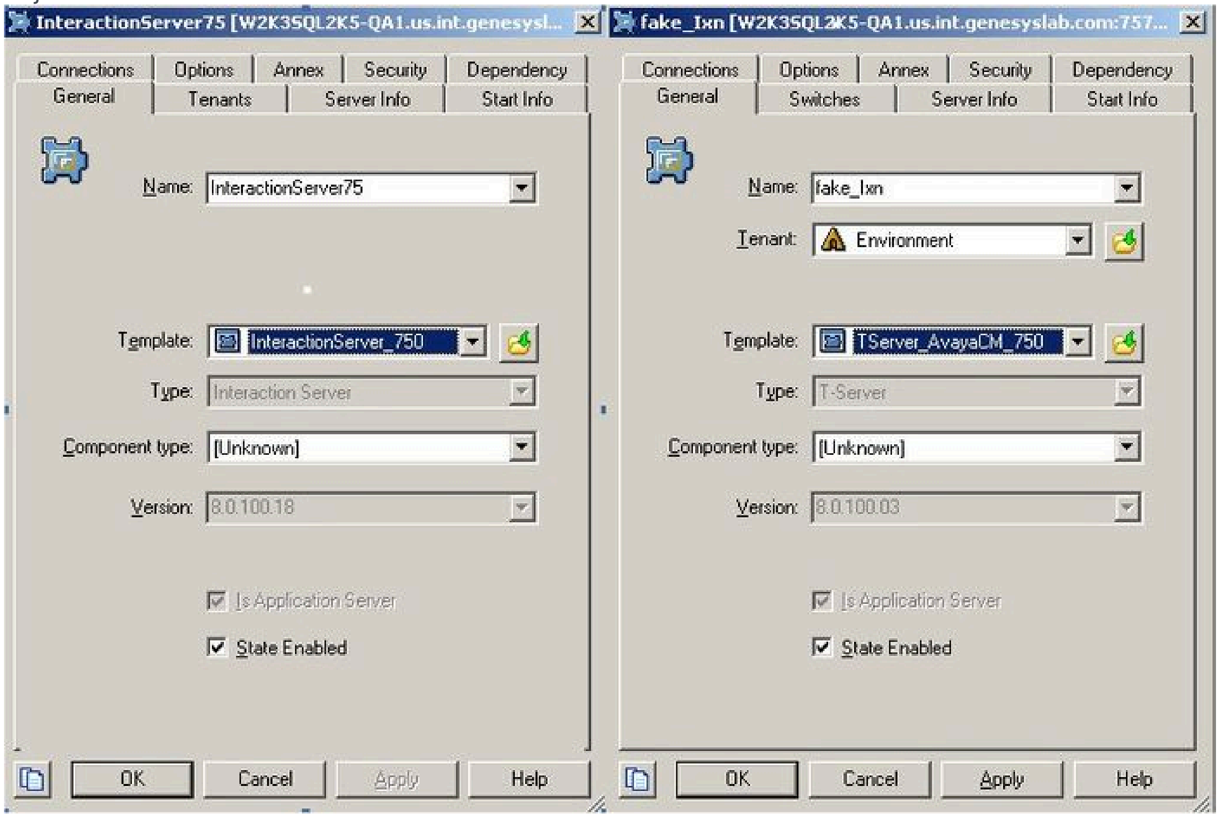
Server	S.	Connection ...	Trace ...
fake_Ixn			Unkno...
InteractionServer75			Unkno...
SIP76			Unkno...
StatServer			Unkno...
URS75			Unkno...

You can think of the first Interaction Server `Application` object as a server and and the second object as its proxy. You need only install and operate one instance of Interaction Server: the server instance. It is not necessary to install the proxy instance because it does not run as an application.

Configuring eServices Application with Type T-Server

To enable ORS to operate with eServices interactions by configuring two Interaction Server Application objects.

1. Import two templates to create the Interaction Server Application objects.
   -
   - Use an Interaction Server Application template for the Application object that you are using for the actual installation.
   - Use a T-Server Application template (with its default configuration) for the second Application object.

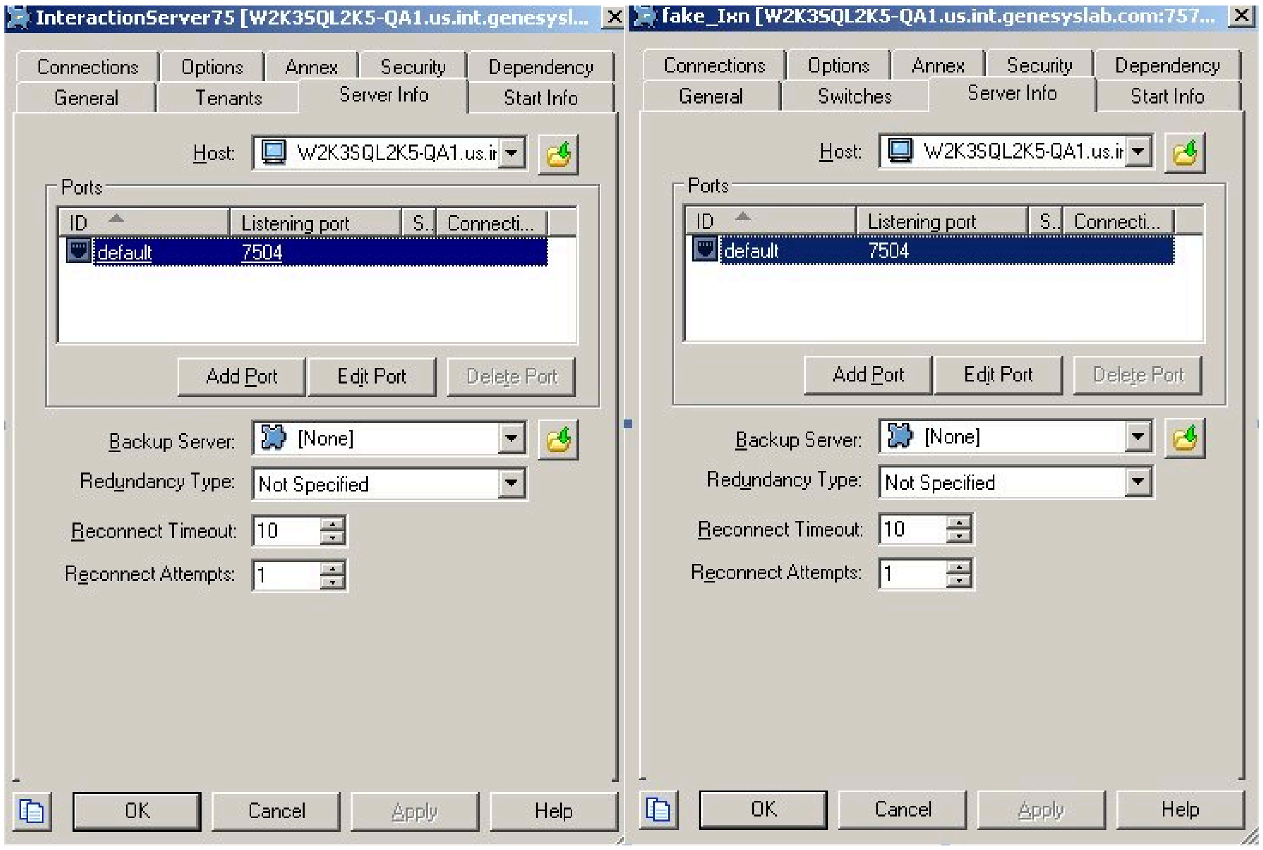

InteractionServer75 [W2K3SQL2K5-QA1.us.int.genesysl...

Name: InteractionServer75
Template: InteractionServer_750
Type: Interaction Server
Component type: [Unknown]
Version: 8.0.100.18
Is Application Server: ✓
State Enabled: ✓

fake_Ixn [W2K3SQL2K5-QA1.us.int.genesyslab.com:757...

Name: fake_Ixn
Tenant: Environment
Template: TServer_AvayaCM_750
Type: T-Server
Component type: [Unknown]
Version: 8.0.100.03
Is Application Server: ✓
State Enabled: ✓

2. Create the Interaction Server Application objects. Ensure that both of the Interaction Server Application objects have different Application object names (see figure above).
3. In the second Application object (based on the T-Server template), specify multimedia switch in the **Switches** Tab. Note: The first Interaction Server Application (created by using the Interaction Server template) has no association in its properties with a multimedia switch. ORS determines this association, based on the Tenant that is specified in **General** tab of the ORS Application. Alternatively, the second the Interaction Server Application (based on T-Server template) must have a multimedia switch configured in its properties, and it must be the same one that is used by the first Interaction Server Application.
4. As shown in the figure below, on the **Server Info** tab of each Interaction Server Application, configure the same port number and ensure both Applications exist on the same Host.



InteractionServer75 [W2K3SQL2K5-QA1.us.int.genesysl...

Host: W2K3SQL2K5-QA1.us.ir

Ports

ID	Listening port	S..	Connecti...
default	7504		

Backup Server: [None]

Redundancy Type: Not Specified

Reconnect Timeout: 10

Reconnect Attempts: 1

fake_Ixn [W2K3SQL2K5-QA1.us.int.genesyslab.com:757...

Host: W2K3SQL2K5-QA1.us.ir

Ports

ID	Listening port	S..	Connecti...
default	7504		

Backup Server: [None]

Redundancy Type: Not Specified

Reconnect Timeout: 10

Reconnect Attempts: 1

5. Save the configuration.

Orchestration Default Interaction Queue

The name of the Orchestration Server default interaction queue is `orchestration_system`. It should be created manually under each Tenant that contains an eServices deployment that the Orchestration Server works with.

Note: The Orchestration SCXML application should not be provisioned for this interaction queue, either directly (via the `orchestration` section in the Annex) or indirectly (via an Interaction Submitter > Interaction View).

Usage

ORS uses the `orchestration_system` queue as a value of the queue attribute, if that attribute is not explicitly defined in the `<ixn:createmessage>` action. Also, if queue was not explicitly defined in that action, ORS will automatically request the selected media server to send a "created" message and to clean up the interaction created by the media server for that message.

The processing of `<ixn:createmessage>` action in ORS if the queue attribute is not defined is as follows:

1. ORS will send an ESP request to create a message of the specified type to the server, defined in the `<ixn:createmessage>` action, with parameter `“Queue“ = “orchestration_system“`.
2. As soon as an ACK response is received with new `InteractionID`, ORS will automatically send an ESP request to send the created message to the server that is defined in the `<ixn:createmessage>` action.

Automatic clean-up of interactions, submitted into interaction queue `orchestration_system`, is as follows:

1. All ORS nodes constantly monitor events for interaction queue `orchestration_system`.
2. Upon processing of `EventInteractionSubmitted`, each node will determine the node that "owns" the interaction by the content of `KVPair "ORSI:...`" in the User Data.
3. The ORS node that owns that interaction will automatically pull it from the `orchestration_system` queue by `InteractionID`.
4. As soon as the interaction is pulled, the ORS node will request Interaction Server to stop processing that interaction and the interaction will be discarded.

ORS Features

This section describes clustering, persistence, high availability (HA), load balancing, and support for multiple Data Centers. It also provides examples of ORS 8.1.3 deployment models.

- Persistence
- Clustering
- High Availability
- Load Balancing
- Multiple Data Centers

Persistence

To achieve persistence, ORS leverages the capability to store and retrieve information for SCXML sessions and SCXML documents. This includes the SCXML session data and the SCXML session state, as well as scheduled activities (such as start and stop).

Whenever required, these stored sessions or documents that have been persisted are fully recovered and used as needed. In some cases, extended sessions are utilized for long durations of time, up to several months or more. ORS 8.1 and later releases use persistence storage exclusively in conjunction with third-party Apache Cassandra (NoSQL Datastore).

The key function of persistence is supporting the retrieval of SCXML session information to restore the session context, as required. Session context is restored, for example, upon:

- a restart of the ORS component.
- the arrival of an event for an SCXML session, which was previously deactivated to reduce the memory required.

Storage and Retrieval Design and Configuration

The key principles of ORS persistence storage, retrieval design, and configuration are as follows:

- During configuration, all ORS instances within a single cluster must be configured to work with the same persistent storage, so that each Orchestration instance has access to all session information.
- The persistence layer allows storage and retrieval of data related to active Orchestration sessions. The amount of stored data is sufficient to restore an Orchestration session fully, and continue its execution.
- SCXML sessions are persisted upon the request by ORS application and only when the last queued event is flushed.
- Additional information is also persisted for ORS operations:
- A list of actions scheduled to be executed at specific times.
- Information that identifies the current SCXML session being handled by an ORS instance.

Persistent Storage Connection

ORS 8.1.2 and earlier releases required a connection to Cassandra. If ORS detects a lost connection, it exited. ORS 8.1.3 and later do not require a connection to Cassandra. If a connection is not present, an alarm is raised.

Be aware that if a connection is not present, the following functionality is not available:

- Session recovery is not available.

- Sending events between sessions may not be available.
- Working with HTTP interface may be unavailable depending on the configuration of the cluster and destination of the message.
- Scheduled session functionality will also not be available.

Persistent Storage Operation

Apache Cassandra provides a decentralized, fault tolerant, elastic, durable, and highly scalable distributed database system. Using Cassandra gives ORS a simpler deployment and installation, and assures scalability and performance of the back-end datastore for high-availability.

Document Persistence

SCXML documents are the basis for session construction. Many sessions may refer to the same document when persisted. The SCXML engine:

- Retrieves the requested document from the specified location. For example, in the Annex tab of a RoutePoint configuration, there would be an Orchestration section with the path to the application, such as: http://test52/Queues_1.scxml.
- Requests the persistence component to store the document information in the persistent storage when the document has been verified and compiled.

The compiled document is serialized and a request to store this document is made to the specified Cassandra cluster.

SCXML Session State and Data-Model Persistence

SCXML sessions are validated and compiled based on an SCXML document, and the session information including the full data model, is stored in the persistent storage. SCXML sessions are persisted at specific points of time to optimize session information integrity as well as system performance. When all events in the event queue have been processed, the SCXML session including the data model is serialized and stored in the persistence storage.

Persistence configuration options are listed in the persistence section, ORS Application object.

Note: Active sessions means that the sessions as defined in the SCXML document have not reached the final state. If this state is reached, the session information is removed from persistent storage.

Persistence Scheduling

SCXML sessions can be started in the future, or hung sessions can be terminated, through the Schedule component.

When ORS receives a request to schedule a session, if the requested time has passed, the session action is processed immediately.

Note: The requested time past is defined as the requested time plus the latency in processing the request. Currently five seconds is allowed for the latency period.

The scheduled session information is serialized and stored in Cassandra within the Keyspace as specified in configuration, in the Schedule SuperColumn.

Session High Availability

For effective High Availability of a session, the Cassandra data model provides persistence by maintaining the following information:

- ORS node currently responsible for processing the session.
- sessionID of the session.

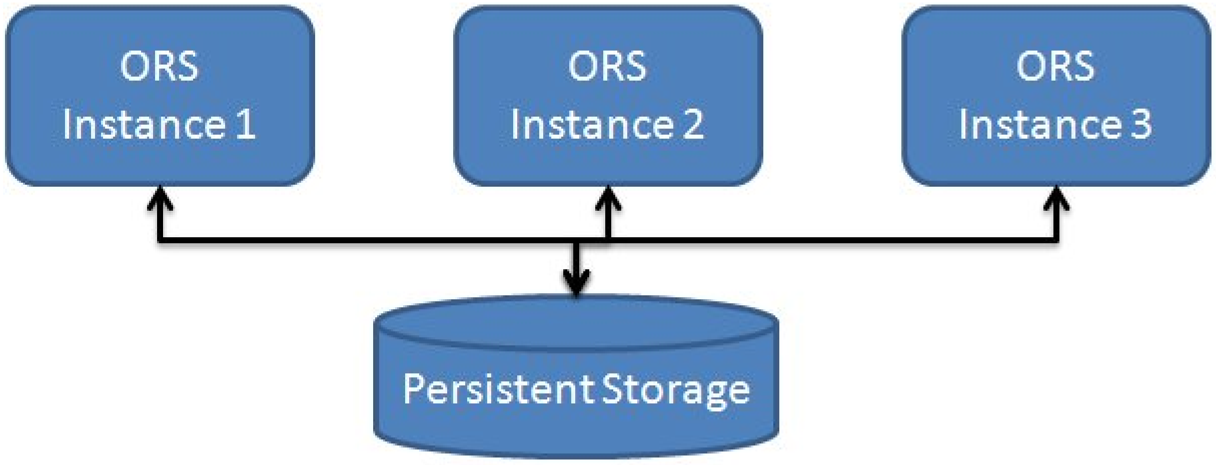
The ORS on which a session is created, persists the sessionID and its Orchestration Node ID, which is the dbid of the Orchestration application.

When the session reaches the state final, the entry for the session is removed from persistence.

The session-to-server node information is placed in the ColumnFamily SessionIDServerInfo, with ColumnName SessionServerInfo with keys of the sessionID. To facilitate the retrieval of the sessions for a given server node, the ColumnFamily SessionIDServerInfoRIndex is employed, with keys that are the string form of the Node ID and Columns that are the sessionids for that node.

Deploying Persistent Storage

In Figure 11, multiple instances of ORS are running and processing sessions and schedules at the same time. Refer to ORS Cluster Architecture for a description of multiple ORS instances configured as clusters. All ORS instances are active, and all work with persistent storage.



Configuring Persistent Storage

Configure persistent storage in the ORS Application object, Options tab, persistence section.

Option Name	Value
`cassandra-listenport`	Set to the value of the Cassandra port.
`cassandra-nodes`	Enter a semicolon-separated list of host names of the Cassandra nodes in the cluster.
`cassandra-keyspace-name`	Specify the name of the Cassandra Keyspace.
`cassandra-schema-version`	Enter the Cassandra schema version.
`cassandra-strategy-class`	Set to `SimpleStrategy` if Cassandra is deployed as a single cluster. Set to `NetworkTopologyStrategy` in the case of a Data Center Cassandra cluster deployment.
`cassandra-strategy-options`	Set the replication factor for a given Keyspace.

Examples

- If `SimpleStrategy`, cassandra-strategy-options = replication_factor:2
- If `NetworkTopologyStrategy`, cassandra-strategy-options = DC1:2;DC2:3

Enabling Persistence in the SCXML Application

Enabling persistence for an SCXML application is controlled with the `<scxml>` tag's attribute `_persist`, which has values of `true` and `false`. Starting with release 8.1.2, the default value for this attribute is `false`, therefore persistence for an SCXML application is disabled by default. A value of `true` activates persistence for an SCXML application.

Also see the `persistence-default` option in the scxml section.

When persistence for an SCXML application is enabled, ORS will regularly try to store the current session state into persistent storage.

Important: All nodes of ORS and Cassandra should have some time synchronization enabled and all should have the same time.

Function Behavior During Failover and Restoration

The information below relates to Orchestration persistence and how SCXML applications should be designed to handle failover and restoration.

