



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Orchestration Server Deployment Guide

Orchestration Server 8.1.4

11/5/2024

Table of Contents

Orchestration Server 8.1.4 Deployment Guide	3
About Orchestration Server	5
New in This Release	11
Architecture	16
Deployment Models	30
SCXML and ORS Extensions	36
General Deployment	38
ORS Features	50
Persistence	51
High Availability	57
Graceful Shutdown	65
Clustering	66
Load Balancing	70
Multiple Data Centers	75
Direct Statistic Subscription	77
Performance Monitoring	82
Hiding Sensitive Data	93
Debug Logging Segmentation	99
Elasticsearch Connector	100
Direct Statistic Definition	124
Configuration Options	128
Application-Level Options	129
Common Log Options	195
Switch gts Section	198
DN-Level Options	200
Enhanced Routing and Interaction Submitter Script Options	203
Interacting with eServices	211
ORS and SIP Server	217
SCXML Application Development	222
Install Start Stop	228
Installation	229
Starting and Stopping	237
Uninstalling ORS	241
Graceful Shutdown	65
Sizing Guidance	243

Orchestration Server 8.1.4 Deployment Guide

Welcome to the Orchestration Server 8.1.4 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:

[ORS Overview](#)

[New in This Release](#)

[Architecture](#)

[Deployment Models](#)

[SCXML and ORS Extensions](#)

General Configuration & Deployment

This section includes information on:

[Prerequisites](#)

[Deployment Tasks](#)

[Cluster Configuration](#)

[Loading an Application](#)

[Interacting with eServices](#)

ORS Features

This section includes information on:

[Persistence](#)

[High Availability](#)

[Clustering](#)

[Load Balancing](#)

[Multiple Data Centers](#)

Options

This section includes information on:

[Application-Level Options](#)

[Common Log Options](#)

[Switch gts](#)

[DN-Level Options](#)

[Enhanced Routing Script Options](#)

ORS and SIP Server

This section includes information on:

Deployment and Debugging

This section includes information on:

- [PrivateServiceRequest Action](#)
- [Attribute Details](#)
- [Target T-Server](#)
- [Child Objects](#)

- [Deploying Voice Applications](#)
- [Manually Deploying a Voice Application](#)
- [Debugging SCXML Applications](#)
- [Samples](#)

Installing

This section includes information on:

- [Install Package Location](#)
- [Installing on Windows](#)
- [Installing on UNIX](#)
- [Starting and Stopping](#)

Starting and Stopping

This section includes information on:

- [Starting](#)
- [Stopping](#)
- [Non-Stop Operation](#)
- [Version Identification](#)

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](#)).

ORS Capabilities

Below is an example of a customer interaction that involves multiple sessions spread out over time.

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 email congratulating him on the approval of his application.

This example, involving voice, IVR, fax, SMS, and email channels, shows how ORS is able to treat all the various transactions as a single service.

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, email servers, web servers, and workflow servers.

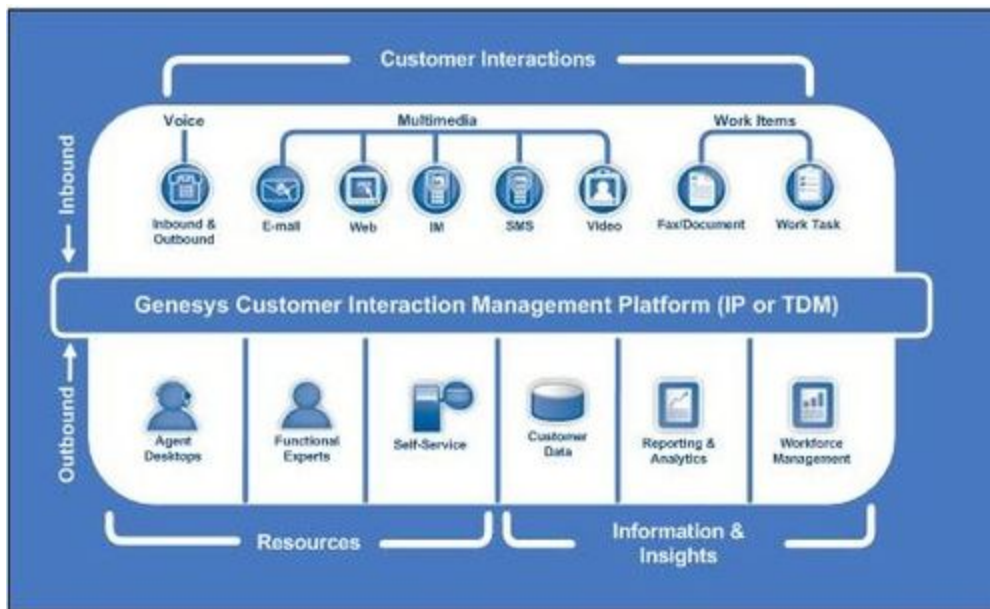
ORS can handle pure-voice, non-voice, and multimedia environments, enabling routing of each media type based on appropriate criteria. Routing strategies and business processes automate interaction routing to the most appropriate agent/resource based on factors such as the type of inquiry, the business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



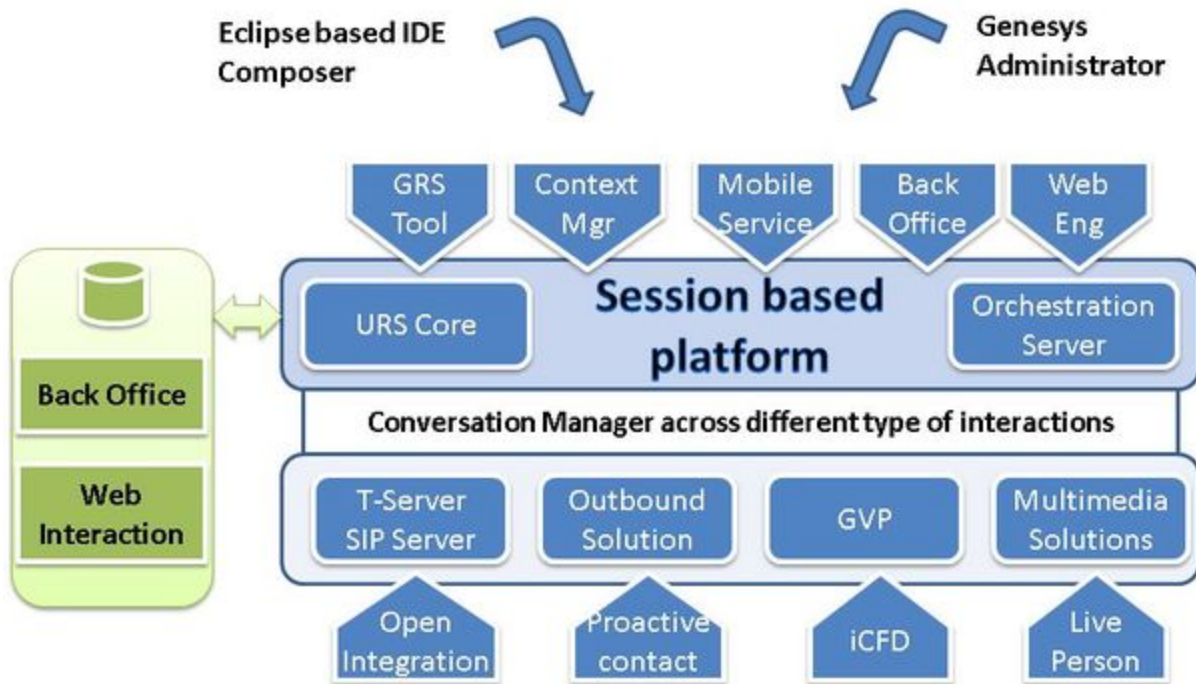
ORS and Multimedia

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 8.1.4 Deployment Guide*, [Application Server Requirements](#).

MIME Types

MIME (Multipurpose Internet Mail Extensions) refers to a common method for transmitting non-text files via Internet email. 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	Configuration 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://.

Orchestration Video

This video introduces you to Orchestration.

What is Orchestration?

[Link to video](#)

New in This Release

This section summarizes the new features in the Genesys Orchestration Server (ORS) 8.1.4 release. The following new features were introduced after the initial Orchestration Server 8.1.4 release. Also see the [Document Change History](#) in the *Orchestration Server Developer's Guide*.

Tip

Detailed information on the features below can be found in [Configuration Options](#) and [ORS Features](#).

- 8.1.400.62—ORS provides the capability of masking sensitive data in T-Library messages of ORS log output in the following attributes: `AttributeCollectedDigits`, `AttributeLastDigits`, `AttributeDTMFDigits`, and `AttributeTreatmentParms`.
- 8.1.400.58—ORS enhances its Elasticsearch real-time reporting capabilities with the addition of explicit mapping of session and performance indexes; new Orchestration SCXML extension, `elasticconnector`; and Kibana Visualization samples.
- 8.1.400.55—The `assembled-cache-reload-threshold` and `max-assembled-cache-age` options may be provisioned in the `Application` section of the Enhanced Routing Script objects, in which case they will override the corresponding options configured in the `scxml` section of the Orchestration Server Application. ORS introduces the `/ors/help` method to provide help for ORS REST APIs. ORS includes additional logging for the Enhanced and Legacy pulling methods of multimedia interactions. The ORS log now prints Warning messages regarding certain pulling conditions.
- 8.1.400.53—This release improves how ORS performs a re-association of an interaction with a remote target session started on different ORS node within a cluster. A new option, `complete-assoc-on-target-node`, supports this feature.
- 8.1.400.51—ORS now supports Cassandra 3.9.
- 8.1.400.49— ORS installation includes an Orchestration Plug-in for a real-time view of sessions currently being executed by ORS. Option `functions-by-urs` is enhanced. A new option, `http-curl-ip-family`, explicitly defines the IPv4 for connections used by the ORS SCXML engine to retrieve SCXML documents from a web server. ORS option `http-ssl-version` supports additional values `TLSv1_0`, `TLSv1_1`, and `TLSv1_2`.
- 8.1.400.48—The default values for some ORS Application level caching options in the `scxml` section have changed as noted in the release notes. ORS now supports Cassandra internal authentication. New persistence section options, `username` and `password`, support this feature. By default, ORS now attaches interaction properties to ESP requests. This release introduces a new `attach_ixn_data` attribute for the `<session:fetch>` Action of the `esp` method.
- 8.1.400.47—Support for Red Hat Enterprise Linux 7.
- 8.1.400.46—If a DN with an alias is created in the Configuration Layer after ORS is started, and no subsequent configuration updates are made to that DN, then an attempt to use that alias in the

`_genesys.statistic.sData()` function for a statistic that is requested directly from Stat Server no longer causes an error in Stat Server.

- 8.1.400.45—Function `_genesys.statistic.sData` allows submission of statistic parameters via a JavaScript object, passed to that function as "statistic" parameter. ORS provides the optional capability to restart processing of voice calls without the usage of session persistence. In support of this feature, ORS introduces two new configuration options: `restart-session-on-switchover` and `session-restart-timeout`.
- 8.1.400.44—A new cookie option defines the value of the Set-Cookie header in web responses. If empty, the ORS will use its own value in the format `ORSSESSIONID=sessionId.nodeId`.
- 8.1.400.43—The default value of the ORS option `scxml\max-state-entry-count` is changed from 100 to 500. ORS now stores two new Session Reporting fields into Elasticsearch: `ani`—The value of the ANI attribute. It applies to voice interactions only. `duration`—The duration of the session in seconds (the difference between end and begin timestamps). It is populated at session termination. The JavaScript execution time can now be restricted by the ORS option `scxml\max-script-duration`, if any ORS extension functions (such as `_genesys.statistic.sData`) has been called from the `<script>` action element.
- 8.1.400.42—ORS can now delegate session creation to the Reserved Node when the Assigned ORS Node that is responsible for call processing can no longer create a session under certain conditions as described in the release note.
- 8.1.400.40—ORS can now use Elasticsearch to store data, such as ORS session, performance, and node data. In support of the Elasticsearch feature, Debug Logging Segmentation adds a new Debug Log Segment header, `ElasticConnector`. The existing `log-trace-segments` option adds `ElasticConnector` as a new value. A new attribute, `_es_store`, for the `<state>` level SCXML Element, can allow ORS to save information about session states into Elasticsearch, which can be used for operational/performance monitoring and analytics. ORS now supports Cassandra 2.2.5.
- 8.1.400.39—New keepalive options, `http-keepalive-idle` and `http-keepalive-intvl`, are added. If TCP keepalive is enabled, these options can be used to configure it on TCP sockets used for document fetches. A new `interaction.property.changed` asynchronous event for multimedia interactions allows ORS to react on an `OnCallInfoChangedEx` notification. You can now define which ORS cluster will process multimedia interactions in a deployment with Legacy Pulling of multimedia interactions.
- 8.1.400.36—You can now define the ORS cluster that will process interactions. This functionality is introduced for deployments with Voice (DN level option `application@<Orchestration cluster name>` and Enhanced Pulling of multimedia interactions (`cluster-id` in Enhanced Routing and Interaction Script options). A new optional Boolean attribute, `associated`, for the `<ixn:createmessage>` action is introduced. When set to `false`, a new interaction is not associated with the session that created it. ORS now supports the `extensions` property of the `hints` attribute in the `<ixn:redirect>` action when applied to a multimedia interaction.
- 8.1.400.35—A new option, `max-script-duration` allows you to set the maximum time in milliseconds that JavaScript within a `<Script>` element is allowed to execute.
- 8.1.400.34—ORS now successfully attaches a parent interaction to a session started by a consultation call in a Network Consult scenario when the call parties are not monitored by ORS.
- 8.1.400.33—For compatibility with ORS 8.1.3, this release introduces the `same-node-for-cons-prim-calls` option. Its purpose is to control call distribution compatibility between ORS nodes when a deployment contains nodes from both 8.1.3 and 8.1.4 releases.

-
- 8.1.400.31—The [Debug Logging Segmentation](#) feature, introduced in Release 8.1.300.52, is enhanced. A new Debug Log Segment header `ScxmlMetricEvalExpr` is added for `METRIC:eval_expr`. The existing `log-trace-segments` option adds `ScxmlMetricEvalExpr` as a new value. A new option, `filter-eval-expr`, which works with `METRIC:eval_expr`, enhances the ORS capability of hiding of sensitive data in logs, which was introduced in Release 8.1.400.30.
 - 8.1.400.30—ORS can now selectively [hide sensitive data](#) in logs when the printing of such sensitive data is not explicitly requested by an SCXML strategy. The Performance Monitoring feature, previously introduced in Release 8.1.400.21, is enhanced. You can now [display ORS Performance Data in a RTME client](#) such as Pulse.
 - 8.1.400.27—ORS supports [pulling interactions from multiple Interaction Servers](#). The Self-Monitoring feature now allows you to define separate log messages for each type of performance counter. You can now create the Interaction Server Application(s) using only the Interaction Server Application template.
 - 8.1.400.26—ORS supports URIs of lengths up to 4,094 bytes.
 - 8.1.400.24—A new function, `_genesys.session.getServerVersion`, allows you to retrieve the Orchestration Server version.
 - 8.1.400.21—You can monitor ORS performance using a set of [performance counters](#) and the ability to configure conditions that will both trigger and clear Management Layer alarms. A new `detach` attribute, for the `<ixn:redirect>` Action Element, described in the [Orchestration Developer's Guide](#), controls whether ORS should detach an interaction from the current session before routing to the specified target.
 - 8.1.400.20—A new option [http-verbose](#) can be used to control the volume of HTTP Request logging.
 - 8.1.400.17—ORS can [request data directly from Stat Server](#) instead of going through Universal Routing Server. A new `resultof` property is added to the `interaction.notcontrolled` event to provide a reason for a session not being able to control an interaction that it owns.
 - 8.1.400.15—A new option `getlistitem-binary-conversion` defines how binary-formatted Transaction List item attributes are treated by the function `_genesys.session.getListItemValue(list, item)` when an item is returned as an OBJECT of key/value pairs.
 - 8.1.400.17—ORS adds support for Red Hat Enterprise Linux 6 and Windows Server 2012 native 64-bit.

When ORS requests a statistic directly from Stat Server, object names for Routing Point and Queue objects may be specified as DN aliases.

ORS can now automatically perform re-registration of unregistered DNs or in scenarios where ORS receives `EventError` for `RequestRegisterAddress`.

An architectural change streamlines the way that ORS obtains statistical data when executing routing strategies.

Previously ORS allowed the function `_genesys.session.setOptions` to override only eight options. This restriction is now removed.

To provide a reason for a session not being able to control a session that it owns, a `resultof` property is added to the `interaction.notcontrolled` event.

- 8.1.400.11:

Option `mcr-pull-cycle-quota` defines how many multimedia interactions ORS will request to be pulled from queues in a given pulling cycle during enhanced pulling.

ORS adds support for Windows Server 2012 native 64-bit.

ORS adds the capability to specify a `reasons` property in the `hints` attribute for the `<ixn:redirect>` Action Element.

ORS supports Debug Logging Segmentation, which provides more precise control of the logging when the `log-trace-segments` option is configured.

Performance improvements in the SCXML documents processing minimize session-creation time.

Options `max-assembled-cache-age` and `assembled-cache-reload-threshold` allow users to fine-tune the behavior of the Assembled Document Cache.

ORS adds support for voice call and Open Media interaction User Data properties represented by a key-value pair of Unicode or Binary type.

8.1.4 Initial Release

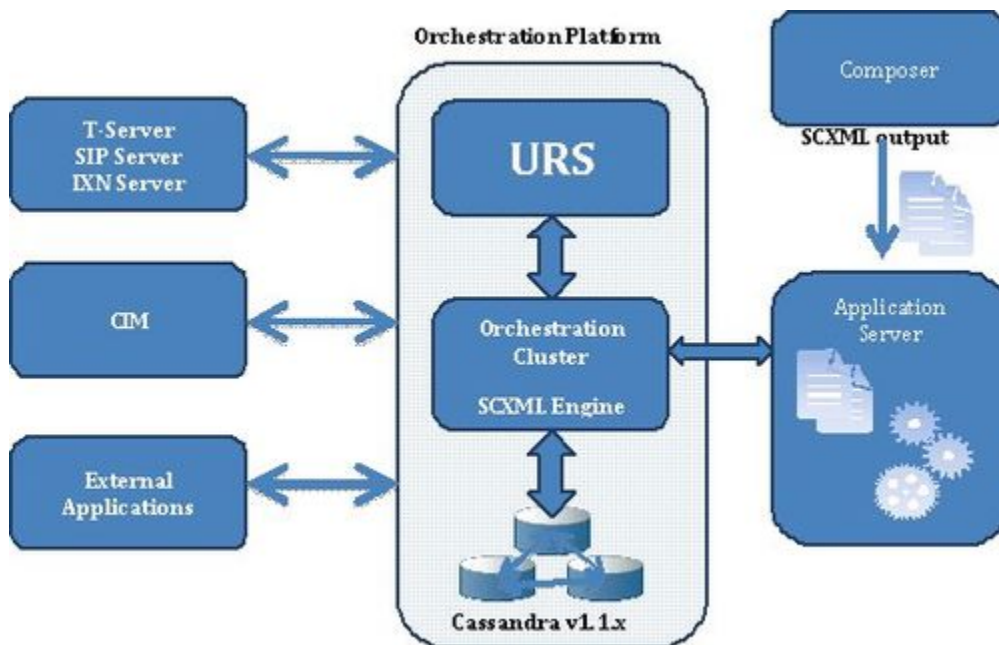
The following new features were introduced in the initial 8.1.4 release of Orchestration Server in November 2014.

- An ORS/Interaction Server `protocol extension` enhances the mechanism of getting the appropriate interactions for processing from Interaction Server. With enhanced pulling, the pull mechanism uses the strategy name and allows different priorities to be set for different types of interactions. This enhanced pull mechanism can result in improvements in performance and reliability.
- ORS supports `Submitter objects` configured in Composer. Submitters supply parameters that control how Interaction Server submits multimedia interactions to ORS.
- An ORS protocol extension supports the `attachment of Virtual Queue data` to multimedia interactions. The attached user data can then be used by a Reporting Solution, such as Genesys Info Mart. This feature requires Universal Routing Server 8.1.4.
- Support for activation/deactivation of any specific routing strategy.
- ORS can directly connect to Configuration Server and pull data. For example, when executing the `GetListItemValue` function, Orchestration Server can pull data without going through Universal Routing Server.
- Orchestration Servers configured as an High Availability pair can automatically `synchronize` voice treatment request data, thereby ensuring a smooth switchover.
- ORS can now terminate a session when the `ixn.redirect` action is successfully completed, and there is no other associated interaction attached to that session during the configured timeout. This functionality is enabled by the new Application-level option, `ixnfm-idle-session-ttl`, in the `orchestration` section.

-
- When upgrading/migrating the Orchestration Server Application to a newer release, you can perform a migration procedure without significant loss of data or impact to business operations. For more information, see [HA ORS 8.1.4 Migration Procedure when Persistence Enabled](#).

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 customer sessions by utilizing Cassandra NoSQL DB, which is packaged and built-in with ORS (see Cassandra in figure above). A *session* refers to a period of time during which ORS executes an SCXML-based document, such as one that defines a routing strategy created in **Composer**. Cassandra stores information regarding active sessions, as well as scheduled activities.

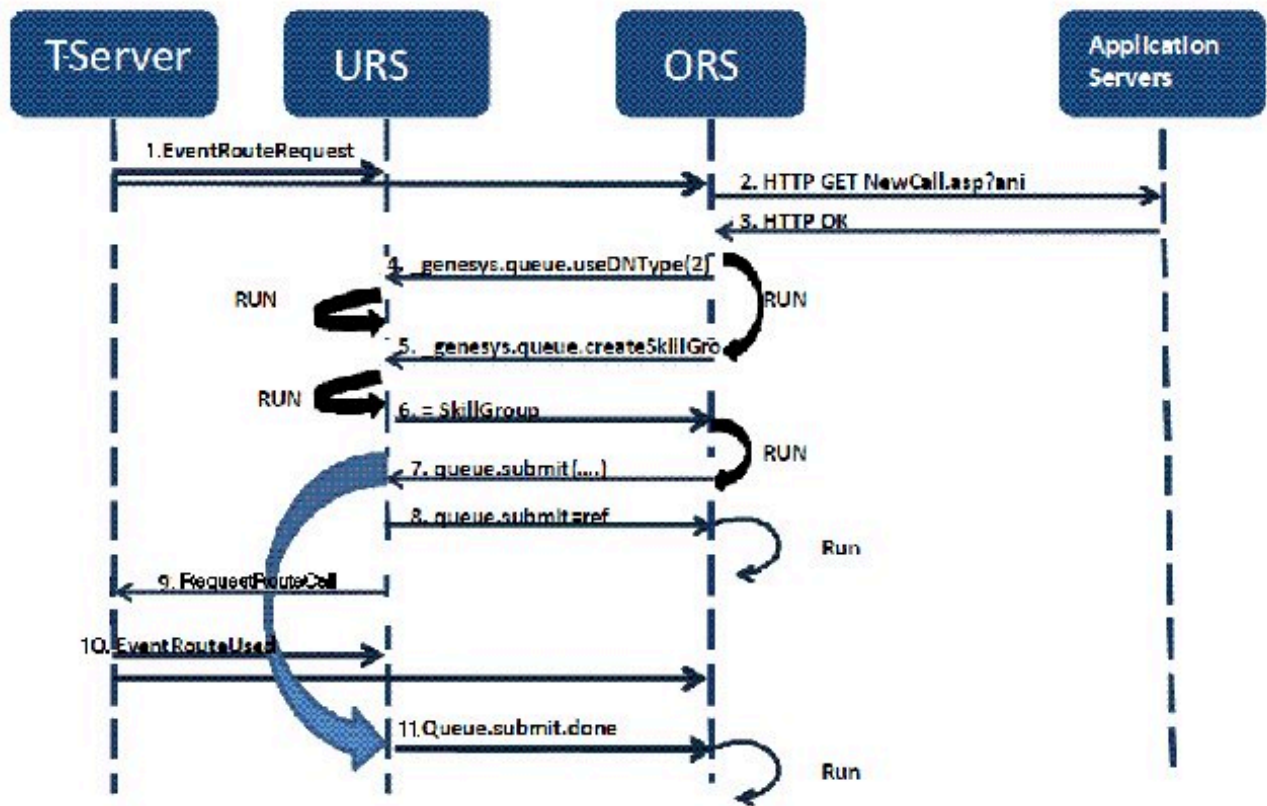
- For information on Cassandra, see the [Cassandra Installation/Configuration Guide](#).

Important

ORS supports only Apache's implementation of Cassandra. All references to Cassandra in ORS documentation refer to Apache Cassandra. The DataStax DSE edition of Cassandra is not supported as of now. ORS uses `gcassandraLib`, which is a C++ Thrift based library to interface with Cassandra.

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.

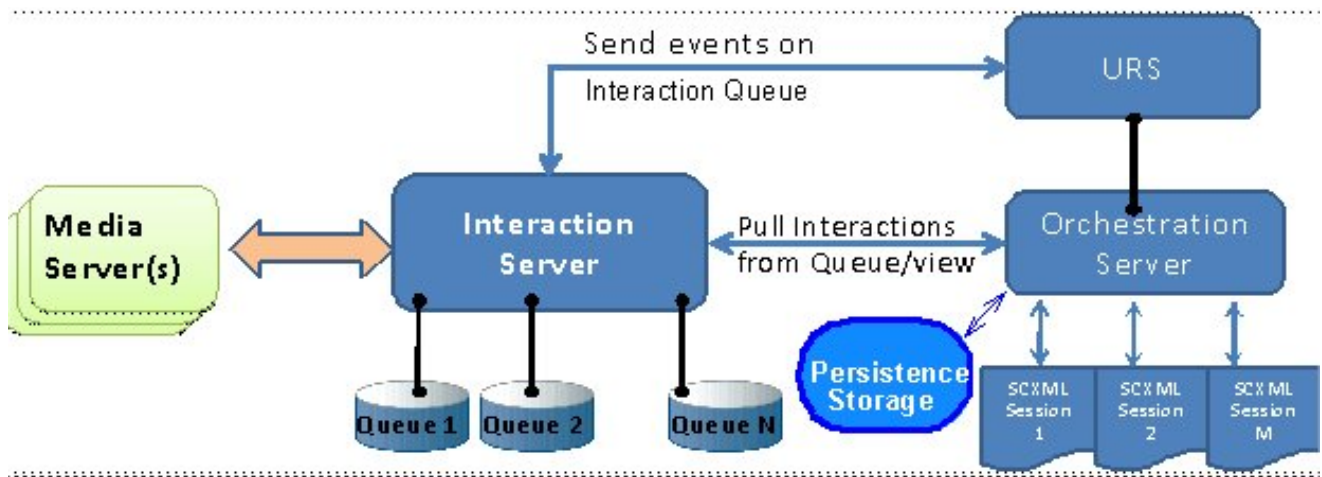
1. A new interaction arrives for T-Server (EventRouteRequest). T-Server notifies URS and ORS.
2. The configuration is enabled to use an ORS application, so ORS asks the Application Server for an SCXML application session.

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. Also, see [Interacting with eServices](#) and [Pulling from Multiple Interaction Servers](#).

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.

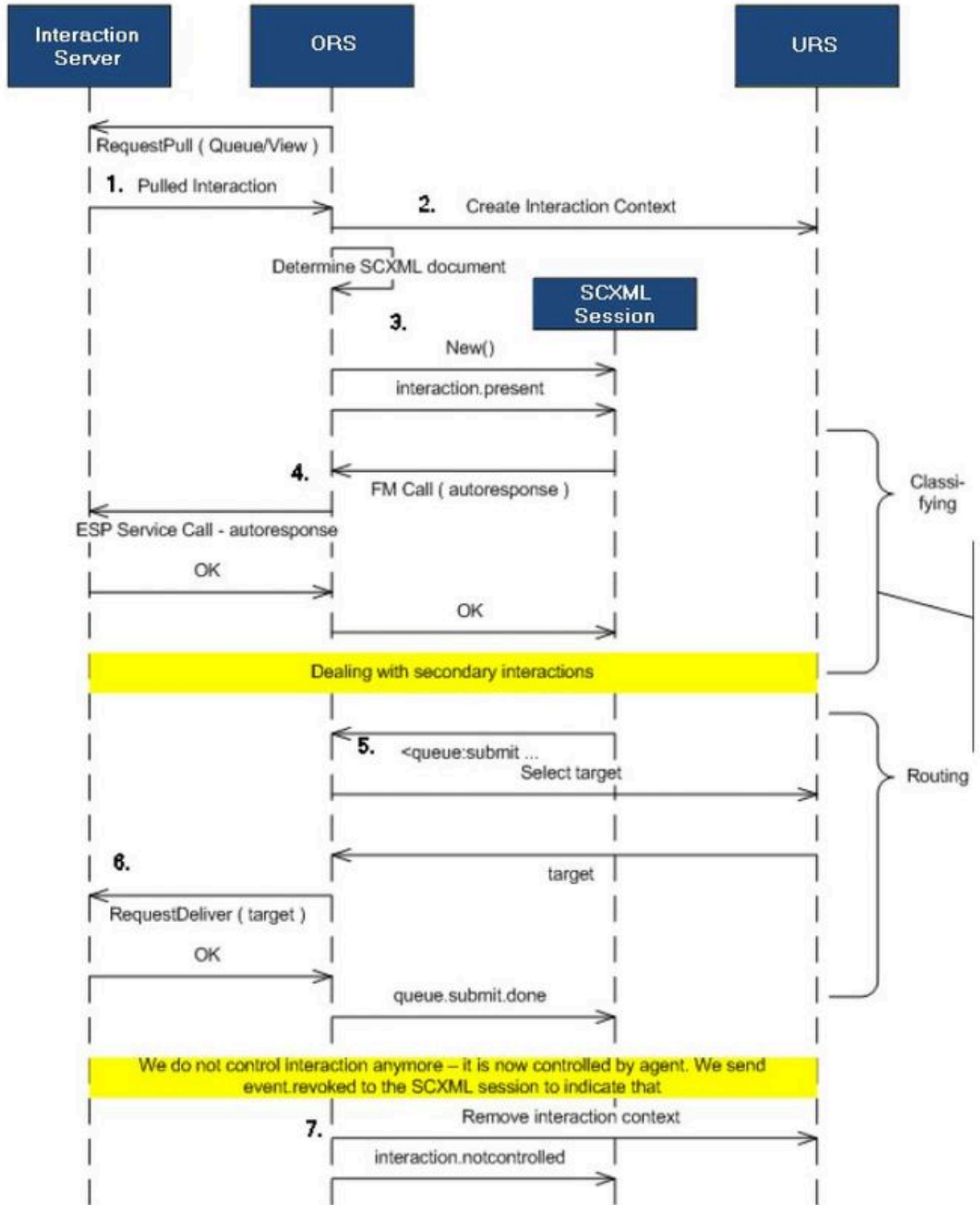
Note: All ORS requests with their attributes, including `RequestPull` can be seen in the Interaction Server log.

- 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.



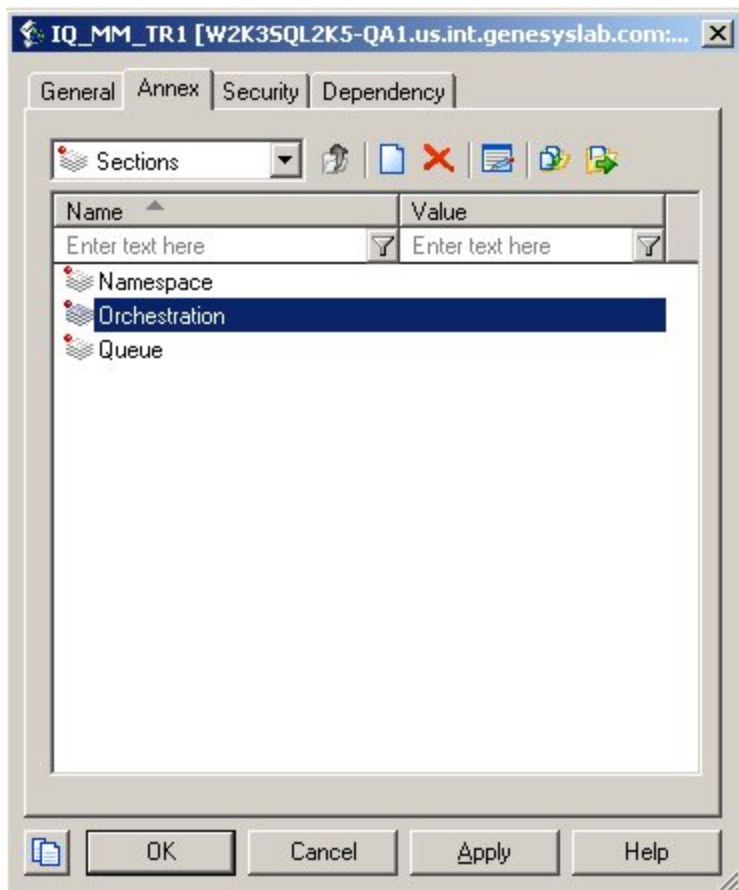
The numbered steps in the preceding figure are identified as follows:

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.

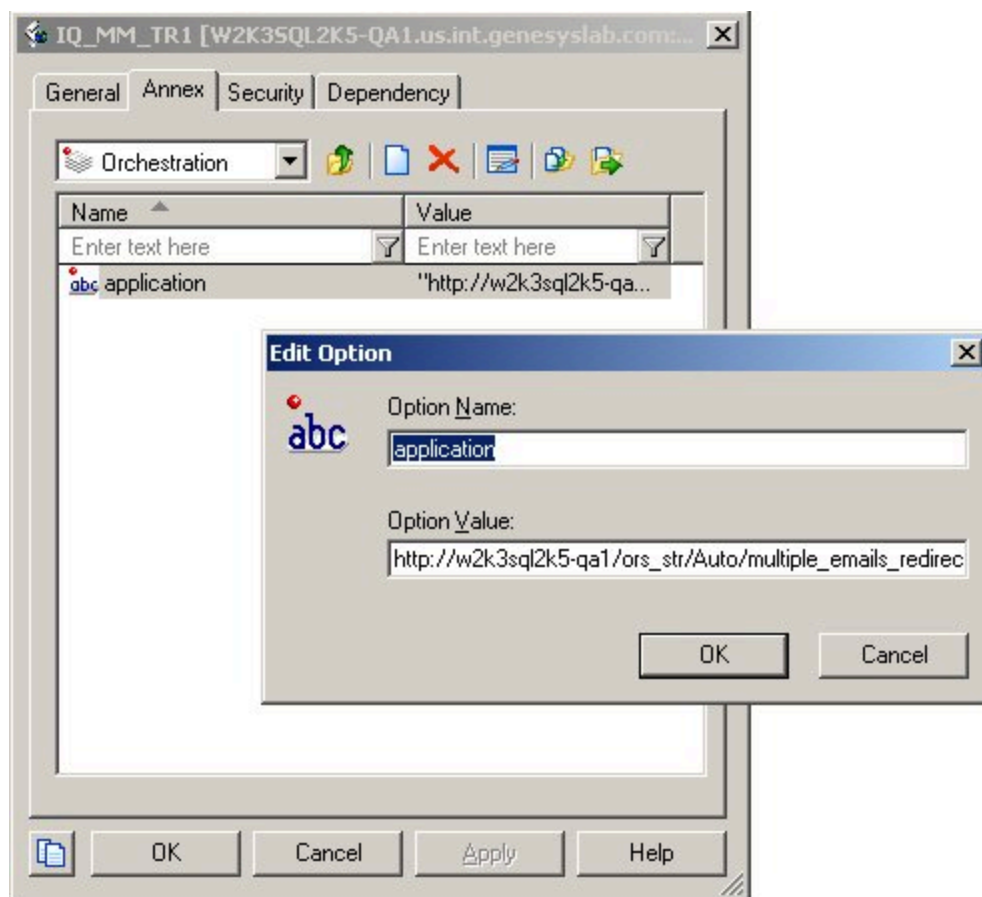
Processing eServices Interactions

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.