ORS functions defined within SCXML documents (e.g., via <script> elements) may not behave as expected following an Orchestration failover and session restoration. Specifically, the scope in which a function executes may change following a session restore. For more information, see the section on Functions and `<script>` Elements in the Extensions and Deviations section of the *SCXML Technical Reference*.

High Availability

Starting with version 8.1.3, Orchestration server can operate in a high-availability (HA) configuration, providing you with redundant systems.

The Framework Management Layer currently supports two types of redundant configurations: warm standby and hot standby. ORS offers the warm standby redundancy type.

Warm Standby

Genesys uses the expression "warm standby" to describe a high-availability configuration in which a Backup-server remains initialized and ready to take over the operations of a Primary server.

HA architecture implies the existence of redundant applications. In the case of ORS, HA is achieved with Primary and Backup ORSs. These applications are configured with redundancy so if one fails, the other takes over its operations without significant loss of data or impact to business operations.

ORS Node

A pair of Primary and Backup Orchestration Servers is called an ORS "Node". An ORS Node has a Node ID, which is a DBID of an ORS `Application` configured as a Primary in Configuration Server.

An ORS application running in Backup mode does not:

- Create a new session
- Serve an existing session
- Perform the pulling of multimedia interactions
- Service the HTTP port

Note: Starting with Release 8.1.3, ORS does not support Super Nodes and Master Super Nodes in the ORS Cluster architecture.

HA Deployment and Session Creation

The ORS Node (Primary/Backup pair) that is responsible for a given interaction processing is the Assigned Node. Within the same cluster, an alternative Node (Primary/Backup pair), called a Reserved Node, can be another Node in single Data Center or another Node in another Data Center.

- When receiving an appropriate event on a DN loaded with a routing strategy, the Primary server/ application of the assigned Node starts call processing by trying to create a session.

- A Backup application and a Reserved Node are watching the Primary application during the timeout specified in the `orchestration/call-watching-timeout` option. This continues until the session is created by Primary application (Assigned Node) or until the timeout expired.

If session for a call was successfully created, then the Primary continues the processing of the interaction.

Session Creation Sequence

If the Primary application failed to create a session during the specified timeout or the Backup application starts running in Primary mode, the new Primary will make an attempt to create a session for a given call. If the Assigned Node (Primary/Backup pair) fail to create a session, the Reserved Node will make an attempt to create a session.

Recovery of Failed Sessions

The ORS 8.1.3 and later HA architecture provides the possibility to recover a failed sessions for a given call. When the Management Layer detects failure of a Primary ORS, it starts running the Backup Orchestration server in Primary mode. The new Primary attempts to recover the persisted sessions.

Note: Not all persisted sessions can be recovered and some session events may be missed.

During an active session, when the Backup ORS application becomes a new Primary, the session can be explicitly restored when a proactive recovery mechanism is enabled in an SCXML application. Enabling proactive recovery for an SCXML application is controlled with the `<scxml>` tag attribute `_sendSessionRecovered`, which has values of `true` and `false`.

When set to `true`, ORS publishes a `session.recovered` event to a session. The SCXML application should specify its own transition to handle session recovery. See the *Orchestration Developer’s Guide* for details.

Note: Composer 8.1.3 allows the `_sendSessionRecovered` attribute to be added to the root `<scxml>` element via the `Extensions` attribute of an interaction process diagram. For information on `_sendSessionRecovered`, see SCXML Elements in the SCXML Language Reference. Also see Interaction Process Diagrams.

Limitations for ORS HA Deployment

- In the case of a multimedia interaction, only the Node that pulls an interaction is allowed to work with it. There are no assigned or reserved Nodes in this case.
- A session that was created on one Node cannot be transferred to another Node.

Clustering

An Orchestration solution supports the ability to run multiple nodes of ORS in a single, logical entity called a *cluster*. With ORS cluster support, you can create a scalable architecture in which the system's capacity can be scaled up.

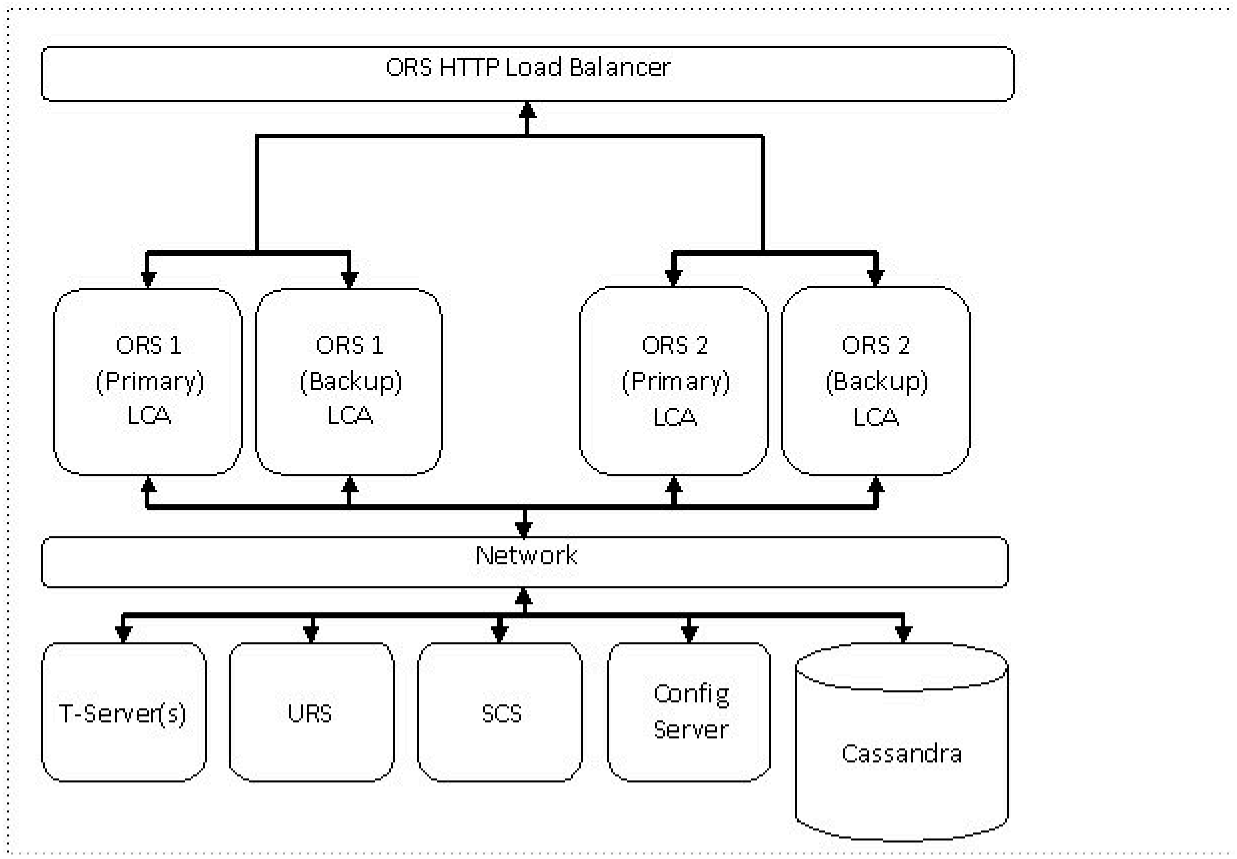
Clustering Requirements

Deployment of ORS cluster requires that:

- Each ORS Node in a cluster should serve the same T-Server/SIP Server.
- Each ORS Node should work with the same persistent storage.
- Each ORS Node should have only one Universal Routing Server in its Connections list.

Note: ORS 8.1.3 cluster deployment allows each ORS Node to have its own Universal Routing Server.

The figure below shows a simple two-Node pair in a cluster.



The above figure illustrates a simple two-ORS Node cluster deployment. It shows four ORS instances configured as two Nodes (Primary/Backup ORS applications). Each Node is running on a separate `Host`. Each node connects to the same T-Server. Each ORS Node connects to the same URS.

Configuring a Cluster/ORS Node

See the section on Configuring an ORS Cluster in General Deployment.

Load Balancing

The ORS **cluster** deployment introduces the possibility to distribute the load of incoming Interactions (voice calls, multimedia and http requests) across all ORS Nodes.

Load Balancing for Voice Interactions

Each Node in cluster serves the same T-Server/SIP Server so all Nodes are aware of all voice calls. However, each Node has responsibility for specific sessions.

The ORS cluster is designed to have all calls distributed across all existing ORS instances uniformly.

Load Balancing for Multimedia Interactions

The nature of the "pulling" mechanism provided by Interaction Server guarantees that new sessions created from this trigger are automatically distributed across the cluster.

Load Balancing of HTTP-Related Interactions

If a session is created via the HTTP interface, the session is created on the ORS running in Primary mode, and the session is assigned to that Node. An HTTP load balancer is placed in front of the ORS cluster and provides load balancing services for HTTP requests.

ORS exposes the RESTful Web Services interface, which is based on HTTP. ORS may accept and execute a set of HTTP requests. This interface uses standard mechanisms of load balancing that are typical for web servers. These mechanisms include:

- DNS-based load balancing
- Hardware-based load balancing.

These methods result in distribution of HTTP requests across all ORS Nodes in a cluster.

Load Balancing Optimization

To enable optimal load-balancing, ORS supports "stickiness" of HTTP sessions. This is achieved by providing a cookie with the SCXML session ID in the HTTP responses. Hardware load balancers may then use this cookie to ensure that HTTP requests are about the same SCXML session.

There could be a scenario when an HTTP request about a specific SCXML session is delivered to an

ORS Node that is not handling this SCXML session. In this case, the ORS Node replies with the HTTP response `307 Temporary Redirect` pointing to the ORS Node that is processing the needed SCXML session. The responsibility of the HTTP Client is to resend the same request to the given URI, which results in the request arriving at the correct Node.

For RESTful interface, you must set up the http port definition under **Server Info** during ORS installation, as described in Creating the ORS Application Object.

SCXML Session Startup Rules

When an existing SCXML session (Session1) starts another SCXML session (Session2), the child SCXML session (Session2) is always started on the same ORS Node as the parent SCXML session (Session1).

When an existing SCXML session (Session1) invokes another SCXML session (Session2), the invoked SCXML session (Session2) is always started on the same ORS Node as the existing SCXML session (Session1).

Load Balancing for RESTful Interface

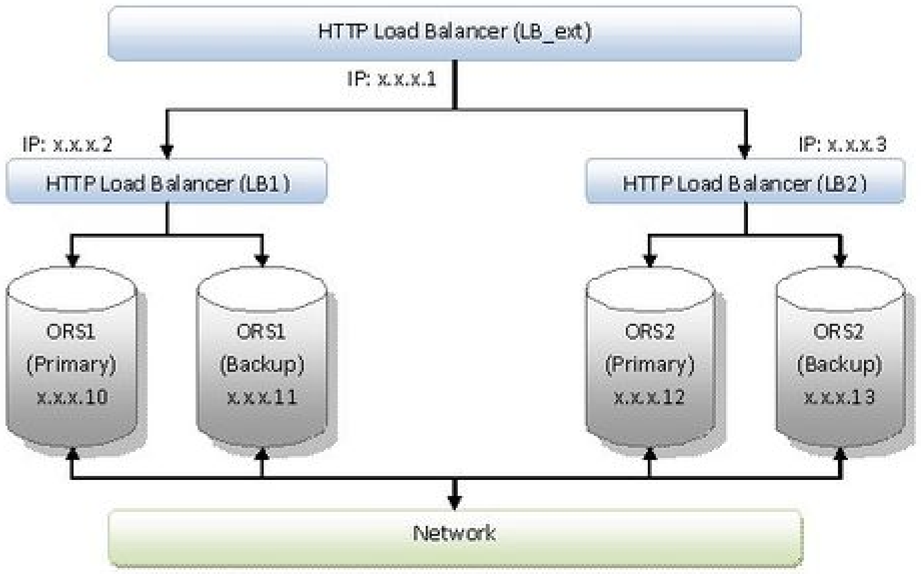
In support of the RESTful web services interface, when ORS accepts and executes a set of HTTP requests, it uses the standard mechanisms of load balancing that are typical for web servers. These mechanisms include:

- DNS-based load balancing
- Hardware-based load balancing

Sessions created via the RESTful web services interface will be assigned to the ORS node that services the initial request. If subsequent HTTP requests targeting the same session (by SCXML session ID) are not delivered to the same ORS Node, they will be redirected with the following priorities:

1. If the recipient of the request is not part of the target ORS Node, it will be redirected to the correct ORS Node using their configured external-url.
2. If the HTTP request was delivered to the ORS instance running in Backup mode (or if the target server has no `external-url` configured), it will be redirected to the ORS instance running in Primary mode by its configured `Host` and port.

Taking the above behaviour into consideration, the following diagram illustrates the recommended deployment of ORS and load balancers:



In the above diagram, ORS1 (both Primary and Backup servers) will have the address of `ORS1 HTTP Load Balancer` configured as the external-url. Similarly, the ORS2 will have `ORS2 HTTP Load Balancer` configured as its `external-url`. The load balancers will distribute HTTP requests across all ORS Nodes in the cluster and the ORS Nodes will automatically ensure that all requests corresponding to one session will be delivered to the correct Node.

To enable optimal load-balancing, ORS supports the following features:

- Session "stickiness" (to ensure requests are delivered to the correct ORS Node).
- Heartbeats/health monitoring (to ensure that requests are not delivered to offline ORS instances or instances running in backup modes).

Configuring Session Stickiness

Session "stickiness" is achieved by providing a cookie with the SCXML session ID in the HTTP responses. Load balancers (such as Apache, or HAProxy) may then use this cookie to ensure that HTTP requests are about the same SCXML session. ORS automatically sets the `Set-Cookie` header in the responses for `Create Session` requests and redirected requests. To configure session “stickiness” in Apache:

- Assuming a deployment similar to the above diagram, in `httpd.conf`, for LB_ext

```
ProxyPass / balancer://orsapcluster1/ stickysession=ORSSESSIONID
<Proxy balancer://orsapcluster1>
        BalancerMember http://apvm1.us.int.genesyslab.com:3482 loadfactor=50
route=<ORS1.node_id>
        BalancerMember http://apvm2.us.int.genesyslab.com:3482 loadfactor=50
route=<ORS2.node_id>
</Proxy>
```

- LB1 (apvm1.us.int.genesyslab.com:3482):

```
<Proxy balancer://node749>
        BalancerMember http://172.21.82.55:7031 route=node749
        BalancerMember http://172.21.82.55:7041 route=node749
</Proxy>
```

- LB2 (apvm2.us.int.genesyslab.com:3482):

```
<Proxy balancer://node753>
        BalancerMember http://192.168.14.245:7041 loadfactor=70 route=node753
        BalancerMember http://192.168.14.245:7031 loadfactor=30 route=node753
</Proxy>
```

- To configure session “stickiness” in HAProxy, assuming a deployment similar to the above diagram, in haproxy.conf, on LB_ext

```
listen my_loadbalancer <my_loadbalancer IP>
        mode http
        appsession ORSSESSIONID len 100 timeout 3h
        server LB_1 <LB_1.address>
        server LB_2 <LB_2.address>
```

- In haproxy.conf, on LB_1

```
listen my_loadbalancer <my_loadbalancer IP>
        mode http
        appsession ORSSESSIONID len 100 timeout 3h
        server ORS1_P <ORS1_P.address>
        server ORS1_B <ORS1_B.address>
```

- In haproxy.conf, on LB_2

```
listen my_loadbalancer <my_loadbalancer IP>
        mode http
        appsession ORSSESSIONID len 100 timeout 3h
        server ORS2_P <ORS2_P.address>
        server ORS2_B <ORS2_B.address>
```

Configuring Server Health Monitoring

ORS provides a special “heartbeat” interface which can be used by external clients to determine whether the ORS instance is operating in Primary or Backup mode. This interface may be accessed via an HTTP GET request to:

http://<ORS_host>:<ORS_HTTP_port>/heartbeat

If the requested ORS instance is running in Primary mode, it will return a `200 OK` response. Otherwise, if the requested ORS instance is running in Backup mode, it will respond with the configured response code as specified in the ORS options (under `orchestration/heartbeat-backup-status`). If no response code is configured ORS will respond with default response code, 503 `Service unavailable`. Load Balancers (such as **F5** or **HAProxy**) can use this interface to avoid redistributing traffic to backup ORS or unresponsive servers. **Note**: The `orchestration/heartbeat-backup-status` option should be configured on both ORS applications-Primary and Backup.

To configure health-monitoring in **F5**:

1. Log into **F5** load balancer web console (or CLI).

2. Go to **Local Traffic** > **Monitors**.
3. Click **Create**.
4. In the new screen, set the **Type** to HTTP.
5. Set the **Send** string to: `GET /heartbeat HTTP/1.0\r\n\r\n`.
6. Set the **Receive** string to: `HTTP/1.[01] 20[0-6]`. The above example matches HTTP status codes 200-206.
7. Save the new monitor.
8. Go to the pool list (assuming that the load balancer pool has already been configured), and apply the monitor to the appropriate pools.