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

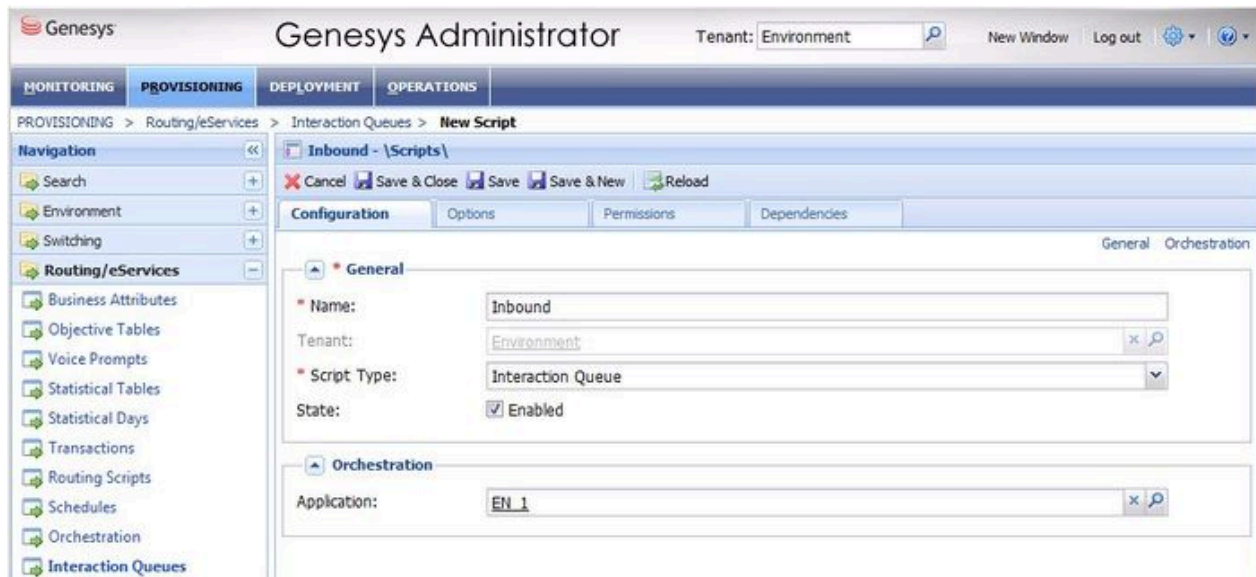


The Orchestration configuration section can contain an application option with a value that is the path to the strategy (SCXML script) that will be executed. The figure below shows the manually created Orchestration section and application option on the Annex tab of the Interaction Queue in Configuration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



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

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



4. For each Orchestration Server dedicated to processing multimedia interactions, you must create a Script object of type Simple Routing with the same name as the Orchestration Server Application object. When this ORS Application is used by multiple Tenants, create such Script objects for each Tenant configured to work with eServices. This is always mandatory when legacy pulling is used and not required with enhanced pulling.

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.

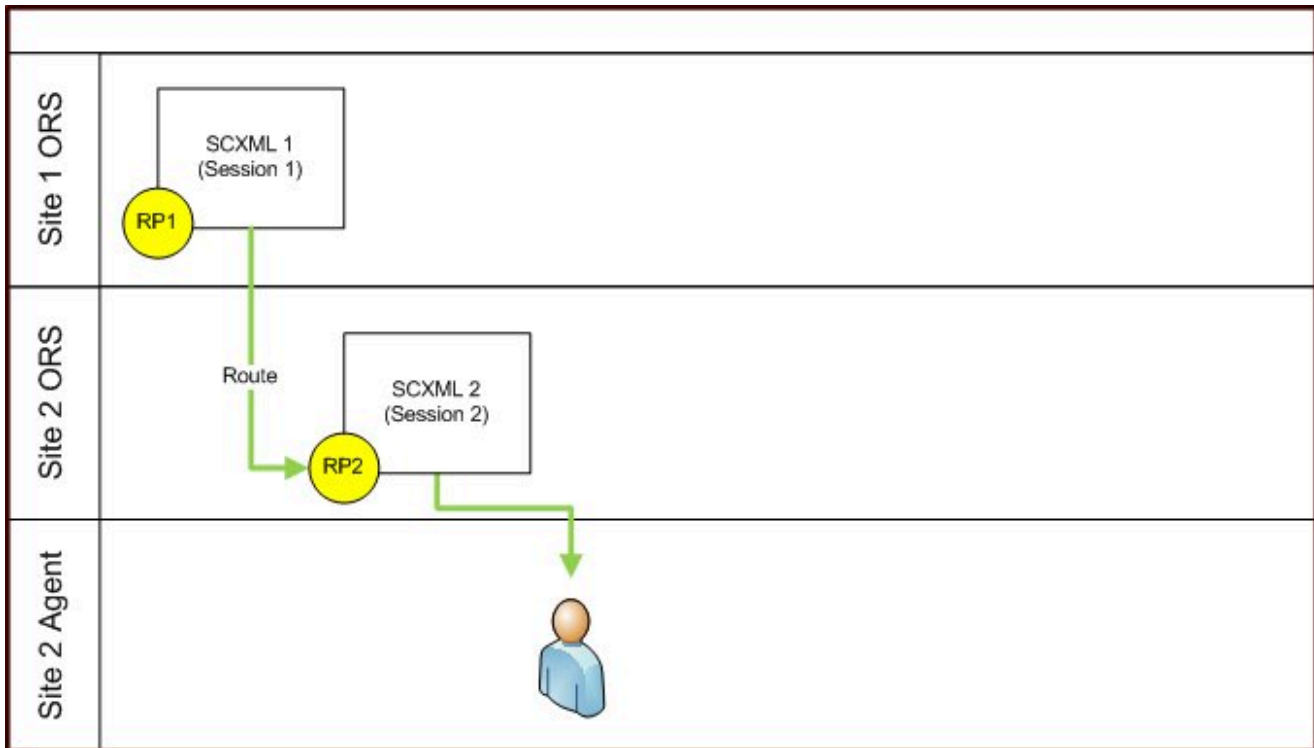
When an interaction is not currently associated with any session, the determination of which SCXML document to execute is based on the existing configuration, and allows for similar structures to be created, as presently done within URS/IRD strategies.

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

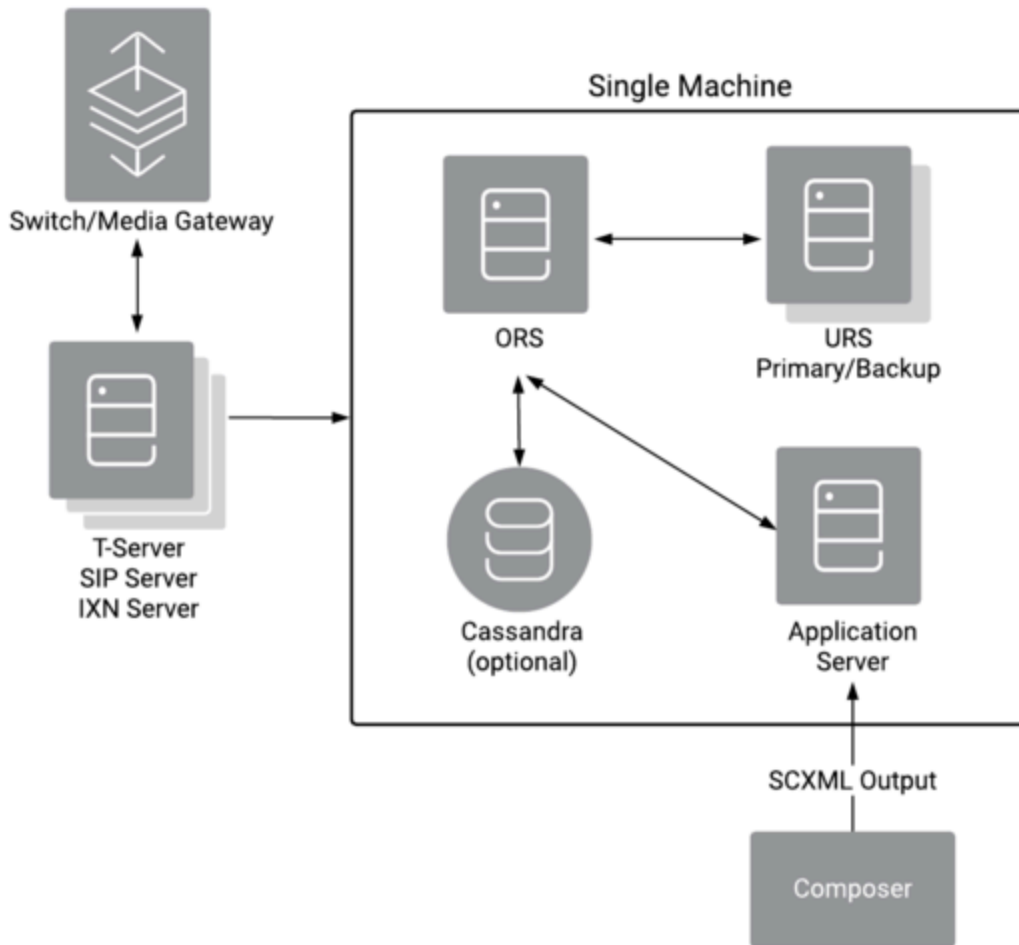
The following section provides examples of various ORS 8.1.3 and later deployment models and provides an overview of behavior, for single site, single site with redundancy, multi-site, and a single Cassandra cluster across sites:

Important

Starting from 8.1.401.10, ORS supports using Redis for session persistence and recovery. For more information, see [ORS with Redis Cluster](#).

Single-Site Deployment Model

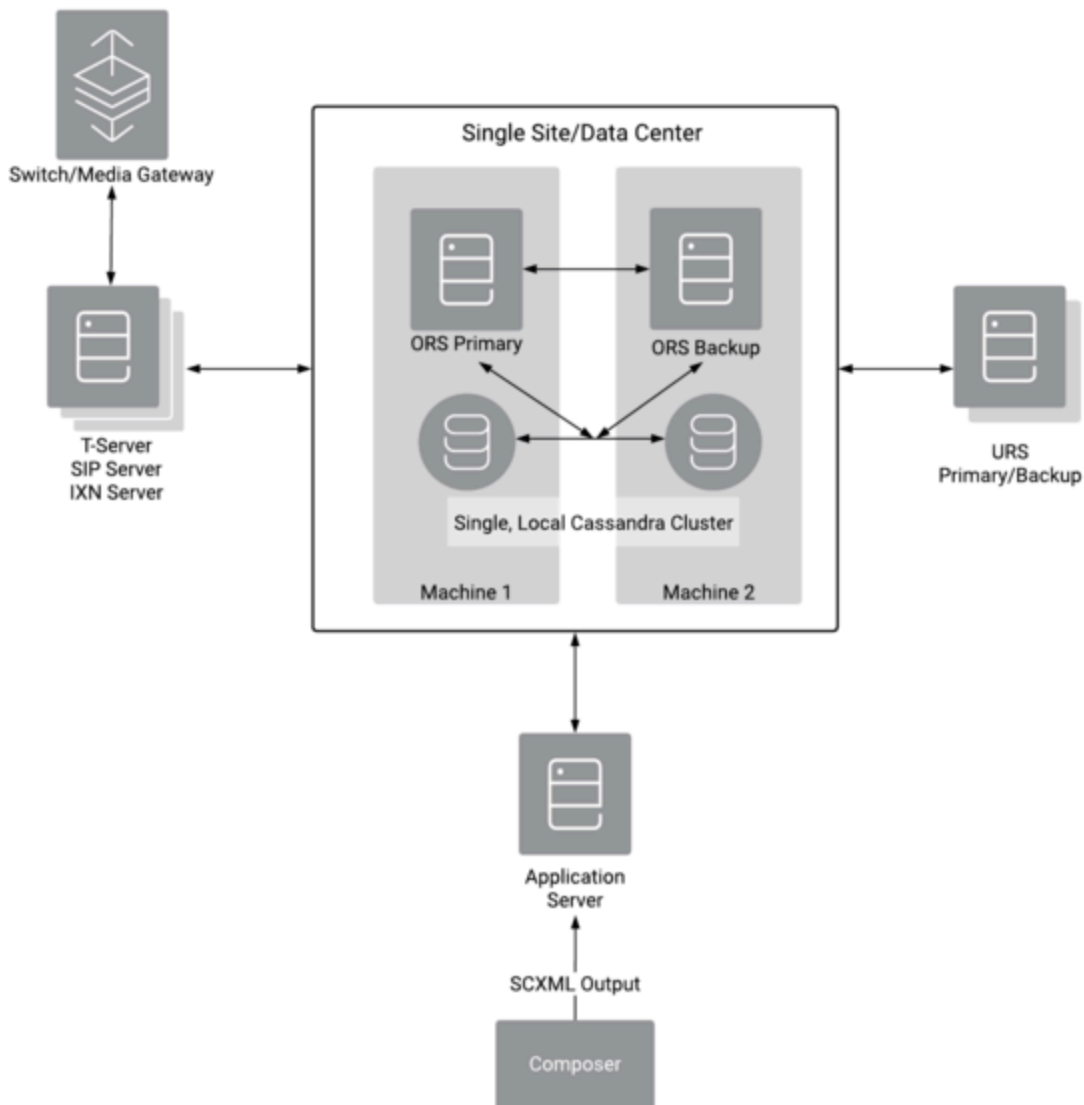
The single-site deployment example assumes that all components are planned to reside within a single box. This deployment does not provide high availability, redundancy, or failure handling. ORS can now be deployed with either Redis or Cassandra. The figure below shows a single site deployment example.



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. ORS can be deployed with either Redis or Cassandra.

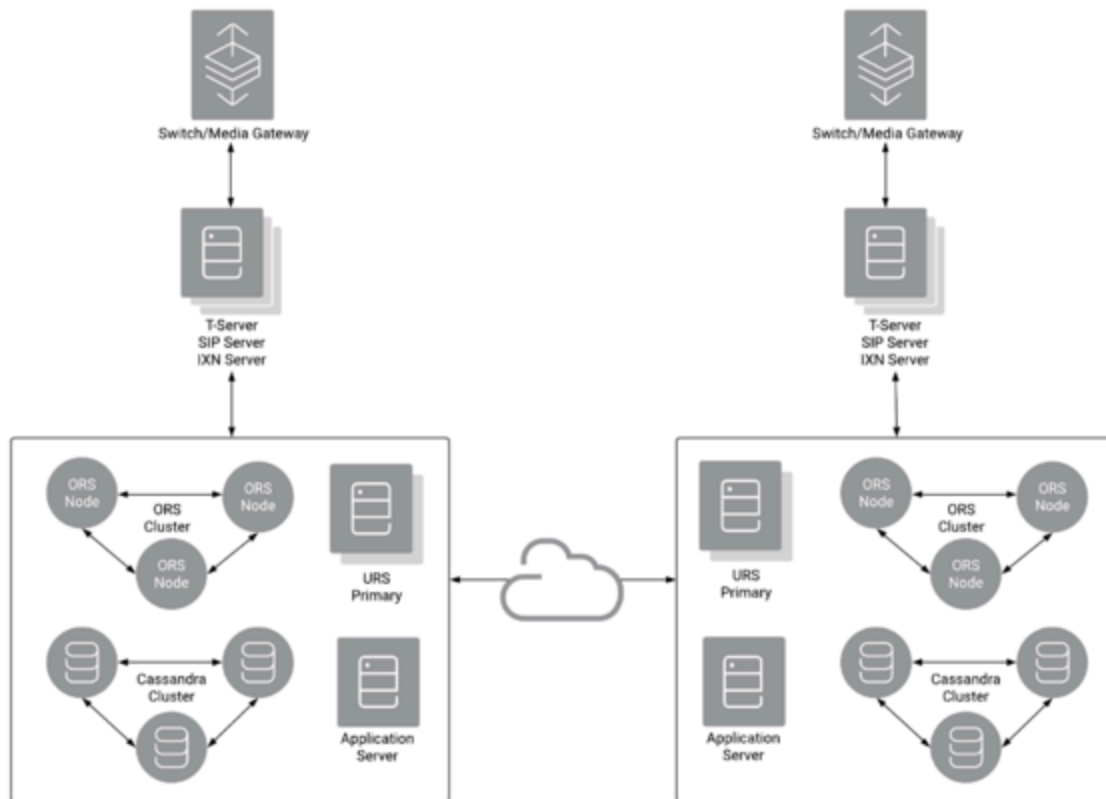
The figure below shows a single site with redundancy deployment example.



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 each ORS cluster can have its own private Cassandra instance or Redis instance, as ORS can be deployed with either Redis or Cassandra.

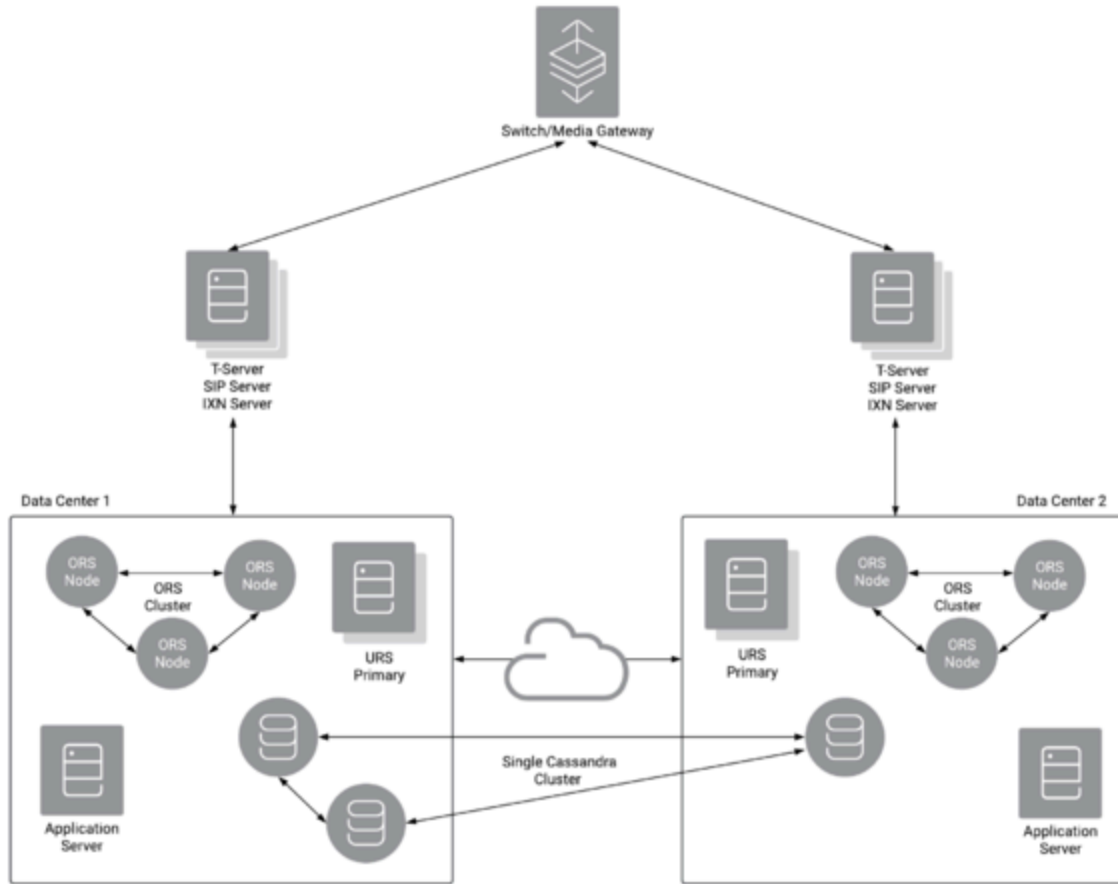
The figure below shows a multiple-site deployment example.



Single Cassandra Cluster across Multiple ORS Data Centers Deployment Model

ORS can be deployed with either Redis or Cassandra.

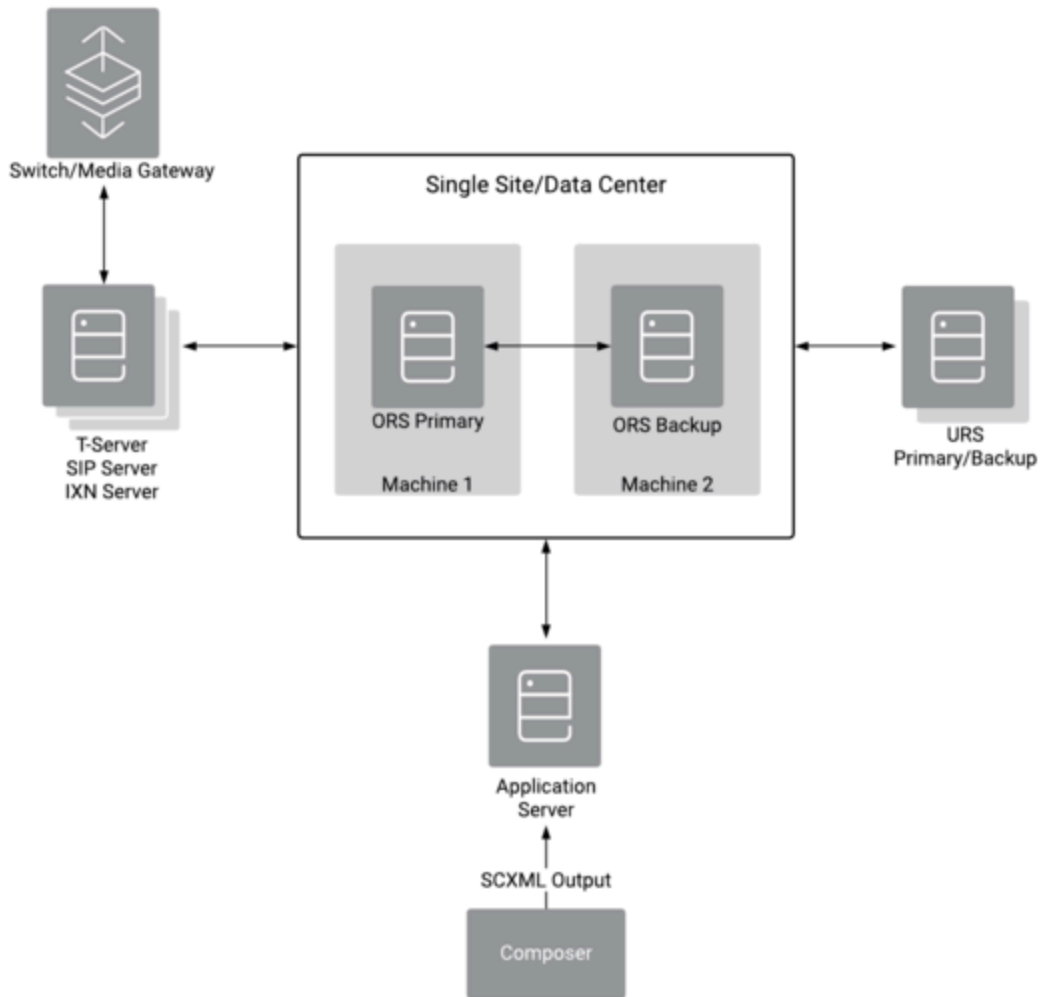
The figure below shows a Single Cassandra Cluster across Multiple-site Deployment example.



Single-Site with Redundancy and No Persistence Deployment Model

The following architecture with software redundancy across multiple boxes represents a deployment without Cassandra (or Redis) persistence/recovery.

The figure below shows a sample deployment with redundancy and without persistence to Cassandra (or Redis).



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

Important

- Genesys no longer recommends using Cassandra for session recovery. Refer to [Recovery of Voice Calls Without Persistence](#) for more information on voice call processing recovery upon an ORS primary/backup switchover without Cassandra persistence.

- Starting from 8.1.401.10, ORS supports using Redis for session persistence and recovery. For more information, see [ORS with Redis Cluster](#).

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/System.

Task Summary: Prerequisites for ORS Deployment

Objective	Related Procedures and Actions
Deploy Configuration Layer and ensure that Configuration Manager or Genesys Administrator 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 .	<ul style="list-style-type: none"> See the Cassandra Installation and Configuration Guide. See ORS with Redis Cluster.

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 in Configuration Manager. 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 wiki.

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

Important

Starting with release 8.1.400.64, you can configure an ORS cluster using a dedicated Transaction object of type List. In addition, options for all ORS applications in a cluster may be configured within that object. This eliminates the need to individually configure the options in every ORS application. For more information, see [Clustering, Enhanced Cluster Configuration](#) section.

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 Procedures or Actions
<p>In Configuration Manager, import the Application Template for ORS.</p> <p>The file name is OR_Server_814.apd.</p>	<p>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.</p>
<p>Create the ORS Application object. Configure Server Info and Tenants tabs.</p>	<p>Configure the Server Info tab:</p> <ul style="list-style-type: none"> • Host: Select name of the Host where the ORS Application will be running. • Port: Enter the available Listening Port of ORS. <p>Configure http port:</p> <ul style="list-style-type: none"> • Port ID: http • Communication port: Enter any available port. • Connection protocol: select http. <p>Configure the Tenants tab:</p> <p>Set up the list of Tenants that ORS works with.</p> <p>See the section on Creating the ORS Application object.</p>
<p>Create a connection for the ORS Application object to the following servers:</p> <ul style="list-style-type: none"> • T-Server(s) • Universal Routing Server • Interaction Server (if needed for multimedia) 	<p>Use the Connection tab of the Application object in Configuration Manager to set up connections to other servers. See Creating the ORS Application object.</p>

<p>Configure the ORS Application to work with persistence storage.</p>	<p>For Redis:</p> <p>See ORS with Redis Cluster.</p> <p>For Cassandra: Go to the Orchestration Server Application object > Options tab > persistence section:</p> <ul style="list-style-type: none"> • <code>cassandra-listenport</code>: Set to the value of Cassandra port. • <code>cassandra-nodes</code>: Enter semi-colon separated list of host names of Cassandra nodes in cluster. • <code>cassandra-keyspace-name</code>: Specify the name of Cassandra keyspace. • <code>cassandra-schema-version</code>: Enter Cassandra schema version. • <code>cassandra-strategy-class</code>: Set to <code>SimpleStrategy</code> if Cassandra is deployed as a single cluster, Set to <code>*NetworkTopologyStrategy</code> in the case of Data Centers Cassandra cluster deployment. • <code>cassandra-strategy-options</code>: Set the replication factor for a given keyspace. <p>Examples:</p> <p>If <code>SimpleStrategy</code>: <code>cassandra-strategy-options = replication_factor:2</code></p> <p>If <code>NetworkTopologyStrategy</code>: <code>cassandra-strategy-options = DC1:2;DC2:3 </code></p>
<p>Configure ORS cluster.</p>	<p>See Clustering.</p>
<p>Manually loading an SCXML application on objects other than ORS.</p>	<p>See the following:</p> <ul style="list-style-type: none"> • 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.
<p>Configure the ORS Application to work with multimedia interactions and/or multiple Interaction Servers (if needed).</p>	<p>If the ORS Application should work with multimedia interactions:</p> <ul style="list-style-type: none"> • Go to the Orchestration Server Application object, Options tab, <code>orchestration</code> section and set <code>mcr-pull-by-this-node</code> to true. • Set up a connection to the Interaction Server Application of Interaction Server type. <p>To support more than one Interaction Server for the same</p>

	<p>Switch:</p> <ul style="list-style-type: none"> • Set the switch-multi-links-enabled option to true in the ORS Application object, orchestration section. • In the ORS Application, set up a connection to each Interaction Server Application of Interaction Server type. <p>For more information, see Pulling from Multiple Interaction Servers.</p>
Set a Redundancy type (if needed).	<p>In the Server Info tab:</p> <ul style="list-style-type: none"> • 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 <p>See High Availability.</p>
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 (legacy method).

Starting with release 8.1.400.64, you can configure an ORS cluster using a dedicated Transaction object of type List. In addition, options for all ORS applications in a cluster may be configured within that object. This eliminates the need to individually configure the options in every ORS application. For more information, see [Clustering](#), Enhanced Cluster Configuration section.

Notes:

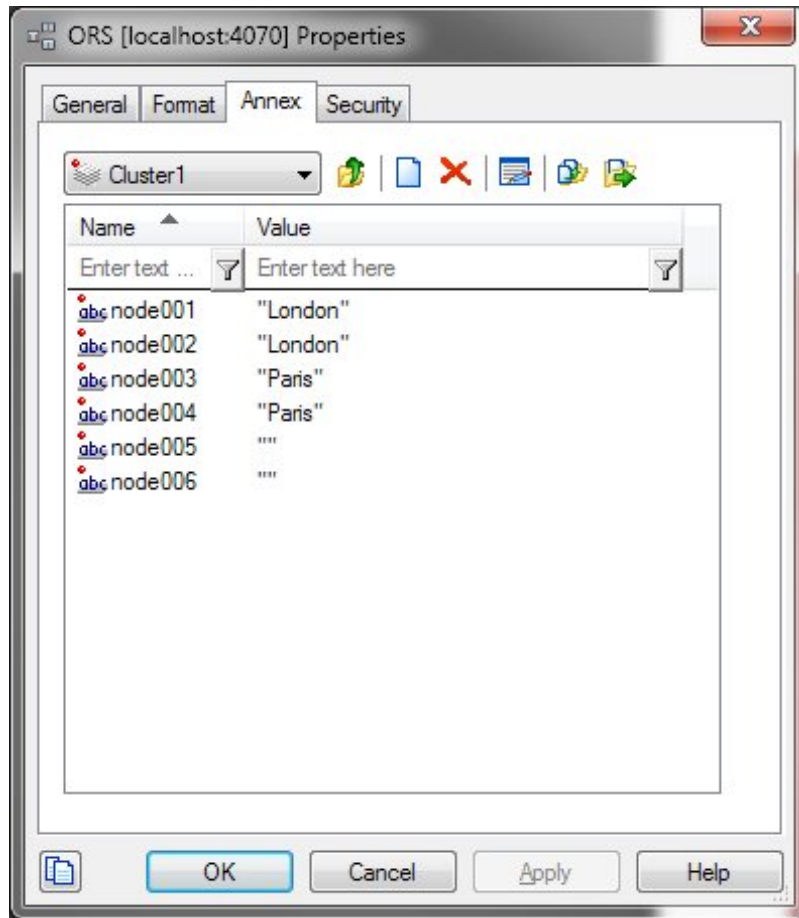
- All ORS nodes with the Data Center set to an empty string will belong to one "nameless" Data Center.
- ORS 8.1.3 and later requires creating an ORS Transaction List even if the deployment has only one ORS node.

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 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 **Up One Level**.
-

12. Repeat Steps 5 - 12 for all clusters.
13. Click **OK** to save and exit. An example is shown below.



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.

Note: In ORS 8.1.3 and later, work allocation happens automatically, based on the configuration of the cluster described above.

Important

ORS 8.1.3 and later requires creating an ORS Transaction List even if 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](#).

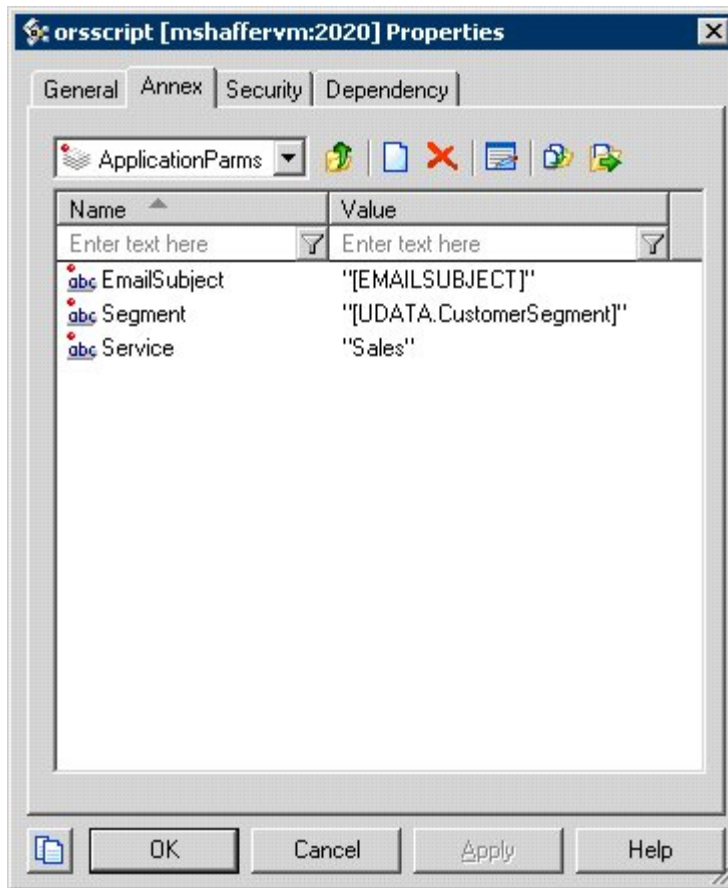
1. In Configuration Manager, select the appropriate Tenant and navigate to the Scripts folder.
2. Open the appropriate Script object of type Enhanced Routing Script (CfgEnhancedRouting).
3. Select the **Annex** tab.
4. Select or add the Application section.
5. Right-click inside the options window and 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.



8. Click **OK** to save.
9. Repeat from Step 5 to add another option in this section.

Orchestration Server REST API Security Considerations and Basic Hardening Steps

Orchestration Server implements RESTful API via HTTP protocol, described in detail [here](#). It is possible that through this RESTful API, sensitive data stored in the JavaScript context of an active routing strategy might be accessed, and ORS can be forced to perform resource-consuming activities (DoS attack).

Major security vulnerabilities of the RESTful API implementation are:

- No authentication of any kind.
- No ability to provision HTTP responses with security headers of any kind.
- No firewall features of any kind (port filtering, etc.).

Hardening Steps for ORS REST API

You can perform the following steps to harden the ORS REST API (that is, secure the ORS REST API by reducing its surface of vulnerability):

1. Provision only one listening port for the REST API (the ORS application in CME must have only one port with the protocol `http` (named `default`) in the **Server Info** tab).
2. Provision TLS/SSL transport-level security for communications via that HTTP port as described in the [Genesys Security Deployment Guide](#).
3. Configure the firewall to allow connections to ORS ports only from 100% trusted zones with no exceptions. This is very important because, access to the ORS HTTP port means access to all features of the ORS REST API.

ORS Features

This section lists the Orchestration Server features.

- Persistence
- High Availability
- Graceful Shutdown
- Clustering
- Load Balancing
- Multiple Data Centers
- Direct Statistic Subscription
- Performance Monitoring
- Hiding Sensitive Data
- Debug Logging Segmentation
- Elasticsearch Connector

Persistence

Important

- Genesys no longer recommends using Cassandra for session recovery. Refer to [Recovery of Voice Calls Without Persistence](#) for more information on voice call processing recovery upon an ORS primary/backup switchover without Cassandra persistence.
- Starting from 8.1.401.10, ORS supports using Redis for session persistence and recovery. For more information, see [ORS with Redis Cluster](#).

Persistence refers to the ORS capability of storing and retrieving (persisting) customer information from SCXML sessions that result from ORS executing SCXML documents, such as routing strategies created in [Composer](#). A single customer interaction may include [multiple sessions](#) that extend over time. The stored data 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.

Important

If the ORS option, `orchestration\sessionid-with-nodeid`, is not configured or is set to `false`, the lifespan of an SCXML session must not exceed 24 hours.

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.