To configure health-monitoring in **HAProxy**:

- Assuming deployment similar to above diagram, in haproxy.conf, on LB_1

```
listen my_loadbalancer <my_loadbalancer IP>
        mode http
        option httpchk GET /heartbeat HTTP/1.1\r\nHost:\ www
        server ORS1_P <ORS1_P.address> check
        server ORS1_B <ORS1_B.address> check
```

- In haproxy.conf, on LB_2

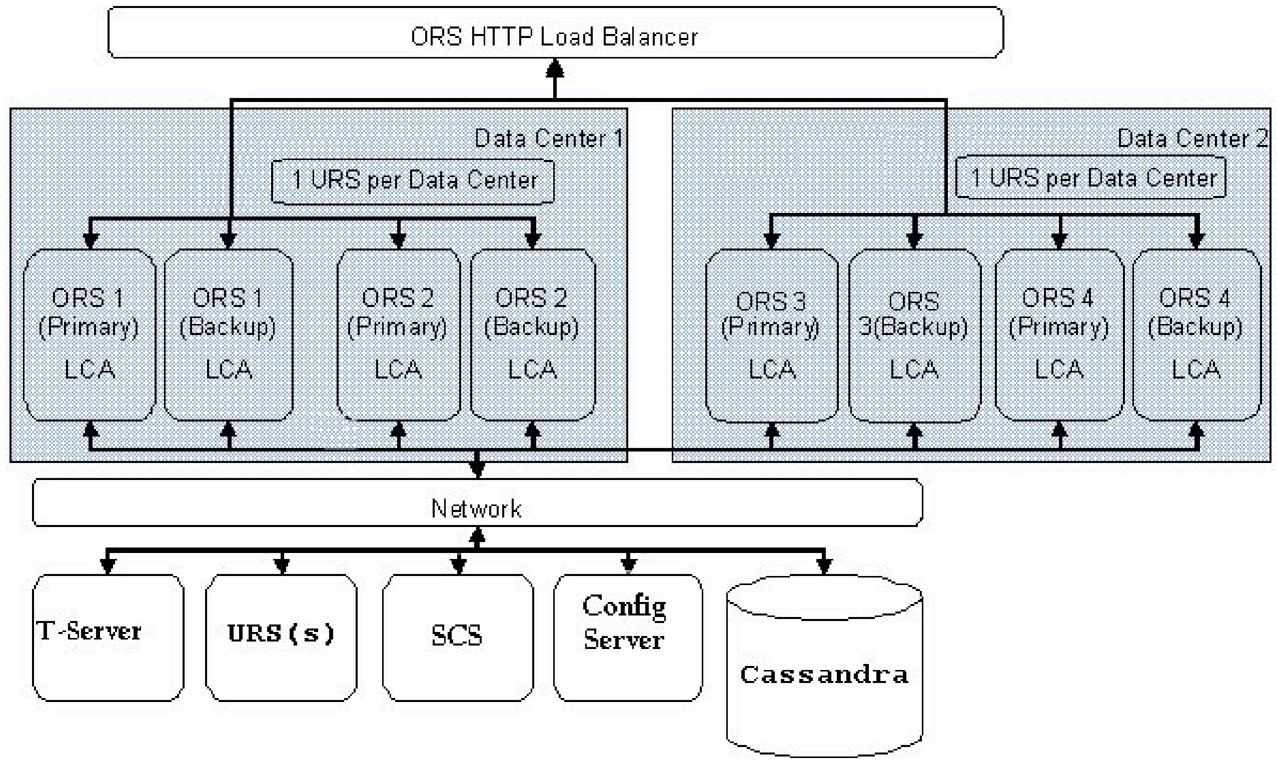
```
listen my_loadbalancer <my_loadbalancer IP>
        mode http
        option httpchk GET /heartbeat HTTP/1.1\r\nHost:\ www
        server ORS2_P <ORS2_P.address> check
        server ORS2_B <ORS2_B.address> check
```

Multiple Data Centers

ORS 8.1.3 and later provides further deployment possibilities and configurations that are supportive of multiple Data Center architectures.

A *Data Center* is a facility used to house computer systems and associated components, such as telecommunications and storage systems. It generally includes redundant or backup power supplies, redundant data communications connections, environmental controls (such as air conditioning, fire suppression) and security devices.

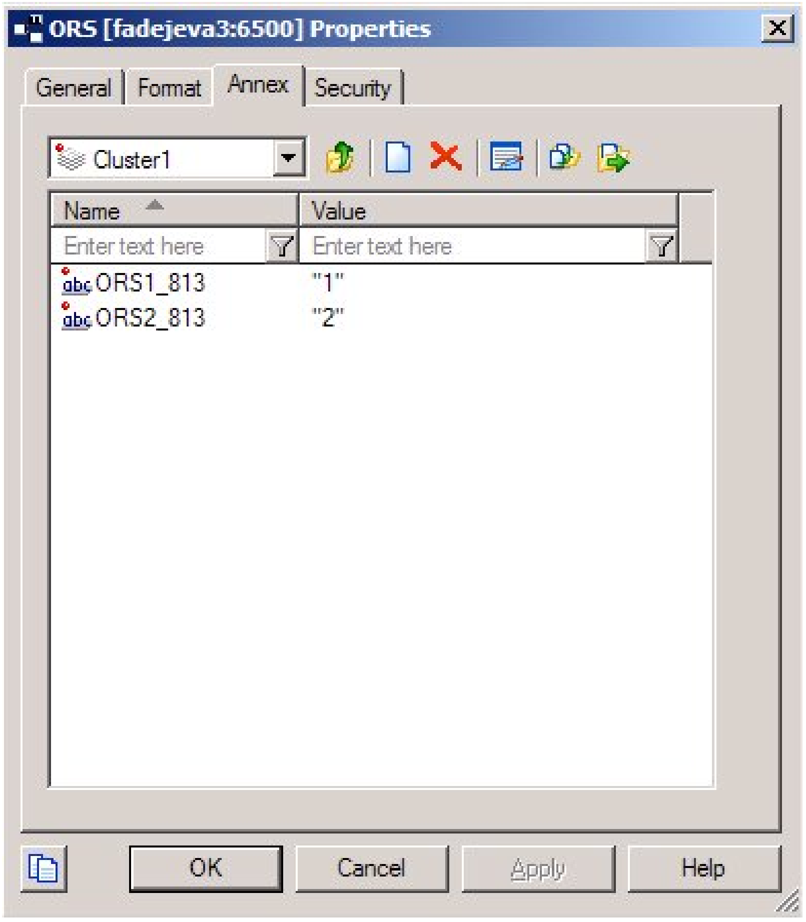
A single Data Center is commonly used. Multiple Data Centers, which are usually geographically separated, are becoming more prevalent. Figure 13 shows an example of multiple Data Centers.



In case of one Data Center failure any sessions currently being processed by this Data Center will be lost but any new sessions will be processed in the second data center.

Note: Genesys recommends having a Primary and Backup pair of Universal Routing Servers configured for each Data Center.

When operating in a cluster environment with one or more Data Centers, ORS obtains the list of ORS applications configured in the cluster from the ORS `Transaction List` which is configured under the `Environment` Tenant in a multi-tenant deployment and under the `Resources` Tenant in single Tenant deployment. For a particular cluster, ORS uses the value of the Primary ORS application to identify ORSs that belong to a particular Data Center.



Name	Value
ORS1_813	"1"
ORS2_813	"2"

Configuration Options

ORS options are placed in the following option folders:

- For the ORS `Application` object, in the `orchestration`, `persistence`, `scxml`, `log`, and `mcr` sections of the `Options` tab of the ORS `Properties` dialog box.
- For DN (`Extension` or `Route Point`), or `Interaction Queue` object, in the `Orchestration` section of the `Annex` tab of the appropriate `Properties` dialog box.
- For `Script` objects of type `Enhanced Routing`, in the `Application` or `ApplicationParms` section of the `Annex` tab of the appropriate `Properties` dialog box.

For detailed option descriptions, see:

Application Level Options
Common Log Options
Switch gts Options
DN Level Options
Enhanced Routing Script Options

Application Level Options

orchestration Section

This section describes `orchestration` section options.

call-watching-timeout

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `5000`

Valid values: `1000 - 300000`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the maximum amount of time in milliseconds that a reserved ORS node waits for the assigned node to start call processing. If the assigned node fails to create a session during this timeout, the reserved node attempts to create a session.

For example: `call-watching-timeout=5000`

external-url

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: None

Valid values: Any valid URL

Value changes: For the next interaction

Use to redirect HTTP requests in scenarios when an HTTP request about a specific session is delivered to an ORS node that is not handling the session. This URL will be populated into the Location header of a `307 Temporary Redirect` response as a redirect target.

Configure this option on the Primary ORS Application object. Set its value to the URL of Load Balancer located in front of the ORS node. Another ORS node will use this option and populate its value into the Location header of a `307 Temporary Redirect` response as a redirect target. It allows delivery of the HTTP request to the session owner.

functions-by-urs

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `true`

Valid values: `true` or `false`

Value changes: Immediately (starting with ORS Release 8.1.300.39)

This option was introduced in ORS Release 8.1.300.36.

Orchestration Server allows you to retrieve information directly from the Configuration Layer without sending requests to Universal Routing Server when using the following _genesys.session functions: `isSpecialDay, timeInZone, dateInZone, dayInZone, listLookupValue,` and `getListItemValue`.

To enable this functionality, use the option `functions-by-urs`.

If set to `true`, Orchestration Server sends requests to Universal Routing Server.

If set to `false`, Orchestration Server retrieves information directly from Configuration Layer.

Note: The `get-list-item-by-urs` option, which covers only one function, is obsolete and should not be used. It is replaced by the `functions-by-urs` option, which covers six functions.

get-list-item-by-urs

This option, which covers only one function, is obsolete and should not be used. It is replaced by the `functions-by-urs` option, which covers six functions.

heartbeat-backup-status

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `503`

Valid values: Valid HTTP response codes

Value changes: Immediately.

This option, introduced in ORS Release 8.1.300.32, allows you to define a valid HTTP response code sent by an ORS instance running in Backup mode to the requestor, after receiving a `heartbeat` query.

This option can be used to provide HTTP health monitoring support to determine whether the ORS instance is operating in Primary or Backup mode.

When an ORS instance running in Primary mode receives a specific GET request for `/heartbeat URI` in the format `http://<ORS_host>:<ORS_HTTP_port>/heartbeat`, it responds with `200 OK`.

If an ORS instance is running in Backup mode, it responds with `503 Service Unavailable` or with the HTTP response code defined in the new configuration option `heartbeat-backup-status`.

Genesys recommends configuring health monitoring in deployments with Load Balancing.

http-pending-max-time

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `600`

Valid values: Any non-negative integer between `1` and `600`

Value changes: Takes effect after restart.

Orchestration Server supports pipelining of HTTP requests according to RFC 2616, Section 8.1.2.2. As a result, when ORS receives several HTTP messages over the same connections, the responses are in the same order. This option and `http-orphan-section-action` implement this functionality.

This option, introduced in Release 8.1.300.43, defines the maximum time, in seconds, that Orchestration Server will process an HTTP request before sending a timeout response (HTTP message with status code 408) to the client.

http-orphan-session-action

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: none

Valid values: `none, terminate, <valid scxml event name>`

Value changes: Immediately upon notification.

This option, introduced in ORS Release 8.1.300.43, defines the Orchestration Server action if a response to an HTTP request to create an SCXML session cannot be sent to the client or when a timeout message is already sent. Valid values:

- none: No action will be taken.
- terminate: The SCXML session will be terminated.
- `<valid scxml event name>`: The event will be sent to the SCXML session.

See `http-pending-max-time` option for additional information on this option.

log-trace-segments

Option section: `orchestration`

Default value: `all/All`.

Valid values: Any combination of comma-separated segment names, case sensitive: `ORSInternal, ORSInit, ORSSynch, Cluster, ItxLink, ItxManager, CallMonitor, ORSURS, Persistence, Schedule, SessionManager, ThreadSync, ScxmlCommon, ScxmlLog, ScxmlMetric, ScxmlQOS, ORSScxml, ScxmlConfig, ScxmlIxnAction, ScxmlOrsAction, DFM,` `ScxmlEvent, FMClassification, FMDialog, FMInteraction, FMQueue, FMResource, FMSession, FMStat, FMWeb, ScxmlProperty, ScxmlIO`

Valid changes: Take effect immediately

This option, introduced in 8.1.300.52, provides more precise control of debug log out by allowing you to define the specific type of debug messages that will be printed into the log files. When set to `all` or `ALL` (the default), all debug messages are printed with the log level defined by the `log/x-server-trace-level` option. To control the log levels for each segment, use syntax: <segment_name>:<log_level>. If <log_level> is not specified or invalid, ORS applies the default value (defined in `x-server-trace-level`). If an unknown segment is specified, ORS ignores it without generating an error.

The debug segment header {<segment_name>:<log_level>} is now added right after the timestamp:

```
10:31:02.249 {ScxmlMetric:3} METRIC <event_queued
sid='ML06K6519D5UPAGSHNE41VJBSS00000M' name='interaction.added' type='external'
thread='15800' />
```

Examples:

SETTING	DESCRIPTION
all, ORSInternal:0, ORSURS:0	Everything is enabled except ORSInternal and ORSURS
ScxmlIO,ThreadSync	Only ScxmlIO and ThreadSync are enabled
all,ORSURS	Everything is enabled (the same as all)
ORSURS:0	Everything is disabled (the same as empty)
CallMonitor,all:0	Everything is disabled (the same as empty)
all,ORSURS:1	Everything is enabled with the default trace level, but the trace level for the ORSURS segment will be changed to 1
(empty)	Everything is disabled (the same as all:0, or x-server-trace-level=0)
all:3	The same as all with x-server-trace-level=3

Initial Setup:

Run ORS 8.1.300.53 or higher with full debug logging enabled for at least one session processed. Then look at the log messages you need – they all contain a prefix with the segment name and message level, so you will immediately know to configure that particular segment in `log-trace-segments`. For example, if you determine that you need only messages from the `ScxmlMetric` segment with level 1 or 2, then set that option as `ScxmlMetric:2`. If then you determine that you also need messages from the `Persistence` segment with level 1 only, add it to that option `ScxmlMetric:2, Persistence:1`, and so on.

max-session-create-time

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `3600`

Valid values: Any integer between `5` and `3600`

Value changes: Immediately

This option, introduced in ORS Release 8.1.300.45, defines the maximum session creation time in seconds. If session creation will take longer, ORS will enter an overload mode and stop creating sessions until all pending session creation requests are completed.

map-composer-log-levels

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true` or `false`

Value changes: Immediately

This option, introduced in ORS Release 8.1.300.29, allows mapping of log levels defined in the `<log>` action element into the specific Genesys log events. If enabled (set to `true`), Orchestration Server will generate log events of the `Alarm, Standard, Interaction, Trace or Debug` levels. The log levels defined in the level attribute of the `<log>` element are mapped as follows:

- `Info (log level 1)` is mapped into the Trace log event 23023
- `Error (log level 2)` is mapped into the Standard log event 23011
- `Warning (log level 3)` is mapped into the Interaction log event 23022
- `Debug (log level 4)` is mapped into the Debug log event 23026
- `Alarm (log level 5)` is mapped into the Alarm log event 23012

Setting the `scxml-log-filter-level` option value to `1, 2, 3, 4 or 5` will override the logging results of the `map-composer-log-levels` option

mcr-pull-by-this-node

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true` or `false`

Value changes: in setting the next timer interval

This option specifies that the pulling of eServices interactions will be allowed to be performed by this node If set to `true`.

Note that by default this option is set to `false`, so if the default is left on all nodes, no pulling will occur.

Note that it is allowed to have more than one node to pull interactions.

For example: `orchestration/ mcr-pull-by-this-node = true`

mcr-pull-interval

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `1000`

Valid values: Any positive integer.

Value changes: in setting the next timer interval

This option provides the number of milliseconds between attempts to pull interactions from the queues/views managed by the ORS.

For example: `orchestration/mcr-pull-interval = 5000`

mcr-pull-limit

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `5000`

Valid values: Any positive integer from `0` to `5000`

Value changes: in setting the next timer interval

This option provides the maximum number of pulled interactions that each node in a cluster (or ORS working in standalone mode) is allowed to have at any given time. A value of 0 disables the pulling of multimedia interactions completely for this instance of ORS.

For example: `orchestration/mcr-pull-limit = 5000`

new-session-on-reroute

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true` or `false`

Value changes: Immediately

This option, introduced in ORS Release 8.1.300.39, controls Orchestration Server behavior when T-Server performs default routing.

In the scenario of T-Server performing default routing, T-Server redirects a call that is processed by some strategy on some Routing Point. The call is then diverted and queued on another Routing Point with `call state 22`.

The option `new-session-on-reroute` allows you to explicitly specify how ORS should process the call. It also enables ORS and Universal Routing Server (URS) to handle this scenario in a consistent fashion.

- If `new-session-on-reroute` is set to `true`, ORS will terminate the existing live session or abort the session recovery process (whatever takes place) when it receives events related to call redirection.

Upon being diverted from the Routing Point (only) with `call state 22` (only), ORS breaks the existing call/session association (if any). As a result, the call becomes free to create a new session upon arriving at the default destination DN loaded with the strategy.

- If `new-session-on-reroute` is set to `false`, ORS reverts to its previous behavior where it does not break the existing call/session association when it receives events related to call redirection. As for live sessions, their execution will be continued, and they will receive the usual events indicating that the initial Routing Point party disappeared. As a result, the live session can handle the situation when the call being handled is moved to another DN by T-Server in the middle of the routing process.

It is necessary to keep the both ORS option `new-session-on-reroute` and Universal Routing Server option `use_ivr_info` synchronized (both are either `true` or `false`).

parse-start-params

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true` or `false`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies whether to enable or disable serialization/deserialization of session parameters. When `true`, parameters delivered to sessions started using invoke or session:start will retain their original type. When `false`, the session parameters will be converted to string.

For example: `parse-start-params = true`

scxml-log-filter-level

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: empty (disabled)

Valid values: any valid combination of custom-defined and Genesys log levels where

Custom log level: An Integer composed of the digits `1, 2, 3, 4, or 5` and less than or equal to 9 digits in length (`minimum of 1 and maximum of 555555555`).

Genesys log levels: `STD, INT, TRC, ALR` and `DEBUG`. These are not case-sensitive.

Value changes: Immediately

This option was introduced in ORS Release 8.1.300.29. If this option is configured and the level parameter of the `__Log` function or level attribute of the `<log>` action element in the SCXML application matches the custom log level value, ORS generates corresponding Genesys log events of the `Standard (23011), Alarm (23012), Interaction (23022), Trace (23023) or Debug (23026) levels. The value of scxml-log-filter-level option can contain:`

- A semi-colon list of colon-delimited pairs of custom and Genesys log levels.
- A semi-colon list of custom log levels (no explicit mapping to Genesys log levels). ORS always generates a log event of the Standard (23011) level.
- Any combination of both value types listed above.

Setting the `scxml-log-filter-level` option value to `1, 2, 3, 4 or 5` will override the logging results of the `map-composer-log-levels` option.

For example, if the SCXML application contains:

```
<log> action element:
                <log expr="'Custom Log Level 45'" label="'Test example'" level="45" />
                <log expr="'Custom Log Level 222'" label="'Test example'" level="222" />

or __Log() function:

                __Log('Custom Log Level 45','Test example','45');
                __Log('Custom Log Level 222','Test example','222');


a.        scxml-log-filter-level = 45:TRC;222:INT
```

ORS will generate following events:

```
Trc 23023 METRIC <log sid='31RJ0AHJQP3851BFQ6QSJR0LA4000001' expr='Custom Log Level 45'
label='Test example' level='45' />
Int 23022 METRIC <log sid='31RJ0AHJQP3851BFQ6QSJR0LA4000001' expr='Custom Log Level 222'
label='Test example' level='222' />

b.        scxml-log-filter-level = 45;222
```

ORS will generate following events:

```
Std 23011 METRIC <log sid='31RJ0AHJQP3851BFQ6QSJR0LA4000001' expr='Custom Log Level 45'
label='Test example' level='45' />
Std 23011 METRIC <log sid='31RJ0AHJQP3851BFQ6QSJR0LA4000001' expr='Custom Log Level 222'
label='Test example' level='222' />

c.        scxml-log-filter-level = 45:TRC;222
```

ORS will generate following events:

```
Trc 23023 METRIC <log sid='31RJ0AHJQP3851BFQ6QSJR0LA4000001' expr='Custom Log Level 45'
label='Test example' level='45' />
```

```
Std 23011 METRIC <log sid='31RJ0AHJQP3851BFQ6QSJR0LA4000001' expr='Custom Log Level 222'
label='Test example' level='222' />
```

Note: For more information on the __Log function, refer to the function description on this page.

sessionfm-fetch-timeout

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `60`

Valid values: any Any integer from `0 to 86400`

Value changes: Immediately

This option, introduced in ORS Release 8.1.300.32, allows you to set a custom value for the timeout attribute used in the `<session:fetch>` action element. The option specifies the timeout in seconds that ORS will use when the timeout attribute is omitted in the `<session:fetch>` action element.