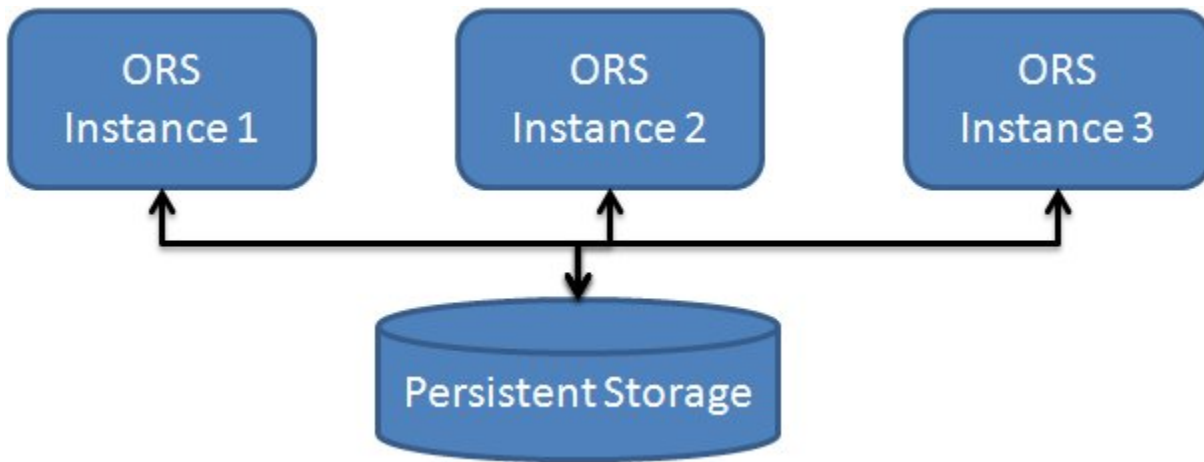
When the session reaches the state final, the entry for the session is removed from persistence.

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 SessionIDServerInfoIndex 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

Redis

See [ORS with Redis Cluster](#).

Cassandra

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.
username	Define the username that will be used to login to the Cassandra cluster.
password	Define the password associated with the username that will be used to login to the Cassandra cluster.

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 *Orchestration Server 8.1.3+ Developer's Guide*, SCXML Language Reference, [Extensions and Deviations](#).

Recovery of Voice Calls Without Persistence

Starting with 8.1.400.45, ORS can be configured for voice call processing recovery upon ORS primary/backup switchover without the Cassandra persistence/recovery feature. See [Recovery of Voice Calls Without Persistence](#).

Disabling Persistence

To completely disable session persistence, perform the following:

1. Configure the `scxml\persistence-default` option to **false** in all ORS application objects.
-

2. Ensure that the `<scxml>` element in all SCXML applications does not have the `_persist` attribute. Or, if the `_persist` attribute is defined, the value of the attribute must be set to **false**.

Note: The value of the `_persist` attribute is set to false using the Session Persistence property of the interaction process diagram in Composer. For more information, see *Session Persistence Property* section on the [Starting a New IPD](#) topic.

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.

Data Synchronization Between Primary and Backup ORS

ORS synchronizes voice treatment request data between primary and backup Orchestration Servers to ensure seamless switchovers. With this feature configured, ORS is able to send the appropriate event (`dialog.playsound.done`) to the session when treatment is ended even after ORS failover/switchover. After ORS failover/switchover, this session can be recovered and continued as normal.

How it Works

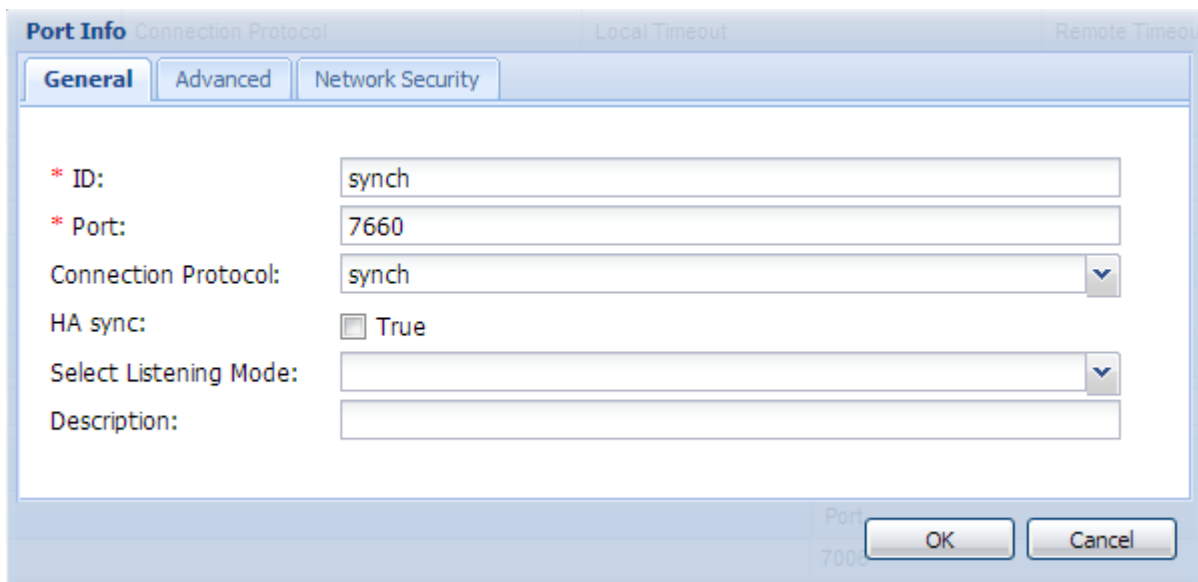
An SCXML session initiates the application of a treatment to a voice call. ORS sends T-Lib `RequestApplyTreatment` to T-Server/SIP Server. When the treatment is applied to a voice call, the information about the associated SCXML session and the treatment applied to a particular call is stored in the primary ORS. The primary ORS then sends this information to the backup ORS. When the treatment is ended, the primary ORS sends the `dialog.playsound.done` event to the appropriate session even after failover/switchover. The persisted session is recovered and the SCXML application continues working normally.

Note: If an HA pair of ORS Application is properly configured except for the absence of the synch port, a warning message is logged.

Configuring the Sync Port

1. In Genesys Administrator, configure a new synch port on the primary ORS Application object, as follows:
 - a. Open the Configuration Tab of the primary ORS Application object.
 - b. In the Server Info section, click Add in the Listening Ports field.
 - c. On the General tab of the Port Info dialog box, configure the synch port as follows:
 - i. ID - Enter synch
 - ii. Port - Enter the port number
 - iii. Connection Protocol - Enter synch
Refer to the example in the diagram below.
 - iv. Click OK

- d. Click **Save** and **Close** to save the new port in the primary ORS.
- 2. Configure a new synch port on the backup ORS Application object, as follows:
 - a. Open the Configuration Tab of the backup ORS Application object.
 - b. In the Server Info section, click **Add** in the Listening Ports field.
 - c. On the General tab of the Port Info dialog box, configure the synch port as follows:
 - i. ID - Enter synch
 - ii. Port - Enter the port number
 - iii. Connection Protocol - Enter synch
Refer to the example in the diagram below.
 - iv. Click **OK**.
 - d. Click **Save** and **Close** to save the new port in the backup ORS.

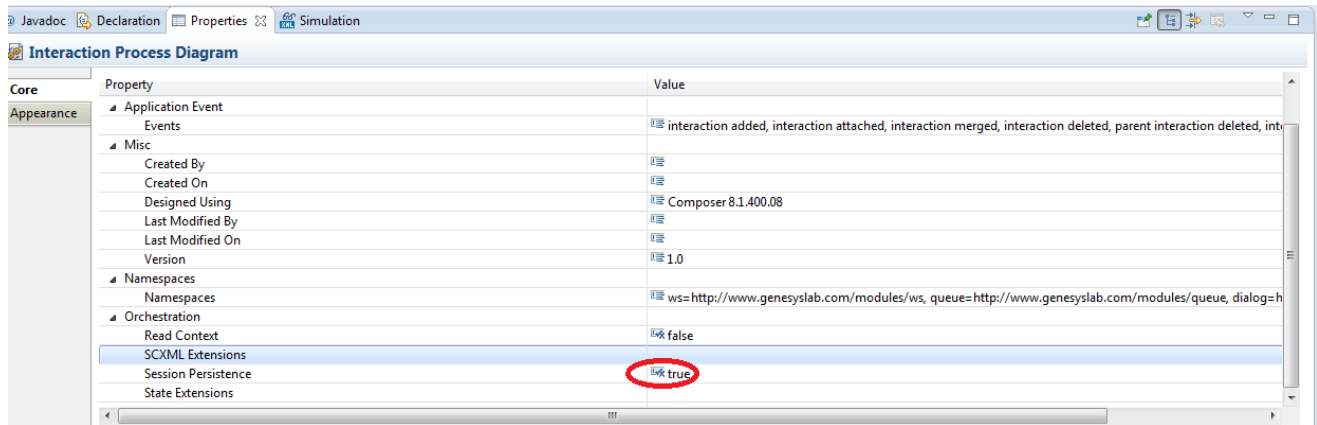


Configuring Persistence in ORS Application

Configure the ORS Application to work with **persistent storage**. Refer to "Configure the ORS Application to work with persistence storage" in the table of **Deployment Tasks**. Also see the **Recovery of Voice Calls Without Persistence** section below.

Configure Persistence in SCXML Session

Configure persistence for the SCXML session. Using **Composer** in your SCXML workflow, set the `_persist` attribute into true in the `<scxml>` element by setting the Session Persistence attribute in a Core Property of the **interaction process diagram** (see the example in the figure below).



Note: The synchronization mechanism is controlled by two Application-level options set in the orchestration section— **backup-synch-max-buffer** and **backup-synch-max-age**. Genesys recommends that you use the default values for these options.

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](#).

Recovery of Voice Calls Without Persistence

Starting with 8.1.400.45, ORS can be configured for voice call processing recovery upon an ORS primary/backup switchover without the Cassandra persistence/recovery feature. Requires Universal Routing Server 8.1.400.27+.

To disable Cassandra persistence:

- Set the `persistence/cassandra-nodes` option to `unknown` in all ORS Applications in the cluster (primary and backup) to disable connections to Cassandra.
- Set the `orchestration/sessionid-with-nodeid` option to `true` in all ORS Applications in the cluster (primary and backup) to enforce ORS to generate the session ID with the ORS node ID.

Configure Primary and Backup Instances

Configure data synchronization between primary and backup instances for all ORS nodes. This includes configuring the sync port and setting the options as described above.

[+] Responsibility of ORS Primary Instance

The responsibility of the ORS primary instance is to:

- Via primary/backup synchronization, notify the ORS backup instance each time a session is created to process some call, or if the session was terminated. The only data to be transmitted in that notification is `CallUUID` (Interaction ID).

Important

A double switchover scenario (that is, when the backup instance that takes over from a primary instance also fails, the primary instance taking over again) is not supported by design.

The restart functionality is based on the call watching mechanism in ORS. A backup ORS instance sees the same events as the primary instance. When the primary instance creates a session, the backup also creates (but not starts) its own session. As session creation requires some time, the backup instance creates the session in advance and is ready to start the session when it is required in the event of a failover.

Session restart on switchover for a backup instance which takes on the role of a primary involves a considerably big load on the system as in addition to handling new incoming calls, it should be able to restore sessions for old ones. A limited worker thread pool resource also imposes some limitations.

So, the session restart period is not the best time for doing additional processing, taking into account that the former primary may not be started or initialized yet.

But sessions for new calls that arrive after the switchover will be secured by the current backup when it is started and ready to work.

[+] Responsibility of ORS Backup Instance

The responsibility of the ORS backup instance is to:

- Monitor all calls that the ORS primary instance is processing.

- Process notifications from the primary ORS instance about session terminations. If a session processing a particular call is terminated, the ORS backup instance may drop all call-related data since processing of that call is finished and there is nothing to recover for that call upon a switchover.

The ORS backup instance creates sessions simultaneously with the ORS primary instance, but starts them upon switchover to the primary role. In addition, the ORS backup instance will also:

- Publish a “session.restart” SCXML event for each session.
- Throttle the recovery process to avoid overloading of the SCXML engine.

Configure ORS Application Options

Important

For voice call processing recovery, all options described below must have the same value in the primary and backup ORS Applications. Synchronization between primary and backup instances must be provisioned for all ORS nodes.

Existing Options

Set the existing ORS options as follows to enable call processing recovery for all ORS nodes in a cluster:

- Set the `new-session-on-reroute` option to `true` in all ORS Applications in the cluster (primary and backup).
- Set the `same-node-for-cons-prim-calls` option to `true` in all ORS Applications in the cluster (primary and backup).

New Options

Configure the following options introduced in release 8.1.400.45 as required:

`restart-session-on-switchover`

Option section: `orchestration` in an ORS Application;
Application in Enhanced Routing Script

Configuration Object: `Enhanced Routing Script, ORS Application`

Default value: `false`

Valid values: `true, false`

Value changes: Immediately, but new value will apply only to sessions created after that change.

Set to true to have the ORS backup instance attempt to restart the processing of voice calls that have been processed by the ORS primary instance and sessions that have not been terminated at the moment of switchover.

session-restart-timeout

Option section: orchestration in an ORS Application;
Application in Enhanced Routing Script

Configuration Object: Enhanced Routing Script, ORS Application

Default value: 50

Valid values: Any integer value starting from 10

Value changes: Immediately.

Specify the timeout (msec) between cycles of session starts upon call processing recovery. The restart quota per cycle is equal to the number of engine-working threads defined with the session-processing-threads option in the ORS scxml section (default = 8).

New session.restart Event

A new [Asynchronous Event](#), session.restart, supports this feature.

Attribute: Interactionid - string. The ID of the interaction whose processing has been restarted. This event is distributed when ORS restarts the processing of calls after an ORS switchover without using persistence. After that event, ORS will distribute all SCXML events from the beginning of call existence while the current ORS instance was running in backup mode.

Strategy Modifications

When recovery without persistence is configured, ORS restarts call processing from the very beginning of the SCXML strategy after an ORS switchover. Depending on call flow and strategy design, it is possible that a strategy may not execute properly upon a switchover. For example, a call may have certain user data already set; some other components such as Media Control Platform may not process the same call a second time. As a result, if using session recovery without persistence, Genesys recommends modifying your SCXML strategies to treat the session.restart event using an adjusted logic to handle such scenarios.

For more information on the session.restart asynchronous event, refer to the [Session Interface](#) section on the *Core Extensions* topic.

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.

Graceful Shutdown

Important

This feature is no longer available, starting with release 8.1.400.75.

Starting with release 8.1.400.64, ORS supports "graceful shutdown".

In general, the "graceful shutdown" feature of Management Framework is described in the [Management Layer User's Guide](#), Graceful Stop section. The information below describes graceful shutdown processing specific to ORS.

Processing Graceful Shutdown Signal from Local Control Agent

After a graceful shutdown signal has been received from the LCA, ORS does the following:

- Changes state to Suspending.
- Stops creating new sessions.
- Finishes processing of existing sessions.
- After all active sessions terminate, changes state to Stopped.
- If ORS receives a signal from the Local Control Agent (LCA) to switch into backup mode while it is in suspending mode, ORS processes it as it does in regular primary mode – terminates all current sessions, and so on.
- While in a suspending state, ORS might receive an "ungraceful" shutdown signal from the LCA. In this case ORS immediately stops.

Processing Incoming Interactions

To enable ORS to continue processing incoming voice interactions when a primary application of an ORS node processes a "graceful shutdown":

- The ORS cluster must consist of more than one node.
- A graceful stop must not be applied to more than one node at a time.
- When one of the ORS nodes from a cluster performs a graceful stop, it delegates creation of new sessions to the Reserved 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.

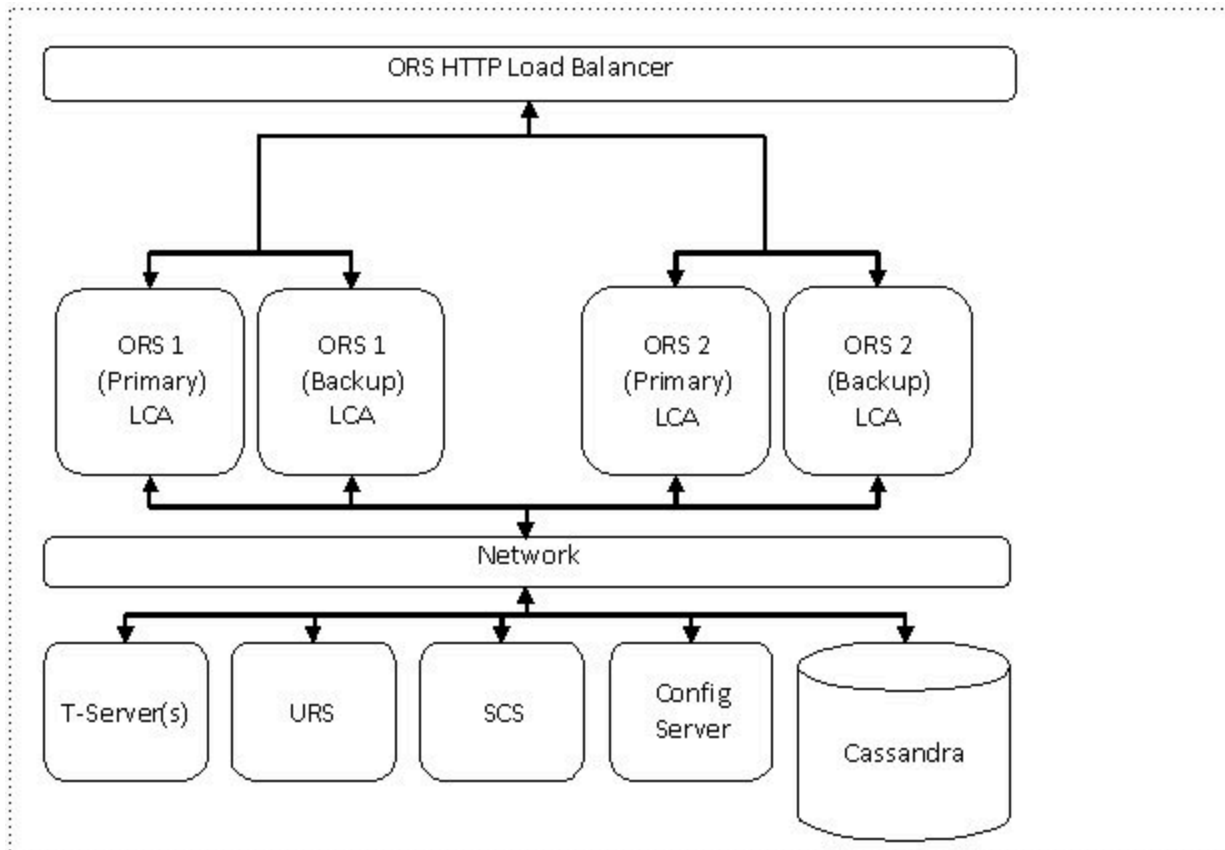
Clustering Requirements

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.

Enhanced Orchestration Cluster Configuration

Starting with release 8.1.400.64, you can configure an ORS **cluster** using a dedicated Transaction object of type List. In addition, options for all ORS applications in a cluster may be configured within that object. This eliminates the need to individually configure the options in every ORS application.

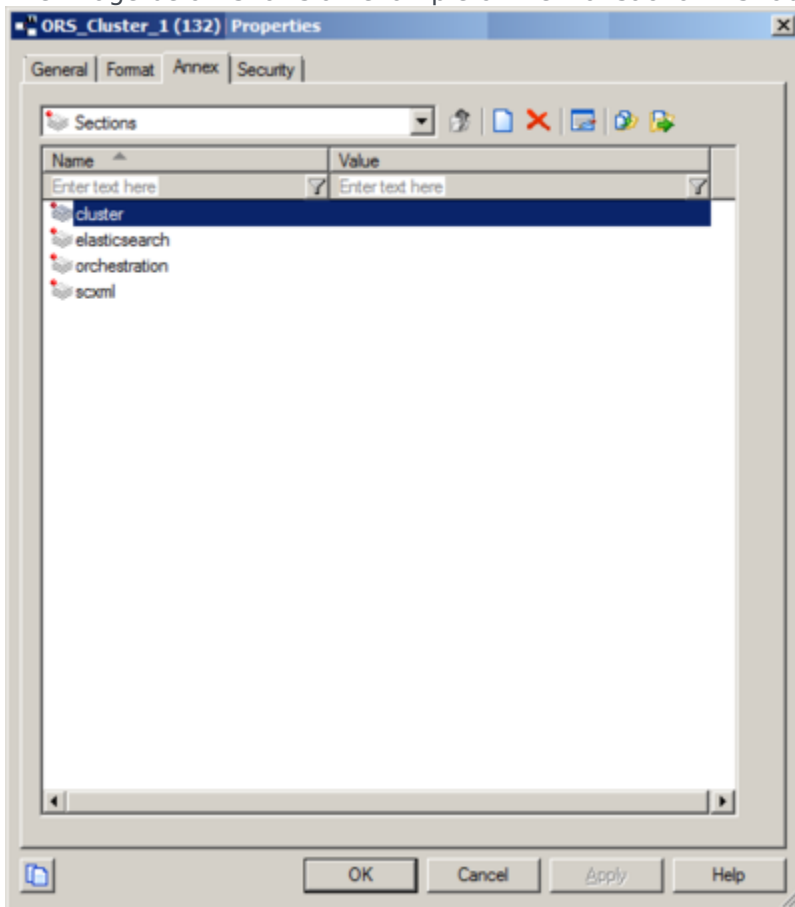
Configuring the Transaction List Object

1. Create a Transaction List object.

1. For a multi tenant deployment, create a Transaction List object under the Environment tenant. For a single tenant deployment, create a Transaction List object under the Resources tenant.
2. Assign an arbitrary cluster name, such as ORS_Cluster_1.
2. In the **Annex** tab, add the section cluster.
 1. List the cluster nodes/data centers in the same way as in the legacy implementation.
 2. For the *Name*, specify the name of the ORS Application.
 3. For the *Value*, specify the Data Center name.
3. Create the sections for option values. Use any one of the following sections: **orchestration**, **scxml**, **persistence**, **elasticsearch**, **performance-alarms**, and **mcr**.
4. Define the options and values for each section.

Sample List Object

The image below shows an example of the Transaction List object sections in Configuration Manager.



Retrieval of Cluster/ORS Application Option Configuration

- Submit the cluster name to ORS as an additional parameter in the command line, `-cluster <cluster name>`. Upon startup, ORS retrieves the cluster option' configuration from the Transaction List object with that name. Example: `-host <host> -port <port> -app <ORS application name> -cluster <cluster name>`.
- If the cluster name is defined in the command line, ORS retrieves the configuration options from the Transaction List object. If the same option is configured in both the ORS Application object and the cluster Transaction List object, the option from the cluster Transaction List object takes precedence.
- If the cluster name is not defined in the command line, ORS retrieves options from the ORS Application object(legacy method).

Limitations

- The **[log]** and other logging-related sections cannot be configured in the cluster transaction list object.
- The **scxml\system-id** option cannot be configured in the cluster transaction list object.

Transaction List Object Documentation

For information on configuring List objects in Genesys Administrator, see:

- Genesys Administrator Help, [Transactions](#).

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

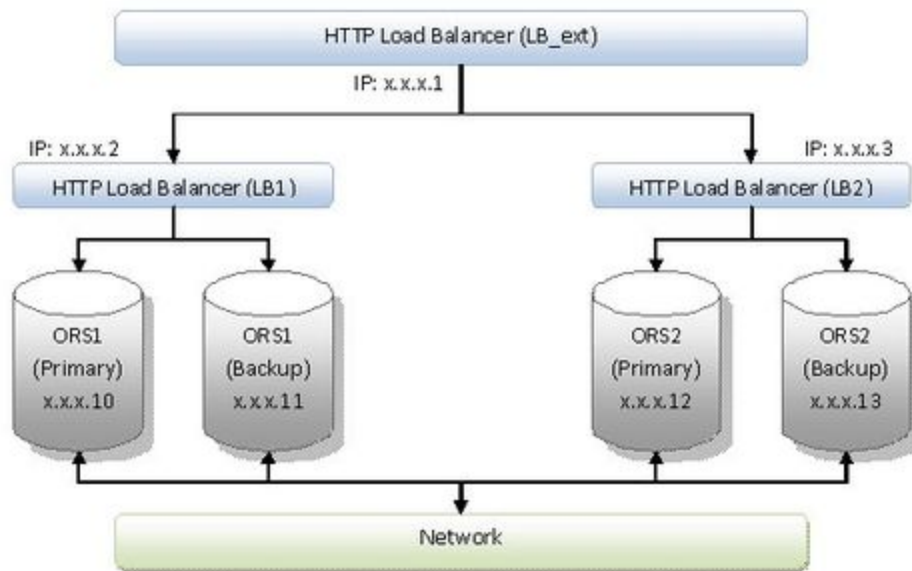
In support of the RESTful web services interface, when ORS accepts and executes a set of HTTP requests, it uses the standard mechanisms of load balancing that are typical for web servers. These mechanisms include:

- 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

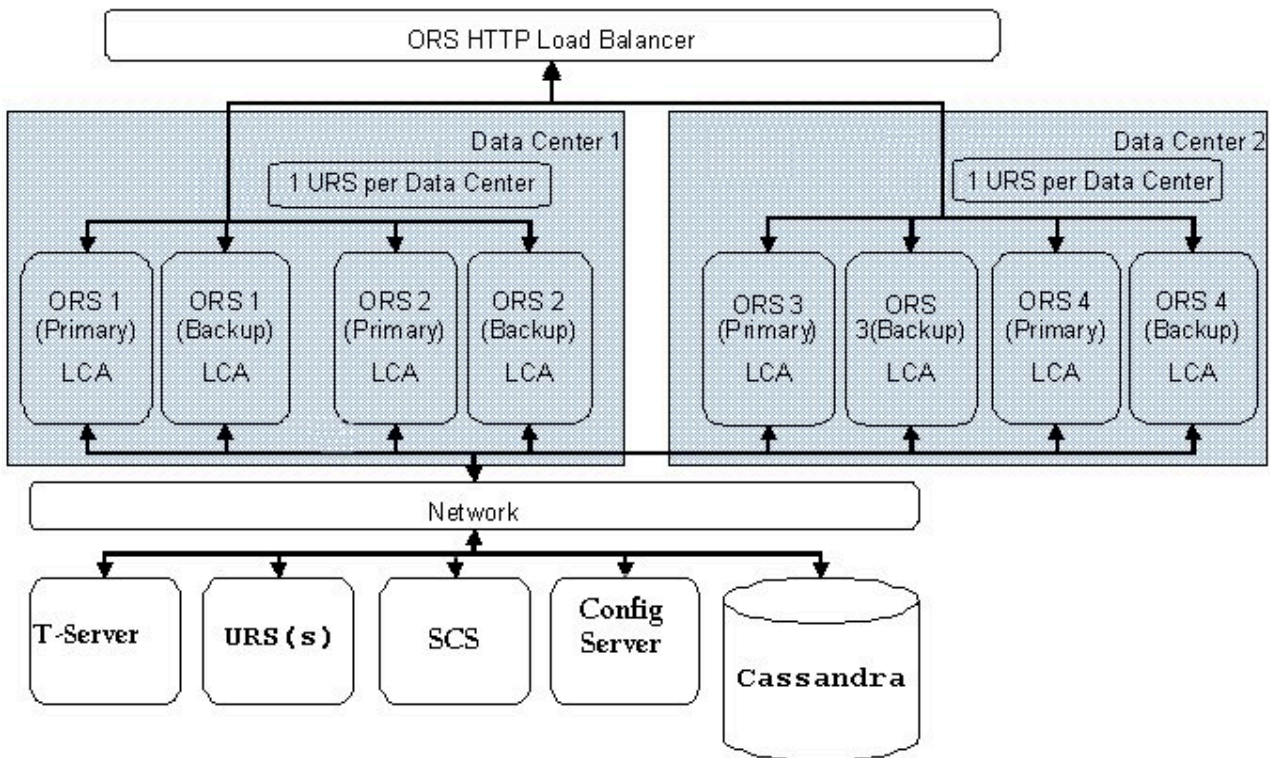
```
listen my_loadbalancer <my_loadbalancer IP>
  mode http
  option httpchk GET /heartbeat HTTP/1.1\r\nHost:\ www
  server ORS2_P <ORS2_P.address> check
  server ORS2_B <ORS2_B.address> check
```

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.

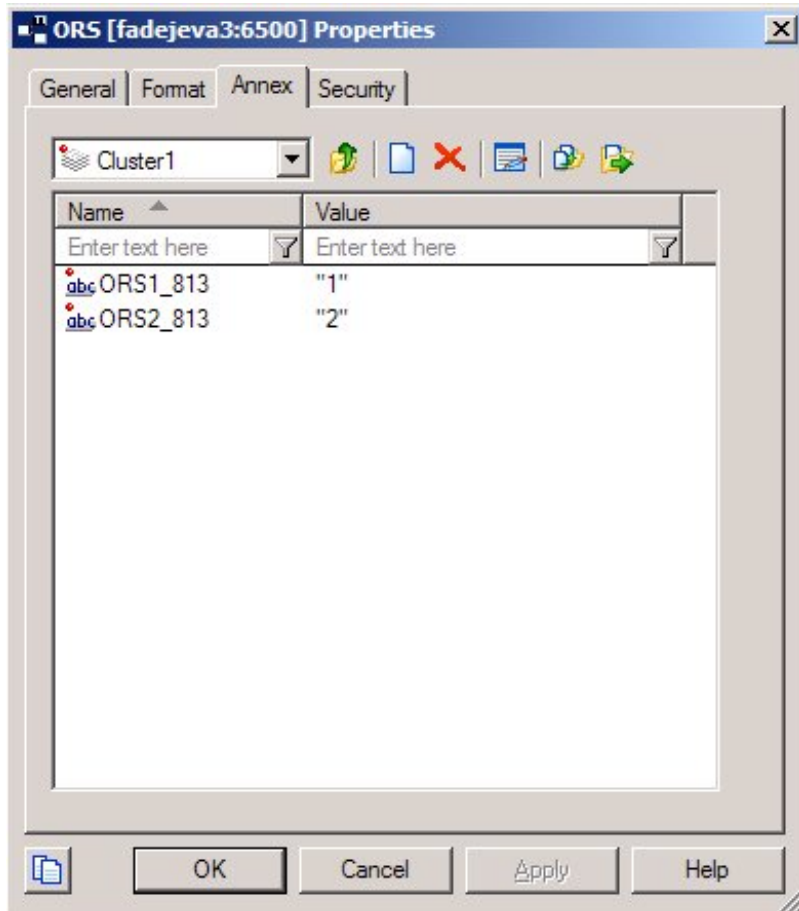


In case of one Data Center failure any sessions currently being processed by this Data Center will be lost but any new sessions will be processed in the second data center.

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.



Direct Statistic Subscription

Starting with Release 8.1.400.17, an architectural change streamlines the way that Orchestration Server obtains statistical data when executing routing strategies. Instead of requesting statistics from **Stat Server** by going through **Universal Routing Server** (URS), ORS can now request data directly from Stat Server.

This capability is only applicable for the `statistic.sData`, `statistic.getAvgData`, `statistic.getMinData`, `statistic.getMaxData` functions described in the **Orchestration Server Developer's Guide** and the `statistic.subscribe` action when used in strategies. It is not related to any other ORS capabilities that indirectly use statistical data (for example, `queue.submit` action). There are no changes to the ORS `statistic.sData`, `statistic.getAvgData`, `statistic.getMinData`, and `statistic.getMaxData` functions and the `statistic:subscribe` action so no changes are required in existing SCXML applications.

Note: This feature does not completely replace URS for obtaining statistic information since there are statistics calculated by URS itself. Specifically, the following statistics must be requested from URS only: `RStatCallsInQueue` and `StatAgentLoading`.

Configuration Summary

How to configure the ORS Stat Server Direct Access feature is summarized in the table below. Each step is then detailed in the sections that follow.

Objective	Key Procedure and Action
Configure the connection to Stat Servers in the ORS Application	Under ORS Application > Connections Add/configure each Stat Server that ORS could query for statistics
Configure Transaction Objects of Statistic Type	Under the Tenants > Routing/eServices > Transactions directory > Transaction objects of Statistic Type > orchestration section Set the statserver-source option to True
Configure the default Stat Server in the ORS Application	Under ORS Application > orchestration section Set the def-statserver-name option to <name_of_Stat_Server_application>
Configure the default object type in the ORS Application	Under ORS Application > orchestration section Set the def-stat-object-type option to <name_of_default_object_type>

Configure Connection to Stat Server

In the ORS `Application` object, add/configure each Stat Server that ORS could query for statistics.

- In `Connections`, enter the name of the Stat Server `Application`.
- If needed, configure additional capabilities for connection to Stat Server (ADDP protocol, security, and so on). For more information, see the [Framework 8.1 Configuration Manager Help](#), topic `Applications:Connections General Tab`. All capabilities, provided by Management Framework for connection to server are supported.

Configure Transaction Objects of Type Statistic

In the Configuration Database, `Transaction` objects of `Statistic` Type contain statistic definitions. For each `Tenant`, define the objects of type `Statistic` that should use ORS direct Stat Server access by configuring the `orchestration/statserver-source` option with a value of `True`.

`statserver-source`

Object: `Transaction of Statistic type`

Option section: `orchestration`

Default value: `False`

Valid values: `False`, `True`

Value changes: Upon ORS restart

This option defines the source for the statistical data in the `Statistic` configuration. If this option is set to `True`, then `Orchestration Server` direct access to `Stat Server` is used for this statistic. If the option is `False` (or the option is not configured, `Orchestration Server` will obtain the statistic data by requesting it from `Universal Routing Server`.

Configure Default Stat Server

Use the `def-statserver-name` option to specify the default Stat Server. If name of `StatServer` is not explicitly specified in the object name (see description of the `._genesys.statistic.sData` function in the [Orchestration Server Developer's Guide](#)), ORS will use default Stat Server for statistic subscription.

`def-statserver-name`

Object: Name of an object in Configuration Layer as described below

Location in Configuration Layer by precedence: `Routing Point`, `T-Server`, `Tenant`, `ORS Application`

Valid value: The name of any available Stat Server

Default value: The first available Stat Server that has the “current” Tenant in its Tenants list

Value changes: In 8.1.400.17, value changes take effect immediately except at Tenant level, when value changes take effect upon restart. In 8.1.400.18, value changes at all levels (including Tenant) take effect immediately.

ORS will look up the `def-statserver-name` option in the following order:

1. Routing Point, current for interaction/session (section `orchestration`)
2. T-Server (section `orchestration`)
3. Tenant (section `orchestration`)
4. ORS (section `orchestration`)

Configure Default Object Type

Use this option to configure the default object for which statistics are being collected.

`def-stat-object-type`

Object: ORS Application

Option section: `orchestration`

Default value: `agent`.

Valid values:

Option Value	Description
<code>agent</code>	Agent
<code>agent_place</code>	Agent Place
<code>agent_group</code>	Group of Agents
<code>place_group</code>	Group of Places
<code>queue</code>	Queue
<code>route_point</code>	Routing Point

Value changes: Take effect immediately

This value will be used by default in case an object type is skipped in the object description in the strategy.

Same Statistic for Multiple Tenants

In the case where a `Transaction` object of the same `Statistic` belongs to more than one `Tenant`, ORS selects the proper statistic definition for the action/function called from SCXML using option `def-statserver-name`.

- On startup, ORS reads all `Transaction` objects of `Statistic` type that belong to the `Tenants` associated with the ORS Application. In Genesys Administrator, the `Tenants` are listed in `Configuration > Server Info > Tenants`.
- If ORS is executing an SCXML application without a `Tenant` context, the statistical value cannot be obtained.

Disconnects/Reconnects

Since Stat Server supports only warm-standby High Availability (HA), there is no difference between scenarios with a standalone Stat Server and a Stat Server HA pair. In both cases, ORS will drop the statistic subscription and create it from scratch upon reconnect. While the connection to Stat Server is not yet established upon reconnect, the result of the `sData` function will be undefined, if ORS is requesting the statistic from that Stat Server. This result is applicable for a Stat Server HA pair as well.

Targeting a Stat Server

As described above, ORS supports simultaneous connections to many Stat Servers, which must be listed in the `Connections` tab of the ORS Application object. Any of these servers could be used to receive statistical data. You can also explicitly and implicitly subscribe to a Stat Server for exact statistical data.

Explicitly Targeting a Stat Server

You can explicitly target a Stat Server in the statistic action/function call as part of the object name. In this case, the component parses as in following code snippet, which contains the target Stat Server and statistic name.

```
<statistic:subscribe object="'SipGr_2@StatSrvName.GA'"
  statistic="'StatSrvStatName'"/>
```

If the `Statistic` name pointing to a specific Stat Server should be used, then ORS uses the cache of the connected Stat Servers to find the corresponding server. If there is no such Stat Server with that name in the connection cache, then ORS will not subscribe for this statistic and prints out the following error message into the log:

```
TFMStatSubscription: Server StatSrvName is not in the connection list
```

If the server with name `StatSrvName` is available, then ORS communicates with it and subscribes to statistic with parameters, configured in `Transaction/statistic` object, named as `StatSrvStatName`.

Implicitly Targeting a Stat Server

Implicitly targeting a Stat Server requires that one Stat Server from the list of connected ones is set as default. To configure a default Stat Server, option `orchestration/def-statserver-name` must point to its name. If this option does not contain the server name or contains an incorrect server name (i.e., the name is not from the list of connected Stat Servers in the ORS Connection list), then ORS uses the default value of the `def-staserver-name` option. An incorrect value of this option is logged via Standard-level message ID=3010. - If the name server of Stat Server is defined and it is the correct name, then ORS can communicate with the targeted Stat Server.

Debug-level logging

The log file sample below contains typical log output for the initial subscription phase: request to open statistic, event 22 (SEventStatOpened), event 2 (SEventInfo) - actual statistic value.

```
13:43:21.897 {FMStat:2} ExecuteSubscribe: Object 'AG_20_AlexK.GA', statistic
'CurrCallsInbound'
13:43:21.897 {FMStat:3} ORSStatServer::SubscribeToStatistic: <<
    RequestID      '1'
    StatServer name 'Stat_Server_812'
    tenant         'sip80'
    statistic      'CurrCallsInbound'
    target         'AG_20_AlexK.GA'
    interval       0
<<
13:43:21.902 {FMStat:3} ORSStS::HandleREvent: <<
    RequestID      '1'
    event          22
    value          0
<<
13:43:21.902 {FMStat:3} ORSStS::HandleREvent: <<
    RequestID      '1'
    event          2
    value          0
<<
```

Performance Monitoring

Starting with Release 8.1.400.21, you can monitor Orchestration Server performance using a set of **performance counters** and configure conditions that will trigger **Management Layer** alarms. The performance data can also be written to the log or **displayed in a Genesys runtime metrics client** (RTME), such as Pulse.

Performance Parameters and Counters

ORS can monitor the following performance parameters:

PARAMETER DESCRIPTION	MEASUREMENT	COUNTER ID
Number of active sessions (sessions that started or recovered on this node)	Current number	active-sessions
Number of pending session creation requests	Current number	pending-sessions
Time, required to create a session	Time in msec	create-session-time
Document processing time	Time in msec (duration from the doc_retrieved metric)	document-processing-time
SCXML Event queuing time	Time in msec	scxml-event-queuing-time
Fetch (HTTP methods) response time	Time in msec	http-fetch-time
Fetch (ESP method) response time	Time in msec	esp-fetch-time
Fetch (URS method) response time	Time in msec	urs-fetch-time
ORS/URS request - initial response delay	Time in msec (between a request and a requestID response)	urs-request-time
ORS-StatServer - SOpenStat-SStatOpened delay	Time in msec (between SOpenStat and SStatOpened)	openstat-time
Cassandra latency	Time in msec (already measured)	cassandra-latency
Function execution time	Time in msec	func-execution-time, param1 is the function name, like <code>_genesys.session.getListItemValue</code>
State time	Time in msec	state-time, where param1 is the id of state
Redirect time	Time in msec	redirect-time
Rate of session creation in sessions per second (available with 8.1.400.27)	Sessions per second	creation-session-rate

PARAMETER DESCRIPTION	MEASUREMENT	COUNTER ID
Rate of exits from state (available with 8.1.400.27)	States per second	state-rate
Rate of function execution (available with 8.1.400.27)	Number per second	func-execution-rate
Rate of fetch action executions (available with 8.1.400.27)	Fetches per second	fetch-rate
Current number of active CTI calls (available with 8.1.400.27)	Current number	active-calls

Moving Average

ORS collects performance data samples and then uses them to calculate a *moving average*, which contains the *latest* measurements. The size of this sample depends on the window during which the sample data is collected. You can define the window as the number of last measurements, or as the number of last seconds during which data should be collected. The size of the window you define (in number of last measurements or in the number of last seconds) has a major impact on the value of the moving average. You define the window size based on the particular parameter and your contact center's needs.

ORS Performance Monitoring Options

Configure the performance monitoring options in the Orchestration Server Application object performance-alarms section. Two new options are introduced to support the performance monitoring functionality: alarm-name and alarm-check-interval.

You can create different alarms with the same counter-id. Two examples are shown below.

```
slow-session-creation_one=counter-id=create-session-time;threshold=4000;
window-type=time; window-size=120
slow-session-creation_two=counter-id=create-session-time;threshold=4000;
window-type=samples; window-size=120
```

alarm-name

For alarm-name, specify a counter ID from the [Performance Parameters and Counters](#) table.

Option section: performance-alarms

Configuration object: ORS Application object

Default value: None

Valid values: Enter the alarm definition using the format and values below.

Value changes: Immediately upon notification.

```
<alarm-name>=counter-id=<counter-id>; threshold=<value>;window-type=<time|samples>;
window-size=<int>;enabled=<true|false>;
debug=<true|false>;param1="value";alarm-msg-id=<default|alarm-msg-00...| alarm-msg-24>
```

where:

- counter-id. Possible values are listed in [Performance Parameters and Counters](#) table. Specifies a performance-related counter that ORS is able to monitor.
- window-type. Possible values time, samples. Specifies the type of the window for moving average: time if the window is measured in time (seconds) or samples if the window is defined as the number of last samples.
- window-size. Size of the window, if window-type=time, then window-size specifies time interval, in seconds, on which the moving average will be calculated. If window-type=samples, then it specifies the number of the samples used to calculate the moving average.
- threshold. Specifies the threshold value. If the moving average exceeds this value, then alarm will be triggered.
- enabled. Optional parameter. Possible values are true (default), or false. If set to false, the alarm is ignored. This options gives the opportunity to temporary disable an alarm without deleting it from the configuration.
- debug. Optional parameter. Possible values true, false (default). If set to true, then the value of the moving average for this performance counter is printed in the log even if it does not exceed the threshold.
- param1. Optional parameter that may be required for some counters.
- alarm-msg-id. Optional parameter, introduced in ORS 8.1.400.27. See [Separate Message Identifiers for Performance Counter Alarms](#) for details.

alarm-check-interval

Option section: performance-alarms

Configuration object: ORS Application object

Default value: 60 seconds

Valid values: Any non-negative integer from 1 to 600

Value changes: Immediately upon notification.

This option is where you specify how often alarm conditions will be checked.

At the time of the check, if the alarm condition is true for at least one counter and it was false at the time of previous check, an alarm-level message will be printed in the format shown below along with names of all triggered performance alarms, their values and threshold values:

```
Alr 23028 Following alarms exceeds threshold values: <alarm-name1>:
<alarm-value> (<threshold-value>), <alarm-name2>:
<alarm-value2> (<threthold-value2>), ...
```

For example:

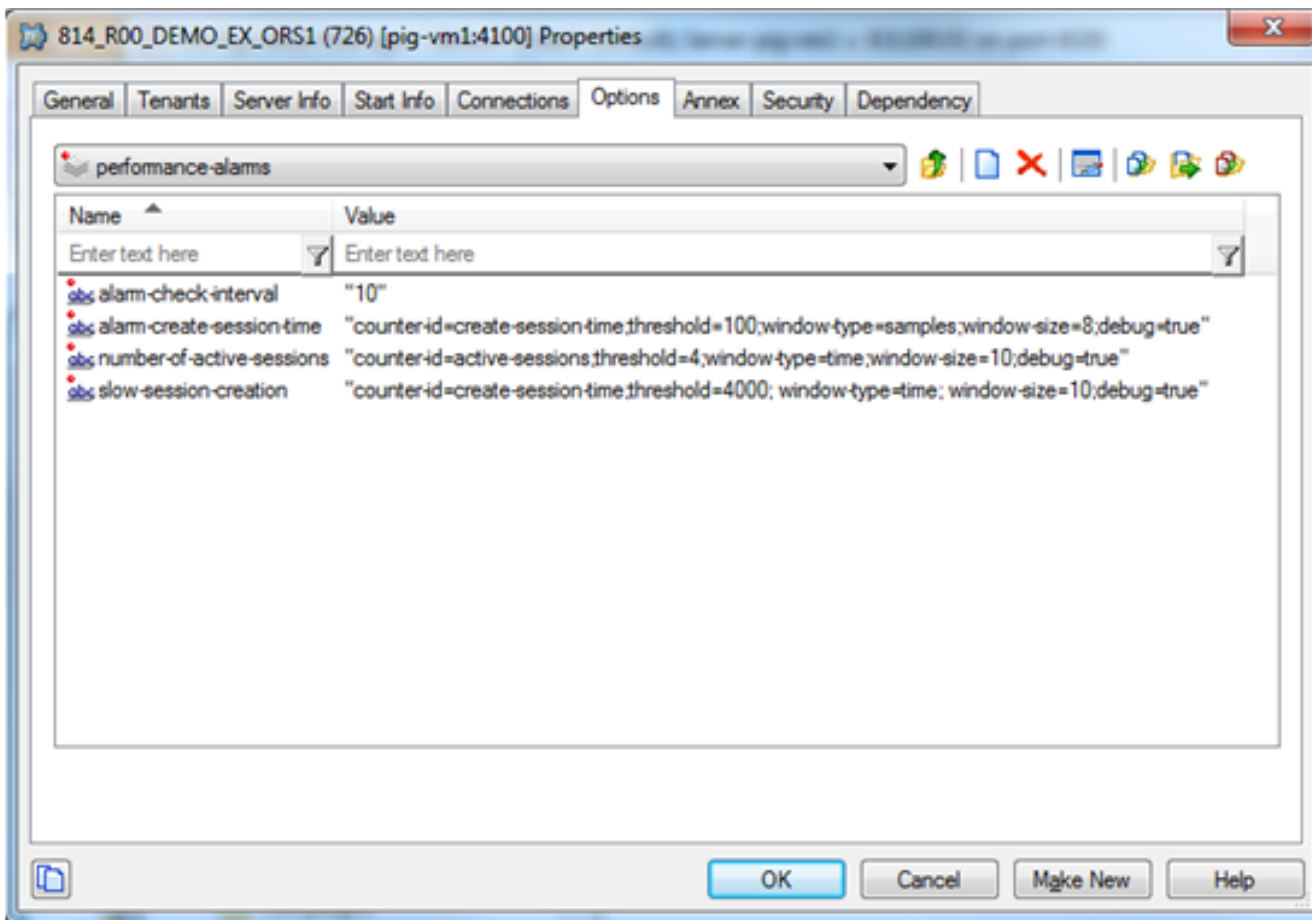
Alr 23028 Following alarms exceeds threshold values: SCXML Queuing Time:
40.9 (1.0) State Time: 45.5 (1.0)

Example Performance Alarm Configuration

Below is an example ORS performance alarm configuration that triggers when average session creation time exceeds 4 seconds if calculated on 2-minute window.

```
slow-session-creation=counter-id=create-session-time;threshold=4000;  
window-type=time; window-size=120
```

An example configuration in the ORS Application object performance-alarms section (prior to specifying the alarm name) is shown below.



Separate Message Identifiers for Performance Counter Alarms

Starting with Release 8.1.400.27, the ORS performance monitoring feature supplies an individual alarm log entry for each performance counter shown in the above table. This enhancement to the performance monitoring feature supplies:

- A default pair of message identifiers for each type of performance counter: an Alarm-level message identifier to trigger an alarm in Solution Control Interface (SCI) (or Genesys Administrator) and a Standard-level message identifier to clear an alarm in SCI (or in Genesys Administrator).
- A set of 25 reserved message identifier pairs, located in the *.lms file, which can be used in a configuration for any performance counter. For example, you could use these identifiers when configuring more than one performance counter for a counter type, such as if configuring func-execution-time counters for several different functions.

Configuring Individual Message Identifiers

When configuring message identifiers, you can use an additional, optional parameter, alarm-msg-id, in the counter configuration. This parameter defines separate log messages for each type of performance counter. The alarm-msg-id values can range from alarm-msg-00 to alarm-msg-24. Or you can use default for the default alarm message identifiers for a particular counter type.

The alarm-msg-id identifies a pair: the Alarm-level message identifier and the Standard-level message identifier from the reserved range to trigger and clear alarms in SCI (or in Genesys Administrator).

To ensure that each alarm with alarm-msg-id has a unique message identifier:

- No two counters may have same non-default alarm identifier.
- No two counters of the same type can have default as a value of the alarm-msg-id parameter.

Example Configuration

```
Active Calls=counter-id=active-calls; threshold=1000;window-type=samples;
window-size=100;enabled=true;debug=true;
alarm-msg-id=alarm-msg-01;
```

Alarms without the alarm-msg-id parameter work as they did prior to ORS Release 8.1.400.27: all alarms are listed in one message. The alarms with alarm-msg-id are printed separately with the corresponding message identifier.

Value of message identifiers when alarm-msg-id=default:

type of alarm	message id to trigger alarm	message id to clear alarm
active-sessions	25000	25001
pending-sessions	25002	25003
create-session-time	25004	25005
document-processing-time	25006	25007
scxml-event-queuing-time	25008	25009
http-fetch-time	25010	25011
esp-fetch-time	25012	25013

urs-fetch-time	25014	25015
openstat-time	25016	25017
cassandra-latency	25018	25019
func-execution-time	25020	25021
state-time	25022	25023
creation-session-rate	25024	25025
redirect-time	25026	25027
urs-request-time	25028	25029
state-rate	25030	25031
func-execution-rate	25032	25033
fetch-rate	25034	25035
active-calls	25036	25037

Value of message identifiers when alarm-msg-id=alarm-msg-xx:

value of alarm-msg-id	message id to trigger alarm	message id to clear alarm
alarm-msg-00	25100	25101
alarm-msg-01	25102	25103
...
alarm-msg-24	25148	25149

Alarm Triggering Conditions

- It may happen that an alarm condition for a counter becomes true during the interval between the checks specified by the alarm-check-interval option, but at the time of the check, the condition is false. In this case, no alarm is reported. If necessary, the alarm-check-interval may be decreased to check for alarm conditions more often.
- No alarm is raised if there is not enough data to calculate a moving average according to parameters specified in the performance alarm configuration. The moving average value will not be used to determine an alarm condition if:
 - For window-type samples, the number of collected data is less than specified in window-size
 - For window-type time, the age of the oldest collected sample is less than the value specified in window-size

For example, no alarm will be triggered under the following conditions:

- The specified window-size is 100 samples, but at the time of the alarm check, only 50 samples were collected.
- The specified window-size is 30 seconds, but all data is collected during last five seconds.

Clearing Alarms

If alarm messages are printed and conditions for all configured performance counters become false during the alarm-check-interval time, the following Standard-level message is printed to clear Management Layer alarms:

```
Std 23028 All performance alarms are cleared.
```

The `alarm-check-interval` option prevents over-populating the log with performance alarm-related messages in cases when improperly configured performance alarms are triggered too often.

For the alarms that do not exceeds threshold values but have parameter `debug=true`, the Debug-level messages are printed in following format:

```
Current alarms values:
    <Counter name1> (<threshold>): <mean> +/- <standard deviation>
    [<min>,<max>] %<current sample size>
    . . .
    <Counter nameN> (<threshold>): <mean> +/- <standard deviation>
    [<min>,<max>] %<current sample size>
```

If the sample size for some counter is zero (no values to calculate average), the following format will be used:

```
<Counter name1> (<threshold>):N/A
```

Example:

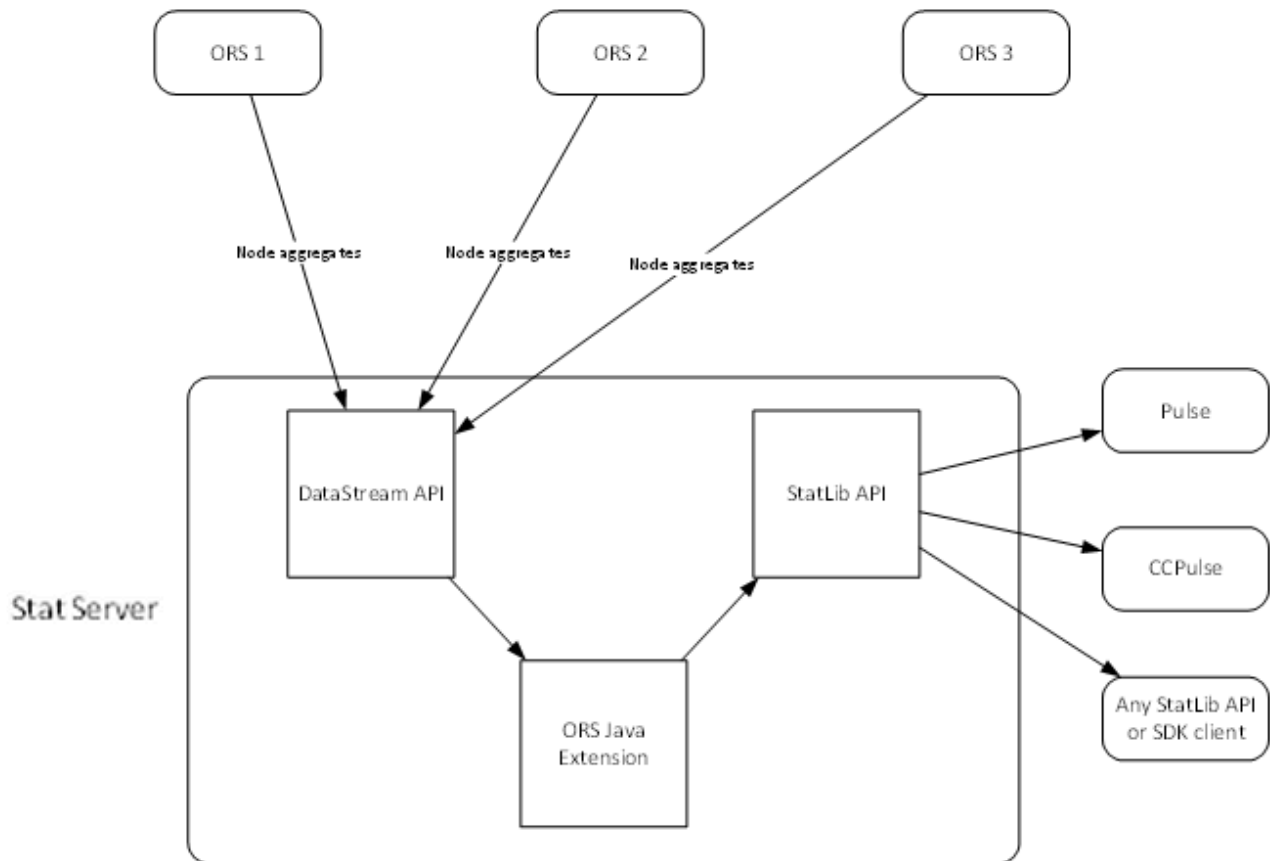
```
Current alarms values:
  Doc Processing Time (1.0):          93.00 +/-22.55 [67.00,122.00] #3
  SCXMP Queuing Time (1.0):         108.00 +/-65.33 [31.00,188.00] #10
  HTTP Fetch (1.0):                 N/A
```

The `alarm-check-interval` option also affects how often these messages will be printed.

Displaying Performance Data in a Genesys RTME Client

Starting with ORS Release 8.1.400.30, you can display ORS performance data in a **Genesys Real-Time Metrics Engine** (RTME) client such as Pulse. Each ORS **Node** can be configured to supply **performance data** to a dedicated Stat Server. An ORS Java Extension then processes, aggregates, and delivers the data (via Stat Server) to the GUI client. Any RTME GUI client (such as Pulse) may be used as a presentation layer.

Process Diagram



A summary of the deployment process is as follows:

1. Install the ORS Java Extension.
2. Provision Stat Server.
3. Configure the performance counters as statistics.
4. Configure the statistic update interval.

Installing the ORS Java Extension

- Install the ORS Java Extension located on the Stat Server Real-Time Metrics Engine 8.5 CD. For more information, see the *Stat Server 8.5 Deployment Guide*, [Manually Installing the Java Extensions](#).
- Set parameter `ORSStatExtension.jar` to `True` in the `java-extensions` section of Stat Server Application object.

Provisioning Stat Server

To send performance counter information to a dedicated Stat Server, set the `perf-counters-report` in the Stat Server Application object to `true`.

`perf-counters-report`

Option section: `orchestration`

Configuration Object: Stat Server Application object

Default value: `false`

Valid values: `true`, `false`

Value changes: Immediately upon change.

If `true`, ORS opens a Data Stream to the ORS Java Extension and submits data. Setting to `true` has no effect if ORS does not have a connection to that Stat Server.

Configuring the Performance Counters as Statistics

To configure a performance counter as a statistic, define an explicit statistical type (Stat Type) configuration for that counter. For additional information on this requirement, see the *Stat Server 8.5 User Guide*, [Statistical Type Sections](#) and [Java Sections](#). As shown below, each **performance counter** must have its own section in the **Options** tab of corresponding Stat Server Application object.

- Counter = <counter name>
- Objects = Tenant
- Category = JavaCategory
- JavaSubCategory = `ORStatExtension.jar:Calculator`
- Nodes = <list of ORS Nodes> See [Setting the Nodes Parameter](#) below.
- Aggregation = <aggregation metric - last, avg, min, max> See [Configuring the Aggregation Function](#) below.

The value of Counter must match the value of **alarm-name** of a particular **performance counter** as it is configured in the Orchestration Server Application.

Setting the Nodes Parameter

The above Nodes parameter defines the set of ORS **Nodes** whose counters will be aggregated to calculate a particular statistic. The value of that parameter may be:

- any
- `cluster:<name of ORS Cluster>`, such as `cluster:ORS_CL1` or a
- Comma-delimited list of ORS application names, such as `ORS_1,ORS_2,ORS_3`. If each ORS node consists of a **High Availability** pair, include only the name of primary ORS Application object.

If the value is any or the parameter does not present at all, the corresponding counter will be aggregated without taking the Node into consideration.

Configuring the Aggregation Function

The above Aggregation parameter defines the Aggregation function to be used to calculate a given statistic. The value may be one of the following:

- last - The value of the statistic is the latest value of the counter received from any of the ORS Nodes from the Nodes list.
- avg - The value of the statistic is the average value of the counter received from any of the ORS Nodes from the Nodes list within the time interval specified in the statistic subscription request.
- min - The value of the statistic is the minimum value of the counter received from any of the ORS Nodes from the Nodes list within the time interval specified in statistic subscription request.
- max - The value of the statistic is the maximum value of the counter received from any of the ORS Nodes from the Nodes list within time interval specified in statistic subscription request.

Example:

An example performance monitoring statistic configuration is shown below. It shows the maximum time required to create and start a session for all ORS nodes in the ORS_CL1 cluster. Assume that alarm name SessionCreateTime, configured for the create-session-time counter, exists in the ORS nodes configuration.

```
[MaxSessionCreationTime]
Aggregation=max
Category=JavaCategory
Counter=SessionCreateTime
JavaSubCategory=ORSStatExtension.jar:Calculator
Nodes=cluster:ORS_CL1
Objects=Tenant
```

Configuring the Data Stream (Statistic Update) Interval

The above [process diagram](#) shows the Data Stream to the ORS Java Extension. An ORS Node opens a Data Stream after connection/registration to a Stat Server dedicated to the ORS Java Extension. Only ORS Nodes in primary mode will open Data Streams. An ORS Node will close a Data Stream if switched to backup mode. Both primary and backup ORS Nodes use the name of the primary Node in SDataStreamInfo.pszDataSourceName upon opening of a Data Stream. ORS submits statistical data to Stat Server periodically, where time between updates may be configured with the following ORS option:

datastream-update-interval

Configuration object: Orchestration Server Application
Option section: performance-alarms
Default value: 1
Valid values: A number between 1 and 120
Value changes: Immediately upon notification

Defines the time interval in seconds between performance counters submissions from ORS to Stat Servers.

General Performance Monitoring Notes

- It is possible to configure more than one trigger for same counter that will have a different window size and threshold value. Specifying a large time interval for window-size may require a considerable amount of memory in which to keep samples if the value of the counter changes quickly.
- For information about Management Layer alarms, see [Alarm-Signaling Functions](#).
- No additional configuration is necessary to have Management Layer alarms triggered when messages related to performance alarms are logged, because the ORS Performance Monitoring functionality uses Alarm-level messaging.

Hiding Sensitive Data

Starting with Release 8.1.400.30, Orchestration Server can selectively hide sensitive data in logs when the printing of such sensitive data is not explicitly requested by an SCXML strategy. Because Orchestration Server prints Event as a whole, sensitive data may appear in ORS logs in the following scenarios.

Type of Logging	Supported?	Source for ORS Configuration Information
Headers of HTTP messages triggered by the <session:fetch> action.	Supported starting with ORS Release 8.1.401.03.	See Hiding Sensitive Data below.
Create/Update ORSURS requests for multimedia interactions.	Supported starting with ORS Release 8.1.400.30.	See Hiding Sensitive Data below.
Incoming Web requests, processed by WebFM and outgoing Web requests from WebFM.	Supported starting with ORS Release 8.1.400.30. Important Incoming/outgoing Web requests may be printed as-is by underlying library (in addition to logging in ORS FMWeb debug logging segment), and that log output cannot be filtered via options, described in this topic. If it is the case, that low-level logging may be suppressed via the ORS option <code>log\X-server-http-trace-level</code> , value less than 3.	See Hiding Sensitive Data below.
Incoming TEvents/Requests from/ to T-Server/SIP Server.	Yes. ORS and other Genesys servers already support this functionality.	See the Genesys Security Deployment Guide . Also see the log-filter-data section of the <i>Platform SDK Developer Guide</i> .

The configuration information below applies to hiding sensitive data in ORSURS and WebFM requests.

Hiding Sensitive Data

To hide ORS log data that may be considered as completely or partially sensitive in ORSURS requests and Web requests (request/response URL, headers and body), ORS implements the usage of **regular expressions**. A regular expression is a pattern describing a certain amount of text. The name "regular expression" is frequently abbreviated to "regex" as described in the option below.

Before printing sensitive data in logs, ORS verifies that string data against all regular expressions defined in the `ors-regex-<name>` option. The portion of a string that satisfies a regular expression will be replaced with "****" symbols or tagged. When filtering and/or tagging data in logs, only the hide and tag capabilities are supported as described under `<filter-type>` below.

A new option supports the hiding of sensitive data in ORSURS request and Web request logging.

ors-regex-<name>[;<filter-type>]

- For <name>, specify a descriptive name describing the rule with the intent of having different option names. Note that ors-regex is case-sensitive.
- For <filter-type>, specify the way of hiding information in the log, when regex (**regular expression**) finds a match. See <filter-type> in the table below.

Object: ORS Application object

Option section: log-filter-data

Default value: No default value

Valid values: Any valid Perl-compatible regular expression (regex).

Value changes: Take effect immediately.

Use this option to define regular expressions for hiding/tagging part of a URL, the header or body of a Web request, or the content of an ORSURS request.

<filter-type>

Valid Value	filter-type Description
hide	Symbols, matched by a regular expression, will be replaced by asterisks.
tag	<p>tag[(<tag-prefix>,<tag-postfix>)]</p> <p>The substring, matched by a regular expression, will be tagged with the prefix specified by <tag-prefix> and the postfix specified by <tag-postfix>. If the two parameters are not specified, the default tags <# and #> are used as prefix and postfix, respectively. To use the default tags, you can use any of the following values:</p> <ul style="list-style-type: none"> • tag • tag() • tag(,) <p>To define your own tags, replace the two parameters in the value with your tags. Your own tag can be any string up to 16 characters in length; any string longer than that will be truncated. If the string includes a blank space or any of the characters , (comma), (, or) as start and stop characters, they will not be counted as part of the length of the string. For information about how to use the hide and tag filter types, refer to the description of default-filter-type option in the Framework 8.5 Configuration Options Reference Manual. See Filtering and/or Tagging Data in Logs.</p>

ORS 8.1.400.31 introduced option, filter-eval-expr, which works with METRIC:eval_expr and Debug Logging Segmentation (see option log-trace-segments).

filter-eval-expr

Option section: orchestration
Configuration object: ORS Application object
Default value: false
Valid values: true or false
Value changes: Immediately upon notification

This option enables hiding of sensitive data in expression and result fields of the `eval_expr` log metric. If set to `true`, ORS verifies that string data against the regular expression defined in the `ors-regex-<name>` option.

Beginning with release 8.1.400.91, ORS does not consider the expression and result fields of the `eval_expr` metric as independent while hiding sensitive data contained in them. If information considered sensitive is found in the expression field (by matching with the regular expression specified in the `ors-regex-<name>` option), the result field is hidden automatically.

Important

One of the reasons for auto-hiding the result field is that at times it is possible to decipher sensitive data contained in the result of a previous expression if a subsequent expression and its result are known, even if the result by itself might not look like it contains sensitive data. Users must consider this when formulating regular expressions.

The following sample illustrates the above change:

In this sample, let us assume 123 is sensitive data.

Original:

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var aaa=123;' ...  
result='123' />
```

Processed and hidden:

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var aaa=****;' ...  
result='*****' />
```

A subsequent metric could implicitly expose the hidden sensitive data:

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var bbb=aaa+100;' ...  
result='223' />
```

As you know the expression `bbb=aaa+100` and its result 223, you can easily infer the value of the `aaa` variable.

Regular expression matching is quite a costly operation in terms of performance. So, you should provide regular expressions as simple as possible but yet be able to hide what you want to hide. And their number should be minimal as well.

In reality, there could be multiple (as many as you need) variables and object properties holding

sensitive data in your strategies. The following tactic is recommended as an example only without any legal obligations:

- All names of variables and object properties holding sensitive data should start with one common prefix that distinguishes them from others. For example, the prefix could be `sdhldr_` that stands for *sensitive data holder*.
- You can follow that with regular expressions that hide everything after the prefix up to the end of the statement or the expression field value, for example:
 - `(?<=sdhldr_)[^;]*`
 - `(?<=sdhldr_).*`

Now, returning to our sample illustration, the original metrics in case of using the first regex provided above:

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var
sdhldr_aaa=123;' ... result='123' />
```

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var
bbb=sdhldr_aaa+100;' ... result='223' />
```

Will be securely treated as:

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var sdhldr_****;'
... result='****' />
```

```
METRIC <eval_expr sid='~1064~003F61K53CF333DP1EFHS2LAES000001' expression='var
bbb=sdhldr_****;' ... result='****' />
```

Both the result fields will be hidden automatically as sensitive data is found in the related expression fields.

Headers of HTTP messages triggered by the `<session:fetch>` action

Beginning with version 8.1.401.03, the Hiding Sensitive Data feature is now applicable to logging of the headers of HTTP messages triggered by the `<session:fetch>` action.

To enable the feature, configure the `ors - regex - <name>` options in the `log-filter-data` section.

If the options or section are not configured, a default rule, `ors-regex-auth;hide=(?<=Authorization:).*` is applied and the value of the Authorization header is hidden in the log. The default rule covers all options with the name, `ors-regex-auth;hide`, and value, `(?<=Authorization:).*`.

Important

Default rules must be included in the ORS configuration template for new installations.

Samples

The samples below show complex regular expressions that can be used to hide substrings recognized as Social Security numbers, phone numbers, and credit card numbers.

Note: Line breaks in log message added for readability.

Regular expressions

[log-filter-data]

Credit Card:

```
(?>^(?<=[\s[:alpha:]](),,.;?!'"`=&))(>4\d{3}|5[1-5]\d{2}|6011|622[1-9]|64[4-9]\d|65\d{2})
([\ -.]|%20)\d{4}([\ -.]|%20)\d{4}([\ -.]|%20)\d{4}(?>$|(?=[\s[:alpha:]](),,.;?!'"`=&))
```

Matches only valid credit card numbers preceded/followed by any whitespace, Latin alphabetic, or any of (),,.;?!'"`=& characters, using positive lookbehind/lookahead.

4-digit groups can be delimited by a single space (or URL encoded as '%20'), period, or dash character. Make sure that this sample covers your cases before using it. Otherwise, modify the sample to fit your needs. Example:

```
http://example.com/send?cn=5555%205555%205555%205555&cvv=666
```

Phone Number using the North American Numbering Plan:

```
(?>^(?<=[\s[:alpha:]](),,.;?!'"`=&))((?:\+?|2[Bb])1([\ -.]|%20))?(?:\(|%28)?
[2-9][0-9]{2}(\|)%29)?([\ -.]|%20)?[2-9][0-9]{2}([\ -.]|%20)[0-9]{4}
(?>$|(?=[\s[:alpha:]](),,.;?!'"`=&))
```

Matches only valid phone numbers preceded/followed by any whitespace, Latin alphabetic, or any of (),,.;?!'"`=& characters, using positive lookbehind/lookahead.

An optional country calling code +1 (or %2B1 url encoded), 3-digit area code (can be surrounded with parentheses or URL encoded '%28'/'%29'), 3-digits exchange code and 4-digits subscriber number can be delimited by a single space (or URL encoded as '%20'), period, or dash character. Make sure that this sample covers your cases before using it. Otherwise, modify the sample to fit your needs. Example:

```
http://example.com/call?phone=%2B1%20%28415%29%20555%205555,
http://example.com/call?phone=%2B1(415)5555555,
http://example.com/call?phone=14155555555
```

Social Security Number:

```
(?>^(?<=[\s[:alpha:]](),,.;?!'"`=&))(!000|666|9)\d{3}([\ - ]|%20)(?!00)\d{2}
([\ - ]|%20)(?!0000)\d{4}(?>$|(?=[\s[:alpha:]](),,.;?!'"`=&))
```

Matches only valid Social Security Numbers preceded/followed by any whitespace, Latin alphabetic, or any of (),,.;?!'"`=& characters, using positive lookbehind/lookahead.