When ORS is operating on Windows, due to the operating system default settings, the timeout reaches a 21-second maximum regardless of the value configured in the `sessionfm-fetch-timeout` option.

session-hung-timeout

This option is obsolete and no longer used.

switch-multi-links-enabled

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true` or `false`

Value changes: after restart

This option determines whether ORS supports multi-link switch configurations (load sharing Network T-Servers or IVR Servers). When set to `true`, ORS enables support of multi-link switch configuration. When set to `false`, ORS does not enable multi-link switch support.

Note: The ORS `switch-multi-links-enabled` configuration option and the `switch-multi-links-enabled` configuration option specified on the load sharing switch object must both be enabled to support this functionality.

Load Sharing Switch Object

The `switch-multi-links-enabled` option, specified on the load sharing switch object (`Switches\"SharedSwitchObject"\Annex\gts`), determines whether the switch is working in load-balancing mode; that is, it is served by multiple Network T-Servers or IVR T-Servers. Orchestration uses this option to determine whether to enable connection to more than one Network T-Server or IVR T-Server serving this switch.

`switch-multi-links-enabled`

Default value: `0`

Value	Description
1	A network or IVR switch in load-balancing mode.
Any other integer	Not a network or IVR switch in load-balancing mode.

Changes Take Effect: After ORS restart.

This option should be used only in a configuration in which Network T-Servers or IVR T-Servers are working in load-balancing mode; that is, when there is no duplication in notification events received in Orchestration via connections to these T-Servers. Currently, load balancing mode is supported only for Network T-Servers and IVR T-Servers. This option was introduced in ORS 8.1.2+.

thread-synch-ipv

Option section: `orchestration`

Configuration object: ORS `Application` object

Default value: `any`

Valid values: `any`, `4`, `6`.

Value changes: During startup. Option changes take effect after restart only.

This option, introduced in ORS Release 8.1.300.45, can improve strategy execution time by preventing delays that can occur when establishing the initial connection to session processing threads. The option specifies the protocol version for inter-thread communication in ORS. The value `any` indicates any version (this mode was used before introducing the option). The value `4` indicates the IPv4 protocol version. The value `6` indicates the IPv6 protocol version. Mixed modes are not allowed, such as `4`, `6` or `6`, `4`.

For example: `thread-synch-ipv=4`

webfm-event-hold-response

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `true`

Valid values: `true` or `false`

Value changes: Immediately

When this option is set to `true`, the HTTP response depends on how the event is processed:

- If a transition has been selected as a result of the event, response contains headers `Status-Code: 200, Reason-Phrase: OK`
- If no transition has been selected, response contains headers `Status-Code: 204, Reason-Phrase: No Content`
- If an error has occurred while processing the event, response contains headers `Status-Code: 400, Reason-Phrase: Bad Request`

When the option is set to `false`, once the event has been successfully published, ORS responds with `200 OK`.

persistence Section

This section describes `persistence` section options.

cassandra-connect-attempt-timeout

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `2000`

Valid values: Any integer greater than or equal to `1`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the maximum time, in milliseconds, allowed for a connection to the Cassandra cluster to complete.

For example: `persistence/cassandra-connect-attempt-timeout = 5000`

Configuring a value for `cassandra-connect-attempt-timeout` above the default value could cause excessive delays in Orchestration startup if the Cassandra store is not available.

cassandra-keyspace-name

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `Orchestration`

Valid values: Strings of alpha-numeric characters and underscores. Must begin with a letter.

Value changes: take effect during startup. Changes to this option are not applied dynamically.

This option specifies the name of the Cassandra keyspace to use for this ORS cluster. This option would be used if you plan to run multiple distinct ORS clusters using the same Cassandra cluster. This option must be unique for an Orchestration cluster. If nodes in different Orchestration clusters inadvertently define the same keyspace, cross-contamination of persisted data will lead to unpredictable operation in a failover scenario.

For example: `persistence/cassandra-keyspace-name = ORS_Cluster_west`

For the most current information on using ORS with Cassandra, see the Cassandra Installation/ Configuration Guide.

cassandra-listenport

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `9160`

Valid values: Any valid socket port.

Value changes: take effect during startup. Changes to this option are not applied dynamically.

This option provides the Cassandra client connection port when using the Cassandra-based persistence method. This option must be supplied with an appropriate value for correct Cassandra-based persistence operation to occur.

For example: `persistence/cassandra-listenport = 9160`

cassandra-max-latency

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `400`

Valid values: Any non-negative integer greater than `100` and less than `5000`

Value changes: take effect during startup. Changes to this option are not applied dynamically.

Maximum request latency allowed before messages are logged warning of excessive delays in response to Cassandra requests. If subsequent requests have a latency less than this value a message is logged indicating the excessive latency condition has cleared. This option is not required, and the default is 400 msec.

cassandra-nodes

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: No default value (empty string).

Valid values: A list of host names for Cassandra nodes in the Cassandra cluster.

Value changes: take effect during startup. Changes to this option are not applied dynamically.

This option provides the Cassandra client connection port when using the Cassandra-based persistence method. This is a semi-colon separated list of host names or IP addresses.

For example: `persistence/cassandra-nodes = DWS;mpswin;test47`

cassandra-read-timeout

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `500`

Valid values: Any integer value from `201` to `4999`

Value changes: Takes effect after restart.

This option, introduced in ORS Release 8.1.300.47, defines the maximum time, in milliseconds, that ORS will wait for a Cassandra connection to become ready to read. If this time is exceeded, ORS considers the connection to be lost.

cassandra-schema-version

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `ORS8130000`

Valid values: This is a required parameter, which may be left at the default value or set to a previous schema version, but it must agree with the Column Family name for the cassandra cluster to which ORS is connecting. This is a string and must be of the form `ORSddddddd`, where `ddddddd` must be numeric

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the schema version that either exists in the Cassandra cluster, or is to be loaded automatically by Orchestration upon the first successful connection to the Cassandra cluster.

On startup and connection to Cassandra, Orchestration verifies that the requested schema agrees with the existing Cassandra schema, if the schema has been previously loaded. Or, it loads the schema automatically at the schema version defined by the default.

If the schema disagrees or is not defined, Orchestration will continue to operate, but without persistence, and therefore, without the ability to restore sessions or scheduled events.

For example: `persistence/cassandra-schema-version = ORS8130000`

cassandra-strategy-class

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `SimpleStrategy`

Valid values: `SimpleStrategy` or `NetworkTopologyStrategy`.

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the strategy class that Cassandra uses for the cluster. `SimpleStrategy` defines a single cluster, without multiple Data Centers.

The `NetworkTopologyStrategy`, which is network strategy in conjunction with the `Cassandra-topology` properties file (located in each Cassandra instance install configuration directory), defines the Data Centers for the Cassandra cluster. Multiple Data Centers typically are geographically dispersed.

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

For example: `persistence/cassandra-strategy-class = SimpleStrategy`

cassandra-strategy-options

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: Unknown

Valid values: `replication_factor:N`, where `N` is the replication factor desired, or `DC1:K;DC2:J`…Where `DC1`, etc. are the Data Center names defined in the topology properties file, and `K,J`, etc. are the replication factors for each Data Center.

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the replication_factor to be configured for the given Orchestration keyspace. The two variations apply to `SimpleStrategy` or `NetworkTopologyStrategy` respectively. See the Cassandra Installation and Configuration Guide.

Examples:

- If SimpleStrategy, cassandra-strategy-options = replication_factor:2
- If NetworkTopologyStrategy, cassandra-strategy-options = DC1:2;DC2:3

cassandra-thread-count

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `8`

Valid values: Any integer greater than or equal to `1`

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the number of threads in the persistence worker thread pool. Recommended value: 2x the number of cores.

For example: `persistence/cassandra-thread-count = 4`

cassandra-write-timeout

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `500`

Valid values: Any integer value from `201` to `4999`

Value changes: Takes effect after restart.

This option, introduced in ORS Release 8.1.300.47, defines the maximum time, in milliseconds, that ORS will wait for a Cassandra connection to become ready to write. If this time is exceeded, ORS considers the connection to be lost.

max-cache-count

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `100000`

Valid values: Any integer greater than or equal to `10000`

Value changes: During startup. Changes to this option are not applied dynamically.

Defines the number of entries allowed within the internal Orchestration persistence cache.

For example: `persistence/max-cache-count = 150000`

For example: persistence/max-cache-count =

When connection to persistent storage is disabled or persistent storage is not available, Orchestration Server removes the cached data of terminated sessions. This enables the processing of long–lived sessions, if the number of current active sessions does not exceed the number specified in the max-cache-count option.

max-cache-size

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `100000000`

Valid values: Any integer greater than or equal to `0`

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the maximum size of the amount of memory (in bytes) that the internal Orchestration persistence cache can use.

For example: `persistence/max-cache-size = 100000`

scxml Section

This section describes `scxml` section options.

debug-enabled

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `Boolean`

Value changes: During startup. Changes to this option are not applied dynamically.

This option allows you to specify whether debugging with Genesys Composer is enabled in the system. If it is enabled, you must also specify the `debug-port`.

debug-port

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `7999`

Valid values: any available port

Value changes: During startup. Changes to this option are not applied dynamically.

This option allows you to specify and enable or disable the port to be used by the debug client Genesys Composer to interact with the SCXML Engine during a debug session.

default-encoding

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `UTF-8`

Valid values: Any valid converter name supported by ICU.

Value changes: After restart

This option defines the source encoding that ORS will use to encode all input String data into Unicode before it is passed into the SCXML engine. Input String data includes:

1. SCXML and JavaScript String literals
2. variables
3. properties of String type
4. values of String type message attributes from the incoming API such as TLib and Ixn
5. values of String type properties of Configuration Objects retrieved from Configuration Server, and so on.

This option also defines the destination encoding that ORS will use to encode all SCXML Actions, function attributes, and parameters of String type from Unicode before those values will be used as attribute values in all ORS external APIs like TLib, Ixn, HTTP, and so on.

fips-enabled

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true` or `false`

Value changes: During startup. Changes to this option are not applied dynamically.

Enables FIPS compliance in the SCXML library.

For example: `scxml/fips-enabled = false`

http-client-side-port-range

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: <empty string>

Valid values: A number between `1 and 65535` followed by a `"-"` and then another number between `1 and 65535`; with the second number larger than the first number.

Value changes: During startup. Changes to this option are not applied dynamically.

Specifies the port range of the socket used for an HTTP client side connection. An empty value, or lack of option, indicates that the default should be used.

For example: `scxml/http-client-side-port-range = 1000-2000`

http-enable-continue-header

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true` or `false`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies whether to enable or disable the `100-continue` header in the `HTTP 1.1 post request`.

For example: `scxml/http-enable-continue-header = true`

http-enable-keepalive

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true` or `false`

Value changes: Changes take effect after restart.

This option, introduced in ORS Release 8.1.300.35, enables keepalive transmissions on TCP sockets used by Orchestration Server. If set to `true`, enables `keepalive` transmissions on TCP sockets for fetching documents from an application server.

http-max-age

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `60`

Valid values: Any integer greater than or equal to `0`

Value changes: During startup. Changes to this option are not applied dynamically.

Defines the time in seconds for how long a fetched SCXML application is cached. When the same SCXML application is to be used for the new session within the configured timeout, the cached version of the document will be used instead of fetching it from the application server.

To disable fetching of SCXML applications from cache, set options `http-max-age` and `http-max-stale` to 0.

If the URL of the SCXML application contains parameters that are unique for each invocation, then the application will be fetched each time regardless of option value.

Note: If `max-age` is configured for an Enhanced Routing Script object, the value from `max-age` takes precedence over the `http-max-age` Application level option.

http-max-age-local-file

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `60000`

Valid values: Any integer greater than or equal to `0`

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the maximum age of the local file in milliseconds.

For example: `scxml/http-max-age-local-file = 65000`

http-max-cache-entry-count

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `1000`

Valid values: Any integer greater than or equal to `0`

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the maximum number of entries that can be stored in the cache.

For example: `scxml/http-max-cache-entry-count = 1250`

http-max-cache-entry-size

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `100000`

Valid values: Any integer greater than or equal to `0`

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the maximum size of each cache entry in bytes.

For example: `scxml/http-max-cache-entry-size = 125000`

http-max-cache-size

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `10000000`

Valid values: Any integer greater than or equal to `0`

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the maximum size of the HTTP cache in bytes.

For example: `scxml/http-max-cache-size = 12500000`

http-max-redirections

Option section: `scxml`

Configuration object: `ORS Application objectt`

Default value: `5`

Valid values: `0 - 100`

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the maximum number of times to follow the `Location:` header in the HTTP response. Set to 0 to disable HTTP redirection.

For example: `scxml/http-max-redirections = 10`

http-max-stale

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `60`

Valid values: Any integer greater than or equal to `0`

Value changes: During startup. Changes to this option are not applied dynamically.

Defines time in seconds to extend the lifetime of cached SCXML application.

For example, if the value of the `http-max-age` option is set to 60 seconds and value of the `max-stale` option is set to 30 seconds, then the fetched document will be cached for the timeframe of 90 seconds.

- If value is set to 0, the lifetime of a cached file will not be extended.
- To disable fetching SCXML applications from the cache, set `http-max-age` and `http-max-stale` to 0.

- If `max-stale` is configured for an Enhanced Routing Script object, the value from `max-stale` takes precedence over the `http-max-stale` Application level option.

http-no-cache-urls

Option section: `scxml`

Configuration object: `ORS Application object`

Default value:`<empty string>`

Valid values: Comma-delimited set of strings

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets a comma-delimited list of substrings that would prevent a response from being cached, if the URL contains any of the substrings.

For example: `scxml/http-no-cache-urls=myserver.com, yourserver.com`

http-proxy

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `<empty string>`

Valid values: Valid IP Address : `Port` or URL: `Port`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the HTTP proxy server (if applicable). If specified, all HTTP fetches done by the ORS will be done via this proxy server.

For example: `scxml/http-proxy = 127.0.0.1:3128`

scxml/http-proxy = myproxy:3128

http-ssl-ca-info

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `<empty string>`

Valid values: Valid file name

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the file name holding one or more CA certificates with which to verify the peer. This is only useful when `ssl-verify-peer=true`.

For example: `scxml/http-ssl-ca-info = myca.cer`

http-ssl-ca-path

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `""`

Valid values: Valid path

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the path holding one or more CA certificates with which to verify the peer. The certificate directory must be prepared using the openssl c_rehash utility.

For example: `scxml/http-ssl-ca-path = c:\ssl\ca`

http-ssl-cert

Option section: `scxml`

Configuration object: `ORS Application object`

Default value:`<empty string>`

Valid values: Valid file name

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the file name of the SSL certificate.

For example: `scxml/http-ssl-cert = mycert.crt`

Spaces are not valid in the directory name that is specified in the file path for this option. For example `C:\Program Files\ Certificates\my_cert.pem` is not a valid path.

http-ssl-cert-type

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `PEM`

Valid values: `PEM, DER`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the SSL Certificate Type: `PEM – PEM encoded certificate DER – DER encoded certificate`

For example: `scxml/http-ssl-cert-type = DER`

http-ssl-cipher-list

Option section: `scxml`

Configuration object: `ORS Application object`

Default value:`<empty string>`

Valid values: String value

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the cipher list as defined by OpenSSL. Paste the following url into your web browser to see valid cipher list formats:

http://www.openssl.org/docs/apps/ciphers.html#CIPHER_LIST_FORMAT

For example: `scxml/http-ssl-cipher-list=RC4-SHA`

http-ssl-key

Option section: `scxml`

Configuration object: `ORS Application object`

Default value:<empty string>

Valid values: Valid path or file name

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the path or file name of the SSL private key. For example: `scxml/http-ssl-key = c:\ssl\mykey.key`

Spaces are not valid in the directory name that is specified in the file path for this option.

For example `C:\Program Files\ Certificates\my_cert.pem` is not a valid path.

http-ssl-key-type

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `PEM`

Valid values: `PEM, DER`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the SSL Key Type:

PEM – PEM encoded key

DER – DER encoded key

For example: `scxml/http-ssl-key-type = DER`

http-ssl-random-file

Option section: `scxml`

Configuration object: `ORS Application object`

Default value:`<empty string>`

Valid values: Valid file name

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the file which is read from in order to see the random engine for SSL. The more random the specified file is, the more secure the SSL connection will become.

For example: `scxml/http-ssl-random-file=c:\ssl\random-seed`

http-ssl-verify-host

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `disable`

Valid values: `disable, common, match`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies how the common name from the peer certificate should be verified during the SSL handshake.

- disable – the connection succeeds regardless of the names in the certificate
- common – the certificate must contain a “`Common Name`” field, but the field’s contents are not validated
- match – the certificate must indicate correct server name, or the connection will fail

For example: `scxml/http-ssl-verify-host = match`

http-ssl-verify-peer

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true` or `false`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies whether the system should verify the peer's certificate. When this is true, either `http-ssl-ca-path` or `http-ssl-ca-info` would be set.

For example: `scxml/http-ssl-verify-peer = true`

http-ssl-version

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `default`

Valid values: `String (default, TLSv1, SSLv2 or SSLv3)`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the SSL version to be used. By default, the system will determine the correct version to use. However, this option may be useful when some servers make it difficult to determine the correct SSL version.

For example: `scxml/http-ssl-version = SSLv2`

https-proxy

Option section: `scxml`

Configuration object: `ORS Application object`

Default value:`<empty string>`

Valid values: Valid IP Address: `Port` or URL: `Port`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the HTTPS proxy server (if applicable). If specified, all HTTPS fetches done by the ORS will be done via this proxy server.

For example: `scxml/https-proxy = 127.0.0.1:3128`

js-preload-files

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: No value

Valid values: A semi-colon delimited list of JavaScript files

Value changes: During startup. Changes to this option are not applied dynamically.

This option provides a list of JavaScript files that are pre-loaded at ORS startup. If the full file path is not provided, ORS will look for files in the ORS working directory. This allows users to specify functions or variables that are available to all sessions.

max-assembled-cache-size

Option section: `scxml`

Configuration object: ORS `Application` object

Default value: `10000000`

Valid values: Minimum = 0, no maximum value

Valid changes: Take effect after restart.

This option was introduced in ORS Release 8.1.300.48. Use `max-assembled-cache-size`, `max-assembled-cached-docs`, and `max-assembled-cached-doc-size` to configure the document cache by storing fully-assembled SCXML documents, eliminating the need for document fetches for `xincludes` in subsequent sessions.

This option determines the maximum allowed size (in bytes) of the assembled document cache. The size of the cache is determined by the total size of all cached documents stored in the cache.

max-assembled-cached-docs

Option section: `scxml`

Configuration object: ORS `Application` object

Default value: 1000

Valid values: Minimum = 0, no maximum value

Valid changes: Take effect after restart.

This option, introduced in ORS Release 8.1.300.48, determines the maximum number of documents to be stored in the assembled document cache. If this number is exceeded, the least referenced cached items will be removed from the cache to make room for new entries.

max-assembled-cached-doc-size

Option section: `scxml`

Configuration object: ORS `Application` object

Default value: `10000000`

Valid values: Minimum = 0, no maximum value

Valid changes: Take effect after restart.

This option, introduced in ORS Release 8.1.300.48, determines the maximum allowed size (in bytes) of each individual entry in the cache. If a fully-assembled SCXML strategy exceeds this size, it cannot be stored in the cache.

max-cache-entry-count

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `1000`

Valid values: Any integer greater than or equal to `0`

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the maximum number of entries that can be stored in the cache. For example: `scxml/http-max-cache-entry-count = 1250`

max-cache-entry-size

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `100000`

Valid values: Any integer greater than or equal to `0`

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the maximum size of each cache entry in bytes.

For example: `scxml/http-max-cache-entry-size = 125000`

max-compiler-cache-size

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `10000000`

Valid values: non-negative integer

Value changes: Takes effect after restart. Changes to this option are not applied dynamically.

The maximum amount of memory (in bytes) allowed for the `<xi:include>` compiler cache.

max-compiler-cached-docs

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `1000`

Valid values: non-negative integer

Value changes: Takes effect after restart. Changes to this option are not applied dynamically.

The maximum number of items that the `<xi:include>` compiler cache can have.

max-compiler-cached-doc-size

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `10000000`

Valid values: non-negative integer

Value changes: Takes effect after restart. Changes to this option are not applied dynamically.

The maximum size, in bytes, allowed to cache the results of the compiled `<xi:include>` document.

max-debug-sessions

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `0`

Valid values: an integer

Value changes: During startup. Changes to this option are not applied dynamically.

When debugging with Genesys Composer, this option allows you to set the maximum number of simultaneous debug sessions within the current SCXML Engine instance. Note that this value must be at least one digit less than the value specified in the `session-processing-threads` option to prevent session thread starvation. If it is not, it will default to 0.

max-includes

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `500`

Valid values: `0 to 10000`

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the maximum number of documents that may be included using <xi:include>.

For example: `scxml/max-includes = 200`

max-microstep-count

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `1000`

Valid values: any integer (no maximum)

Value changes: Immediately

For infinite loop protection, this option is used along with `max-state-entry-count`. Both options may be configured globally in ORS configuration, or may be configured per session by setting the attribute in the SCXML document:

`<scxml _microStepLimit="1000" . />`

<scxml _stateEntryLimit=100" . />

For example, `max-microstep-count = 500`

This option specifies the maximum number of times in which transitions may be taken during a session without the processing of a new event. Once this value has been exceeded, the session is exited gracefully before being terminated by the system. A value of `0` indicates that the session is not to be forcibly terminated.

max-pending-events

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `100`

Valid values: positive integer between `30 and 100000`, inclusive.

Value changes: Changes take effect During startup. Changes to this option are not applied dynamically.

This option specifies the maximum number of events allowed to be queued to a session (inclusive of `internal, external, delayed` and `undelivered` events). If this number is reached, ORS shall attempt to exit the session. This feature cannot be disabled.

For example: `scxml/max-pending-events = 1000`

max-preprocessor-cache-size

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `10000000`

Valid values: Any integer greater than or equal to `0`

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the amount of memory, in bytes, that the `<xi:include>` pre-processor cache can use.

For example: `scxml/max-preprocessor-cache-size = 20000000`

max-preprocessor-cached-docs

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `1000`

Valid values: Any integer greater than or equal to `0`

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the maximum number of items that the `<xi:include>` pre-processor cache can have.

For example: `scxml/max-preprocessor-cache-docs = 2000`

max-preprocessor-cached-doc-size

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `10000000`

Valid values: non-negative integer

Value changes: Takes effect after restart. Changes to this option are not applied dynamically.

The maximum size, in bytes, of the cached results of the preprocessed `<xi:include>` documents.

max-session-age

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `604800`

Valid values: A positive integer

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the maximum age in seconds that an ORS session should exist. If this age is reached, ORS shall attempt to exit the session.

To disable this feature, set the `max-session-age` to 0. The SCXML Engine supports `_maxtime` as an attribute on the `<scxml>` element, and also supports the option `max-session-age`, which is intended to queue a terminate event for the session after the time expired.

The value specified in the `_maxtime` attribute overrides the value in the `max-session-age` configuration option.
A connection to Cassandra is required for this functionality.

Orchestration Server sets the time to live in Cassandra for information required to support proactive session recovery. The time to live for the information in Cassandra is the configured `scxml/max-session-age` plus 3600 seconds. After this period, the data within Cassandra will be marked as deleted.

max-state-entry-count

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `100`

Valid values: any integer (no maximum)

Value changes: Immediately

For example, `max-state-entry-count = 250`

For infinite loop protection, this option is used along with `max-microstep-count`. Both options may be configured globally in ORS configuration, or may be configured per session by setting the attribute in the SCXML document:

```
<scxml _microStepLimit="1000" . />
```

<scxml _stateEntryLimit="100" . />

This option specifies the maximum number of times that a transition may be taken to a target state. Every state element within an SCXML document keeps an individual count of the number of times entered via a targeted transition. Once this value has been exceeded, the session is exited gracefully before being terminated by the system.

A value of `0` indicates that the session is not to be forcibly terminated.

password

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: <empty string>

Valid values: String value

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the SSL key password.

For example: `scxml/password = agent007`

persistence-default

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true` or `false`

Value changes: Takes effect after restart. Changes to this option are not applied dynamically.

This option defines the default session persistence. If set to `true`, then Orchestration will persist the session if Cassandra is available. If set to `false`, Orchestration will not attempt to persist the session.

Note: In your routing strategy, you can set the `_persist` attribute in the `<scxml>` element, which will take precedence over the value set in the `persistence-default` Application level option.

persistence-max-active

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `10000`

Valid values: `100 - 1000000`

Value changes: During startup. Changes to this option are not applied dynamically.

This option allows setup of a maximum number of active sessions that the SCXML engine will keep in memory.

For example: `scxml/persistence-max-active = 5000`

process-event-timeout

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `10000`

Valid values: A positive integer

Value changes: During startup. Changes to this option are not applied dynamically.

This option specifies the maximum time (in milliseconds) allotted for the processing of the event queue. The processing of one event may lead to additional events being queued. Processing of the event queue does not complete until the event queue is empty. This feature sets an upper bound to the amount of time dedicated to processing these events. If the timeout is reached, ORS attempts to exit the session. To disable this feature, set the `process-event-timeout` to 0.

For example: `scxml/process-event-timeout = 60000`

session-processing-threads

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `8`

Valid values: Integer from `1 to 128` (inclusive)

Value changes: During startup. Changes to this option are not applied dynamically.

This option sets the number of threads in the thread pool. The recommended value is two times the number of cores.

For example: `scxml/session-processing-threads = 16`

system-id

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `-1`

Valid values: Any integer greater than or equal to `-1`

Value changes: During startup. Changes to this option are not applied dynamically.

This option allows setup of the ORS system ID in order to have unique session IDs across ORS instances. If `-1` is specified, ORS creates a session ID based on the IP address of the host running ORS.

For example: `scxml/system-id = 11`

log Section

This section describes log options specific to ORS

x-server-trace-level

Option section: `log`

Configuration object: `ORS Application object`

Default value: `0`

Valid values: `0 - 3`

Value changes: as soon as committed to Configuration Server

This option specifies the level of tracing to be enabled for the ORS.

For example: `log/x-server-trace-level = 2`

Also see log-trace-segments.

x-server-gcti-trace-level

Option section: `log`

Configuration object: `ORS Application object`

Default value: `0`

Valid values: `0 - 3`

Value changes: as soon as committed to Configuration Server

This option specifies the level of GCTI tracing to be enabled for the ORS. It controls how much detail should be in the logs for GCTI-related events, such as those from T-Server.

For example: `log/x-server-gcti-trace-level = 2`

x-server-config-trace-level

Option section: `log`

Configuration object: `ORS Application object`

Default value: `0`

Valid values: `0 - 3`

Value changes: as soon as committed to Configuration Server

This option specifies the level of configuration tracing to be enabled for the ORS. It controls how much detail should be in the logs for Configuration-related events, such as reading from Configuration Server and reacting to dynamic changes.

For example: `log/x-server-config-trace-level = 2`

x-print-attached-data

Option section: `log`

Configuration object: `ORS Application object`

Default value: `0`

Valid values: `0, 1`

Value changes: as soon as committed to Configuration Server

This option specifies whether or not attached data should be formatted and printed in the logs.

`0—Suppress printing attached data`

1—Format and print attached data

For example: `log/x-print-attached-data = 1`

mcr Section

This section describes mcr section options.

om-memory-optimization

Option section: `mcr`

Configuration object: `ORS Application object`

Default value: `true`

Valid values: `true` or `false`

Value changes: take effect immediately.

This option specifies whether memory optimization will be in effect for the current ORS. When the value is set to true, ORS will remove passive multi-media interactions from memory cache.

For example: `mcr/om-memory-optimization = true`

om-max-in-memory

Option section: `mcr`

Configuration object: `ORS Application object`

Default value: `100`

Valid values: `1 to 2000`

Value changes: take effect immediately.

This option specifies a maximum number of interactions to store in memory cache before interactions will begin to be removed from the cache (oldest first), when `om-memory-optimization` is set to true. The value is in thousands, so that the default value of `100` represents 100,000 interactions that are to be stored in memory cache. The maximum value for this option is `2000`, which represent 2,000,000 interactions.

For example: `mcr/om-max-in-memory = 100`

Note: For high volume loading, Genesys recommends the default value of `100`.

om-delete-from-memory

Option section: `mcr`

Configuration object: `ORS Application object`

Default value: `1`

Valid values: `1 to 2000000`

Value changes: take effect immediately.

This option specifies how many interactions should be deleted when the value of `om-max-in-memory` has been reached.

For example: `mcr/om-delete-from-memory = 1`

Common Log Options

Configure the common log options in the same section as the ORS-specific log options.

For information on the log options listed below, see the Framework 8.1. Configuration Options Reference Manual.

log Section

This section must be called `log`.

For applications configured via a configuration file, changes to log options take effect after the application is restarted.

- buffering
- check-point
- compatible-output-priority
- expire
- keep-startup-file
- memory
- memory-storage-size
- message_format
- messagefile
- print-attributes
- segment
- spool
- time_convert
- time_format
- verbose

Log Output Options

To configure log outputs, set log level options (`all, alarm, standard, interaction, trace, and/or debug`) to the desired types of log output (`stdout, stderr, network, memory`, and/or `[filename]`, for log file output).

You can use:

- One log level option to specify different log outputs.
- One log output type for different log levels.
- Several log output types simultaneously, to log events of the same or different log levels.

You must separate the log output types by a comma when you are configuring more than one output for the same log level.

If you direct log output to a file on the network drive, an application does not create a snapshot log file (with the extension `*.snapshot.log`) in case it terminates abnormally.

Directing log output to the console (by using the `stdout` or `stderr` settings) can affect application performance. Avoid using these log output settings in a production environment.

The log output options are activated according to the setting of the `verbose` configuration option.

- `all`
- `alarm`
- `standard`
- `interaction`
- `trace`
- `debug`

Log File Extensions

You can use the following file extensions to identify log files that an application creates for various types of output:

- `.log—Assigned` to log files when you configure output to a log file. For example, if you set `standard = confservlog` for Configuration Server, it prints log messages into a text file called `confservlog.<time_stamp>.log`.
- `.qsp—Assigned` to temporary (`spool`) files when you configure output to the network but the network is temporarily unavailable. For example, if you set `standard = network` for Configuration Server, it prints log messages into a file called `confserv.<time_stamp>.qsp` during the time the network is not available.
- `.snapshot.log—Assigned` to files that contain the output snapshot when you configure output to a log file. The file contains the last log messages that an application generates before it terminates abnormally. For example, if you set `standard = confservlog` for Configuration Server, it prints the last log message into a file called `confserv.<time_stamp>.snapshot.log` in case of failure.

Note: Provide `*.snapshot.log` files to Genesys Customer Care when reporting a problem.

- `.memory.log—Assigned` to log files that contain the memory output snapshot when you configure output to `memory` and redirect the most recent memory output to a file. For example, if you set `standard = memory` and `memory = confserv` for Configuration Server, it prints the latest memory output to a file called `confserv.<time_stamp>.memory.log`.

Switch gts Section

The <createcall> and <consult> interaction action elements have a limitation for the attribute “to” (where a destination is specified). Only digits can be specified in the “to” attribute. The limitation can be removed by configuration of `valid-digits` and `max-digits` options on the `Switch object > Annex tab> gts` section:

Option: `valid-digits`

Value: An explicit set of all acceptable symbols that can be dialed.

Example: `0123456789_RPO`

Option: `max-digits`

Value: The maximum number of digits that can be dialed.

Example: `25`

DN Level Options

This section lists DN-level options configured in the `orchestration` section.

application

Option section: `orchestration`

Configuration object: `DN (Extension or RoutePoint or Interaction Queue)`

Default value: none (this is a required option)

Valid values: any valid URL

Value changes: For the next interaction that hits this resource

This option specifies the URL of the SCXML document to load. The URL can be any one of the following protocols:

- `file:<path>`
- `http://<url>`
- `script:<name of script object>` The use of script: allows an indirect reference to a script object of type `CfgEnhanced Routing`, which can contain the application URL, parameters, and other configuration values.

The URL can also contain parameters which will be passed to the Application server. The values shown in the URL Parameter Elements for application below are substituted at run-time based on the information in the interaction.

URL Parameter Elements for application

- `[DNIS]` The `DNIS` attribute of the interaction
- `[ANI]` The `ANI` attribute of the interaction
- `[DN]` The `ThisDN` attribute of the interaction
- `[CED]` The `CollectedDigits` attribute of the interaction
- `[EMAILFROM]` E-mail from address
- `[EMAILTO]` E-mail to address
- `[EMAILSUBJECT]` E-mail subject
- `[UDATA]` Expanded to the entire user data of the interaction in the format: `name1=value1&name2=value2&…`
- `[UDATA.*]` Expanded to the entire user data of the interaction in the format: `name1:value1,name2:value2,…`
- `[UDATA.name]` Expanded to the value of a specific user data key of the interaction value.

For example: `&servicetype=[UDATA.ServiceType]` could resolve to:`&servicetype=CreditCards`.

Examples

- `application = http://xserver.genesyslab.com:80/NewCallReq.asp`
- `application= http://xserver.genesyslab.com:80/NewCallReq.asp?ani=[ANI]&dnis=[DNIS]&servicetype=[UDATA.ServiceType]`
- `application = script:orsscript`

Users can provide an alternate URL utilizing the following rules. The value of the application section can be done as follows:

`application = <URL1><single space><URL2>`

For example:

- `application = http://host1/RouteToDN1.scxml http://host1/RouteToDN2.scxml`
- `application = http://host1/RouteToDN1.scxml http://host1/RouteToDN2.scxml`

The same is applicable for `file:<path>`. A single space is used between the two file paths. </toggledisplay>

send-retries

Option section: `orchestration`

Configuration object: `DN (Extension or RoutePoint or Interaction Queue)`

Default value: 2

Valid values: `0-10`

Value changes: During startup. Changes to this option are not applied dynamically.

The option specifies the number of times ORS tries to resend an event if the first attempt does not succeed.

Example: `orchestration / send-retries = 1`

Enhanced Routing Script Options

Application Section

This section describes `Enhanced Routing Script` object Application section options.

alternate-url

Option section: `Application`

Configuration object: `Script object (CfgEnhancedRouting)`

Default value: none

Valid values: any valid URL

Value changes: For the next interaction that hits this resource

This option specifies the URL of the SCXML document to load. This is attempted if the value of the `url` option cannot be fetched or compiled (for example: application server is temporarily down, network error, script is malformed, and so on).

This is the option to use for `<callback> URI` functionality, as a failover mechanism to provision the SCXML application from an ORS Web Server.

The URL can be any one of the following protocols:

- `file:<path>`
- `http://<url>`
- `https://<url>`

The URL can also contain parameters which will be passed to the Application server. The values shown below are substituted at run-time based on the information in the interaction.

Parameter Element Descriptions

- `[DNIS]` The `DNIS` attribute of the interaction
- `[ANI]` The `ANI` attribute of the interaction
- `[DN]` The `ThisDN` attribute of the interaction
- `[CED]` The `CollectedDigits` attribute of the interaction
- `[EMAILFROM]` E-mail from address
- `[EMAILTO]` E-mail to address

- `[EMAILSUBJECT]` E-mail subject
- `[UDATA]` Expanded to the entire user data of the interaction in the format: `name1=value1&name2=value2&…`
- `[UDATA.*]` Expanded to the entire user data of the interaction in the format:`name1:value1,name2:value2,…`
- `[UDATA.name]` Expanded to the value of a specific user data key of the interaction value. For example: `&servicetype=[UDATA.ServiceType]` could resolve to `&servicetype=CreditCards`

Simple case

```
alternate-url = http://xserver.genesyslab.com:80/NewCallReq.asp<br>
```

With parameters:

```
alternate-url =  https://xserver.genesyslab.com:80/
NewCallReq.asp?ani=[ANI]&dnis=[DNIS]&servicetype=[UDATA.ServiceType]
```

Parameter Substitution

Orchestration Server allows substitution of parameters, provided in the `URL` of an SCXML application, with predefined values. There are several ways to utilize this functionality.

- Configure parameters in an `Enhanced Routing Script` object in Configuration Layer. The `URL` of the SCXML application specified in the `url` option or `alternate-url` option may contain parameters in the format `$$parameter-name$$` where `parameter-name` should match the name of the option configured in the `ApplicationParms` section. When this `parameter-name` is found, the parameter will be substituted for its value. For example, configure the following options in the `ApplicationParms` section:

  ```
  APPSERVER1=http://appserver:8080
   REGION1=CA
   REGION2=NY
  ```

- Configure the `url` and `alternate-url` options in the Application section:

  ```
  url=$$APPSERVER1$$/service/$$REGION1$$/Workflow1.scxml
   alternate-url=$$APPSERVER1$$/service/$$REGION2$$/Workflow2.scxml
  ```

  The `url` will be substituted at run-time with:

  ```
  http://appserver:8080/service/CA/Workflow1.scxml
  ```

  The `alternate-url` will be substituted at run-time with:

  ```
  http://appserver:8080/service/NY/Workflow2.scxml
  ```

- Sessions started using an HTTP request with parameters. For example:

  ```
  http://localhost:17011/scxml/session/start?
   src=http://appserver:8080/service/CA/
   Workflow1.scxml&APPSERVER1=http://appserver:8080®ION1=CA
  ```

  These parameters can be utilized in a started SCXML application as a value of the `hrf`attribute in an `<xi:include>` action element. For example:

```
<xi:include href="$$APPSERVER1$$/service/$$REGION1$$/RouteToAgGroup.scxml" …/>
```

At run-time, the `href` value will be substituted with:

```
"http://appserver:8080/service/CA/RouteToAgGroup.scxml"
```

- Session started with parameters specified in `<session:start>` action element. For example:

```
<session:start sessionid="_data.NewID" src="_data.Path + '/Workflow1.scxml'" >
 <param name="APPSERVER1" expr="'http://appserver:8080'"/>
 <param name="REGION1" expr="'CA'"/>
 </session:start>
```

Notes:

- Orchestration Server now sends the `HTTP fetch` request to the address specified in the `alternate-url` option after the `fetch-timeout` has expired.
- `parameter-name` can be used in all subroutines started from the main SCXML session. To avoid unexpected outcomes, Genesys recommends avoiding a `parameter-name` with a space at the beginning, end, or in the middle. `parameter-name` is case-sensitive.