3-digits, 2-digits, and 4-digits groups can be delimited by a single space (or URL encoded as '%20') or the dash character. Make sure that this sample covers your cases before using it. Otherwise, modify the sample to fit your needs. Example: <http://example.com/find?ssn=078%20051120&v=12>

Hiding of Certain Properties in a JSON-Encoded Object

Initial Log Message:

```
05:35:30.049 {ORSURS:3} CreateVirtualCall: <<
refID          1
tenant         'Automation'
call           'da377034140aacad'
context
'{"MediaType":"email","UserData":{"ReceivedAt":"2015-07-30T12:35:28Z","InteractionId":"da377034140aacad","Media
"PlacedInQueueAt":"2015-07-30T12:35:28Z","InteractionType":"Inbound",
"InteractionSubType":"InboundNew","Queue":
"Qaart_EmailInboundQueueAutomation","SubmittedBy":
"QAART_Media_Server_email_Automation",
"SubmittedAt":"2015-07-30T12:35:28Z",
"MovedToQueueAt":"2015-07-30T12:35:28Z","TenantId":101,
"InteractionState":0,
"IsOnline":0,"IsLocked":0,"next_kvlist":{"str":"some str","int":3},
"test_name":"eServiceTerminate"}}'
<<06:25:11.525 {FMWeb:3} Send pending HTTP response:
Session ID = 9F7V6Q4L8P00P7FUD0I8FDM4F4000001, Send ID = 2,
Result Code = Schedule_01_session_start_done,
Content = {"PhoneNumber":"+79211122333"}
```

Configuration:

```
[log-filter-data]
ors-regex-hide-testname=(?<="test_name":).*?(?=")
ors-regex-hide-mediatype;tag=(?<="MediaType":).*?(?=")
ors-regex-hide-part-of-phone-number=(?<="PhoneNumber":"\+7921).*?(?=")
```

Log Message After Hiding:

```
05:35:30.049 {ORSURS:3} CreateVirtualCall: <<
refID          1
tenant         'Automation'
call           'da377034140aacad'
context
'{"MediaType":"email","UserData":{"ReceivedAt":"2015-07-30T12:35:28Z","InteractionId":"da377034140aacad","Media
"<#email#>","PlacedInQueueAt":"2015-07-30T12:35:28Z","InteractionType":
"Inbound","InteractionSubType":
"InboundNew","Queue":
"Qaart_EmailInboundQueueAutomation","SubmittedBy":
"QAART_Media_Server_email_Automation",
"SubmittedAt":"2015-07-30T12:35:28Z",
"MovedToQueueAt":"2015-07-30T12:35:28Z",
"TenantId":101,"InteractionState":0,"IsOnline":0,
"IsLocked":0,"next_kvlist":{"str":"some str","int":3},"test_name":"****"}}'
<<
06:25:11.525 {FMWeb:3} Send pending HTTP response:
Session ID = 9F7V6Q4L8P00P7FUD0I8FDM4F4000001,
Send ID = 2, Result Code = Schedule_01_session_start_done,
Content = {"PhoneNumber":"+7921****"}
```

Debug Logging Segmentation

Starting with Release 8.1.300.52, ORS supports Debug Logging Segmentation, which provides more precise control of the logging when the `log-trace-segments` option is configured. The debug segment header `{<segment_name>:<log_level>}` is now added right after the timestamp. For example: `10:31:02.249 {ScxmlMetric:3} METRIC <event_queued sid='ML06K6519D5UPAGSHNE41VJBSS00000M' name='interaction.added' type='external' thread='15800' />`

Use this option to specify the 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.

Example:

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

Note: Starting with ORS 8.1.400.31, a new Debug Log Segment header `ScxmlMetricEvalExpr` is added for `METRIC:eval_expr`. In a log file, it will appear as follows:

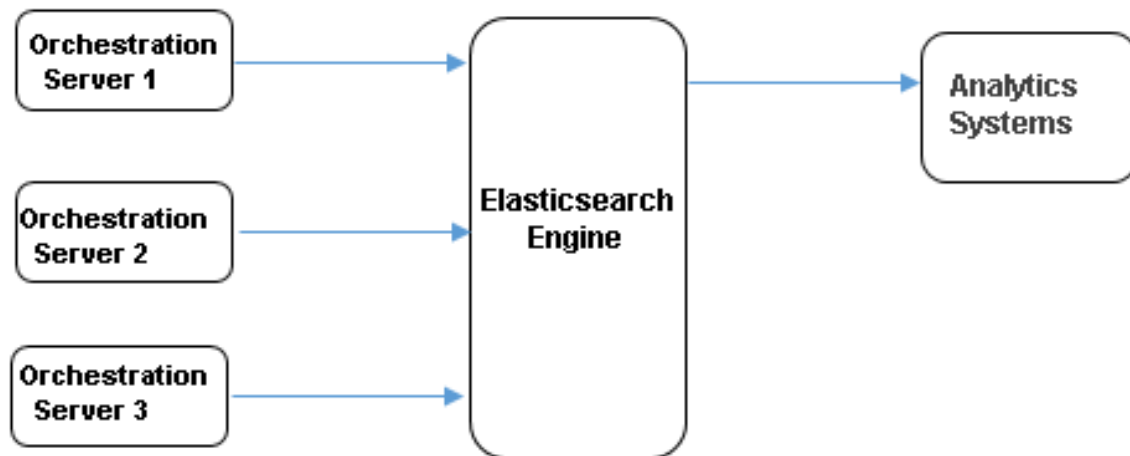
```
14:47:11.414 [T:8436] {ScxmlMetricEvalExpr:3} METRIC <eval_expr sid='ors2521'
expression='global._data =
new Object();' result='[object Object]' thread='8436' />
```

As a result, the `log-trace-segments` option adds `ScxmlMetricEvalExpr` as a new value.

Also see `filter-eval-expr`.

Elasticsearch Connector

Starting with Release 8.1.400.40, Orchestration Server uses [Elasticsearch 2.x](#) to store data, such as operational and performance data. The data may then be used with Kibana (default) or a custom visualization tool to monitor Orchestration Server performance and routing session processing in near real time.



Starting with release 8.1.400.64, ORS supports Elasticsearch releases 5.x, starting from 5.3.0. There have been significant changes between major Elasticsearch releases, in relation to APIs and settings. ORS functionality has been enhanced such that it dynamically adjusts in accordance with the version of the Elasticsearch node that it operates with.

Important

Secure connections to Elasticsearch 5.x are supported using TLS 1.2 only.

General Setup/Configuration

1. Download and install [Elasticsearch](#).
2. [Configure Elasticsearch](#) for Orchestration.
3. Configure the ORS Elasticsearch options for [session](#), [performance](#), and [node](#) reporting. You do not have to configure each type of reporting (although you can); you can configure only the reporting options that you wish to use.
4. Configure [Security options](#).

Configuring Elasticsearch for Orchestration

[+] Configuring Elasticsearch for Orchestration

The purpose of the information below is to provide basic recommendations for Elasticsearch 2.2+ configuration that will be more or less optimal in an Orchestration-specific scenario. It is not a complete Elasticsearch configuration guide of any kind. To get complete information about Elasticsearch installation, go to www.elastic.co.

Note: The URLs shown below were obtained from the Elasticsearch website on 03/31/16.

Specifics of Orchestration Usage of Elasticsearch

The Orchestration scenario is very similar to a “Logstash” scenario – very frequent index writes and not very frequent queries. Orchestration uses two daily indexes, “session-*” and “performance-*”, that experience frequent write operations, mostly via the bulk API of Elasticsearch. To ensure that Elasticsearch will successfully handle data from Orchestration, some important adjustments of the default configuration of Elasticsearch data nodes are required.

Elasticsearch Clustering

Generally speaking, in most cases, even a single properly configured Elasticsearch data node may handle the load produced by an Orchestration cluster in deployments with 5-10 sessions per second. However, we suggest provisioning of an Elasticsearch cluster with at least two data nodes and one load-balancing (client) node.

Suggested Configuration for Elasticsearch Data Nodes

JVM Heap Size

The recommended size of JVM heap is at least 6GB. For detailed instructions for setting on a particular platform, look in <https://www.elastic.co/guide/en/elasticsearch/guide/current/heap-sizing.html>

Disable Memory Swapping

In the `elasticsearch.yml` file:

- `bootstrap.mlockall: true` (before ES version 2.4)
- `bootstrap.memory_lock: true` (starting from ES version 2.4)

For details, look in <https://www.elastic.co/guide/en/elasticsearch/reference/2.2/setup-configuration.html#setup-configuration-memory>

Transaction Logging Configuration

In the `elasticsearch.yml` file:

- `index.translog.durability: async`
- `index.translog.flush_threshold_ops: 5000`

The first setting is the most important one – it enforces Elasticsearch to fsync and commit a transaction log in the background every 5 seconds (default value of `index.translog.sync_interval`), instead of doing that after each operation (default setting). For details, look in https://www.elastic.co/guide/en/elasticsearch/reference/2.2/index-modules-translog.html#_translog_settings

Thread Pool Configuration

In `elasticsearch.yml` file:

- `threadpool.bulk.size`: # availableProcessors
- `threadpool.bulk.queue_size`: 500
- `threadpool.index.size`: # availableProcessors
- `threadpool.index.queue_size`: 500

In most cases, explicit configuration of the “size” parameter is not required because Elasticsearch will detect the number of processors and use it “under the hood.” However, sometimes that number is not properly retrieved, so it is better to configure it by hand. The number of processors, detected by Elasticsearch, may be checked via “`_nodes/os`” API. For details of thread pool configuration, check <https://www.elastic.co/guide/en/elasticsearch/reference/2.2/modules-threadpool.html>

Indices Settings

In the `elasticsearch.yml` file:

- `indices.memory.index_buffer_size`: 30%
- `indices.memory.min_shard_index_buffer_size`: 12mb
- `indices.memory.min_index_buffer_size`: 96mb

The purpose of these settings is to slightly increase the size of JVM heap used to buffer newly indexed documents. For details, look in <https://www.elastic.co/guide/en/elasticsearch/reference/2.2/indexing-buffer.html>

Cache Sizes

In the `elasticsearch.yml` file:

- `indices.fielddata.cache.size`: 15%

Defines the % of node VM heap that is used as a field data cache. For details, check <https://www.elastic.co/guide/en/elasticsearch/reference/2.2/modules-fielddata.html>

Index Refresh Interval

In the `elasticsearch.yml` file:

- `index.refresh_interval`: 5s

This is the time between index refreshes, when newly created documents become available for search operations. Since refresh is somewhat expensive, it is better to increase it from the default 1s. For details, check <https://www.elastic.co/guide/en/elasticsearch/reference/2.2/index-modules.html#dynamic-index-settings>

Inline Indices Support

Support of scripting for “update” API has to be enabled.

In the `elasticsearch.yml` file:

- `es.script.engine.groovy.indexed.update=true` # (for ES 2.x)

Final Note

From the suggested Elasticsearch configuration changes, only three are mandatory:

- Increased JVM heap size
- Disabled swapping (bootstrap.mlockall: true or bootstrap.memory_lock: true)
- Asynchronous translog commit (index.translog.durability: async)

For all other configuration parameters, we suggest to monitor real resource utilization by Elasticsearch nodes (via `_nodes/stats` API) and use Elasticsearch recommendations and your own good judgment. The best source of information is “ElasticSearch: The Definitive Guide” <https://www.elastic.co/guide/en/elasticsearch/guide/current/index.html>

Elasticsearch 5.3 Configuration Recommendations

The following options must be configured in the `elasticsearch.yml` file if Elasticsearch 5.3 is being used:

- `bootstrap.memory_lock: true`
- `indices.memory.index_buffer_size: 30%`
- `script.engine.painless.stored.update: true`

Configuring ORS Options for Elasticsearch

Session Reporting

ORS stores SCXML session data into Elasticsearch where documents are created in a “session” daily index. The Orchestration session ID is used as index key. ORS saves information about session states if `_es_store=true` in the state description in the strategy. Example: `<state id="routing" _es_store="true">`.

[+] Session Reporting Document

Note: This document is predefined and not configurable.

ORS stores following session data:

```

“ani”: “<The value of the ANI attribute. It applies to voice interactions only>”,
“application”: “<URL of the SCXML document>”,
“begin”: “<date and time in UTC format when that session has been started>”,
“cluster”: “<name of ORS cluster>”,
“connectionID”: “<ConnID for primary voice call, empty otherwise>”,
“device”: “<for voice, name of DN where call arrival initiated session creation, for
multimedia, name of queue that interaction was pulled from>”,
“end”: “<date and time in UTC format session has been terminated>”,
“interactionID”: “<interaction ID>”,
“media”: “<name of media type>”
“node”: “<name of ORS node>”,
“states”:
  “<state name>”:
    “total_duration”: <sum of durations in that state, msec>,
    “count”: <number of times scxml entered that state>
“duration”: “<The duration of the session in seconds (the difference between end and begin
timestamps). It is populated at session termination>”,

```

Notes:

- If "<state name>" contains dot symbols (.), the state name stored in Elasticsearch will contain underscore symbols instead of dots.
- Starting with release 8.1.400.43, ORS stores two new Session Reporting fields into Elasticsearch: `ani` and `duration`.

Session reporting uses the following options:

`ors-es-session-report`

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: `false`

Valid values: `true`, `false`

Value changes: Take effect immediately upon notification.

This option specifies if session reporting is enabled or disabled. When set to `true`, session reporting is enabled. When set to `false`, session reporting is disabled.

If an `ors-es-session-report` value is changed from `false` to `true` in the middle of a session, ORS will send an update request for a non-existing session document. Elasticsearch will respond with a "document missing" exception, which ORS will ignore.

`ors-es-bulk-write-period`

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: 5 seconds

Valid values: An Integer value in the range of 5 to 120 seconds.

Value changes: Take effect immediately upon notification.

Used for both session and performance reporting. The word "bulk" refers to the possibility of having ORS accumulate a configurable amount reporting data and then sending the data to Elasticsearch instead of sending one reporting request at a time.

This option specifies the maximum time interval between the sending of two bulk requests to Elasticsearch. The bulk mechanism is based on an internal buffer that stores all requests that should be sent to Elasticsearch. In cases where there are no buffered requests, the actual time between two bulk requests could exceed the `ors-es-bulk-write-period`. However, when there is a steady stream of the requests to be handled by single bulk operation, the actual time between two bulk requests will be equal to the time defined by this option.

ors-es-bulk-max-size

Option section: elasticsearch

Configuration object: ORS Application object

Default value: 10000

Valid values: Integer value in range from 1000 to 100000000

Value changes: Take effect immediately upon notification.

Used for both session and performance reporting. This option specifies the maximum number of actions inside a single bulk request. If reporting is enabled, ORS stores all requests in its internal bulk requests cache. This option can be set to control cache overgrowth. If the maximum number of actions in the cache is reached, ORS removes the oldest actions and inserts the newest.

Note: Genesys recommends clustering Elasticsearch nodes with several master nodes to prevent a potential loss of reporting data should an extended node disconnect occur (unless all available Master nodes in an Elasticsearch cluster are excluded).

Performance Reporting

If **performance monitoring** is enabled, ORS stores performance data into Elasticsearch where documents are created in a “performance” daily index.

[+] Performance Reporting Document

Note: This document is predefined and not configurable.

ORS stores following session data performance data:

```
“cluster”: “<name of ORS cluster>”,
  “node”: “<name of ORS node>”,
  “timestamp”: “< date and time in UTC format when that performance counter has been
submitted by ORS>”,
“counters”: {
  “name”: “<counter name>”: <counter value>}
```

Performance reporting uses the following options:

ors-es-perfsnapshot-report

Option section: elasticsearch

Configuration object: ORS Application object

Default value: false

Valid values: true, false

Value changes: Take effect immediately upon notification.

This option specifies if performance reporting is enabled via Elasticsearch.

- If set to `true`, performance reporting via the Elasticsearch is enabled.
- If set to `false`, performance reporting via Elasticsearch is disabled.

ors-es-bulk-write-period

Used for both session and performance reporting. Option `ors-es-bulk-write-period` allows you to submit session or performance reporting to Elasticsearch via a "bulk" mode. For information on this option, see the description in [Session Reporting](#).

Node Reporting

ORS stores node information into Elasticsearch where a document is created in a "nodes" index. The document is created during node startup and updated when ORS switches to primary mode.

[+] Node Reporting Document

Note: This document is predefined and not configurable.

ORS stores the following node data:

```

    "cluster": "<name of ORS cluster>",
    "default_port": <ORS default port>,
    "host": "<hostname where application is running>",
    "http_port": <ORS http port>,
    "node": "<name of ORS node>",
    "synch_port": <ORS synch port>,
    "application": "<name of application that currently is in primary mode>",
    "sw_start": "<date and time in UTC format when ORS switches to primary mode>"

```

If the port is not configured in the Application, then ORS reports the value -1.

Node reporting uses the following options:

ors-es-node-info-report

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: `false`

Valid values: `true`, `false`

Value changes: Take effect immediately upon notification.

This option specifies if node reporting via the Elasticsearch engine is enabled. If, upon ORS startup,

`ors-es-node-info-report` has a value of `false`, and the value is subsequently changed to `true`, the report information will not be sent.

`ors-es-nodes`

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: `<empty string>`

Valid values: String containing a semicolon-separated list of `http(s)://host:port` of Elasticsearch servers.

Value changes: Take effect after restart.

This option is used for connectivity to Elasticsearch. It specifies the addresses of Elasticsearch nodes. Configure each ORS to work with an Elasticsearch node. For valid values:

- If the node is specified as `http://<ES host>:<port>`, then a non-encrypted connection will be used for communication with Elasticsearch.
- If the node is specified as `https://<ES host>:<port>`, then an encrypted connection will be used for communication with Elasticsearch.

Reporting for ORS Elasticsearch Clusters

ORS nodes can send report requests to either a single Elasticsearch node or to an Elasticsearch cluster. It does not matter whether Elasticsearch is installed in single mode or in cluster mode. ORS communicates with Elasticsearch using the `host:port` specified in option `ors_es_nodes`.

- If Elasticsearch is configured as a single node setup, then ORS sends report requests directly to this node.
- In the case of an Elasticsearch cluster, each ORS Application can communicate with a specific node of the cluster, or all ORS Applications can communicate with a dedicated load-balancing node.
- The dedicated load-balancing node is the Elasticsearch node that is configured to receive documents and distribute them to the data nodes of the cluster.
- Upon startup, ORS connects to the first node in the `ors-es-nodes` list. If a disconnect from this Elasticsearch node occurs, ORS will try to connect to the next Elasticsearch node specified in the list.

ORS uses the following options for connectivity:

`ors-es-reconnect-timeout`

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: 5 seconds

Valid value: 1 to 20 seconds.

Valid changes: Take effect immediately upon notification.

This option specifies the timeout period that ORS uses when trying to reconnect to an Elasticsearch node. If a connection to Elasticsearch is lost, then ORS will try to reconnect to the next Elasticsearch node in the `ors-es-nodes` list.

`ors-es-node-exclude-timeout`

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: 600 seconds

Valid values: Integer value in range from 5 to 86400 seconds.

Valid changes: Take effect immediately upon notification.

After successfully connecting to an Elasticsearch node, ORS checks the node status. In case of a red status, ORS disconnects from the node and excludes the node from the reconnection procedure for the time period defined by the option `ors-es-node-exclude-timeout`.

Security Configuration

ORS communicates with Elastic search via the Elasticsearch REST API. The following security features are supported:

- TLS/SSL—ORS connects to Elasticsearch using SSL or TLS. The connection is supported via proprietary security layer in Genesys Framework libraries. For details, see the [Genesys Security Deployment Guide](#), chapter "Protection of Data in Transit".

To enable a TLS/SSL connection to an Elasticsearch node, specify it in the `ors-es-nodes` option with the `https://` prefix.

- Authentication (basic)—ORS supports authorization and authentication based on standard HTTP protocol mechanisms.

Configure the following basic authentication options:

`username`

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: <empty string>

Valid value: String value

Valid changes: Take effect immediately upon notification.

This option specifies the username that should be used for basic authentication on the Elasticsearch server.

password

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: <empty string>

Valid value: String value

Valid changes: Take effect immediately upon notification.

This option specifies the password that should be used for basic authentication on the Elasticsearch server.

Limitations

- All ORS Applications that belong to the same ORS cluster must be connected to master nodes of the same Elasticsearch cluster.
- Also see the Note in [ors-es-bulk-max-size](#).

Configuring ORS Options for Elasticsearch 5.3

Users can configure the following options as required, if Elasticsearch 5.x is being used. All other existing options are supported for Elasticsearch 5.3 also.

Dynamic Index Settings

Starting from Elasticsearch 5.3, configuration of the index module for each Elasticsearch node in the `elasticsearch.yml` file is not possible. Settings for the index module can only be applied per-index via the Elasticsearch index or settings API. ORS uses the API to explicitly define the required index module settings for each index it creates.

The following option is available to configure the index module settings:

`index-settings-<version>`

Option section: `elasticsearch`

Default value: `{"index":{"translog":{"durability":"async"},"refresh_interval":"5s"}}`

Valid values: Elasticsearch API request content in the form of a JSON string.

Value changes: After ORS restarts.

This option is to define the Elasticsearch version-specific index settings based on which settings will be applied to the indexes that ORS creates. Each ORS application dynamically updates index settings after any operation that may cause the creation of a new index.

Important

Elasticsearch version in the above option can be specified completely, as in "5.4.0", or partially, as in "5" or "5.3". For example, if it is specified as "5", the index settings from this option's value will be applied if the version of Elasticsearch starts with "5."

Index Settings for Custom Indexes

A new attribute, **settings**, has been added to specify index settings for custom indexes created using the `<elasticconnector:createdoc>` or `<elasticconnector:updatedoc>` actions. If not specified, ORS uses the settings value specified in the `index-settings-<version>` option. If the Elasticsearch version is less than 5.x, this attribute is ignored.

If the `<elasticconnector:createdoc>` action or the `<elasticconnector:updatedoc>` action is executed without the settings attribute, the result is the same as executing the action with the value defined by the current Elasticsearch node's **index-settings-<version>** option.

Important

New index settings are sent for the custom index only if they are different from the previous settings sent for that particular index, except if the Elasticsearch node was disconnected or ORS was restarted.

Orchestration Plugin

Starting with Release 8.1.400.49, ORS installation includes a plugin for a real-time view of active sessions currently being executed. With the Orchestration Plug-in integrated into [Kibana](#) you can:

- Display a list of active sessions.
- Use different filter criteria to refine the list of active sessions.
- View session-related information.
- Terminate an active session (for example, a "stuck" session).

You can filter using the following criteria:

- time interval (from/to)
- session age
- node (one or more)
- cluster, (one or more)
- device (routing point or interaction queue)
- media type (voice, email, chat, and so on)

ORS Application Configuration

In the ORS Application object, Options tab, configure the Elasticsearch options in the elasticsearch section as described above.

Plugin Installation

- Install Elasticsearch 2.4.1 and Kibana 4.5.0.
- Install ORS 8.1.400.49.
- Navigate to elasticserach\kibana-plugins\Orchestration subfolder and unpack the plugin.zip file.
- Follow the instructions provided in the installation guide PDF to launch Elasticsearch and Kibana.

[+] Plugin Installation Guide

Important

Starting with release 8.1.400.70, the Orchestration plugin is not compatible with versions of Kibana prior to 5.6.0.

PREREQUISITES:

Existing Kibana installation. This manual uses <Kibana root folder> instead of C:\bin\kibana for Windows and ~/<Kibana root folder> for Unix as the root path of your Kibana installation. This manual assumes that the current folder for the commands is the one where this document resides in the ORS package.

INSTALLATION STEPS:

1. Stop Kibana if it is running. Refer to the Kibana manual for the installed version.
2. Unpack the plugin.

On Windows:

- Open file plugin.zip in Windows Explorer.
- Click the “Extract all files” button and then follow the instructions on screen.
- Or use any archiving tool of your choice.

On Unix:

```
tar -xvzf plugin.tgz
```

3. Delete previous versions of the Orchestration plugin. Skip this step if this is the first installation of this plug-in. Note: you may need to have administrator privileges for this step.

On Windows:

- Open <Kibana root folder>\installedPlugins
- Delete the “orchestration” folder
- Open <Kibana root folder>\optimize\bundles
- Delete all files with names ending with .bundle.js, .entry.js, and .style.css

On Unix:

```
rm -rf ~/<Kibana root folder>/installedPlugins/orchestration
rm -rf ~/<Kibana root folder>/optimize/bundles/*.bundle.js
rm -rf ~/<Kibana root folder>/optimize/bundles/*.entry.js
rm -rf ~/<Kibana root folder>/optimize/bundles/*.style.css
```

4. Copy the plugin code to the Kibana plugin folder. Note: you may need to have administrator privileges for this step.

On Windows:

Copy the “orchestration” folder extracted on step 2 to the folder <Kibana root folder>\installedPlugins. If you have used Windows Explorer for unpacking, “orchestration” is under the “plugin” folder.

On Unix:

```
cp -R orchestration ~/<Kibana root folder>/installedPlugins/
```

5. Set the correct plugin version.

- Open the package.json file in the orchestration folder under the plugins folder of your Kibana installation folder, in any text editor.
- Look for the version field and change the value from the default 0.0.0 to the version of your Kibana installation.
- If unsure of your installation version, check the package.json file in the root folder of the Kibana installation.

6. Cleanup the Kibana index in ElasticSearch (if you ES instance was never connected to Kibana with the plugin (prior to version 8.1.400.63), skip this step).

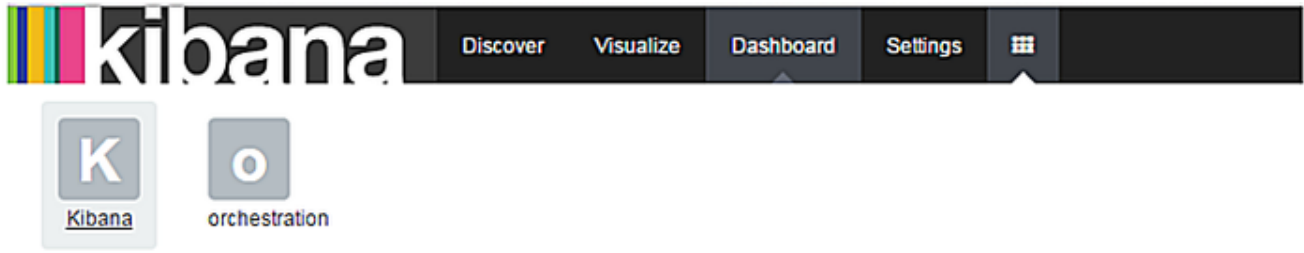
- Edit the Powershell script reindex.ps1.
- If necessary, change the Kibana index name from the default .kibana to the one set in your installation.
- Check the Kibana configuration file if you are unsure of the index name.
- Run the edited script with parameters -h and -p specifying the host and port of your ElasticSearch instance. (Both parameters are optional. The defaults are localhost for -h and 9200 for -p. For example, powershell .\reindex.ps1 -h example.com -p 1234. Powershell version 3 or higher iss required.)

7. Start Kibana. Refer to the Kibana manual for the installed version. Starting takes a little time because Kibana has to process the plugin code.

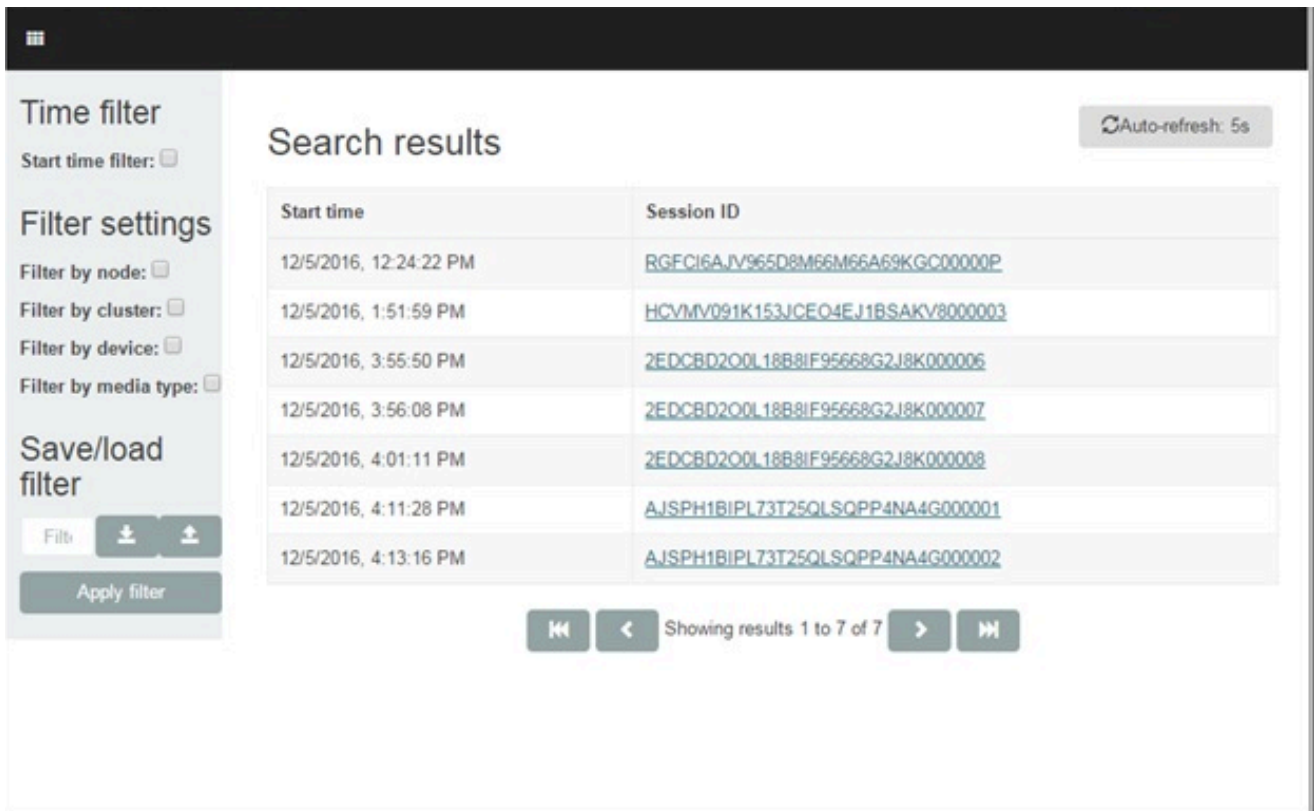
CHECK THE INSTALLATION:

Open Kibana in a browser (yourdomain:5601 by default). When Kibana is loaded, an application switching button should appear to the right of **Settings** tab. When clicked, an application list will appear.

- Access the Kibana dashboard through your browser. Kibana adds an **orchestration** icon.



- Select the orchestration icon to bring up the plugin interface. The example below shows a list of active sessions.



- To the left, you can filter by session start time interval, session age, node, cluster, device or media type. You can also save and load filters.
- To the right, the interface displays search results: the **Start Time** and **Session ID** of active sessions. **Session ID** is the link to the Session information dialog box (shown below), which contains a control to terminate a session.
- An **Auto-refresh** on/off button appears at the top right.
- Forward and back buttons appear for navigating through search results that exceed the page.

Applying a Filter

Below is an example of filtering for sessions within a specific time frame.

1. Select the **Start time filter** check box.
2. Define the from/to time frame using **Start time interval** button.
3. Select the **Apply filter** button. Sessions filtered within this time frame display under the **Session ID** column.

Retrieving Session Information

Click particular session identifier. A Session information dialog box summarizes the selected session (see example below).

Terminating a Session

Click the **Terminate** link in the Session Information dialog box. An HTTP request to terminate the session is sent to the corresponding ORS node.

Session information ×

Session ID AJSPH1BIPL73T25QLSQPP4NA4G00000F

Start time 12/6/2016, 3:46:22 PM

End time Currently active. Terminate.

Cluster Cluster1

Node ORS1_814

Application http://localhost/scxml/workspace/src-gen/IPD_ORS_RouteToAgent.scxml

Device RP_sip1

ANI 450

ConnectionID 01e60291fad34134

InteractionID VO6N3BGU8H71F3D7RR8PH33JN800009K

Media type voice

States information

State name	Duration	Entered that state
_composer_init	15	1
RouteToAgent_Entry1	16	1
RouteToAgent_Assign1	0	1
RouteToAgent_UserData1	47	1
RouteToAgent_PlaySound1	1201	1
RouteToAgent_RaiseEvent1	0	1
RouteToAgent_WaitEvent1	1014	1

Plugin Limitations

The ability to terminate a session via the Orchestration Plugin feature is not supported:

1. In a deployment using [Recovery of Voice Calls Without Persistence](#).
2. If an ORS node is restarted.

Elasticsearch Connector Enhancements

Starting with 8.1.400.58, ORS enhances its Elasticsearch real-time reporting capabilities with the addition of the following new features:

- Explicit mapping of session and performance indexes
- A new Orchestration SCXML extension, `elasticconnector`
- Samples of Kibana dashboards and visualizations

Explicit Mapping

ORS now uses **Explicit mapping** instead of Dynamic mapping for **session** and **performance** indexes via the creation of index templates (`tpl_session` and `tpl_performance`). From a performance standpoint, Explicit mapping is more optimal and supports all data types. ORS creates Session and Reporting indexes as soon as it starts and switches to Primary mode.

New SCXML Extension

A new **Orchestration SCXML extension**, `elasticconnector`, provides simplified access to Elasticsearch APIs which allows communication with Elasticsearch without the usage of `<session:fetch>`. You can add new properties into a session document or create custom documents right from a strategy. To use `elasticconnector` actions, configure the corresponding namespace manually in Composer.

ORS supports the following new Action elements that can be used with Composer's **SCXML State block**:

[+] createindextemplate

Name	Required	Type	Def. value	Valid values	Description
requestid	False	Location expression	none	Any valid location expression, which represents a string.	This is the location for the request ID that is returned as part of this request. Standard attribute of all Orchestration actions.
name	True	Value expression	none	Any value expression that returns a valid string.	template name
index	True	Value expression	none	Any value expression that returns a valid string.	Elasticsearch index name (pattern).
order	False	Value expression	0	Any value expression that returns a valid	Elasticsearch template order.

				integer.	
type	True	Value expression	none	Any value expression that returns a valid string.	Elasticsearch type name.
mapping	True	Value expression	none	Any valid ECMAScript object.	Object that represents the whole body of Elasticsearch property of mappings.type in the mapping request.
timeout	False	Value expression	0	Any value expression that returns a valid integer.	The integer returned must be interpreted as a time interval in milliseconds. This interval begins when action is executed. A failed and timed out fetch returns the error.elasticconnector.createindex event.

[+] createdoc

Name	Required	Type	Def. value	Valid values	Description
requestid	false	Location expression	none	Any valid location expression which represents a string.	This is the location for the request ID that is returned as part of this request. Standard attribute of all Orchestration actions.
id	false	Value expression	none	Any value expression that returns a valid string	Elasticsearch document within an index. If not specified, ID will be automatically generated. Explicit ID assignment and automatic ID generation cannot be mixed within same index.
index	true	Value expression	none	Any value expression that returns a valid string.	Elasticsearch index name
type	true	Value expression	none	Any value expression that returns a valid string.	Elasticsearch type name
request	true	Value expression	none	Any valid ECMAScript object	Object that represents the whole body of an Elasticsearch create document API request.

bulk	false	Boolean expression	false	true,false	If true, Elasticsearch request will be just added to the ORS bulk request buffer and the elasticconnector.createdoc.done event will be raised immediately. If false, a request to create document will be sent to Elasticsearch and the corresponding event and elasticconnector.createdoc.done or error.elasticconnector.createdoc will be raised after response from Elasticsearch.
timeout	false	Value expression	0	Any value expression that returns a valid integer.	The integer returned must be interpreted as a time interval in milliseconds. This interval begins when action is executed. A failed and timed out fetch returns the error.elasticconnector.createdoc event.