</toggledisplay>

fetch-timeout

Option section: `Application` in an Enhanced Routing Script object

Configuration object: `Enhanced Routing Script object`

Default value: `5000`

Valid values: Any integer greater than `0`

Value changes: After restart

This option specifies the time (in milliseconds) before a document fetch operation is abandoned. If the SCXML document cannot be retrieved within this timeout value, the fetch operation is abandoned and a fetch operation for the `alternate-url` is attempted.

Orchestration Server sends the `HTTP fetch` request to the address specified in the `alternate-url` option after `fetch-timeout` has expired.

http-useragent

Option section: `Application`

Configuration object: `Script object (CfgEnhancedRouting)`

Default value: `GOES/8.0`

Valid values: Any string

Value changes: For the next document fetch

Specifies the value of the `User-Agent` header that ORS includes into originating HTTP requests that it sends when fetching an SCXML document (script) from an application server. For example: `http-User-Agent = MYAGENT`

http-version

Option section: `Application`

Configuration object: `Script object (CfgEnhancedRouting)`

Default value: `1.1`

Valid values: `1.0, 1.1`

Value changes: For the next document fetch

This option specifies the HTTP version to use when fetching an SCXML document from an application server.

For example: `http-version = 1.1`

max-age

Option section: `Application`

Configuration object: `Script (CfgEnhancedRouting) object`

Default value: `0`

Valid values: Any integer greater than or equal to `0`

Value changes: For the next change of the SCXML document

Defines time in seconds for how long a fetched SCXML application is cached. When the same SCXML application is to be used for the new session within the configured timeout, the cached version of the document will be used instead of fetching it from the application server.

- If the option value is empty or set to `0`, fetching this document from the cache will be disabled.
- If `max-age` is configured for an Enhanced Routing Script object, the value from `max-age` takes precedence over the `http-max-age` Application level option.
- If the URL of SCXML application contains parameters that are unique for each invocation, then the application will be fetched each time regardless of the option value.

max-stale

Option section: `Application`

Configuration object: `Script (CfgEnhancedRouting) object`

Default value: `0`

Valid values: Any integer greater than or equal to `0`

Value changes: For the next fetch of the SCXML document

Defines the time in seconds to extend the lifetime of a cached SCXML application. For example, if the value of the `max-age` option is set to 60 seconds and the value of the `max-stale` option is set to 30 seconds, then the fetched document will be cached for the timeframe of 90 seconds.

- If the option value is empty or set to `0`, fetching this document from the cache will be disabled.
- If `max-stale` is configured for an Enhanced Routing Script object, the value from `max-stale` takes precedence over the `http-max-stale` Application level option.

url

Option section: `Application`

Configuration object: `Script (CfgEnhancedRouting) object`

Default value: none (this is a required option)

Valid values: any valid URL

Value changes: For the next interaction that hits this resource

This option specifies the URL of the SCXML document to load. The URL can be any one of the following protocols:

- `file:<path>`
- `http://<url>`

The URL can also contain parameters which will be passed to the Application server. The values shown in Parameter Element Descriptions below are substituted at run-time based on the information in the interaction.

Parameter Element Descriptions

- `[DNIS]` The `DNIS` attribute of the interaction
- `[ANI]` The `ANI` attribute of the interaction
- `[DN]` The `ThisDN` attribute of the interaction
- `[CED]` The `CollectedDigits` attribute of the interaction
- `[EMAILFROM]` E-mail from address
- `[EMAILTO]` E-mail to address
- `[EMAILSUBJECT]` E-mail subject
- `[UDATA]` Expanded to the entire user data of the interaction in the format: `name1=value1&name2=value2&…`
- `[UDATA.*]` Expanded to the entire user data of the interaction in the format:`name1:value1,name2:value2,…`

- `[UDATA.name]` Expanded to the value of a specific user data key of the interaction value. For example: `&servicetype=[UDATA.ServiceType]` could resolve to `&servicetype=CreditCards`

For example:

```
url = http://xserver.genesyslab.com:80/NewCallReq.asp
```

```
url = http://xserver.genesyslab.com:80/
NewCallReq.asp?ani=[ANI]&dnis=[DNIS]&servicetype=[UDATA.ServiceType]
```

Orchestration Server allows substitution of parameters, provided in the URL of an SCXML application, with predefined values. For more information, see the `alternate-url` option.

{Parameter Name}

Option section: `ApplicationParms`

Configuration object: `Script object (CfgEnhancedRouting)`

Default value: none

Valid values: Any string

Value changes: For the next interaction that hits this resource

This option specifies a string that represents a parameter value to be passed to the application.

The `ApplicationParms` section contains the values for data elements that may be referred to within the SCXML application. The parameters can be statically defined for each application (for example, `Service = Sales`) or contain substitution values that will be substituted at run-time based on the information in the interaction.

For example: `Segment = [UDATA.CustomerSegment]`

Parameter Element Descriptions

- `[DNIS]` The `DNIS` attribute of the interaction
- `[ANI]` The `ANI` attribute of the interaction
- `[DN]` The `ThisDN` attribute of the interaction
- `[CED]` The `CollectedDigits` attribute of the interaction
- `[EMAILFROM]` E-mail from address
- `[EMAILTO]` E-mail to address
- `[EMAILSUBJECT]` E-mail subject
- `[UDATA.name]` Expanded to the value of a specific user data key of the interaction value. For example: `&servicetype=[UDATA.ServiceType]` could resolve to `&servicetype=CreditCards`

For example:

```
ApplicationParms =
Service = Sales
EmailSubject = [EMAILSUBJECT]
Segment = [UDATA.CustomerSegment]
AppServer = http://myappserver:8080
```

ORS and SIP Server

Orchestration Server supports the SIP Server Advice of Charge (AoC) feature by implementing the business logic to insert charge messages. This feature enables SIP Server to act as a Charging Determination Point (CDP) to specify the charges for using a service. SIP Server sends the CDP data to a Charging Generation Point (CGP) in Secure SIP (SIPS) INFO messages action. The charge information is initiated by using the PrivateServiceRequest action. Functionality for this feature is based on the 3GPP Specification, TS 32.280.

PrivateServiceRequest Action

Used in conjunction with SIP Server 8.1.0, the `<privateservice>` action enables users to implement the Advice of Charge (AoC) feature in their solution. This action enables an application to pass data and request services that are supported only by certain T-Servers and are not covered by general feature requests. Request services can include Set Feature, AoC, change T-Server behavior, and so on. The `<privateservice>` action is equivalent to the `TPrivateService` T-lib request. See the applicable T-Server documentation for information about the request `TPrivateService` request.

Attribute Details

The table below contains the `<privateservice>` action attributes and their descriptions. There are no default values for any of these attributes.

Attribute Name	Type	Valid Values	Description
`requestid` (not required)	Location expression	Any valid location expression	Represents the location of the request ID that is returned in the request. Any data model expression evaluating to a data model location. The location's value is configured as an internally generated unique string identifier that is associated with the action that is sent. If this attribute is not specified, the event identifier is dropped. This identifier can be tested by the completion event handler to distinguish between multiple

			outstanding requests. If this attribute is not specified, the identifier can be acquired from the fetch completion event. Every request must receive a unique identifier.
`serviceid` (required)	Value expression	Any value expression that returns a valid integer	A value expression that returns an integer to indicate the type of information that is passed or the service that is requested. It is specific to the T-Server that is handling the call. Before you configure this value, check the T-Server documentation for your switch.
`Interactionid` (required)	Value expression	Any value expression that returns a valid integer	A value expression that returns the `_genesys.FMname.interactions[x].g` interaction ID that is associated with this request. Non voice interactions can result in the generation of an `error.voice.privateservice error`.
`resource` (not required)	Value expression	Any valid string or resource object	A value expression that returns the DN of the controlling agent or route point for which the information is provided. This attribute corresponds to the `thisDN` parameter within the TLib `TPrivateService` method. Before you configure this value, check the T-Server documentation for your switch. Orchestration Server distributes RequestPrivateService to SIP Server when the resource attribute is not specified in the <ixn:privateservice> action element when an inbound call diverted from a Routing Point to an external destination is answered. When

			integrated with SIP Server, the extensions parameter of the <ixn:privateservice> action element should contain key-value pair AOC-Destination-DN with the value referring to the party in established state where the AoC (Advice of Charge) notification should be delivered. Note: This feature is available when integrated with SIP Server beginning with version 8.1.100.93.
`udata` (not required)	ECMAScript object	Any valid ECMAScript object	An ECMAScript object that contains the list of key/value pairs are attached to the call in question.
`reasons` (not required)	ECMAScript object	Any valid ECMAScript object	An ECMAScript object that contains the list of key/value pairs that provide additional information associated with this Private Service request, and is intended to specify reasons for and results of actions taken by the user.
`extentions` (not required)	ECMAScript object	Any valid ECMAScript object	An ECMAScript object that contains the list of key/value pairs that provide an additional data structure that is intended to take into account the switch-specific features that cannot be described by other parameters, or the list of key/value pairs in the original structure of the user data that is associated with this Private Service request. See the resource attribute for additional information.
`hints` (not required)	Value expression	Any valid ECMAScript object	A value expression that returns the ECMAScript object that contains information that might

			be used by the implementing functional module while performing this action. This information might consist of protocol-specific parameters, protocol selection guidelines, or other related data. The meaning of these hints is specific to the implementing functional module.

Note: For more information, see the State Chart XML (SCXML): State Machine Notation for Control Abstraction, W3C Working Draft 7 May 2009 W3C Specification.

Determining the Target T-Server

The target T-Server to which the Private Service request is submitted is determined by the following factors:

- If the resource attribute contains both a switch and DN, the switch is used to locate the T-Server to which the request is submitted.
- If the resource attribute contains only the switch, the switch is used to locate the T-Server to which the request is submitted and the thisDN parameter value of the underlying TLib TPrivateService request is not populated.
- If the resource attribute contains only a DN, the switch and associated T-Server are determined by the interaction ID. The switch is determined, based on the first party that references the DN resource.
- If the resource is not provided, the T-Server is determined by the last party within the associated party entries for the associated interaction. In this case, the target T-Server does not provide a resource (thisDN parameter).

If the target T-Server cannot be determined by using the provided information, an error.voice.privateservice error is generated. See the following example:

```
<state id="do_private_service">

<datamodel>

<data id="reqid"/>

</datamodel>

<onentry>

          <script>
```

```
                var myuserdata = {details : {name: "Smith, John", age : 45} };
                var myreasons = {code: "New Update"};
                var myextensions = { keyname : "Its value"};
                var myhints = {handle_responses : "false"};
          </script>
          <ixn:privateservice requestid="_data.reqid"
          serviceid="1234" interactionid="_genesys.ixn.interactions[0].g_uid"
          resource="’9000’"
          udate = "myuserdata"
          reasons = "myreasons"
          extensions = "myextensions"/>
</onentry>
<transition event="voice.privateservice.done" target="statex"/>
<transition event="error.voice.privateservice" target="statey"/>
</state>
```

Children

There are no child objects.

Events

The following events can be generated as part of this action, but are specific to the service and T-Server implementation. Therefore, Genesys recommends that you refer to the appropriate T-Server manual for information about how and when to generate them.

- `voice.privateservice.done`—This event is sent when the request is accepted and sent by Orchestration Server. It is not an indication that the T-Server handled or accepted the event.
- `error.voice.privateservice`—This event is sent if, for any reason, the request itself fails.

SCXML Application Development

You can create SCXML-based applications by using the following methods:

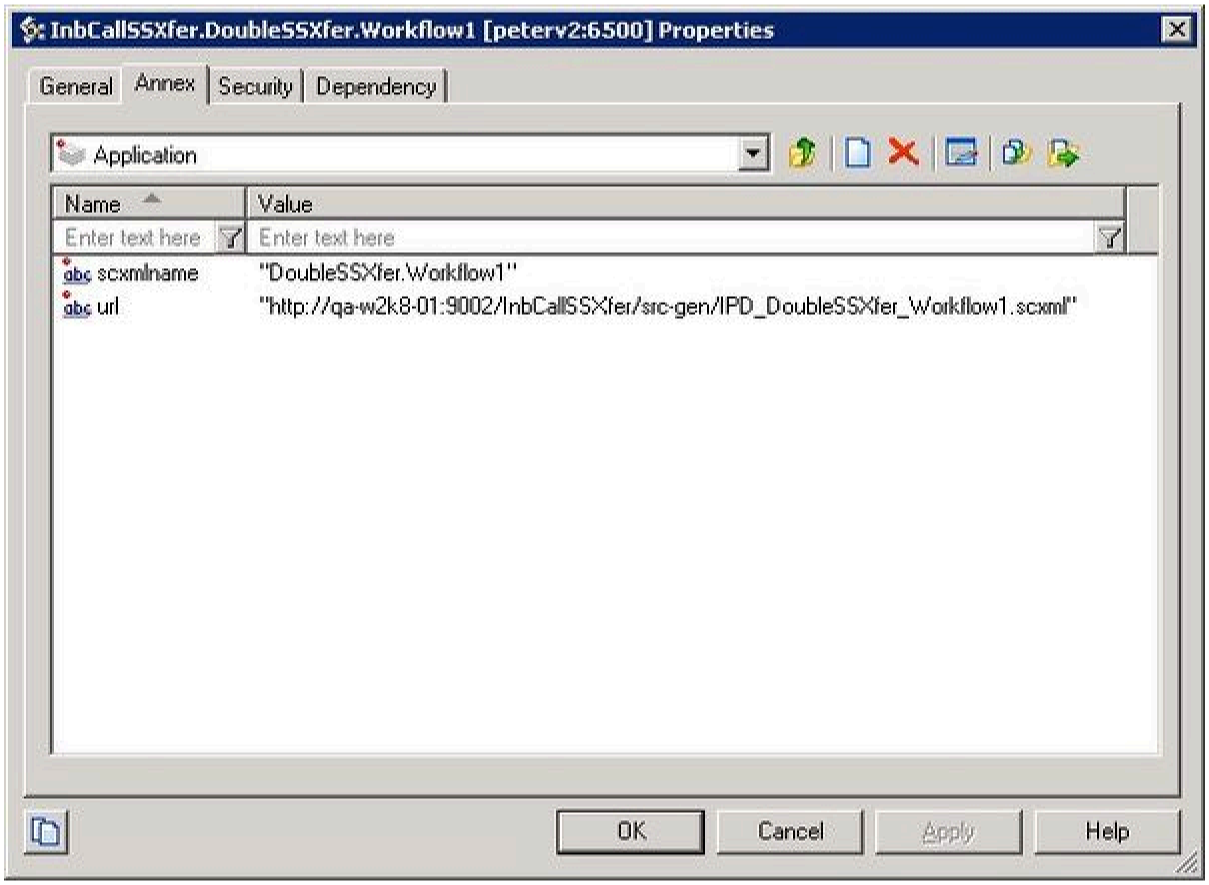
- Any simple text editor such as Notepad or an XML-based editor, with which you are already comfortable. Use the Orchestration Server Developer’s Guide for the details.

- Using Genesys Composer, which is an Integrated Development Environment based on Eclipse. Composer provides both drag-and-drop graphical development of voice applications (or "callflows") and routing strategies (or "workflows") as well as syntax-directed editing of these applications.

For more information on using this GUI, see the Composer 8.1.x Help. Also see the Composer 8.1.3 Deployment Guide.

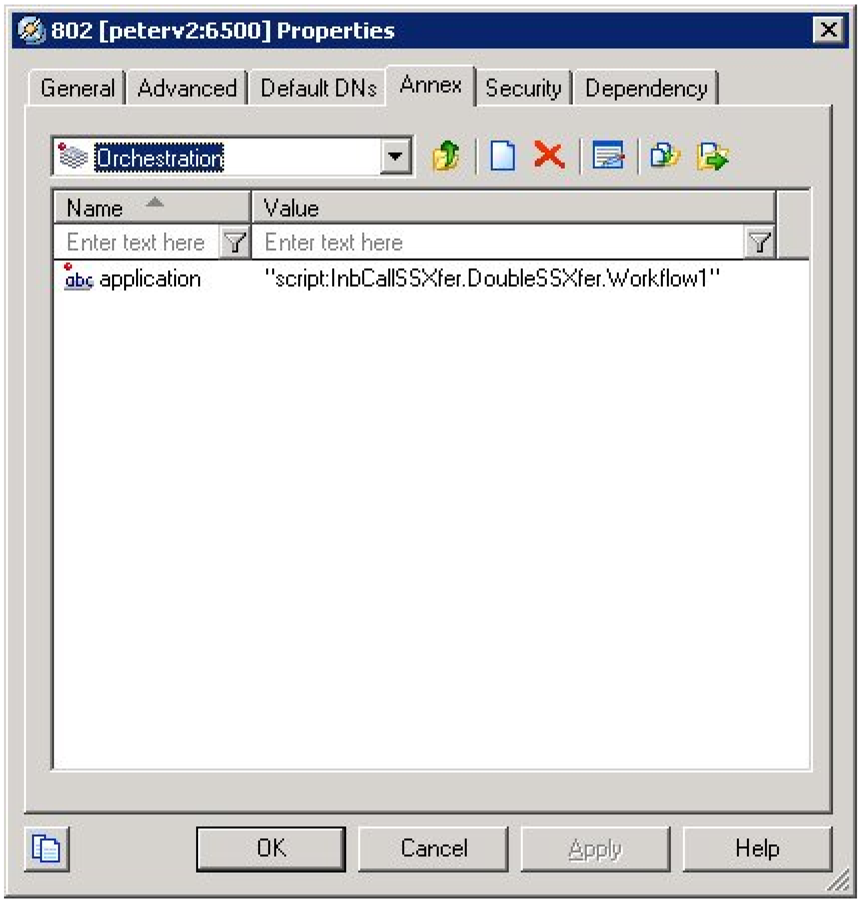
Deploying Voice SCXML Applications

1. Within Composer, click toolbar icon to create a new Java Composer Project.
2. Specify the `Route` type. Specifying `Route` indicates the development ORS/URS routing strategies.
3. Build the interaction process diagram, which was created automatically in step 1 above (default name: `default.ixnprocess`).
4. In the interaction process diagram (`default.ixnprocess`), locate the `Resource` property, and point to the workflow (`Workflows/default.workflow`), which can be developed later.
5. Within Composer, connect to Configuration Server so you can view/select objects, such as agents, places, and so on.
6. Develop your workflow diagram (routing strategy) using the diagram-building blocks from the palette, such as from the `Flow Control` and `Routing block` categories.
7. Save the workflow and interaction process diagrams.
8. Validate the workflow and interaction process diagrams.
9. View any warnings or problem messages and correct.
10. Generate the code. As a result, two SCXML files are generated and can be viewed in an SCXML editor.
11. Publish the interaction process diagram to Configuration Server. An `Enhanced Routing Script` object will be created. The figure below shows an example. The figure below shows an `Enhanced Routing Script` object with option url set to the location of the SCXML application.