[+] updatedoc

Name	Required	Type	Def. value	Valid values	Description
requestid	false	Location expression	none	Any valid location expression which represents a string.	This is the location for the request ID that is returned as part of this request. Standard attribute of all Orchestration actions.
id	true	Value expression	none	Any value expression that returns a valid string.	Elasticsearch document ID within an index.
index	false	Value expression	Current session daily index.	Any value expression that returns a valid string.	Elasticsearch index name. If not specified, current session daily index will be used. Also, in this case, the type attribute must be not specified, or the action will fail.
type	false	Value expression	"session"	Any value expression that returns a valid string.	Elasticsearch type name. If not specified, the "index" attribute should be not specified as well and "session" type will be used.
request	true	Value expression	none	Any valid ECMAScript object	Object that represents the whole body of an Elasticsearch update document API request.
bulk	false	Boolean expression	true	true,false	If true, Elasticsearch request will be just added to ORS bulk request

					buffer and elasticconnector.updatedoc.done event will be raised immediately. If false, request to update document will be sent to Elasticsearch and corresponding event elasticconnector.updatedoc.done or error.elasticconnector.updatedoc will be raised after response from Elasticsearch.
timeout	false	Value expression	0	Any value expression that returns a valid integer.	The integer returned is interpreted as a time interval in milliseconds. This interval begins when the action is executed. A failed and timed out fetch returns the error.elasticconnector.updatedoc event.

[+] deletedoc

Name	Required	Type	Def. value	Valid values	Description
requestid	false	Location expression	none	Any valid location expression which represents a string.	This is the location for the request ID that is returned as part of this request. Standard attribute of all Orchestration actions.
id	true	Value expression	none	Any value expression that returns a valid string.	Elasticsearch document ID within an index.
index	true	Value expression	none	Any value expression that returns a valid string.	Elasticsearch index name
type	true	Value expression	none	Any value expression that returns a valid string.	Elasticsearch type name
bulk	true	Boolean expression	true	true,false	If true, Elasticsearch request will be added to ORS bulk request buffer and elasticconnector.deletedoc.done will be raised immediately. If false, request to delete document will be sent to Elasticsearch and corresponding elasticconnector.deletedoc.done or error.elasticconnector.deletedoc

					event will be raised after response from Elasticsearch.
timeout	false	Value expression	0	Any value expression that returns a valid integer.	The integer returned is interpreted as a time interval in milliseconds. This interval begins when the action is executed. A failed and timed out fetch returns the error.elasticconnector.deletedoc event.

Events

The following Events are supported:

[+] Events

}

Name	Attributes	Description
elasticconnector.createindextemplate.done		This event indicates the success of the request.
	requestid	This is the ID of the request.
error.elasticconnector.createindextemplate		This indicates that an error occurred while trying to perform the createindextemplate request.
	requestid	This is the ID associated with the request.
	error	This is the type of error that occurred. The following is a specific error code: protocol.errorcode - This represents the protocol-specific errors that occur when the attempting the request.
	description	This is a more detailed description of the error
elasticconnector.createdoc.done		This event indicates the success of the request.
	id	This is ID of created document.
	requestid	This is the ID of the request.

error.elasticconnector.createdoc		This indicates that an error occurred while trying to perform the createdoc request.
	requestid	This is the ID associated with the request.
	error	This is the type of error that occurred. The following is a specific error code: protocol.errorcode - This represents the protocol-specific errors that occur when the attempting the request.
	description	This is a more detailed description of the error
elasticconnector.updatedoc.done		This event indicates the success of the request.
	Requestid	This is the ID of the request.
error.elasticconnector.updatedoc		This indicates that an error occurred while trying to perform the updatedoc request.
	Requestid	This is the ID associated with the request.
	Error	This is the type of error that occurred. The following is a specific error code: protocol.errorcode - This represents the protocol-specific errors that occur when the attempting the request.
	Description	This is a more detailed description of the error
elasticconnector.deletedoc.done		This event indicates the success of the request.
	Requestid	This is the ID of the request.
error.elasticconnector.deletedoc		This indicates that an error occurred while trying to perform the deletedoc request.
	Requestid	This is the ID associated with the request.
	Error	This is the type of error that occurred. The following is a specific error code: protocol.errorcode - This represents the protocol-specific errors that occur when the attempting the request.
	Description	This is a more detailed description of the error

Configuration for Elasticsearch

To configure these enhancements in Elasticsearch, edit the `elasticsearch.yml` file:

- `es.script.engine.groovy.indexed.update: true.`

See [Configuring Elasticsearch for Orchestration](#).

Session and Performance Enhancement Limitations

Elasticsearch v 1.7.3 is not supported for these enhancements as it does not support the Date Format that ORS uses.

Kibana Visualization Samples

Starting with Release 8.1.400.58, ORS installation includes samples of Kibana dashboards and visualizations of data stored by ORS Elasticsearch Connector into Elasticsearch. ORS installation brings two sets of samples in folders located in the directory where ORS is installed:

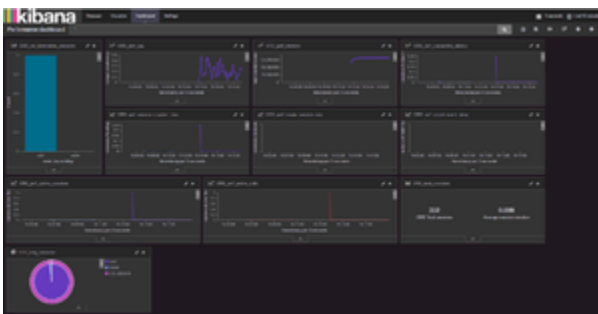
1. Composer [Project](#) sample located in `\elasticsearch\kibana_dashboard_sample\composer`. The provided Composer project contains SCXML strategies configured to store data into Elasticsearch that will be retrieved and presented in the Kibana visualizations.
2. Samples of Kibana dashboards and visualizations located in `\elasticsearch\kibana_dashboard_sample\kibana`.

To load the dashboards/visualizations, execute script `\elasticsearch\kibana_dashboard_sample\kibana`

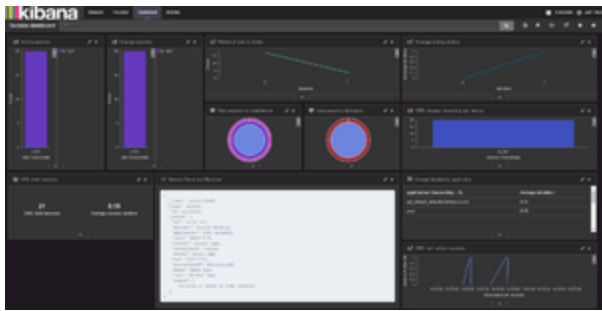
- Unix: `./load.sh -url "http://<elasticsearch host>:9200"`
- Windows: `.\load.ps1 -url "http://<elasticsearch host>:9200"`

The performance visualizations are based on predefined `<alarm-name>(s)` configured in the [performance-alarms](#) section imported from the ORS Application template.

Click to open expanded views of the samples. Click the "X" to close.



Performance Dashboard



Session Dashboard

Direct Statistic Definition

Starting with ORS 8.1.400.45, the `_genesys.statistic.sData` function is enhanced to allow you to directly define statistic parameters in the SCXML routing strategy code via a JavaScript object passed as an input parameter. This enhancement is useful when strategy logic requires statistic parameters to dynamically change.

Background: Configuration Server currently stores statistic parameters as Transaction/Statistic objects in its database. Previous to this enhancement, if a strategy requested a statistic, it could only use the `_genesys.statistic.sData` function with the name of the Transaction/Statistic object in the Configuration Database defined as a string input parameter.

Statistic Parameters in JavaScript Object

[+] Statistic Parameters in JavaScript Object

JavaScript Object Properties

```
{
    Params: {},
    Extensions: {}
}
```

Properties of "Params":

Property	StatServer API meaning	Mandatory	Type
MainActionsMask Value must be defined as a string containing Stat Server Action names, delimited by a comma. Action names are the same as used in the configuration of Statistical Types in Stat Server. Wildcard(*) and logical NOT(~) character are supported. See the <i>Stat Server 8.1 User's Guide</i> , section "Statistical Type Sections", table 9, "MainMask" and "RelMask".	SRequestParams.MainActionsMask	Yes (if not statistic of JavaCategory)	String
RelActionsMask Same as above for Value	SRequestParams.RelActionsMask	Yes	String
Category	SRequestParams.Category	Yes	String

Property	StatServer API meaning	Mandatory	Type
Values are the same as used in the configuration of Statistical Types in Stat Server. All rules, applicable for corresponding options are supported. See the <i>Stat Server 8.1 User's Guide</i> .			
Interval Contains the interval type as a string. See the <i>Stat Server 8.1 User's Guide</i> , "TimeProfiles section", table 1) - Sliding, Growing, Selection, SinceLogin.	SRequestParams.Interval	Yes	String
TimeRange2Start Starting point of time range in seconds. See the "Stat Server 8.1 User's Guide", "TimeRanges section".	SRequestParams.tmStart	Yes for ServiceFactor category, not applicable otherwise.	Number
TimeRange2End End of time range in seconds. See the <i>Stat Server 8.1 User's Guide</i> , "TimeRanges section".	SRequestParams.tmEnd	Yes for ServiceFactor category, not applicable otherwise.	Number
Subject	SRequestParams.Subject	Yes (if not statistic of JavaCategory)	String
TimeRangeStart Starting point of time range in seconds. See the <i>Stat Server 8.1 User's Guide</i> , "TimeRanges section".	SRequestParams.tmLeftRangeLimit	Yes, if category requires it. Optional otherwise.	Number
TimeRangeEnd End of time range in seconds. See the <i>Stat Server 8.1 User's Guide</i> , "TimeRanges section".	SRequestParams.tmLeftRangeLimit	Yes, if category requires it. Optional otherwise	Number

Properties of "Extensions":

Property	StatServer API meaning	Mandatory	Type
DynamicFilter Contains a filter expression. See the <i>Stat Server User's Guide</i> ,	SOpenStatEx, pExtensions->"DynamicFilter"	No	String

Property	StatServer API meaning	Mandatory	Type
"Filters section".			
DynamicTimeProfile Contains a time profile expression. See the <i>Stat Server User's Guide</i> , "TimeProfiles section".	SOpenStatEx, pExtensions->"DynamicTimeProfile"	No	String
DynamicUserDataFormula See the <i>Stat Server 8.1 User's Guide</i> , "Custom Formulas" section.	SOpenStatEx, pExtensions->"DynamicUserDataFormula"	No	String
DynamicFormula Not supported in current version of Stat Server.	SOpenStatEx, pExtensions->"DynamicFormula"	No	String
JavaStats	SOpenStatEx, pExtensions->"JavaStats".Type shall be described as JavaScript object with the structure as shown Under Type .	Yes, if category requires it. Not applicable otherwise.	{ JavaExtension: "", JavaSubCategory: "" }

Sample Statistic Definitions

[+] Sample Statistic Definitions

_genesys.statistic.sData Function with Statistic Definition:

```
<script>
    var s_Stat = {
        Params: {
            Category: "TotalNumber",
            MainActionsMask: "CallEntered",
            Subject: "DNAction",
            Interval: "Growing"
        },
        Extensions: {
            DynamicFilter: "ANI=\\"7181234567\\"
        }
    };
    var s_Val = _genesys.statistic.sData('sip_a_vq1_Switch_sip1.Q', s_Stat);
</script>
```

With Extensions:

```
{
    Params: {
```

```
    MainActionsMask: "CallAnswered",
    Category: "TotalNumber",
    Interval: "Growing",
    Subject: "DNAction"
  },
  Extensions: {
    DynamicFilter: "DNIS=\"701\""
  }
}
```

Without Extensions:

```
{
  Params: {
    MainActionsMask: "CallAnswered",
    Category: "TotalNumberInTimeRange",
    Interval: "Growing",
    Subject: "DNAction",
    TimeRangeStart: 0,
    TimeRangeEnd: 20
  }
}
```

JavaStats:

```
{
  Params: {
    Category: "JavaCategory",
    Interval: "Growing"
  },
  Extensions: {
    JavaStats: {
      JavaExtension: "eServiceContactStat.jar",
      JavaSubCategory: "age of oldest email"
    }
  }
}
```

Configuration Options

ORS options are placed in the following option folders:

- For the ORS Application,
 - In the orchestration, persistence, scxml, log, elasticsearch, and mcr sections of the Options tab of the ORS Application object's Properties dialog box (legacy method).
 - In the orchestration, persistence, scxml (except system-id option), elasticsearch, and mcr sections of the Annex tab of the Transaction List object for Enhanced Orchestration Cluster Configuration (introduced in 8.1.400.64). For more information, see [Clustering](#), Enhanced Orchestration Cluster Configuration section.
- For DN (Extension or Routing 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.

backup-synch-max-age

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `600`

Valid values: Any integer greater than `10`

Value changes: Immediately upon notification

This option specifies the maximum age (in seconds) of the data in the synchronization buffer of the primary ORS. This synchronization data is sent to the backup ORS to ensure data between the two HA servers is synchronized.

Note: Genesys recommends that you use the default values for this option.

backup-synch-max-buffer

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `100`

Valid values: Any integer greater than `100`

Value changes: Immediately upon notification

This option specifies the maximum size (in KB) of the synchronization buffer in the primary ORS. This buffer is used to hold synchronization data before it is sent to the backup ORS to ensure data between the two HA servers is synchronized.

Note: Genesys recommends that you use the default values for this option.

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`

`complete-assoc-on-target-node`

Option section: `orchestration`

Configuration object: ORS Application object

Default value: `false`

Valid values: `true`, `false`. This option must have the same value for all ORS nodes in a cluster.

Value changes: Changes to this option are applied dynamically.

Introduced in 8.1.400.53, this option improves how ORS performs a re-association of an interaction with a remote target session started on different ORS node within a cluster. It defines the way ORS handles the `interaction.added` and `interaction.present` events when associating an interaction with a remote target session running on an ORS node other than the node executing the `<ixn:associate>` action. Previously, in a multimedia deployment, ORS may wrongly distribute the `interaction.present` event to the remote target session before the interaction was pulled.

When this option is set to `true`, the remote ORS generates the `interaction.added` and `interaction.present` events after the interaction is associated with the target session. This is the proper mode for handling interactions associated with a remote target session. Genesys recommends it for new deployments, for clusters where ORS nodes can be upgraded simultaneously, or for clusters where all ORS nodes are already upgraded to version 8.1.400.53 or higher.

When this option is set to `false`:

- For voice interactions, ORS sends both the `interaction.added` and `interaction.present` events to the remote target session.
- For multimedia interactions, ORS sends the `interaction.added` event to the remote target session. The remote ORS node generates an `interaction.present` event.

Use this mode in deployments where ORS nodes in the cluster are upgraded gradually. Set the option value to `true` as soon as the upgrade of all nodes is completed.

Please note that if you want the remote target session to be able to work effectively with a re-associated interaction, be sure to explicitly place this interaction in any interaction queue that ORS works with. ORS does not automatically do this placement because it does not know strategy developer intentions and subsequent actions.

def-stat-object-type

Object: ORS Application

Option section: `orchestration`

Default value: `agent`.

Valid values:

Option Value	Description
<code>agent</code>	Agent
<code>agent_place</code>	Agent Place
<code>agent_group</code>	Group of Agents
<code>place_group</code>	Group of Places
<code>queue</code>	Queue
<code>route_point</code>	Routing Point

Value changes: Take effect immediately

This option, introduced in Release 8.1.400.17, can be used for [Direct Statistic Subscription](#). This value will be used by default in case an object type is skipped in the object description in the strategy.

def-statserver-name

Object: Name of an object specified below in Configuration Layer as described below.

Location in Configuration Layer by precedence: `Routing Point`, `T-Server`, `Tenant`, `ORS Application`

Valid value: The name of any available Stat Server

Default value: The first available Stat Server that has the “current” Tenant in its Tenants list

Value changes: In 8.1.400.17, value changes take effect immediately except at Tenant level, when value changes take effect upon restart. In 8.1.400.18, value changes at all levels (including Tenant) take effect immediately.

ORS will look up the `def-statserver-name` option in following order:

1. `Routing Point`, current for interaction/session (section `orchestration`)
2. `T-Server` (section `orchestration`)
3. `Tenant` (section `orchestration`)
4. `ORS` (section `orchestration`)

This option, introduced in Release 8.1.400.17, can be used for [Direct Statistic Subscription](#).

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.

filter-eval-expr

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true` or `false`

Value changes: Immediately upon notification

ORS 8.1.400.31 introduced option, `filter-eval-expr`, which works with `METRIC:eval_expr` and Debug Logging Segmentation (see option `log-trace-segments`). This option enables hiding of sensitive data in expression and result fields of the `eval_expr` log metric. If set to `true`, ORS verifies that string data against the regular expression defined in the `ors-regex-<name>` option.

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.

`getlistitem-binary-conversion`

This option, introduced in Release 8.1.400.15, defines how Transaction List item attributes stored in binary format will be treated by function `_genesys.session.getListItemValue(list, item)` when `item` is returned as an OBJECT of key/value pairs.

Option section: `orchestration`

Default value: `ignore`

Valid values: `ignore`, `string`, `array`

Value changes: Immediately upon notification.

When set to `ignore`, no value is returned (as if it does not exist in the List).

When set to `string`, the value is converted to string. If there are bytes with a zero value, then the string will contain bytes only until the first zero byte.

When set to `array`, each byte is converted to element of array. Previously, binary attributes were converted to array.

Also, if Transaction List item attributes are stored in non-string format and the `_genesys.session.getListItemValue(list, item, key)` function retrieves the value of a particular key, it is returned as string. Previously, in this scenario, the `_genesys.session.getListItemValue()` function ignored non-string attributes.

`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-session-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.

ixnfm-idle-session-ttl

Option section: `orchestration` (in Application) or Application (in Enhanced Routing Script)

Configuration Object: Enhanced Routing Script, ORS Application

Default value: 0

Valid values: Any non-negative integer between 0 and 3600.

Value changes: Immediately upon notification.

Defines the maximum time (in seconds) that an SCXML session may exist only if interactions have been successfully redirected and there are no other interactions subsequently attached to that session. If configured in Enhanced Routing Script, the value will override value from the ORS Application for sessions created from that Script. Value 0 disables this feature.

Use to eliminate the possibility of the scenario where an SCXML session gets "stuck" in memory caused by an unsuccessful detach action execution and/or incorrect strategy design. This scenario can prevent the creation of new sessions if the number of sessions reaches the limit of concurrent active sessions per ORS Application.

log-trace-segments

Option section: `orchestration`

Default value: `all`

Valid values: Any combination of these comma-separated segment names (case-sensitive):

`OrsInternal, OrsInit, OrsSynch, Cluster, ItxLink, ItxManager, CallMonitor, ORSURS, Persistence, Schedule, SessionManager, ThreadSync, ScxmlCommon, ScxmlSession, ScxmlMetric, ScxmlQOS, ORSScxml, ScxmlConfig, ScxmlIxnAction, ScxmlOrsAction, ScxmlDFM, ScxmlEvent, ScxmlClassification, ScxmlFMDialog, ScxmlFMInteraction, ScxmlFMQueue, ScxmlFMResource, ScxmlFMSession, ScxmlFMStat, ScxmlFMWeb, ScxmlProperty, ScxmlIO, SelfMonitoring, ScxmlMetricEvalExpr, ElasticConnector, ScxmlCacheStats`

Value changes: Immediately upon notification.

Use this option for Debug Log Segmentation.

Debug Log Segmentation

Starting with Release 8.1.300.52, ORS supports Debug Log Segmentation, which provides more precise control of the logging when the `log-trace-segments` option is configured. The debug segment header `{<segment_name>:<log_level>}` is now added right after the timestamp. For example: `10:31:02.249 {ScxmlMetric:3} METRIC <event_queued sid='ML06K6519D5UPAGSHNE41VJBSS00000M' name='interaction.added' type='external' thread='15800' />`

The `log-trace-segments` option, using a comma separated list, specifies the types of information that are written to the Debug log files. Setting this option logs all messages that fall into the specified

segment or list of segments. This option works with the `x-server-trace-level` option, which must be set to 1, 2 or 3.

Background: Orchestration Server produces a large number of Debug level log messages. The sheer volume of log messages can make it difficult to isolate the specific messages needed to troubleshoot different types of deployments. The ORS Debug Log Segmentation feature groups log messages by functional segments, such as common module, API, or feature group. Segment names are added at the beginning of each log message.

- Use the `!` character before a segment to exclude messages falling under this segment.
- The `all` keyword lists all segments (used as regular segment). All debug messages are printed with the log level defined by the `log/x-server-trace-level` option.

To control the log levels more selectively for each segment, use this syntax:
`<segment_name>:<log_level>`.

- If no log level is specified, the default value (`x-server-trace-level`) is applied.
- The log level specified with the excluded segment (with `!` prefix) will be ignored.

In the case of a syntax error, the default value will be applied.

Important

If an unknown segment is specified, ORS ignores it without generating an error.

Segment Descriptions

N	Segment Name	Functionality
1.	OrsInternal	Processing of ORS internal objects and events
2.	OrsInit	ORS instance initialization
3.	OrsSynch	Synchronization between primary and backup instances
4.	ScxmlQOS	Message for SCXML engine load/session creation time upon each session start
5.	Cluster	Messages related to ORS node initialization within ORS cluster (cluster assignment, cluster configuration, etc.)
6.	AgentFM	Reserved
7.	ItxManager	Low-level messages for eServices communication
8.	ItxLink	Low-level messages for eServices communication
9.	CallMonitor	Call/Interaction processing events/requests
10.	AgentMonitor	Reserved
11.	ORSURS	Communication with URS
12.	Persistence	Operations with Cassandra persistent storage
13.	Schedule	Scheduling via persistence layer
14.	SessionManager	Session management-related messages
15.	ThreadSync	Internal inter-thread communication
16.	ScxmlCommon	Common operations within SCXML engine (such as HTTP doc requests preparation and execution, caching, and so on)
17.	ScxmlSession	Session-related non-metric SCXML messages
18.	ScxmlMetric	SCXML METRIC messages
19.	ORSScxml	ORS core messages about SCXML sessions created/terminated
20.	ScxmlConfig	Configuration options used by SCXML engine
21.	ScxmlIxnAction	Messages related to SCXML actions execution
22.	ScxmlOrsAction	Messages related to internal processing of SCXML action data

```
(validation, compilation)
23.ScxmlDFM           Messages from Distributed Function Module
24.ScxmlEvent         Messages about low-level SCXML event generation/processing
25.ScxmlClassification Messages from Classification Function Module
26.ScxmlFMDialog      Messages from Dialog Function Module
27.ScxmlFMInteraction Messages from Interaction Function Module
28.ScxmlFMQueue       Messages from Queue Function Module
29.ScxmlFMResource    Messages from Resource Function Module
30.ScxmlFMSession     Messages from Session Function Module
31.ScxmlFMStat        Messages from Statistics Function Module
32.ScxmlFMWeb         Messages from Web Function Module
33.ScxmlProperty      Messages about low-level SCXML object properties processing
34.ScxmlIO            Messages from SCXML engine internal fetching module (low-level HTTP
doc operations)
35.SelfMonitoring    Messages related to Performance Monitoring
36.ScxmlMetricEvalExpr eval_expr log metrics
37.ElasticConnector  Messages from ElasticConnector Function Module
38.ScxmlCacheStats   SCXML internal cache statistics messages
```

Examples

- `all,!0rsInternal,!ORSURS` - All segments enabled except `0rsInternal` and `ORSURS`.
- `ScxmlIO,ThreadSync` - Enable `ScxmlIO` and `ThreadSync` only.
- `all,ORSURS` - All segments enabled (the same as `all`).
- `!ORSURS` - All segments disabled (the same as empty)
- `CallMonitor,!all` - All segments disabled (the same as empty).
- `all,ORSURS:1` - All segments enabled with default trace level, but trace level for `ORSURS` segment will be changed to 1.
- `""` - All segments disabled (the same as `!all`, or `x-server-trace-level=0`).
- `all,!ORSURS:3` - All segments enabled except `ORSURS`.
- `all:3` - The same as `all` with `x-server-trace-level=3`.

Debug Log Segment Header Updates

Starting with ORS 8.1.400.31, a new Debug Log Segment header `ScxmlMetricEvalExpr` is added for `METRIC:eval_expr`. In a log file, it will appear as follows:

```
14:47:11.414 [T:8436] {ScxmlMetricEvalExpr:3} METRIC <eval_expr sid='ors2521'
expression='global._data =
new Object();' result='[object Object]' thread='8436' />
```

As a result, the `log-trace-segments` option adds `ScxmlMetricEvalExpr` as a new value. Also see `filter-eval-expr`.

Starting with ORS 8.1.400.40, in support of the [Elasticsearch Connector](#) feature, a new Debug Log Segment header, `ElasticConnector` is added. Log file example:

```
14:30:10.259 {ElasticConnector:1} Response from Elasticsearch host:port:
```

```
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-after-error-timeout

Option section: orchestration

Default value: 120 (seconds)

Valid values: Any non-negative integer from 60 to 3600

Value changes: Immediately upon notification.

This option is associated with [pulling from multiple Interaction Servers](#). It can work with `switch-multi-links-enabled`. Option `switch-multi-links-enabled` is mandatory when pulling from multiple Interaction Servers; `mcr-pull-after-error-timeout` is optional. The `mcr-pull-after-`

`error-timeout` option specifies the time interval, in seconds, when the `RequestPull` for the same strategy or view/queue will be re-sent to the same Interaction Server, if the previous request triggered a response with one of the specific errors listed below.

Interaction Server Error Messages

If Interaction Server receives `RequestPull` and some object (such as a `strategy\submitter\view\queue`) that is necessary to process `RequestPull` is not visible/accessible for this Interaction Server, it will respond with specific error(s). Depending on object, Interaction Server will respond with one of the following errors:

```
__we_error_unknown_view_in_request (42)
__we_error_view_definition_invalid (83)
__we_error_strategy_does_not_exist (107)
__we_error_strategy_has_no_views (108)
__we_error_strategy_is_disabled (109)
```

If ORS receives such an error from an Interaction Server, it waits for the specified `mcr-pull-after-error-timeout` timeout before repeating the `RequestPull`. ORS then sends all subsequent interaction-related requests to the same Interaction Server that this interaction was pulled from. Consult the [eServices](#) documentation for configuration information in these cases.

`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-cycle-quota`

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `100`

Valid values: Any non-negative integer less than 50000

Value changes: Immediately upon notification

This option, introduced in 8.1.400.11, allows you to define how many multimedia interactions ORS will request to be pulled from queues in a given pulling cycle during enhanced pulling. This number is divided equally and rounded up to the nearest integer among the routing strategies that have no outstanding pull request. If there is an outstanding pull request for a strategy, then no pull request will be sent and this strategy will not be used in the calculation of the per-strategy quota for this particular pulling cycle.

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 50000

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. This option is not applicable to the [enhanced pulling of multimedia interactions](#) feature. The [eServices](#) options control the number of pulled interactions for enhanced pulling.

mcr-queue-on-fails

Option section: orchestration

Configuration object: ORS Application object

Default value: An empty string

Valid values: Name of a valid interaction queue or an empty string

Value changes: Immediately

This option specifies the interaction queue into which an interaction is placed if session creation for the multimedia interaction fails, instead of returning it to the same interaction queue. The feature is enabled if the option value is not an empty string.

`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`

`same-node-for-cons-prim-calls`

Option section: `orchestration`

Configuration object: ORS Application object

Default value: `false`

Valid values: `true` or `false`

Value changes: Changes to this option will take effect immediately, but only for consult calls created after that change.

For compatibility with ORS 8.1.3, Release 8.1.400.33 introduced the `same-node-for-cons-prim-calls` option. Its purpose is to control call distribution compatibility between ORS nodes when a deployment contains nodes from both 8.1.3 and 8.1.4 releases.

When set to `false`, ORS 8.1.4 and 8.1.3 call distribution works the same: ORS determines the ownership of consult calls in the same way as it determines the ownership of any other call. The only exceptions are ORS versions 8.1.400.30 through 8.1.400.32, where compatibility with 8.1.3 was broken. When set to `true`, consult calls will be handled by the same node that handled the related primary call.

`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 55555555).

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='31RJ0AHJQP3851BFQ6Q5JR0LA4000001' expr='Custom Log Level 45'
label='Test example' level='45' />
Int 23022 METRIC <log sid='31RJ0AHJQP3851BFQ6Q5JR0LA4000001' 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='31RJ0AHJQP3851BFQ6Q5JR0LA4000001' expr='Custom Log Level 45'
label='Test example' level='45' />
Std 23011 METRIC <log sid='31RJ0AHJQP3851BFQ6Q5JR0LA4000001' 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='31RJ0AHJQP3851BFQ6Q5JR0LA4000001' expr='Custom Log Level 45'
label='Test example' level='45' />
Std 23011 METRIC <log sid='31RJ0AHJQP3851BFQ6Q5JR0LA4000001' 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.

statsserver-source

Object: Transaction of Statistic type

Option section: `orchestration`

Default value: `False`

Valid values: `False`, `True`

Value changes: Upon ORS restart.

This option, introduced in Release 8.1.400.17, defines the source for the statistical data in the Statistic configuration. For more information, see [Direct Statistic Subscription](#).

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 can be used as follows:

- It 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.
- This option must be set to `true` to enable support for [pulling from multiple Interaction Servers](#). Note: For multiple Interaction Server support, it is not necessary to set this option on the load sharing Switch object as described below.

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+.

`statsserver-source`

Object: Transaction of `Statistic` type

Option section: `orchestration`

Default value: `False`

Valid values: `False`, `True`

Value changes: Immediately upon notification.

Starting with Release 8.1.400.17, ORS can now request data directly from Stat Server instead of going through Universal Routing Server. This option defines the source for the statistical data in the `Statistic` configuration. For more information, see [Requesting data directly from Stat Server](#).

`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 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`.

`print-cache-stats-interval`

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `60 (seconds)`

Valid values: Any positive integer greater than 10

Value changes: `Immediately upon notification`

This option specifies the time interval, in seconds, for logging level 3 SCXML internal cache statistics messages in the `ScxmlCacheStats` log segment.

`sessionid-with-nodeid`

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: false

Valid values: true, false

Value changes: After ORS restart

This option is used to enable the new session ID generation scheme, introduced in 8.1.400.70. The scheme minimizes Cassandra lookups to determine node/session relations wherever possible. Custom session IDs are not affected. ORS works with old style session IDs, even if the new ID generation scheme is enabled. Different nodes and/or ORS applications in a cluster can use different ID generation schemes. Introduced in 8.1.400.70.

act-on-behalf-of

Option section: orchestration

Configuration object: ORS Application object

Default value: An empty string

Valid values: List of excluded ORS Node IDs separated by a comma or semicolon

Value changes: Immediately (this is a dynamic option)

Introduced in 8.1.400.70, this option is used to designate an existing node in the ORS cluster to act on behalf of an excluded node. It is the user's responsibility to ensure that only one existing node is acting on behalf of any excluded node. This option must be used only in unavoidable exceptional situations.

multivalued-prop-in-udata

Option section: orchestration

Configuration object: ORS Application object or Enhanced Routing Script object

Default value: false

Valid values: true, false

Value changes: Immediately; new value will take effect for sessions created after option change. When configured in an Enhanced Routing Script object, that option value takes precedence over the option configured in the ORS Application object.

Introduced in 8.1.400.70, this option is used to allow presentation of all values of key-value pairs having the same key name in an interaction's user data. All the values are provided in the form of an array ordered the same way the values were presented in the interaction user data. ORS does not provide a way to modify the multivalued key-value pairs in the interaction user data. But it does allow to use this form of multivalued property in other actions (for example, in the `createcall` action or the `<session:fetch method="esp">` action, where input parameters are specified as an ECMAScript object)). Using this option on an Enhanced Routing Script object may help avoid problems with SCXML strategies that expect only a single-value representation for a multivalued key.

web-start-accept-url

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `true`

Valid values: `true, false`

Value changes: Dynamically

Introduced in 8.1.400.74, this option controls if an arbitrary URL can be used as a value for the `src` parameter in a `<session:start>` action or an `/scxml/session/start` web request, when starting a new session. When this option is set to `true`, ORS allows a URL to be used as a value for the `src` parameter. When this option is set to `true`, ORS allows an URL to be used as a value for the `src` parameter. When set to `false`, ORS accepts data only in the `script:<ERScript name>` format as a value for the `src` parameter. If data is not in the specified format, an HTTP error message error is returned.

use-only-enabled-script

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true, false`

Value changes: Dynamically

Introduced in 8.1.400.79, this option controls whether or not ORS uses Enhanced Routing Script objects for session creation, based on object state. If this option is set to `true`, ORS uses only enabled Enhanced Routing Script objects. Otherwise, any Routing Script object can be used regardless of its state.

fin-acts-on-calldeleted

Option section: `orchestration`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true, false`

Value changes: Dynamically

Introduced in 8.1.400.79, this option enables ORS to finalize all pending actions associated with a call upon receiving a `CallDeleted` event after the call is disconnected. In a blind transfer scenario to Route Point, ORS now generates an `error.interaction.redirect` event after it receives an `EventError` for the `<ixn:redirect>` action applied to a consultation call which is deleted due to transfer completion. To enable ORS to generate an `error.interaction.redirect` event in such

scenarios, set this option to `true`.

esp-response-binary-conversion

Option section: `orchestration`
Configuration object: ORS Application object
Default value: `string`
Valid values: `ignore`, `string`, `array`
Value changes: Immediately

Introduced in 8.1.400.87, this option allow users to define the way binary attributes of an ESP response message are converted to the JavaScript (JS) property of the corresponding SCXML event.

- `ignore` - binary attribute value is not added as a JS property in the SCXML event;
- `string` - value is converted to string and control characters like CR, LF, and so on, are converted to escaped characters;
- `array` - each byte is converted to a numeric element of an array.

keep-original-sesid-on-switchover

Option section: `orchestration`
Configuration object: ORS Application object
Default value: `false`
Valid values: `true`, `false`
Value changes: Immediately (new value will apply only to sessions that are created after the change)

Beginning with release 8.1.400.87, this option provides a possibility to retain the original session ID for backup voice sessions which have been restarted on ORS switchover. Set this option to `true` to allow other VXML/SCXML sessions to communicate with the restarted session. The existing `restart-session-on-switchover` option must also be set to `true`.

gcti-block-redirect-trans-on-queued

Option section: `orchestration`
Configuration object: ORS Application object
Default value: `0`
Valid values: `0`, `1`
Value changes: After restart

Beginning with release 8.1.400.93, this option allows handling redirect scenarios when a call is queued at the Routing Point with `AttributeCallState 23`. Setting this option's value to `1` successfully performs a `<ixn:redirect>` of such calls. Use this option to enable or disable the special mode protecting the stuck *redirect* transactions on `EventQueued (AttributeCallState 23)`.

- `1` - allows to perform either `<ixn:redirect>` or `<queue:submit>` (with `route=true`) actions to distribute calls from the Routing Point.
- `0` - is used for backward compatibility and preserves the old ORS behavior. Only the `<queue:submit>` action (with `route=true`) action can be used to distribute calls from the Routing Point.

restart-session-on-switchover

Option section: orchestration in an ORS Application;
Application in Enhanced Routing Script

Configuration Object: Enhanced Routing Script, ORS Application

Default value: false

Valid values: true, false

Value changes: Immediately, but new value will apply only to sessions created after that change.

Introduced in 8.1.400.45. See [Recovery of Voice Calls Without Persistence](#). Set to true to have the ORS backup instance attempt to restart the processing of voice calls that have been processed by the ORS primary instance and sessions that have not been terminated at the moment of switchover.

session-restart-timeout

Option section: orchestration in an ORS Application;
Application in Enhanced Routing Script

Configuration Object: Enhanced Routing Script, ORS Application

Default value: 50

Valid values: Any integer value starting from 10

Value changes: Immediately.

Introduced in 8.1.400.45. See [Recovery of Voice Calls Without Persistence](#). Specify the timeout (msec) between cycles of session starts upon call processing recovery. The restart quota per cycle is equal to the number of engine-working threads defined with session-processing-threads option in the ORS scxml section (default = 8).

sessionfm-fetch-esp-timeout

Option section: orchestration

Configuration Object: ORS Application object

Default value: 60

Valid values: Any integer from 10 to 86400

Value changes: Immediately.

Introduced in 8.1.401.04. Use this option to specify the timeout in seconds that ORS will use when the timeout attribute is omitted in the <session:fetch> action with the esp method. Previously, for such requests ORS used a hardcoded value of 60 seconds.

user-data-for-esp-hints-tpserver

Option section: orchestration

Configuration Object: ORS Application object

Default value: false

Valid values: true, false

Value changes: Immediately upon notification from Configuration Server.

Introduced in 8.1.401.04. Use this option to ensure that an interaction's user data is added to the ESP request sent to the third-party server specified in the `tpserver` property of the `hints` object of the `<ixn:terminate>` action. Previously, the ESP request was sent without user data.

persistence Section

This section describes persistence section options.

type

Option section: persistence

Configuration object: ORS Application object

Default value: cassandra

Valid values: cassandra, redis

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

This option specifies what type of session recovery must be used. Only supported types are Cassandra and Redis Cluster. If the option value is different from `redis`, ORS will try connect to Cassandra. Changing this option after startup will not have an effect.

redis-nodes

Option section: persistence

Configuration object: ORS Application object

Default value:

Valid values: A list of host names and ports for redis nodes in the format of `<HOST>:<PORT>; [<HOST>:<PORT>...]`

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

This option provides hosts and corresponding ports when using the Redis Cluster based persistence type. This is a semi-colon separated list of host and port of Redis Cluster nodes in the format, <HOST>:<PORT>. A host can be a hostname or an IP address.

For example: `persistence/redis-nodes = rhost:6379;143.244.220.15:8301;test47:1234.`

It is sufficient to configure a single alive node for ORS to connect to all cluster nodes.

username

Option section: `persistence`

Configuration object: `ORS Application object`

Default value:

Valid values:

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

Specifies the username that is used to authenticate with Cassandra or Redis Cluster.

password

Option section: `persistence`

Configuration object: `ORS Application object`

Default value:

Valid values:

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

Specifies the password that is used to authenticate with Cassandra or Redis Cluster.

redis-connect-timeout

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `100`

Valid values: Any integer value

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

Specifies the timeout in milliseconds, that Redis++ client library will use for establishing connection. It is important to configure this timeout as a small value to avoid blocking ORS worker thread execution if Redis becomes unavailable and reconnection is attempted as part of executing the Redis request.

If set to 0, the timeout is disabled.

redis-socket-timeout

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `100`

Valid values: Any integer value

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

ORS communicates with Redis synchronously. This option defines the timeout in milliseconds, Redis++ client library will use to send read or write request to Redis. It is important to configure this timeout as a small value to avoid blocking ORS worker thread execution if Redis becomes unavailable.

Note: Actual max time for execution of Redis request can exceed this option value.

If set to `0`, request to Redis will never timeout and block ORS worker thread until completed successfully or get an error.

tls-enabled

Option section: `persistence`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true,false`

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

If set to `true`, connections to the Redis instances will be encrypted. For details regarding Redis TLS(SSL), refer to Redis documentation, <https://redis.io/docs/connect/cli/>.

Note: TLS is not supported on Windows platform.

tls-cert

Option section: `persistence`

Configuration object: `ORS Application object`

Default value:

Valid values:

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

If the option **tls-enabled** is set to `true`, you can specify in this option a file name of TLS certificate. Example: `/path/to/client/certificate`

tls-key

Option section: persistence

Configuration object: ORS Application object

Default value:

Valid values:

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

If the option **tls-enabled** is set to `true`, you can specify in this option a corresponding file name of TLS private key to the certificate configured in **tls-cert**. Example: `/path/to/private/key/file`

tls-cacert

Option section: persistence

Configuration object: ORS Application object

Default value:

Valid values:

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

If the option **tls-enabled** is set to `true`, you can configure a trusted root certificate bundle using this option. Example: `/path/to/CA/certificate/file`

tls-cacertdir

Option section: persistence

Configuration object: ORS Application object

Default value:

Valid values:

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

If the option **tls-enabled** is set to `true`, you can configure a trusted root certificate directory instead of **tls-cacert**.

tls-sni

Option section: persistence

Configuration object: ORS Application object

Default value:

Valid values:

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

If the option **tls-enabled** is set to `true`, you can specify (optional) in this option, a Server Name Indicator (SNI) value.

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: unknown

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: 100,000,000

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`

Overload Protection Logic for Large SCXML Strategies

Beginning with version 8.1.401.02, ORS offers overload protection when an assembled SCXML document's size exceeds the assembled or compiled cache entry size. This is applicable only for digital interactions and enhanced pulling.

This feature is used to slowly start loading strategies that have not been executed before (or not executed after any changes related to strategy configuration).

- This feature is applicable for all strategies (which are represented by Enhanced Routing Script objects) that are used for enhanced pulling.
- ORS creates a *pulling* set where strategies can be marked as confirmed or non-confirmed
- When a pulling set is created - all strategies are non-confirmed
- A strategy is confirmed if ORS successfully creates a session for that strategy.
- When a session is started for a strategy, the strategy is confirmed.
- When session creation fails for a strategy, the strategy is non-confirmed.
- For confirmed strategies, the ORS pull behavior is unchanged, that is, it pulls according to the cycle quota.
- For non-confirmed strategies, ORS pulls only one interaction per strategy per pull cycle.
- When an Enhanced Routing Script object representing a strategy is changed, the strategy changes to non-confirmed
- When ORS is switched from the backup mode, all strategies are in the non-confirmed state.

The following new option is introduced to control the overload protection logic:

`mcr-pull-gradual-start`

Option section: `orchestration` in an ORS Application object

Configuration object: ORS Application object

Default value: `true`

Valid values: `true`, `false`

Value changes: Immediately upon notification from the Configuration Server

When set to `true`, ORS starts pulling with only 1 interaction per cycle for the strategy, until sessions for that strategy are successfully started (that is, the strategy is confirmed). Then, ORS starts pulling for that strategy according to the cycle quota. If creation of session fails (strategy is non-confirmed), the pulling rate for this strategy drops to 1 until the strategy is confirmed again. Any changes in strategy configuration (Enhanced Routing Script) will change the status of the strategy to non-confirmed.

Controlling session lifetime functionality for deployments without Cassandra

persistence

Beginning with version 8.1.401.08, controlling session lifetime functionality is extended for deployments without Cassandra persistence. Previously, a connection to Cassandra was required for this functionality. The maximum session age in seconds is configured in the `max-session-age` option or in the `_maxtime` attribute in the `<scxml>` action element.

The following new ORS options are introduced to support this enhancement:

extend-max-session-age-action

Option section: `orchestration` in an ORS Application object

Configuration object: ORS Application object

Default value: `false`

Valid values: `true`, `false`

Value changes: Dynamically

This option is used to enable or disable session termination. It is semi-dynamic for sessions started before changing the value of the option and fully dynamic for new sessions. That is, changes take effect immediately for sessions that are started after the value of this option is updated.

max-session-age-notification-timeout

Option section: `orchestration` in an ORS Application object

Configuration object: ORS Application object

Default value: 15

Valid values: Any integer from 0 to 86400 (seconds)

Value changes: Dynamically (for new sessions when `extend-max-session-age-action` is set to `true`) This option controls when ORS notifies a session in advance that it is about to be terminated using the `session.terminating.forcedly` event by default. This option allows to specify the time in seconds before reaching max session age at which to send the event.

This option value should be less than the `max-session-age` or `_maxtime` values. Otherwise, the option is ignored (set to 0), and no notification event will be sent.

override-notification-event-name

Option section: `orchestration` in an ORS Application object

Configuration object: ORS Application object

Default value: An empty string (the default event name is used if no value is provided)

Valid values: Any notification event name

Value changes: Dynamically (for new sessions when `extend-max-session-age-action` is set to `true`)

This option is used to specify any notification event name instead of the default event. It is not recommended to override the notification event with something that starts with `error.`, because this may potentially break backward compatibility. For some specific strategies generated by recent

versions of Designer for Azure, you can override this recommendation.

If you rely on `_genesys_terminate_event` processing when Cassandra is in use, set this option to `_genesys_terminate_event`.

scxml Section

This section describes `scxml` section options.

assembled-cache-reload-threshold

Option section: `scxml`

Configuration object: ORS Application object

Default value: 45

Valid values: Any non-negative integer

Value changes: Takes effect after restart

Options `max-assembled-cache-age` and `assembled-cache-reload-threshold` allow you to fine-tune the behavior of the Assembled Document Cache. This option determines the period of time, in seconds, after which the pre-emptive re-validation of a cached entry occurs. When the age of a cached document in the Assembled Document Cache exceeds the `assembled-cache-reload-threshold`, the next request re-validates the document and updates the cache.

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.

fetch-timeout

Option section: `scxml` in an ORS Application object; Application in an Enhanced Routing Script object

Configuration object: ORS Application 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.

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`

Note: With the newer OpenSSL 3.x that comes supported with ORS 8.1.401.11+, in addition to setting the option `scxml\fips-enabled`, additional configuration steps are necessary to enable FIPS. For more information, see [Configuring FIPS for ORS on Windows](#).

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-curl-dns-cache-timeout

Introduced in version 8.1.400.80, this option defines the TTL value for entries in the internal DNS cache (in seconds). If set as 0, internal DNS caching is disabled. If set to -1, cached DNS entries will have an infinite TTL.

Option section: `scxml`

Configuration object: ORS Application object

Default value: 60

Valid values: Any positive integer or 0 or -1

Value changes: After restart

http-curl-ip-family

Introduced in version 8.1.400.49, this option is used to explicitly define the IPv4 for connections used by the ORS SCXML engine to retrieve SCXML documents from a web server.

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `<empty string>`

Valid values: `ipv4`, `ipv6`

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

Example: `scxml/http-curl-ip-family = ipv4`.

This option specifies the type of IP address to be used in the SCXML library for host name resolution upon SCXML document retrieval via HTTP protocol or execution of the SCXML action of the HTTP method (GET, POST, and so on). Use this option to eliminate any possible delays that may occur upon opening a connection to a web server if the host has enabled more than one version of IP protocol for both interfaces; such as both IPv4 and IPv6.

Starting with ORS 8.1.400.50, IPv6 is supported as a valid value.

`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 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-keepalive-idle

Option section: scxml

Configuration object: ORS Application object

Default value: 7200

Valid values: Any valid integer

Value changes: Take effect after restart.

This option, introduced in Release 8.1.400.39, defines the time between keepalive transmissions in idle condition for TCP sockets for fetching documents from an application server.

http-keepalive-intvl

Option section: scxml

Configuration object: ORS Application object

Default value: 75

Valid values: Any valid integer

Value changes: Take effect after restart.

This option, introduced in Release 8.1.400.39, defines the time interval between keepalive re-transmissions, if an acknowledgement to the previous keepalive transmission was not received.

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 value of the max-age directive of the cache-control header in HTTP requests for documents that the SCXML library retrieves from the web server upon SCXML application assembly. Also, if the response from the server does not change it, the value of that option defines the time interval for how long a cached copy of a document will be used without re-validation. However, since an SCXML application usually consists of multiple documents, that option cannot be used to change the interval until the next re-validation of the entire SCXML application. For that purpose, please use options assembled-cache-reload-threshold and max-assembled-cache-age.

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-redirects

Option section: `scxml`

Configuration object: `ORS Application object`

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-redirects = 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`

`http-verbose`

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `true`

Valid values: `true, false`

Value changes: Take effect after restart.

The option, released in 8.1.400.20, allows you to configure the amount of information written to the log when ORS reuses the connection for a new HTTP request.

`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: `ors.js`

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.

The 8.1.400.21 release changed the default value for the `js-preload-files` option to `ors.js`. The `ors.js` file must always be defined to insure successful execution of SCXML applications if the `ixnid` parameter is not explicitly provided in the request.

max-assembled-cache-age

Option section: `scxml`

Configuration object: `ORS Application. object`

Default value: `60`

Valid values: Any non-negative integer

Value changes: Takes effect after restart

Options `max-assembled-cache-age` and `assembled-cache-reload-threshold` allow you to fine-tune the behavior of the Assembled Document Cache. This option determines the expiration time, in seconds, of an entry in the Assembled Document 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-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-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: `sxml`

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: `sxml/http-max-cache-entry-size = 125000`

max-compiler-cache-size

Option section: `sxml`

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: `sxml`

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: `sxml`

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

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-script-duration`

Option section: `scxml` (in the `ORS Application object`) or `Application` (in an `Enhanced Routing Script object`)

Configuration object: `ORS Application object` and `Enhanced Routing Script object`

Default value: `2,000` milliseconds

Valid values: `500 - 10,000` milliseconds

Value changes: When configured on an `ORS Application object`, takes effect during startup. When configured on an `Enhanced Routing Script object`, takes effect for sessions that started after that update. When configured in an `Enhanced Routing Script object`, that option takes precedence over the option configured in the `ORS Application object`.

If the execution time for JavaScript within a `<Script>` element exceeds the configured limit, execution will be interrupted, an Alarm-level message will be printed in ORS log, and the `error.script` Script event will be raised for the corresponding session.

`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.

Note: Beginning with version 8.1.401.08, controlling session lifetime functionality is extended for deployments without Cassandra persistence. Previously, a connection to Cassandra was required for this functionality. Additional options, `extend-max-session-age-action`, `max-session-age-notification-timeout`, and `override-notification-event-name`, are available to support this ability. See [Controlling session lifetime functionality for deployments without Cassandra persistence](#) for option descriptions.

max-state-entry-count

Option section: `scxml`

Configuration object: ORS Application object

Default value: 500

Valid values: any integer (no maximum)

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

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

Starting with release 8.1.400.43, the default value of this option is changed from 100 to 500.

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`

Optimized Initial Loading of Large SCXML Strategies

Beginning with version 8.1.401.02, ORS is improved to optimize initial loading of large SCXML strategies and thereby enhance performance. The functionality optimizes initial loading strategies by:

- Providing an option to increase performance of retrieving SCXML strategies via HTTP, for socket and CURL buffers.
- **Implementing overload protection** logic during initial start of SCXML strategy.
- Providing an **option** that causes session creation to fail if the SCXML strategy's size exceeds the size of compiled cache.

The following three new configuration options are introduced to increase performance of retrieving SCXML strategies via HTTP. Use these options to increase the send buffer for sockets used by CURL, and the internal CURL receive buffer.

http-buffer-size

Option section: `scxml` in an ORS Application object
Configuration object: ORS Application object
Default value: 131072
Valid values: Any integer between 1024 and 524288 (`CURL_MAX_READ_SIZE`)

Value changes: After restart

This option sets the `CURLOPT_BUFFERSIZE` option that represents the read buffer size, in bytes.

curl-socket-rcv-buffer-size

Option section: `scxml` in an ORS Application object
Configuration object: ORS Application object
Default value: None
Valid values: Determined by OS; 0 indicates that the option will not be set

Value changes: After restart

This option sets the `S0_RCVBUF` option (receive buffer size) for the socket used by CURL. Option will be set only if the new value is bigger than the current socket buffer size.

curl-socket-snd-buffer-size

Option section: `scxml` in an ORS Application object
Configuration object: ORS Application object
Default value: None
Valid values: Determined by OS; 0 indicates that the option will not be set

Value changes: After restart

This option sets the `S0_SNDBUF` option (send buffer size) for the socket used by CURL. Option will be set only if the new value is bigger than current the socket buffer size.

fail-session-if-not-cached

Beginning with version 8.1.401.02, you can also now configure ORS to fail session creation when the size of an SCXML strategy exceeds the compiled cache size. The following option is introduced to facilitate this feature:

Option section: `scxml` in an ORS Application object

Configuration object: ORS Application object

Default value: `false`

Valid values: `true`, `false`

Value changes: After restart

When this option is set to `true`, session creation fails if an SCXML document cannot be placed into the compiled document cache.

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`

See [log-trace-segments](#) for information on segmenting log output.

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`

`x-print-treatment-attr`

Option section: `log`

Configuration object: `ORS Application object`

Default value: `1`

Valid values: `0, 1`

Value changes: Immediately

Beginning with release 8.1.400.93, this option enhances hiding of sensitive data in TEvent attributes. The option is used to specify if the T-lib attributes, `AttributeCollectedDigits`, `AttributeLastDigit`, `AttributeDTMFDigits`, and `AttributeTreatmentParms`, are printed in the log.

- 1 - enables printing treatment attributes to the log.
- 0 - Suppresses printing treatment attributes to the log.

log-filter-data Section

Starting with ORS Release 8.1.400.30, ORS can now selectively hide sensitive data in logs when the printing of such sensitive data is not explicitly requested by an SCXML strategy. A new option supports the hiding of sensitive data in ORSURS request and Web request logging.

`ors-regex-<name>[;<filter-type>]`

For more information on this option, see [hiding sensitive data](#).

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: `50`

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`

performance-alarms Section

This section describes `performance-alarms` section options.

`alarm-name`

The name of this option is the alarm name.

Option section: `performance-alarms`

Configuration object: `ORS Application object`

Default value: None

Valid values: Specify an alarm definition string.

Value changes: Immediately upon notification.

Starting with Release 8.1.400.21, ORS supports Performance Monitoring. You can use a set of performance-related counters and configure conditions that will trigger Management Layer alarms. This option defines the alarm name and the performance monitoring parameters. For more information, see [Performance Monitoring](#).

alarm-check-interval

Option section: performance-alarms

Configuration object: ORS Application object

Default value: 60

Valid values: Any non-negative integer from 1 to 600

Value changes: Immediately upon notification.

datastream-update-interval

Configuration object: Orchestration Server Application

Option section: performance-alarms

Default value: 1

Valid values: A number between 1 and 120

Value changes: Immediately upon notification

Introduced in Release 8.1.400.30 to support [displaying ORS Performance Data in a RTME client](#) such as Pulse. Defines the time interval in seconds between performance counters submissions from ORS to Stat Servers.

Starting with Release 8.1.400.21, ORS supports Performance Monitoring. You can use a set of performance-related counters and configure conditions that will trigger Management Layer alarms. This option defines the time interval, in seconds, to use when checking for alarm conditions. For more information, see [Performance Monitoring](#).

elasticsearch Section

See [Elasticsearch Connector](#) for more information.

ors-es-bulk-max-size

Option section: elasticsearch

Configuration object: ORS Application object

Default value: 10000

Valid values: Integer value in range from 1000 to 10000000

Value changes: Take effect immediately upon notification.

Introduced in ORS 8.1.400.40.

The word "bulk" refers to the possibility of having ORS accumulate a configurable amount reporting data and then sending the data to Elasticsearch instead of sending one reporting request at a time.

Used for both Elasticsearch session and performance reporting. This option specifies the maximum

number of actions inside a single bulk request. If reporting is enabled, ORS stores all requests in its internal bulk requests cache. This option can be set to control cache overgrowth. If the maximum number of actions in the cache is reached, ORS removes the oldest actions and inserts the newest.

Note: Genesys recommends clustering Elasticsearch nodes with several master nodes to prevent a potential loss of reporting data should an extended node disconnect occur (unless all available Master nodes in an Elasticsearch cluster are excluded).

ors-es-bulk-write-period

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: 5 seconds

Valid values: An Integer value in the range of 5 to 120 seconds.

Value changes: Take effect immediately upon notification.

Introduced in ORS 8.1.400.40.

Used for both Elasticsearch session and performance reporting.

This option specifies the maximum time interval between the sending of two bulk requests to Elasticsearch. The bulk mechanism is based on an internal buffer that stores all requests that should be sent to Elasticsearch. In cases where there are no buffered requests, the actual time between two bulk requests could exceed the `ors-es-bulk-write-period`. However, when there is a steady stream of the requests to be handled by single bulk operation, the actual time between two bulk requests will be equal to the time defined by this option.

ors-es-node-exclude-timeout

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: 600 seconds

Valid values: Integer value in range from 5 to 86400 seconds.

Valid changes: Take effect immediately upon notification.

Introduced in ORS 8.1.400.40.

After successfully connecting to an Elasticsearch node, ORS checks the node status. In case of a red status, ORS disconnects from the node and excludes the node from the reconnection procedure for the time period defined by the option `ors-es-node-exclude-timeout`.

ors-es-node-info-report

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: `false`

Valid values: `true`, `false`

Value changes: Take effect immediately upon notification.

Introduced in ORS 8.1.400.40.

This option specifies if node reporting via the Elasticsearch engine is enabled. If, upon ORS startup, `ors-es-node-info-report` has a value of `false`, and the value is subsequently changed to `true`, the report information will not be sent.

`ors-es-nodes`

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: `<empty string>`

Valid values: String containing a semicolon-separated list of `http(s)://host:port` of ElasticSearch servers.

Value changes: Take effect after restart.

Introduced in ORS 8.1.400.40.

This option is used for connectivity to Elasticsearch. It specifies the addresses of Elasticsearch nodes. Configure each ORS to work with an Elasticsearch node. For valid values:

- If the node is specified as `http://<ES host>:<port>`, then a non-encrypted connection will be used for communication with Elasticsearch.
- If the node is specified as `https://<ES host>:<port>`, then an encrypted connection will be used for communication with Elasticsearch.

`ors-es-perfsnapshot-report`

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: `false`

Valid values: `true`, `false`

Value changes: Take effect immediately upon notification.

Introduced in ORS 8.1.400.40.

This option specifies if performance reporting is enabled via Elasticsearch.

- If set to `true`, performance reporting via the Elasticsearch is enabled.
- If set to `false`, performance reporting via Elasticsearch is disabled.

ors-es-reconnect-timeout

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: 5 seconds

Valid value: 1 to 20 seconds.

Valid changes: Take effect immediately upon notification.

Introduced in ORS 8.1.400.40.

This option specifies the timeout period that ORS uses when trying to reconnect to an Elasticsearch node. If a connection to Elasticsearch is lost, then ORS will try to reconnect to the next Elasticsearch node in the `ors-es-nodes` list.

ors-es-session-report

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: `false`

Valid values: `true`, `false`

Value changes: Take effect immediately upon notification.

Introduced in ORS 8.1.400.40.

This option specifies if Elasticsearch session reporting is enabled or disabled. When set to `true`, session reporting is enabled. When set to `false`, session reporting is disabled.

If an `ors-es-session-report` value is changed from `false` to `true` in the middle of a session, ORS will send an update request for a non-existing session document. Elasticsearch will respond with a "document missing" exception, which ORS will ignore.

password

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: `<empty string>`

Valid value: String value

Valid changes: Take effect immediately upon notification.

Introduced in ORS 8.1.400.40.

This option specifies the password that should be used for basic authentication on the Elasticsearch

server.

username

Option section: `elasticsearch`

Configuration object: ORS Application object

Default value: `<empty string>`

Valid value: String value

Valid changes: Take effect immediately upon notification.

Introduced in ORS 8.1.400.40.

This option specifies the username that should be used for basic authentication on the Elasticsearch server.

scxml-doc-request-headers Section

Starting with release 8.1.400.72, this new section is used for defining arbitrary headers for the HTTP requests generated by the SCXML engine to retrieve SCXML strategy documents. Users can define additional headers in this section using the format specified below:

`<header name>`

Option section: `scxml-doc-request-headers`

Configuration object: ORS Application object or annex of Enhanced Routing Script object (if the same header is configured in both objects, the header value in the Enhanced Routing Script object takes precedence).

Default value: None

Valid values: `<header value>`

Value changes: Immediately

Important

The option name is the `<header name>` and must not contain a ':' symbol. You cannot define multiple headers with the same name. ORS does not validate the consistency of the additional header definitions; they are added to the HTTP requests as is.

gts Section

Starting with release 8.1.400.73, the following two new options have been added to support configurable DN registrations:

dn-reg-chunk-size

Option section: `gts`

Configuration object: `ORS Application` object or `Switch` object (if both objects are configured, the `Switch` object takes precedence)

Default value: 500

Valid values: 1 to 2147483647

Value changes: On next registration attempt

This option is used to define the maximum number of consecutive Register Address requests to one T-Server.

dn-reg-chunk-timeout

Option section: `gts`

Configuration object: `ORS Application` object or `Switch` object (if both objects are configured, the `Switch` object takes precedence)

Default value: 1

Valid values: 1 to 2147483647

Value changes: On next registration attempt

This option is used to define the timeout in seconds between chains of consecutive Register Address requests.

elasticsearch-request-headers Section

Starting with release 8.1.400.89, this new section is introduced to define arbitrary headers for HTTP requests to Elasticsearch (excluding those created via the `<session:fetch>` action).

Users can define the additional headers as shown below:

`<header name>`

Option section: `elasticsearch-request-headers`

Configuration object: `ORS Application` object

Default value: None

Valid values: <header value>

Value changes: Immediately

Important

- The <header name> option must not contain a : symbol.
- You cannot define multiple headers with the same name.
- ORS does not validate the consistency of the additional header definitions; they are added to the HTTP requests as is.

If the arbitrary header has the same name as a header that ORS adds by itself (for example, Connection or User-Agent), the arbitrary value takes precedence except for the following headers:

- X-Opaque-Id
- Authorization (if Elasticsearch/username configured)
- Content-Type
- Content-Length

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](#).
- For information on hiding sensitive key-value pairs in the logs see the "Client-Side Port Definition" section of the [Genesys Security Deployment Guide](#).

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:

`valid-digits`

Option section: `gts`

Configuration object: `Switch object`

Default value: `0123456789`

Valid values: Any string, which contains the set of signs/digits/characters that can be used in dialing numbers (example, `0123456789_RP0`), with a maximum length of 255 characters

Value changes: Immediately

`max-digits`

Option section: `gts`

Configuration object: `Switch object`

Default value: `25`

Valid values: Any integer from 1 to 255

Value changes: Immediately

`gcti-re-registration-tout`

Starting with 8.1.400.21, Orchestration Server now automatically performs re-registration of unregistered DNs or in scenarios where ORS receives `EventError` for `RequestRegisterAddress`. To support this functionality, you can configure the `gcti-re-registration-tout` Switch-level configuration option. The re-registration functionality is available for voice Switches only.

Option section: `gts`

Configuration object: `Switch object`

Default value: `Part_A: 0, Part_B: 10`

Valid values: `Part_A: 0`, any integer between 10 and 1800 (seconds), `Part_B: any integer between 10 and 1800 (seconds)`

Value changes: Immediately

For a complete option description, refer to the *Interaction Concentrator Deployment Guide* or the [Configuration Options](#) section on the Genesys Documentation website.

Starting with release 8.1.400.73, the following two new options have been added to support configurable DN registrations:

dn-reg-chunk-size

Option section: gts

Configuration object: ORS Application object or Switch object (if both objects are configured, the Switch object takes precedence)

Default value: 500

Valid values: 1 to 2147483647

Value changes: On next registration attempt

This option is used to define the maximum number of consecutive Register Address requests to one T-Server.

dn-reg-chunk-timeout

Option section: gts

Configuration object: ORS Application object or Switch object (if both objects are configured, the Switch object takes precedence)

Default value: 1

Valid values: 1 to 2147483647

Value changes: On next registration attempt

This option is used to define the timeout in seconds between chains of consecutive Register Address requests.

DN-Level Options

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

`application@<Orchestration cluster name>`

Option section: `Orchestration`

Configuration object: `DN (Extension or Routing Point), Interaction Queue for Legacy Pulling`

Default value: `None`

Valid values: Any valid URL

Value changes: For the next interaction that uses this resource.

- Starting with release 8.1.400.36, ORS provides the capability to define which ORS cluster will process interactions. This release introduces this functionality for deployments with Voice interactions.
- Starting with release 8.1.400.39, ORS provides this capability in a deployment with Legacy Pulling of multimedia interactions, which you configure on the Interaction Queue.

The `application@<Orchestration cluster name>` option specifies the URL of the SCXML document to load for ORS nodes that belong to a specific Orchestration cluster as defined in the option name. When this option is configured, only the ORS nodes that belong to the defined cluster will process calls. If this option is not configured, ORS will load and execute the SCXML document configured in the `application` option.

If you need to dynamically transition the processing of interactions to the new Orchestration cluster, perform following steps:

1. In Configuration Layer, modify the `Orchestration` section in the Annex of all Routing Points that `Orchestration Cluster_A` currently serves:

- Configure option `application@<Cluster_A>` with the same value specified in the existing `application` option.
- Delete the `application` option.

2. Deploy `Orchestration Cluster_B` and start the ORS nodes of `Cluster_B`.

3. In the Annex of `RoutingPoint_B` that will be served by `Orchestration Cluster_B`:

- Configure option `application@<Cluster_B>` with the same value as specified in the existing `application@<Cluster_A>` option.
- Delete the `application@<Cluster_A>` option.

Starting from this moment, only ORS nodes from `Cluster_B` will process calls arriving on `RoutingPoint_B`.

application

Option section: `Orchestration`

Configuration object: `DN (Extension or Routing Point 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.

Enhanced Routing and Interaction Submitter Script Options

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

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@I0N1=CA
```

These parameters can be utilized in a started SCXML application as a value of the href 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.

fetch-timeout

Option section: `scxml` in an ORS Application object; `Application` in an Enhanced Routing Script object

Configuration object: ORS Application 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: `G0ES/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`

`ixnfm-idle--session-ttl`

Option section: `orchestration` (in `Application`) or `application` (in `Enhanced Routing Script`)

Configuration Object: `Enhanced Routing Script`, `ORS Application`

Default value: `0`

Valid values: Any non-negative integer between 0 and 3600

Value changes: Immediately upon notification.

Defines the maximum time that an SCXML session may exist only if interactions have been successfully redirected and there are no other interactions subsequently attached to that session. If configured in `Enhanced Routing Script`, the value will override value from the `ORS Application` for sessions created from that `Script`.

Use to eliminate the possibility of the scenario where an SCXML session session gets "stuck" in memory caused by an unsuccessful detach action execution and/or incorrect strategy design. This scenario can prevent the creation of new sessions if the number of sessions reaches the limit of concurrent active sessions per `ORS Application`.

Value 0 disables this feature.

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-script-duration

Option section: `scxml` (in the ORS `Application` object) or `Application` (in an Enhanced Routing Script object)

Configuration object: `ORS Application` object and `Enhanced Routing Script` object

Default value: `2,000` milliseconds

Valid values: `500` - `10,000` milliseconds

Value changes: When configured on an `ORS Application` object, takes effect during startup. When configured on an `Enhanced Routing Script` object, takes effect for sessions that started after that update. When configured in an `Enhanced Routing Script` object, that option takes precedence over the option configured in the `ORS Application` object.

If the execution time for JavaScript within a `<Script>` element exceeds the configured limit, execution will be interrupted, an Alarm-level message will be printed in ORS log, and the `error.script` Script event will be raised for the corresponding session.