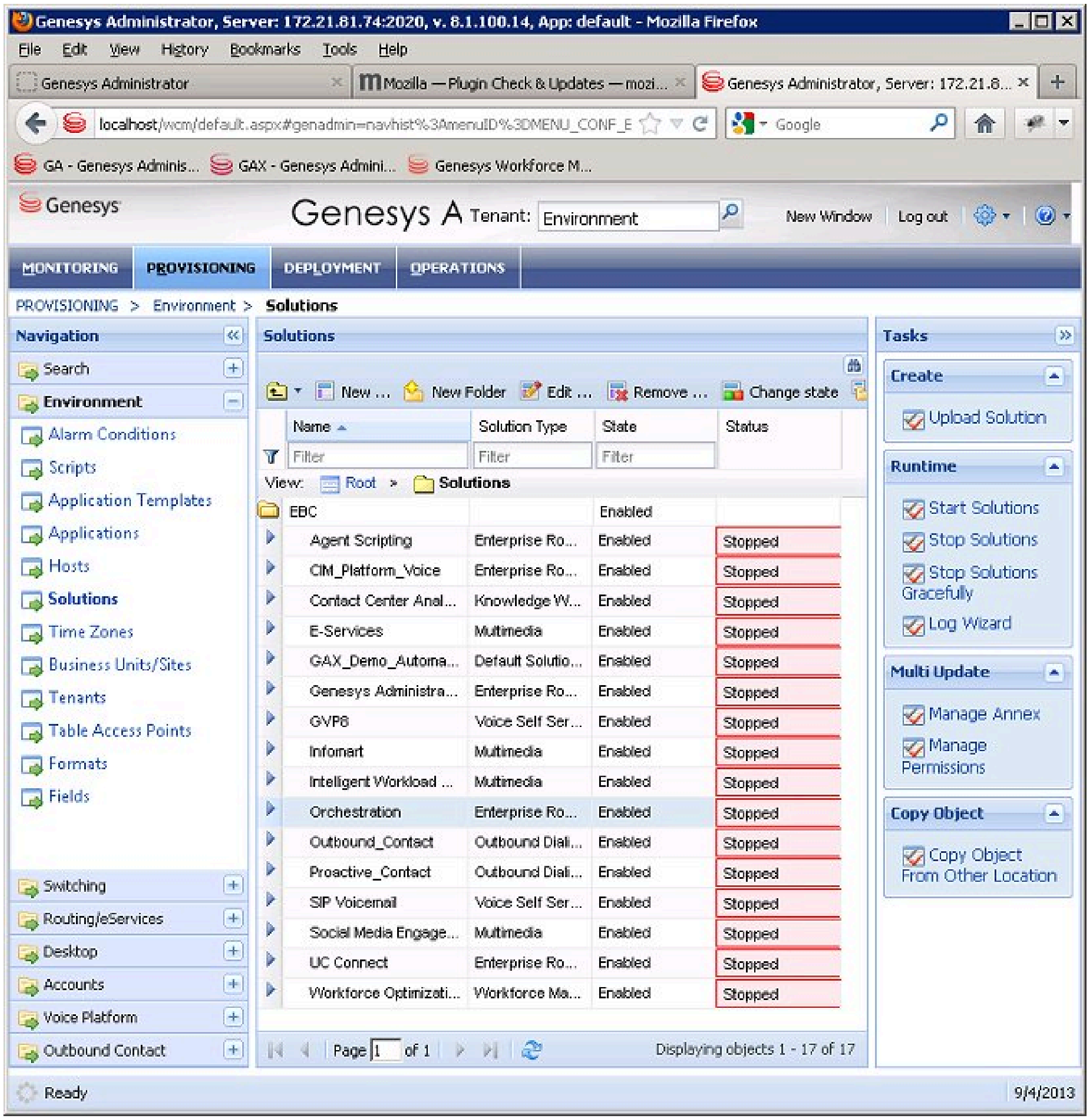
Name	Value
scxmlname	"DoubleSSXfer.Workflow1"
url	"http://qa-w2k8-01:9002/InbCallSSXfer/src-gen/IPD_DoubleSSXfer_Workflow1.scxml"

12. Load the published application to a Routing Point. The figure below shows the Routing Point with the application option set to the name of the `Enhanced Routing Script` object.



Name	Value
application	"script:InbCallSSXfer.DoubleSSXfer.Workflow1"

- Log into Genesys Administrator.
- Navigate to `Provisioning>Routing/eServices >Orchestration`


Genesys Administrator, Server: 172.21.81.74:2020, v. 8.1.100.14, App: default - Mozilla Firefox
File Edit View History Bookmarks Tools Help
localhost/wcm/default.aspx#genadmin=navhist%3AmenuID%3DMENU_CONF_E
MONITORING PROVISIONING DEPLOYMENT OPERATIONS
PROVISIONING > Environment > Solutions
Navigation
Search
Environment
Alarm Conditions
Scripts
Application Templates
Applications
Hosts
Solutions
Time Zones
Business Units/Sites
Tenants
Table Access Points
Formats
Fields
Switching
Routing/eServices
Desktop
Accounts
Voice Platform
Outbound Contact
Tasks
Create
Upload Solution
Runtime
Start Solutions
Stop Solutions
Stop Solutions Gracefully
Log Wizard
Multi Update
Manage Annex
Manage Permissions
Copy Object
Copy Object From Other Location
New ... New Folder Edit ... Remove ... Change state
View: Root > Solutions
Page 1 of 1
Displaying objects 1 - 17 of 17
Ready
9/4/2013

Name	Solution Type	State	Status
EBC		Enabled	
Agent Scripting	Enterprise Ro...	Enabled	Stopped
CIM_Platform_Voice	Enterprise Ro...	Enabled	Stopped
Contact Center Anal...	Knowledge W...	Enabled	Stopped
E-Services	Multimedia	Enabled	Stopped
GAX_Demo_Automa...	Default Solutio...	Enabled	Stopped
Genesys Administra...	Enterprise Ro...	Enabled	Stopped
GVP8	Voice Self Ser...	Enabled	Stopped
Infomart	Multimedia	Enabled	Stopped
Intelligent Workload ...	Multimedia	Enabled	Stopped
Orchestration	Enterprise Ro...	Enabled	Stopped
Outbound_Contact	Outbound Diali...	Enabled	Stopped
Proactive_Contact	Outbound Diali...	Enabled	Stopped
SIP Voicemail	Voice Self Ser...	Enabled	Stopped
Social Media Engage...	Multimedia	Enabled	Stopped
UC Connect	Enterprise Ro...	Enabled	Stopped
Workforce Optimizati...	Workforce Ma...	Enabled	Stopped

- Locate the Enhanced `Routing Script` object.
- Click on **DNs** tab.
- Add your DN (as in the above Routing Point example).
- Load this application to this Routing Point.

13. Test that the application is successfully provisioned and deployed. Make a test call.

Manually Deploying a Voice SCXML Application

1. Create SCXML in the editor of your choice.
2. Manually deploy your SCXML application on an application server.
3. Load deployed application to a routing point.

Debugging SCXML Applications with Composer

The SCXML Real Time Debugging feature of Composer provides the possibility for developers to debug their SCXML scripts in a manner similar to what is provided by Visual Studio and Eclipse.   You can examine variables and properties, as well as pause and resume the flow of execution of a script, by setting breakpoints in the code, viewing standard SCXML Engine metrics (log entries). The ORS feature works using a client-server configuration where the client is the Genesys Composer integrated development environment. The entire RTD system is designed in such a way that it stops at every possible statement, and it is the responsibility of the client to determine whether to resume execution of the script right away or to pause and give control to the developer; in other words, determine whether or not the developer has set a breakpoint to this particular statement.

Debugging can be started on an existing session or it can wait for the next session that runs the application at a given URL.

To enable Orchestration Routing's debugging functionality, configure the debugging options in the SCXML section of the ORS application. See the following options:

- `debug-port`
- `debug-enabled`
- `max-debug-sessions`

The debugging SCXML applications functionality is supported in Composer starting from v8.1.2. For more information, see the Composer Deployment Guide.

Limitation

The `<createcall>` and `<consult>` interaction action elements have a limitation for the attribute "to" (where a destination is specified). Only digits can be specified in the "to" attribute.

The limitation can be removed by configuration of `valid-digits` and `max-digits` options on the Switch object > Annex tab> `gts` section:

Option: **valid-digits**
Value: An explicit set of all acceptable symbols that can be dialed.
Example: `0123456789_RPO`

Option: **max-digits**

Value: The maximum number of digits that can be dialed.
Example: 25

Samples

Orchestration Server supports only SCXML documents. The *Orchestration Developer's Guide* contains samples of SCXML routing applications. The samples are not designed for use in a Production environment. Instead, use them to get started configuring your own applications, subroutines, and list objects. Consider them as guides when developing your own objects adjusted to your company’s specific business needs.

Install Start Stop

This section explains how to:

- Install Orchestration Server
- Start and stop Orchestration Server
- Uninstall Orchestration Server

Installation

- For a list of supported operating systems and databases, see the Genesys Supported Operating Environment Reference Guide, which is available on the Genesys Documentation website at `docs.genesys.com`.
- Genesys does not recommend installation of its components via a Microsoft Remote Desktop connection. The installation should be performed locally.

Installation Package Location

The installation package, whether on CD or from an FTP site, contains a setup folder for Orchestration Server.

When FTP delivery is used, there are separate setup folders for Windows and Linux. Linux compatibility packages always should be installed before installing Genesys IPs.

On the Orchestration Server CD, the setup file is located as follows:

Find the file in the folder named for your operating system:

`\\solution_specific\orchestration_server\linux`

`\\solution_specific\orchestration_server\windows`

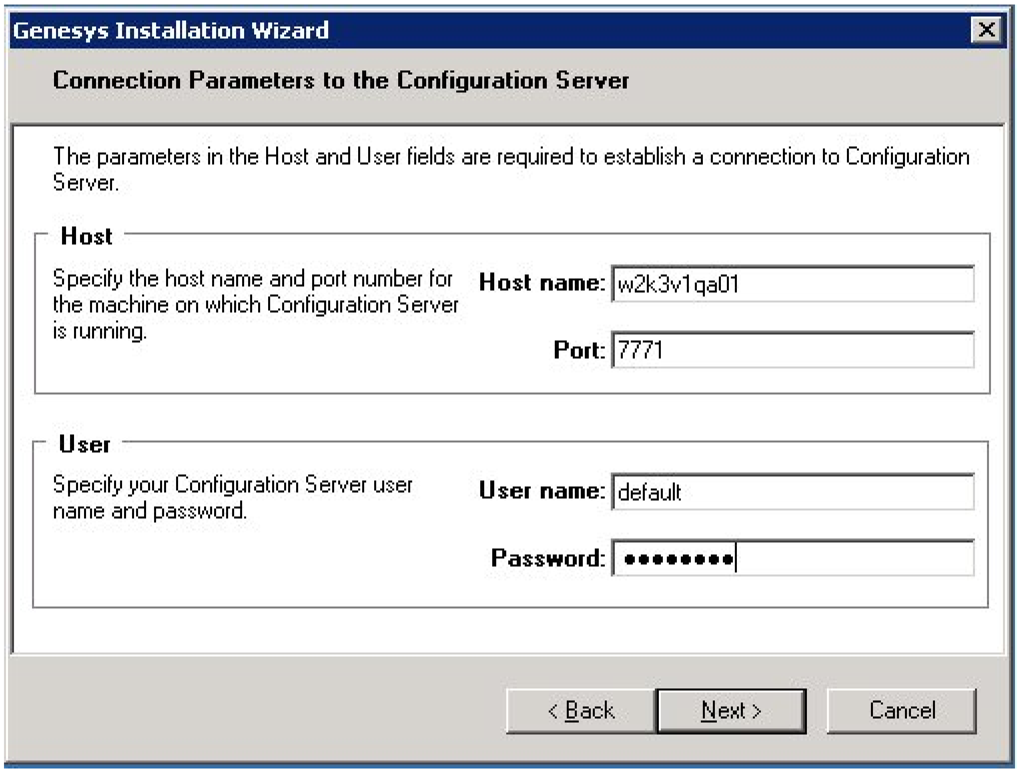
Note: If you install several instances of Orchestration Server component on the same computer, a separate shortcut is created for each one, based on the Application name stored in Configuration Layer.

Installing on Windows

The installation process does not present the option of installing a server component as a service. By default, starting with 7.5, all server components (excluding Genesys Desktop and Multimedia/eServices components) are installed as services in automatic startup mode.

Using the Installation Wizard

1. Double-click `setup.exe`.
2. On the Orchestration Server CD, this file is located in the following folder: `\\solution_specific\orchestration_server\windows`. If Orchestration Server was downloaded from an FTP site, the file is located in the download directory. The Install Shield opens the **Welcome** screen.
3. Click **Next**. The **Connection Parameters to the Configuration Server** dialog box appears.


Genesys Installation Wizard

Connection Parameters to the Configuration Server

The parameters in the Host and User fields are required to establish a connection to Configuration Server.

Host

Specify the host name and port number for the machine on which Configuration Server is running.

Host name: w2k3v1qa01

Port: 7771

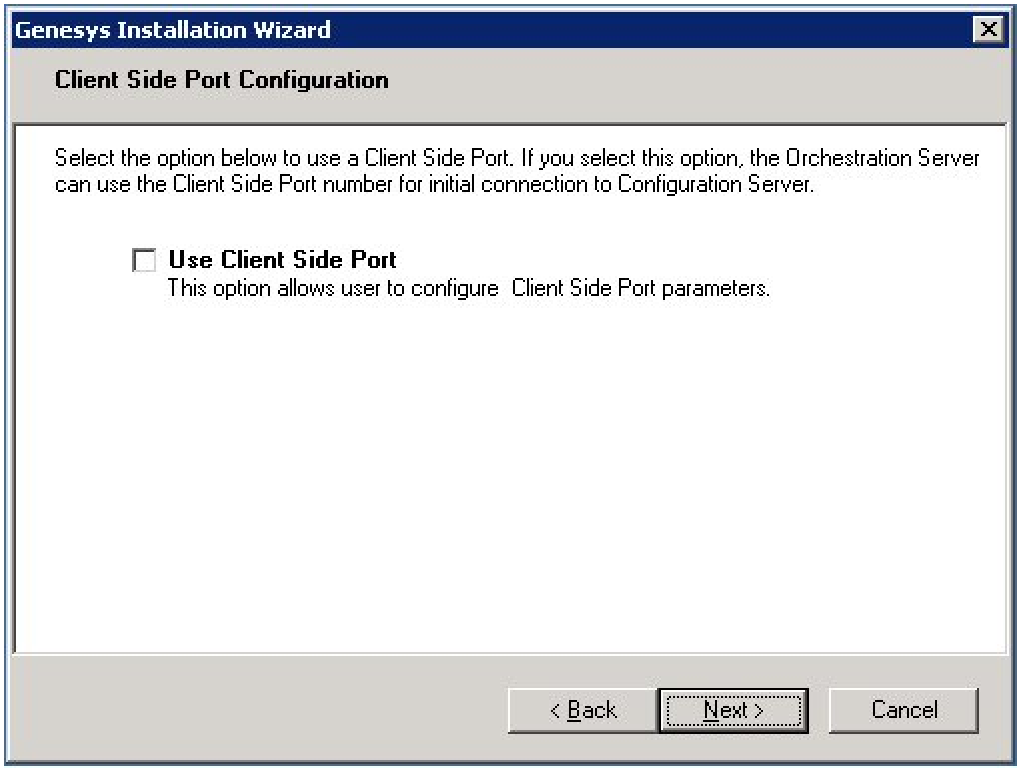
User

Specify your Configuration Server user name and password.

User name: default

Password: ●●●●●●●●

4. Under **Host**, specify the `Host` name and port number for the computer on which Configuration Server is running. This is the main "listening" port entered in the **Server Info** tab for Configuration Server, which is used for authentication in the Configuration Manager login dialog box.
5. Under **User**, enter the user name and password used for logging on to Configuration Server.
6. Click **Next** to open the **Client Side Port Configuration** dialog box.

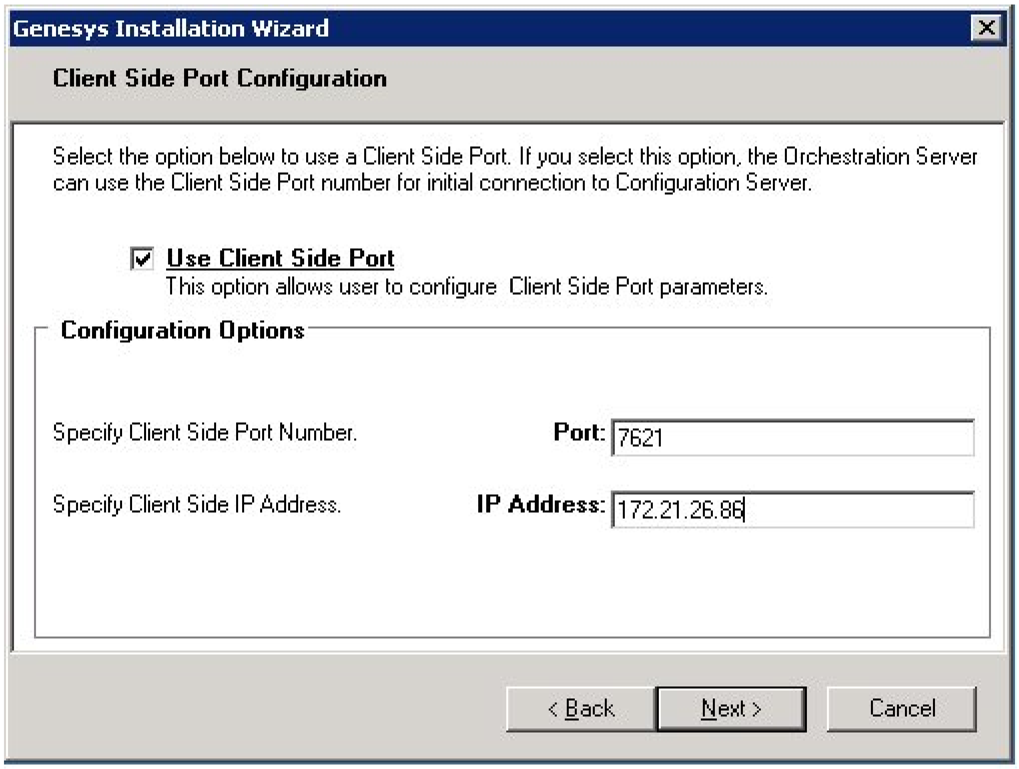


Client Side Port Configuration

Select the option below to use a Client Side Port. If you select this option, the Orchestration Server can use the Client Side Port number for initial connection to Configuration Server.