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
```

Interaction Submitter Script Options

This section describes `Interaction Submitter Script` object options.

Orchestration Section

`cluster-id`

Option section: Orchestration in Annex of the `Interaction Submitter Script` object

Configuration object: `Interaction Submitter Script` object

Default value: None

Valid values: List of ORS clusters separated by `' ; '`

Value changes: Immediately upon notification. Applicable to subsequent pull requests.

Starting with release 8.1.400.36, ORS provides the capability to define which ORS cluster will process interactions. For deployments with **Enhanced Pulling** of multimedia interactions from interaction queues, you configure the `cluster-id` option.

When option `cluster-id` is configured, it restricts usage of the Submitter to only the ORS nodes that belong to the clusters listed in this option. If this option is not configured in a Submitter, then the Submitter can be used by any ORS node. Also see Submitter Objects in the Composer **Interaction Queue Block** topic.

Interacting with eServices

For an architecture diagram, see [eServices Interaction Flow](#).

Starting with ORS 8.1.400.27, in your Configuration Environment, the ORS Application, as a **client** of **eServices'** Interaction Server, must have connections to each Interaction Server Application of Interaction Server type in its **Connections** list. Genesys recommends this method for interacting with eServices/Interaction Server. In addition, this method enables ORS to **support multiple Interaction Servers**. Two **options** support this feature: **switch-multi-links-enabled** (mandatory) and **mcr-pull-after-error-timeout** (optional).

Note: Previously, the configuration for interacting with eServices/Interaction Server involved importing two templates to create two Interaction Server Application objects:

- An Interaction Server Application template for the Application object that you were using for the actual installation.
- A T-Server Application template (with its default configuration) for the second Application object.

Starting with ORS 8.1.400.27, you create the Interaction Server Application(s) using only the Interaction Server Application template. There is no need to create an Interaction Server Application using a T-Server Application Template for the second Application object. For backward compatibility, both methods of deployment are supported.

Configuring eServices Application with Type T-Server

The information below describes the Legacy method of enabling ORS to operate with **eServices** interactions by configuring two Interaction Server Application objects. The recommended method is described in [Pulling from multiple Interaction Servers](#).

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.
2. Create the Interaction Server Application objects. Ensure that both of the Interaction Server Application objects have different Application object names.
3. In the second Application object (based on the T-Server template), specify the multimedia Switch in the **Switches** tab.

Note: The first Interaction Server Application (created by using the Interaction Server Application Template) has no association in its properties with a multimedia Switch. ORS determines this association, based on the Tenant that is specified in the **General** tab of the ORS Application. Alternatively, the second Interaction Server Application (based on the 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. On the **Server Info** tab of each Interaction Server Application, configure the same port number and

ensure both Applications exist on the same Host.

5. Save the configuration.

Pulling Multimedia Interactions from Interaction Queues

Note: Composer 8.1.4 and Interaction Server 8.5.1 are required for Enhanced Pulling of multimedia interactions. Other eServices components can be 8.5.0.

Starting with 8.1.4, an ORS/Interaction Server protocol extension enhances the mechanism of getting the appropriate interactions for processing from Interaction Server to ORS. With Enhanced Pulling, the pull mechanism uses the strategy name and allows Interaction Server to set different priorities for different types of interactions (see Interaction Server Options section below). You can also use Composer's Workflow Block, Maximum Interactions property to configure the limit for each strategy. With the old pull mechanism, interactions were pulled for a specific view.

As a result of Enhanced Pulling, ORS does not need to constantly pull Interaction Server to determine if any interactions are present. This enhanced pull mechanism can result in improvements in performance and reliability for both ORS and URS.

Enhanced Pulling and Interaction Queues

As a result of Enhanced Pulling, ORS 8.1.4 supports two types of interaction queues:

1. "Enhanced" where SCXML-based applications are assigned to views instead of being assigned to interaction queues (as in previous releases) and an object called a Submitter is used for this (described below). Composer-created interaction process diagrams (associated with SCXML-based routing application) must be published into Configuration Server using Composer 8.1.4. The `Orchestration` section in the Annex of the Interaction Queue Script object is not present.
2. "Legacy" where SCXML-based applications are assigned to interaction queues and ORS 8.1.4 pulls interactions equally from all interaction queues as occurred in 8.1.3. The `Orchestration` section in the Interaction Queue Script object contains option `application` and value of that option specifies name of the Enhanced Routing Script object that handles interactions pulled from any View (filter) associated with the interaction queue. Composer-created interaction process diagrams (associated with SCXML-based applications) are published into Configuration Server using Composer 8.1.3 or earlier.

Starting with release 8.1.400.36, ORS provides the capability to define which ORS cluster will process interactions. For deployments with Enhanced Pulling of multimedia interactions, you configure the `cluster-id` option as described in Configuration Options, Interaction Submitter Script Options.

Starting with release 8.1.400.55, ORS includes additional logging for the Enhanced and Legacy pulling methods of multimedia interactions. The ORS log now prints Warning messages regarding certain pulling conditions.

Interaction Server Options

See the [eServices 8.5 Reference Manual](#) for information about required configuration options for Interaction Server.

Note: The Interaction Server pull options are in addition to the push parameters you supply in Composer's View Properties dialog box. For more information, see the Views property in Composer's [Interaction Queue block](#).

Submitters

Orchestration Server supports Submitter objects configured in Composer. In the Composer GUI, the connection between an interaction queue View and a Workflow is represented by a Submitter object. A Submitter supplies parameters that control how Interaction Server submits interactions to Orchestration Server and subsequently to Universal Routing Server. For more information, see Submitter Objects in the Composer [Interaction Queue Block](#) topic.

Pulling from Multiple Interaction Servers

Starting with Release 8.1.400.27, Orchestration Server supports the pulling of interactions from multiple [Interaction Servers](#) belonging to the same [Tenant](#) with one multimedia [Switch](#). In order to support more than one Interaction Server for the same Switch, the ORS Application level option, [switch-multi-links-enabled](#) in the orchestration section, must be set to true. This feature is supported for both [Enhanced Pulling](#) and the Legacy Pulling mechanism.

Pulling Interactions

When this feature is enabled, ORS will send pull requests to each Interaction Server on its Connections list belonging to the same Tenant. In order to limit access by any Interaction Server to some interaction Queue\View\Submitter\EnhancedRoutingScript, you must configure permissions on those objects only for the Accounts and Access Groups that were configured on the Interaction Server Application.

Important

ORS cannot pull interactions from Interaction Server if that Interaction Server has an association with more than one Tenant.

Queuing Interactions

If two Interaction Servers are configured, ORS will pull interactions from both servers and the routing strategy (workflow) will place those interactions into different queues. In this case, for a particular interaction, ORS will send RequestPlaceInQueue to same Interaction Server that it pulled that interaction from.

Routing Interactions

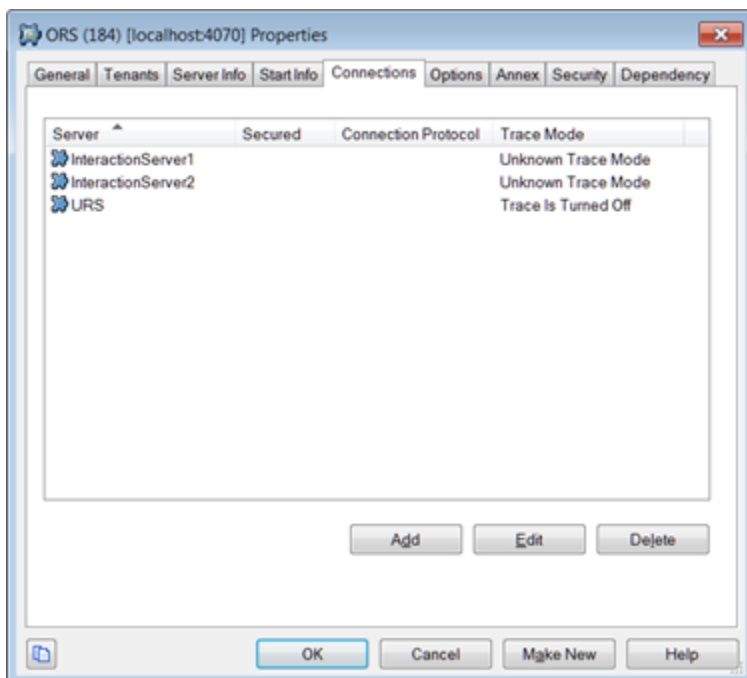
If two Interaction Servers are configured (one for e-mail, one for chat) and one set of agent targets is logged in to handle e-mail interactions, and another set of agents is logged in to handle chat

interactions, ORS pulls interactions from both Interaction Servers and the routing workflow routes those interactions to the targets. In this case, for a particular interaction, ORS will route the interaction to the target on same Interaction Server from which it pulled the interaction.

Connections to Interaction Servers

ORS can be connected to Interaction Servers in one of the following ways:

- Previous to 8.1.400.27, as described in [Configuring eServices Application with Type T-Server](#), you configured an Interaction Server Application Template and a T-Server Application Template.
- Starting with 8.1.400.27, you configure ORS to directly connect to Interaction Server Application objects.



Configuring Options

Two options are associated with this feature: `switch-multi-links-enabled` and `mcr-pull-after-error-timeout`. Option `switch-multi-links-enabled` is mandatory; `mcr-pull-after-error-timeout` is optional.

- The existing Application level option, `switch-multi-links-enabled`, must be set to `true` in the ORS Application object.
- An optional Application level option, `mcr-pull-after-error-timeout`, can be set.

mcr-pull-after-error-timeout

Option section: orchestration

Default value: 120 (seconds)

Valid values: Any non-negative integer from 60 to 3600

Value changes: Immediately upon notification.

This option specifies the time interval, in seconds, when RequestPull for the same strategy or View/Queue will be re-sent to the same Interaction Server, if the previous request triggered a response with one of the specific errors listed below.

Interaction Server Error Messages

If Interaction Server receives RequestPull and some object (such as a strategy\Submitter\View\Queue) that is necessary to process the RequestPull is not visible/accessible for this Interaction Server, it will respond with specific error(s). Depending on object, Interaction Server will respond with one of the following errors:

```
__we_error_unknown_view_in_request (42)
__we_error_view_definition_invalid (83)
__we_error_strategy_does_not_exist (107)
__we_error_strategy_has_no_views (108)
__we_error_strategy_is_disabled (109)
```

If ORS receives such an error from an Interaction Server, it waits for the specified mcr-pull-after-error-timeout before repeating the RequestPull. ORS then sends all subsequent interaction-related requests to the same Interaction Server that this interaction was pulled from. Consult the [eServices](#) documentation for configuration information in these cases.

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.

Reporting on Virtual Queues

Starting with 8.1.4, ORS/Universal Routing protocol allows attaching of Virtual Queue data to multimedia interactions. The attached user data is provided by Universal Routing Server and can then be used by a Reporting Solution, such as a Solution created with [Genesys Info Mart](#). The user data is attached to multimedia interactions and partial data is provided in virtual queue T-Library events.

The list of keys applicable to multimedia interactions includes: RVQID, RVQDBID, RTargetRule, RTargetAgentGroup, RTargetPlaceGroup, RVPQID, Reason, RTargetRuleSelected, RTargetTypeSelected, RTargetObjectSelected, RTargetAgentSelected, RTargetPlaceSelected, RTenant, RRequestedSkillCombination.

For example data, see the [Contact Server 8.1 API Reference](#), Query Interactions topic, Examples section.

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

			<p>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.</p>
<p>serviceid (required)</p>	<p>Value expression</p>	<p>Any value expression that returns a valid integer</p>	<p>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.</p>
<p>Interactionid (required)</p>	<p>Value expression</p>	<p>Any value expression that returns a valid integer</p>	<p>A value expression that returns the <code>_genesys.FMname.interactions[x].g</code> interaction ID that is associated with this request. Non voice interactions can result in the generation of an <code>error.voice.privateService</code> error.</p>
<p>resource (not required)</p>	<p>Value expression</p>	<p>Any valid string or resource object</p>	<p>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 <code>thisDN</code> parameter within the <code>TLib TPrivateService</code> method. Before you configure this value, check the T-Server documentation for your switch. Orchestration Server distributes <code>RequestPrivateService</code> to SIP Server when the resource attribute is not specified in the <code><ixn:privateService></code> action element when an inbound call diverted from a Routing Point to an external destination is answered. When</p>

			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.
<p>udata (not required)</p>	ECMAScript object	Any valid ECMAScript object	An ECMAScript object that contains the list of key/value pairs are attached to the call in question.
<p>reasons (not required)</p>	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.
<p>extentions (not required)</p>	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.
<p>hints (not required)</p>	Value expression	Any valid ECMAScript object	A value expression that returns the ECMAScript object that contains information that might

			<p>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.</p>
--	--	--	--

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'"
update = "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:

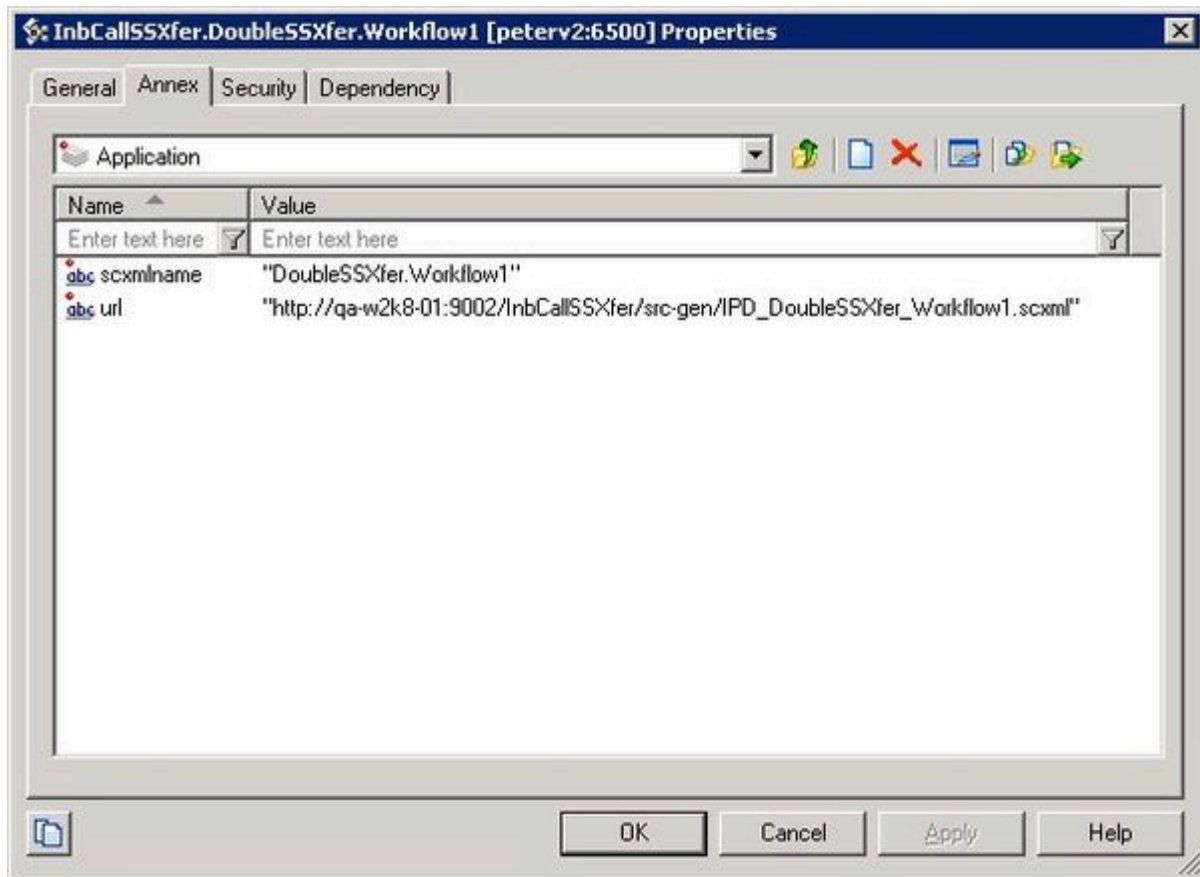
- Any simple text editor such as Notepad or an XML-based editor, with which you are already comfortable. Use the [Orchestration Server Developer's Guide](#) for the details.
- Using [Genesys Composer](#), which is an Integrated Development Environment based on Eclipse. Composer provides both drag-and-drop 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 8.1.4 Deployment Guide](#).

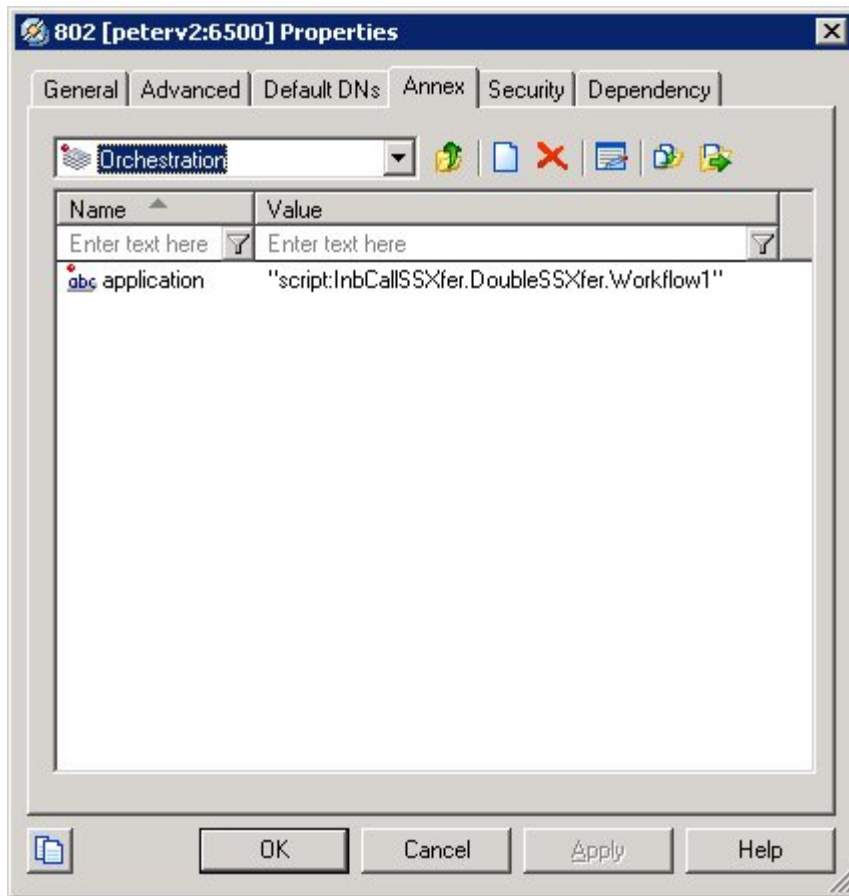
Note: You are not required to upgrade Composer when migrating from ORS 8.1.3 to ORS 8.1.4. ORS 8.1.4 is compatible with Composer 8.1.3. You only need to upgrade to Composer 8.1.4 if you are planning to use [enhanced pulling of multimedia interactions](#).

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 `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 in the Configuration Database. The figure below shows an Enhanced Routing Script object with option [url](#) set to the location of the SCXML application.



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.



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

The screenshot shows the Genesys Administrator interface. The main content area displays a table of Solutions. The table has columns for Name, Solution Type, State, and Status. The 'Status' column for all solutions is 'Stopped'.

Name	Solution Type	State	Status
EBC		Enabled	
Agent Scripting	Enterprise Ro...	Enabled	Stopped
CRM_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 Dial...	Enabled	Stopped
Proactive_Contact	Outbound Dial...	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_RP0

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 [sample 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.

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.

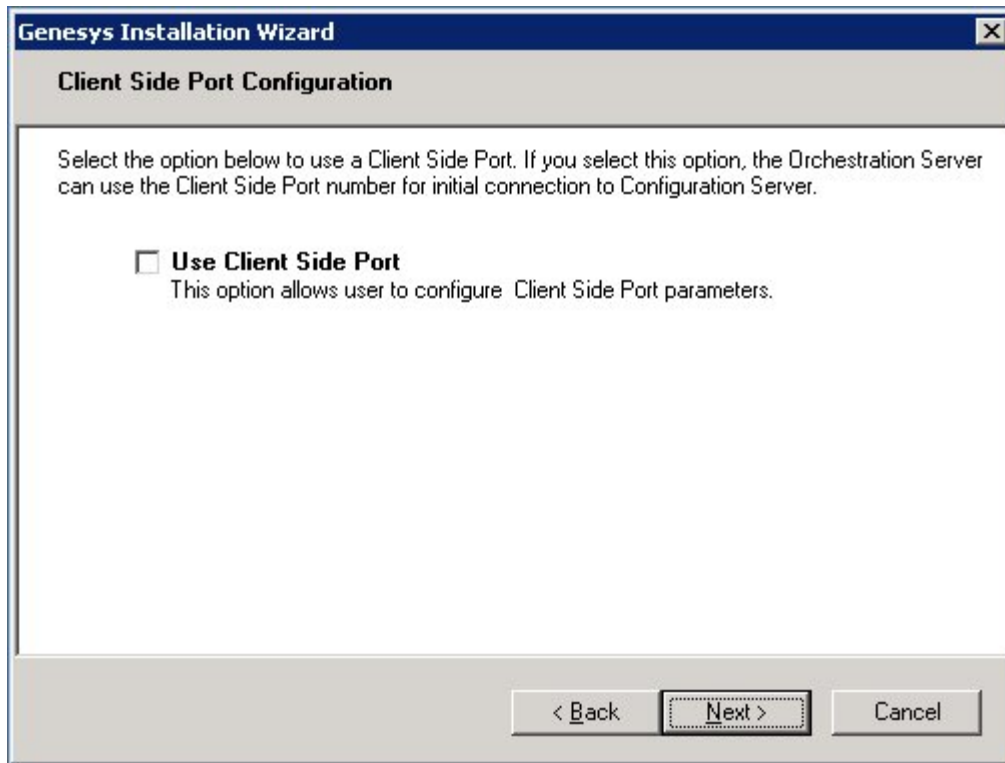
Host name: w2k3v1qa01
Port: 7771

User
Specify your Configuration Server user name and password.

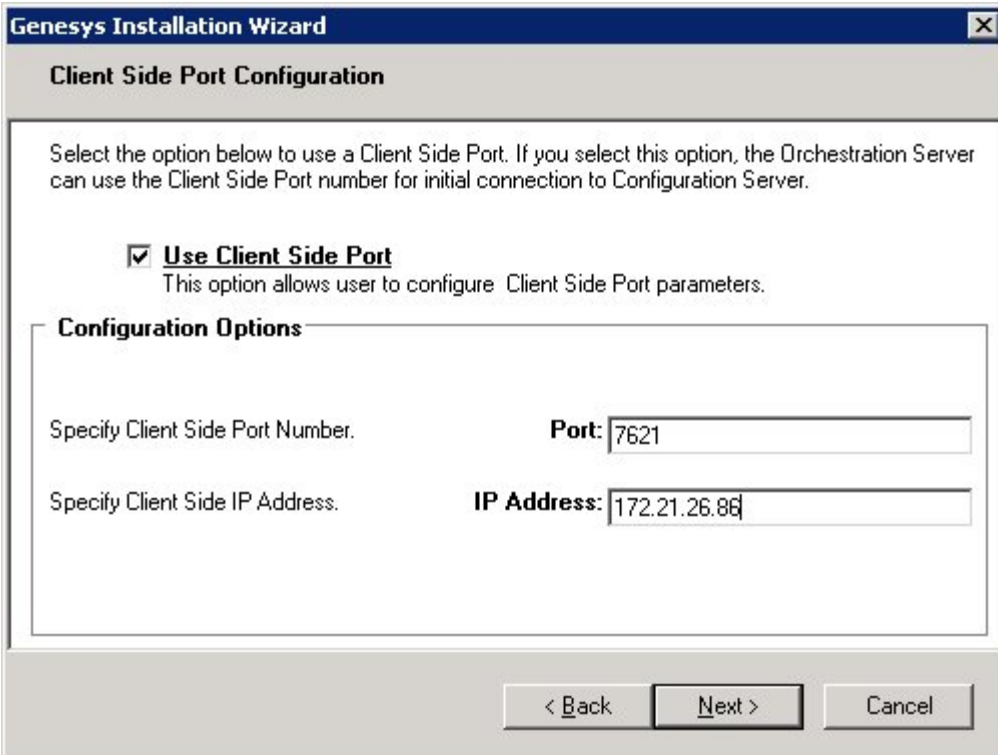
User name: default
Password: ●●●●●●●●

< Back Next > Cancel

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.



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.



The screenshot shows a dialog box titled "Genesys Installation Wizard" with a close button (X) in the top right corner. The main title is "Client Side Port Configuration". Below the title, there is a paragraph of text: "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." Below this text, there is a checked checkbox labeled "Use Client Side Port" with a sub-note: "This option allows user to configure Client Side Port parameters." Underneath, there is a section titled "Configuration Options" which contains two input fields: "Specify Client Side Port Number." with a "Port:" label and a text box containing "7621", and "Specify Client Side IP Address." with an "IP Address:" label and a text box containing "172.21.26.86". At the bottom of the dialog, there are three buttons: "< Back", "Next >", and "Cancel".

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.

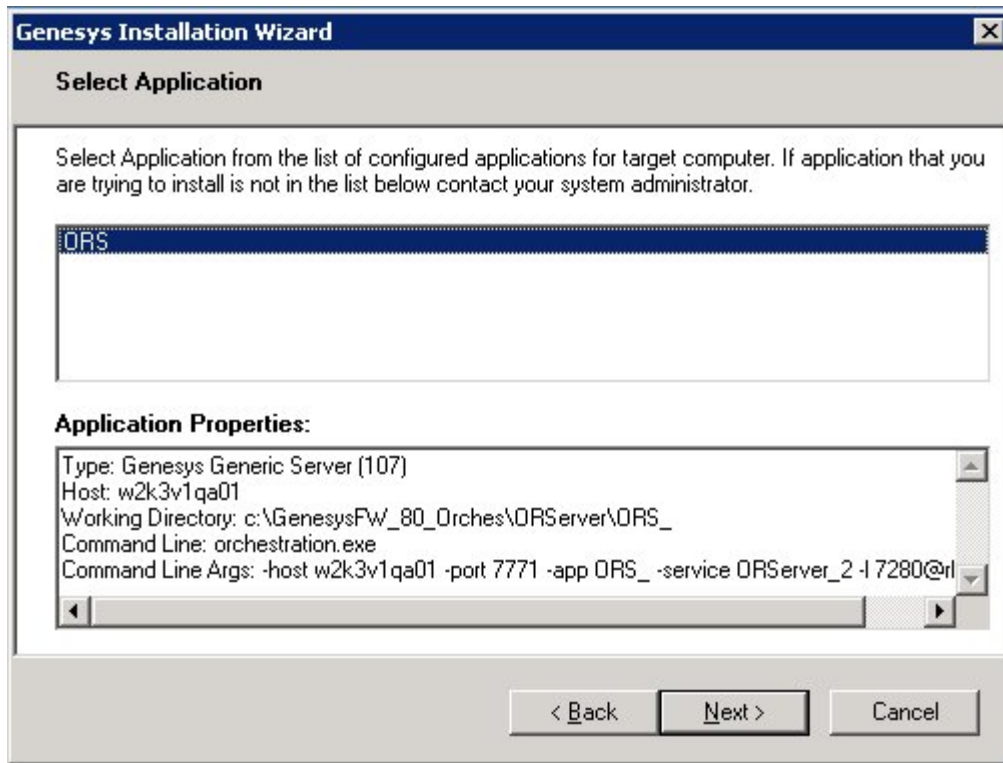
9. Click **Next**. The **Enhanced Orchestration Cluster Configuration** dialog box appears.

The screenshot shows a dialog box titled "Genesys Installation Wizard" with a sub-header "Enhanced Orchestration Cluster Configuration". The main text reads: "Select the option below to use an Enhanced Orchestration Cluster configuration. Orchestration Cluster will be provisioned via dedicated Transaction List object and allow to configure most application options in one place". Below this, there is a checked checkbox labeled "Use Enhanced Orchestration Cluster Configuration" with the subtext "This option allows user to define Enhanced Orchestration Cluster parameters". A section titled "Transaction List Object" contains a text box with the label "Specify name of the Transaction List object." and a "Name:" label followed by an empty input field. At the bottom, there are three buttons: "< Back", "Next >", and "Cancel".

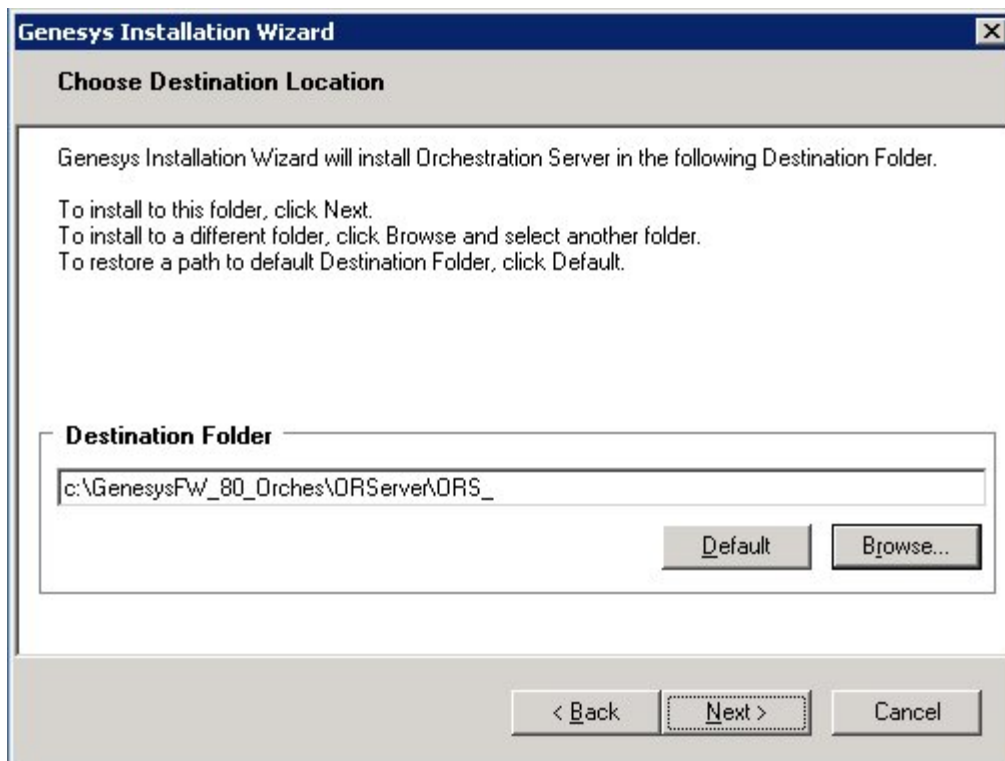
10. Specify the following parameters:

- Select the **Use Enhanced Orchestration Cluster Configuration** checkbox to allow users to define cluster configuration options at one place and reflect across nodes.
- Specify the name of the Transaction List object used to define the cluster configuration options.

11. Click **Next**. The **Select Application** dialog box appears. Example entries are shown below.



12. 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.
13. Click **Next**. The **Choose Destination Location** dialog box appears.



14. Under **Destination Folder**, keep the default or browse for the installation location for Orchestration Server.
15. Click **Next**. The **Ready to Install** dialog box appears.
16. Click **Next**. The Genesys Installation Wizard indicates it is performing the requested operation for Orchestration Server. When through, the **Installation Complete** dialog box appears.
17. 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. When prompted, type either Y for yes or N for no, to use the Enhanced Cluster Configuration feature.

Do you want to use Enhanced Orchestration Cluster Configuration?

This option will provision the Orchestration Cluster via dedicated Transaction List object. (y/n)?

10. If you had typed Y in the previous step, enter the name of the Transaction List object when prompted.

Please enter the name of Transaction List object

11. 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

Extracting tarfile: data.tar.gz to directory:

12. 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.

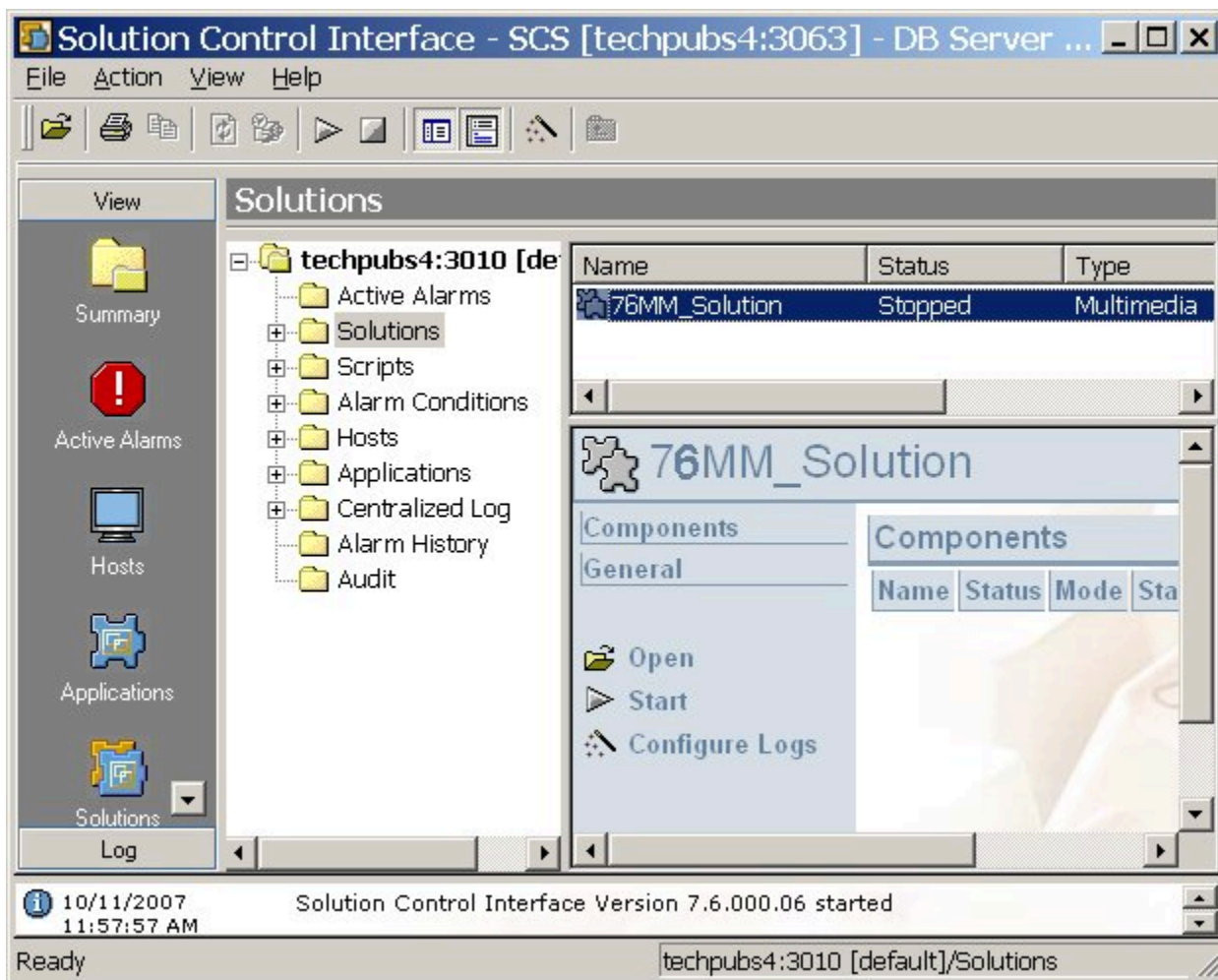
As soon as the installation process is finished, a message appears announcing that installation was successful. The process created a directory, with the name specified during the installation, containing Orchestration Server.

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.



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 running this file which contains:

```
./orchestration -host <name of Configuration Server host> -port <Configuration Server port number> -app <name of ORS Application>
```

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
```

Quotation marks are required before and after the name of the ORS application.

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:

<code>-nco xcount/xthreshold</code>	Where <code>xcount</code> (exception counts) is the number of faults allowed during a specified interval before the application exits and <code>xthreshold</code> (exception threshold) is the time interval in seconds. The values must be separated by a slash.
<code>-nco</code>	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.

Graceful Shutdown

Important

This feature is no longer available, starting with release 8.1.400.75.

Starting with release 8.1.400.64, ORS supports "graceful shutdown".

In general, the "graceful shutdown" feature of Management Framework is described in the [Management Layer User's Guide](#), Graceful Stop section. The information below describes graceful shutdown processing specific to ORS.

Processing Graceful Shutdown Signal from Local Control Agent

After a graceful shutdown signal has been received from the LCA, ORS does the following:

- Changes state to Suspending.
- Stops creating new sessions.
- Finishes processing of existing sessions.
- After all active sessions terminate, changes state to Stopped.
- If ORS receives a signal from the Local Control Agent (LCA) to switch into backup mode while it is in suspending mode, ORS processes it as it does in regular primary mode – terminates all current sessions, and so on.
- While in a suspending state, ORS might receive an "ungraceful" shutdown signal from the LCA. In this case ORS immediately stops.

Processing Incoming Interactions

To enable ORS to continue processing incoming voice interactions when a primary application of an ORS node processes a "graceful shutdown":

- The ORS cluster must consist of more than one node.
- A graceful stop must not be applied to more than one node at a time.
- When one of the ORS nodes from a cluster performs a graceful stop, it delegates creation of new sessions to the Reserved node.

Sizing Guidance

For guidance on hardware sizing refer to the [Genesys Sizing Calculator](#). The Genesys Sizing calculator helps you evaluate resource utilization based on your requirements and make the proper hardware choice.

Important

For more sizing related information and resources, refer to the [Genesys Sizing Tools System Level Guide](#). Also, refer to the **Genesys Orchestration Platform** chapter in the [Genesys Hardware Sizing Guide](#).

Orchestration Persistence - System Configuration and Storage

Orchestration Persistence is not commonly used, but this section provides you some sample guidelines to help understand the factors that must be considered in terms of system configuration and Cassandra sizing when persistence is enabled.

Consider a contact center with around 4.5 million interactions per year, 250 working days a year, and 12 business hours a day. That will approximately work out to around 25 interactions per minute. For this minimal traffic, the following basic system configuration is recommended:

- three nodes for a quorum
- each node with 8GB for the JVM
- multiple cores (2+)
- local secondary storage

Tip

It is good to have more number of cores as Cassandra is CPU intensive, especially during the compaction process.

Sample Disk Space Storage Calculation

The following disk space storage calculation table provides a sample estimate for:

- 5000 active sessions with session persistence enabled,
- average session JSON size of 45K,
- and 2 Cassandra nodes with a replication factor of 2.

NR (Number of Rows)	Assumes 5,000 active sessions	5,000	
NC (Number of Columns) - per Row	1 for Session, 2 for SessionIDServerInfo	3	
CNS (Column Name Size) - in bytes	UTF(8)	8	
CVS (Column Value Size) - in bytes	Average of 45K for a session	45,000	
RKS (Row Key Size) - in bytes		50	
RF (Replication Factor)		2	
Number of Nodes in Cluster		2	
IPS (From column_index_size_in_kb in YAML file)		64	
TNC (Total Number of Columns)	NR*NC	15,000	
Column Data (Amount of actual data in your cluster)	TNC*CVS	675,000,000	45,000 (CVS)
Column Overhead	TNC*(15+CNS)	345,000	23 (15+CNS)
Column Subtotal	Column Data+Column Overhead	675,345,000	
Row Header Overhead	NR*(23+RKS)	365,000	73 (Row Header Overhead/NR)
Row Bloom Filter	NR*((2+8)+CEILING+(((NC*4)+20/8),1))	70,000	14 (Row Bloom Filter/NR)
Row Index	(4+(IF((CEILING(((Column Subtotal/NR)/IPS),1)>1),CEILING(((Column Subtotal/NR)/IPS),1),0))*((CNS*2)+20))*NR	380,000,000	76,000 (Row Index/NR)
Row Subtotal	Row Header Overhead+Row Bloom Filter + Row Index	380,435,000	
Data File Subtotal	Column Subtotal + Row Subtotal	1,055,780,000	
Per Node	Data File Subtotal/ Number of Nodes (Includes replication and should be comparable to summation of all SSTABLES on disk for a single node)	1,055,780,000	
SSTABLE Index	NR*(10+RKS)	300,000	

SSTABLE Bloom Filter	$((NR*15)+20)/8$	9,378	
Base Storage	(Data File Subtotal+SSTABLE Index)+SSTABLE Bloom Filter	1,056,089,378	
Additional Replicas	Base Storage*(RF-1)	1,056,089,378	
Compaction Overhead	Base Storage*RF	2,112,178,755	
Total Storage	(Base Storage+Additional Replicas)+Compaction Overhead	4,224,357,510	
Per Node	Total Storage/Number of Nodes	2,112,178,755	

Important

The above table is only a sample illustration. For additional guidance specific to your requirements, contact your account representative.

For more information on configuring and deploying Cassandra on Intel Architecture, refer to this [document](#) by Intel.