☐ Use Client Side Port
This option allows user to configure Client Side Port parameters.

7. If you are setting up client-side port configuration for the initial connection to Configuration Server as described in the *Genesys 8.1 Security Deployment Guide*, select the Use **Client Side Port** check box to reveal additional fields.



Select the option below to use a Client Side Port. If you select this option, the Orchestration Server can use the Client Side Port number for initial connection to Configuration Server.

☑ **Use Client Side Port**
This option allows user to configure Client Side Port parameters.

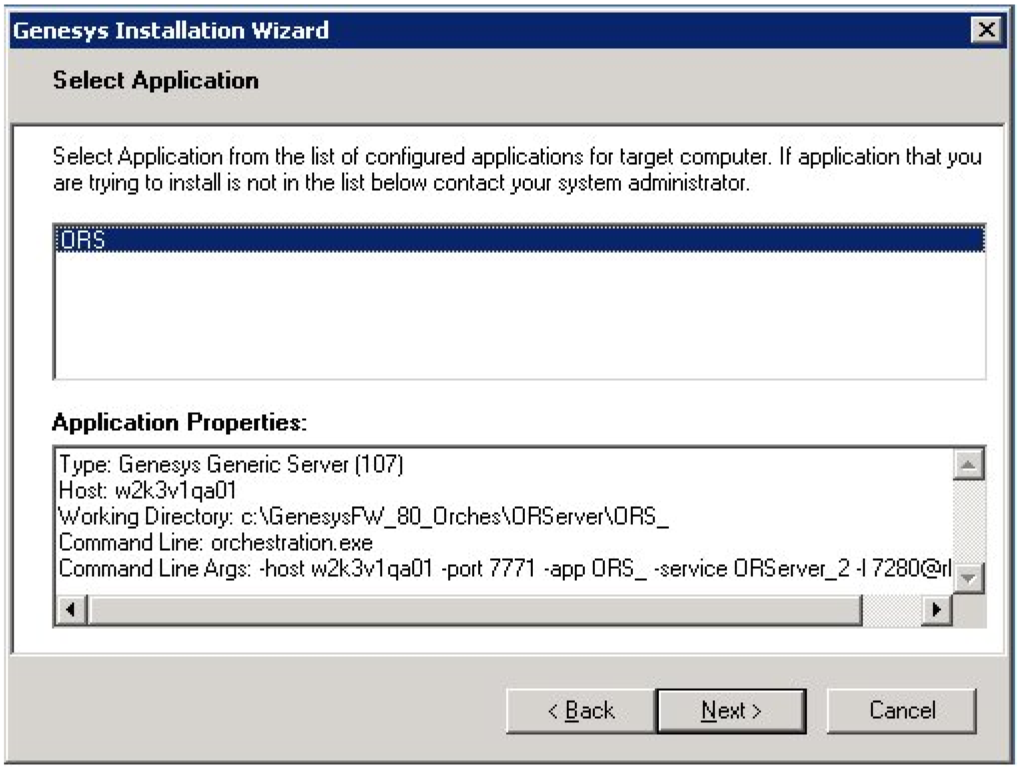
Configuration Options

Specify Client Side Port Number. **Port:** 7621

Specify Client Side IP Address. **IP Address:** 172.21.26.86

8. Specify the following parameters:
   - Enter any free port number (this is not the Listening port in the **Server Info** tab of the Orchestration Server `Application` object).
   - Enter the IP Address of the computer on which you are installing and running the Orchestration Server `Application`.

Note: After entering this information, the installation process will add the necessary command line arguments (`-transport-address and -transport-port`) for connecting to Configuration Server during Application startup.



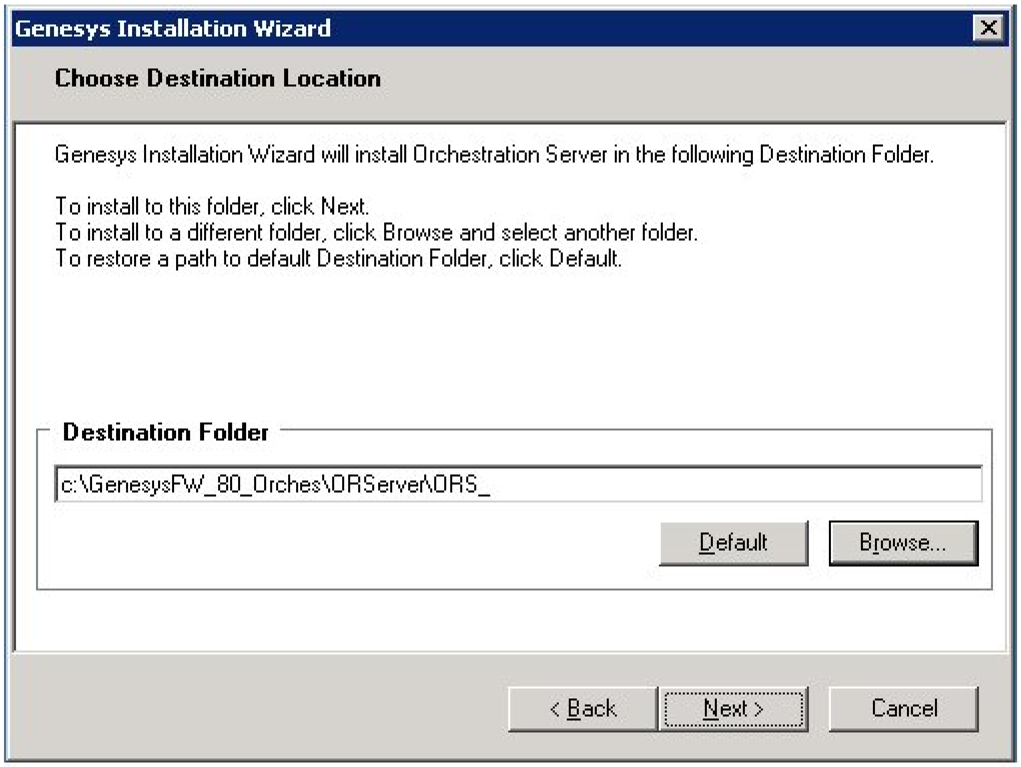
Select Application from the list of configured applications for target computer. If application that you are trying to install is not in the list below contact your system administrator.

ORS

Application Properties:

Type: Genesys Generic Server (107)
Host: w2k3v1qa01
Working Directory: c:\GenesysFW_80_Orches\ORServer\ORS_
Command Line: orchestration.exe
Command Line Args: -host w2k3v1qa01 -port 7771 -app ORS_ -service ORServer_2 -l 7280@rl

9. Click **Next**. The **Select Application** dialog box appears. Example entries are shown below.
10. Select the Orchestration Server `Application` that you are installing. The **Application Properties** area shows the `Type`, `Host`, `Working Directory`, `Command Line` executable, and `Command Line Arguments` information previously entered in the **Server Info** and `Start Info` tabs of the selected Orchestration Server `Application` object.
11. Click **Next**. The **Choose Destination Location** dialog box appears.



Choose Destination Location

Genesys Installation Wizard will install Orchestration Server in the following Destination Folder.

To install to this folder, click Next.
To install to a different folder, click Browse and select another folder.
To restore a path to default Destination Folder, click Default.

Destination Folder

c:\GenesysFW_80_Orches\ORServer\ORS_

12. Under **Destination Folder**, keep the default or browse for the installation location for Orchestration Server.
13. Click **Next**. The **Ready to Install** dialog box appears.
14. Click **Next**. The Genesys Installation Wizard indicates it is performing the requested operation for Orchestration Server. When through, the **Installation Complete** dialog box appears.
15. Click **Finish**.

Installing on UNIX-Based Platforms

Before you start the installation, make sure all instances of Orchestration Server are already installed on your computer and shut down. If you do not do this, you will not be able to back up your files if you want to use the same installation directory for another version of those components. Linux compatibility packages should be always installed before installing Genesys IPs.

1. Go to the directory where the installation is created.
2. Copy all files to a temporary directory. **Note:** Files included in the installation package require permission to execute.
3. Run the installation script by typing `./install.sh`
4. When prompted, enter the host name of the computer where Orchestration Server will be installed or press the `Enter` key for the supplied entry.
5. When prompted, enter the following information about your Configuration Server:

- - `Configuration Server Host name`
  - `Network port`
  - `User name`
  - `Password`

  Prompts appear regarding securing connections between Orchestration Server and Configuration Server.

  ```
  Client Side Port Configuration
   Select the option below to use a Client Side Port. If you select this option,
   the application can use Client Side Port number for initial connection to Configuration
   Server.
   Do you want to use Client Side Port option (y/n)
  ```

6. When prompted, type either Y for yes or N for no. The instructions below assume you typed Y.
7. Enter an IP address or press Enter for the supplied entry after the following prompt:

   ```
   Client Side IP Address (optional), the following values can be used:
   ```

8. Choose the Orchestration Server Application to install after this prompt (which may list several Orchestration Servers):

   ```
   Please choose which application to install:
    1: ORS_application
   ```

9. Enter the destination directory for the installation after this prompt:

   ```
   Please enter full path of the destination directory for installation:
   ```

   After you enter the destination directory, the installation continues. A message appears that starts with

   ```
   xtracting tarfile: d
   ```

10. When this instruction appears:

    ```
    There are two versions of this product available: 32-bit and 64-bit.
    ```

    Enter 32 or 64 according to the Linux (Solaris)type that you use.

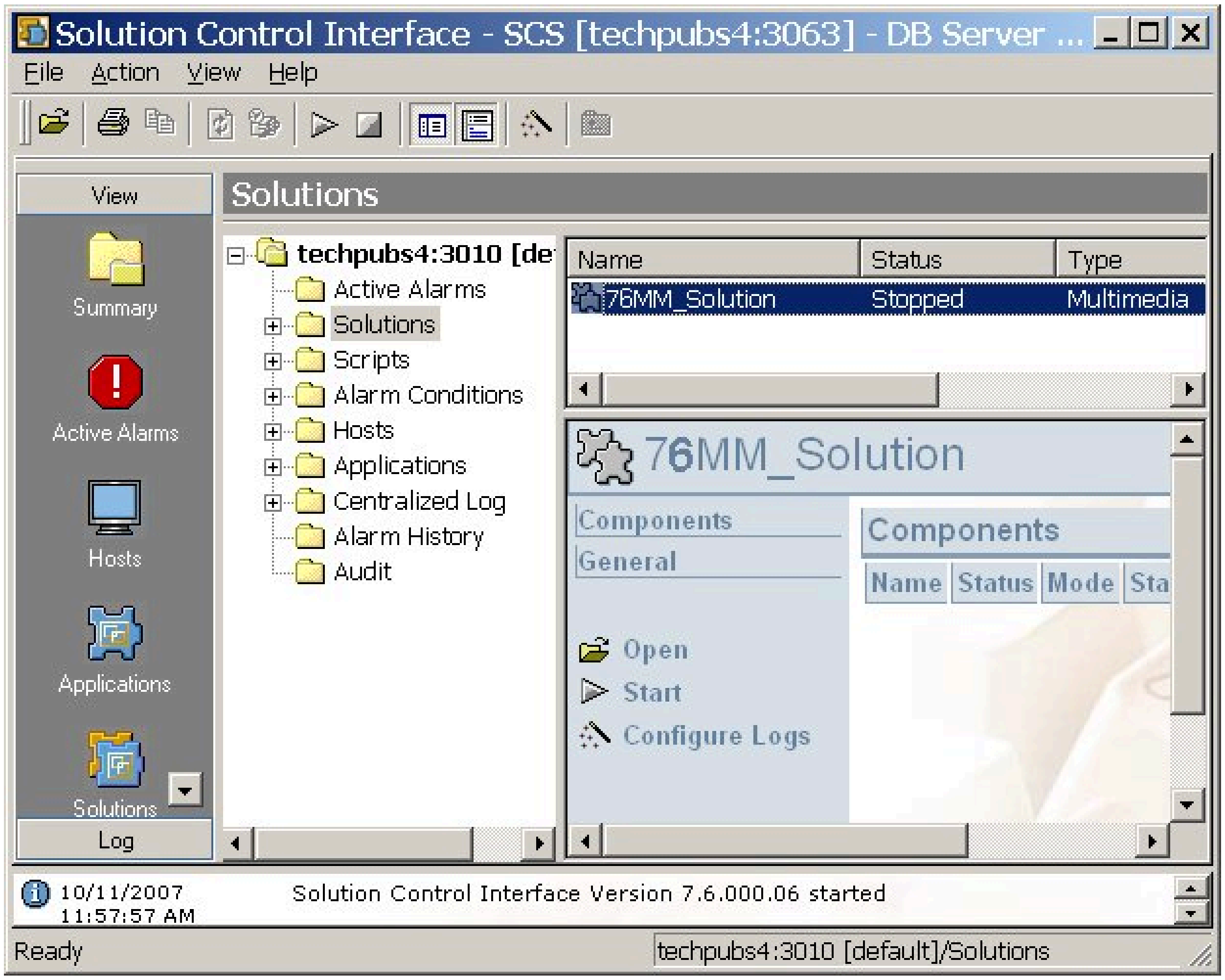
As soon as the installation process is finished, a message appears announcing that installation was successful. The process created a directory, with the name specified during the installation, containing Orchestration Server.

Starting and Stopping

You can start ORS using Solution Control Interface or manually.

Starting ORS with Solution Control Interface

The figure below shows an example that could include Orchestration Server as well as other server applications.



Name	Status	Type
76MM_Solution	Stopped	Multimedia

10/11/2007 11:57:57 AM Solution Control Interface Version 7.6.000.06 started

1. Go to the **Solutions** view.
2. Right-click on the desired solution and select **Start** from the shortcut menu OR
3. Select the desired solution and choose **Action** > **Start** on the menu bar.

Starting Orchestration Server Manually

For information on manually starting Framework components necessary for using Orchestration Server, see the Framework 8.1 Deployment Guide.

- To start Orchestration Server manually, select Start > All Programs > **Genesys Solutions** > **Routing** > **Orchestration Server** > **ORS** > **Start Orchestration Server**. This path is the default location. If you installed the software at a different location, navigate to the appropriate location to start ORS.
- You can also start Orchestration Server as follows: **Programs** > **Administrative Tools** > **Services** > **Genesys Orchestration Server**, and choose **Run** or **Stop**.
- You can also start from the command line or the **Start Info** tab for ORS in Solution Control Interface.

ORS runs automatically after rebooting Windows. To change this: choose ORS in `Services > Properties`,startup type “manual”.

Starting ORS on UNIX-Based Platforms

Installation of ORS creates a `run.sh` file. You can start Orchestration Server on UNIX platforms by just running this file which contains:

```
./orchestration -host <name of Configuration Server host> -port <name of
Configuration Server port> -app <name of ORS Application> -l <the full path and name
of license file>
```

1. Open a terminal window.
2. Log in.
3. Choose the appropriate directory.
4. Run the `run.sh` file.

The text in angle brackets (<text>) above indicates the variables you enter that are unique to your environment and are required. Your information should replace the text and the brackets. See the example below for clarification. The following is an example of a `run.sh` file:

```
./orchestration -host Daemon -port 5010 -app "OR_Server” -l "/FLEXlm /license.dat”
```

Quotation marks are required before and after the name of the ORS application. The license file path must also be enclosed in quotation marks.

When ORS is started, a window opens and messages are sent regarding its status. ORS also establishes connections to all servers listed in the Connections tab of the Orchestration Server Application object.

Stopping

Orchestration Server should be stopped using the Solution Control Interface (SCI).

1. Start the Solution Control Interface.
2. Go to the **Applications** view.
3. Right-click on the desired application and select **Stop** from the shortcut menu OR
4. Select the desired application and choose **Action** > **Stop** on the menu bar.

The command to stop the application is sent to Solution Control Server, which uses Local Control Agent to terminate the application. SCI reports a successful stop of the application. When stopped, Orchestration Server's status changes from `Started` to `Stopped`.

Non-Stop Operation

The `non-stop operation` (NSO) feature enables ORS to continue to run even if it encounters problems. NSO prevents a shutdown in the event of failures. This works by allowing ORS to operate on two levels that are designated by the command-line parameters described below.

Built-in NSO provides the option of running ORS in non-stop operation mode (NSO).

`Note:` When ORS is started, non-stop operation is disabled by default.

The command-line parameter -nco is used to control non-stop operation. ORS built with NSO support runs in NSO only if one of the following arguments is specified in the command line:

`-nco xcount/xthreshold`	Where xcount (exception counts) is the number of faults allowed during a specified interval before the application exits and xthreshold (exception threshold) is the time interval in seconds. The values must be separated by a slash.
-nco	Starts NCO with default parameters (six faults in 10 seconds)

Examples

```
orchestration -host ra -port 2000 -app orchestration -nco
```

```
orchestration -host ra -port 2000 -app orchestration -nco 100/1
```

See the Framework documentation on T-Servers for more information about faults.

Version Identification

To print the ORS version number to the log, use `-v, -version, or -V` in the command line. This option does not actually start ORS. It just prints the version number to the log and then exits.

Uninstalling ORS

This section describes how to remove the Orchestration Server component with Genesys Administrator.

Removing the ORS with Genesys Administrator

If you are using Genesys Administrator in your environment, you can uninstall the Orchestration Server component directly from the Genesys Administrator interface.

1. Log in to Genesys Administrator.
2. Locate the Orchestration Server component that you wish to remove.
3. Click **Uninstall**.

Removing ORS Manually

This section describes how to manually remove the Orchestration Server Component.

Manually Removing ORS on Windows

Do the following on each machine that hosts Orchestration Server:

1. From the Windows Start menu, open the Control Panel and click **Add** or **Remove Programs**.
2. At the **Add or Remove Programs** dialog box, select Genesys Orchestration Server <version number> [ORS].
3. Click **Remove**, and follow the instructions to remove Orchestration Server.
4. Using Windows Explorer, browse to the `GCTI` main directory and delete the complete Orchestration Server subdirectory, including all subfolders, if any folders or files remain.

Manually Removing ORS on UNIX-Based Platforms

For the directory in which each Orchestration Server component is installed, run the following command: rm -R If an instance of the component is running, the command will fail.