



Orchestration Server 8.1

Deployment Guide

The information contained herein is proprietary and confidential and cannot be disclosed or duplicated without the prior written consent of Genesys Telecommunications Laboratories, Inc.

Copyright © 2010-2013 Genesys Telecommunications Laboratories, Inc. All rights reserved.

About Genesys

Genesys is the world's leading provider of customer service and contact center software - with more than 4,000 customers in 80 countries. Drawing on its more than 20 years of customer service innovation and experience, Genesys is uniquely positioned to help companies bring their people, insights and customer channels together to effectively drive today's customer conversation. Genesys software directs more than 100 million interactions every day, maximizing the value of customer engagement and differentiating the experience by driving personalization and multi-channel customer service - and extending customer service across the enterprise to optimize processes and the performance of customer-facing employees. Visit www.genesyslab.com for more information.

Each product has its own documentation for online viewing at the Genesys Customer Care website or on the Documentation Library DVD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Genesys and the Genesys logo are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other company names and logos may be trademarks or registered trademarks of their respective holders. © 2013 Genesys Telecommunications Laboratories, Inc. All rights reserved. The Crystal monospace font is used by permission of Software Renovation Corporation, www.SoftwareRenovation.com.

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Customer Care from Genesys

If you have purchased support directly from Genesys, please contact [Genesys Customer Care](#). Before contacting Customer Care, please refer to the [Genesys Care Program Guide](#) for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the [Genesys Licensing Guide](#).

Released by

Genesys Telecommunications Laboratories, Inc. www.genesyslab.com

Document Version: 81ors_dep_09-2013_v8.1.30F.0€



Table of Contents

List of Procedures	7
Preface	9
About Orchestration Server	9
CIM Platform	10
Intended Audience	12
Making Comments on This Document	13
Contacting Genesys Customer Care	13
Document Change History	13
New in Document Version 8.1.301.00	13
Chapter 1	Orchestration Server Overview 15
	Orchestration Server and Routing 15
	Orchestrating Customer Service 16
	SCXML and Orchestration Applications 16
	Composer and SCXML 17
	SCXML + ECMAScript 17
	Functional Modules 19
	Action Elements 20
	Supported Application Servers 20
	MIME Types 21
	Orchestration Platform Capabilities 21
	Modeling/Managing the Customer Service Process 21
	Coordinating Customer Interactions and Dialogues 22
	Orchestrating Resources and Product Services 22
	Service History Binding 23
	New Features 23
	Release 8.1.3 23
	Release 8.1.2 24
	Release 8.1.1 25
	Release 8.1.0 25
	Security 26
	Client-Side Port Definition 26

Chapter 2	Orchestration Server Architecture and Interaction Flows.....	27
	Orchestration Platform Architecture	27
	Composer	27
	Cassandra	28
	Voice Interaction Flow	29
	eServices Interaction Flow	30
	Design Principles	30
	Basic Multimedia Interaction Routing	33
	How ORS Processes eServices Interactions.....	34
	Memory Optimization for Multimedia Interactions.....	36
	Enhanced Orchestration Multi-Site Support	37
	Interactions and SCXML Sessions	37
Chapter 3	Persistence; High Availability; Load Balancing; Multiple Data Centers	
	" "	41
	Persistence and SCXML Sessions.....	41
	Overview of Persistence Design and Configuration	42
	Persistent Storage Operation.....	43
	Session High Availability.....	44
	Deploying Persistent Storage	44
	Configuring Persistent Storage	45
	Enabling Persistence in the SCXML Application	45
	High Availability/Redundancy	46
	Warm Standby	46
	ORS Node	46
	HA Deployment and Session Creation	46
	ORS Cluster Architecture	48
	Configuring a Cluster/ORS node	49
	Load Balancing Among ORS Nodes	50
	Support for Multiple Data Centers	51
	Deployment Models.....	53
	Single-Site Deployment Model	54
Chapter 4	ORS General Deployment.....	59
	Prerequisites.....	59
	Framework Components.....	59
	Orchestration Server and Local Control Agent	60
	Persistent Storage	60
	Supported Platforms	60
	Task Summary: Prerequisites for ORS deployment	60
	About Configuration Options.....	61
	ORS Deployment Tasks	62

	Configure ORS Application Object	65
	Configuring an ORS Cluster	67
	Adding ORS Application to Cluster and Data Center.....	67
	Manually Loading an SCXML application.....	68
	Configuring ORS to Interact with eServices	71
	Connection to eServices Application	71
Chapter 5	Configuration Options	75
	Options Tab	75
	ORS Application Sections.....	76
	Option List for ORS Application	77
	Option List for DNS	83
	Option List for Enhanced Routing Script Objects.....	83
	Application Level Options	84
	Orchestration Section	84
	Persistence Section	89
	scxml Section.....	93
	log Section	109
	Common Log Options.....	110
	log Section	110
	Log Output Options.....	117
	Log File Extensions	120
	mcr Section	121
	Switch Object, gts Section.....	122
	DN-Level Options	122
	Orchestration section.....	122
	Enhanced Routing Script Options	124
	Application Section	124
	{Parameter Name}.....	129
Chapter 6	Business Logic for Advice of Charge.....	131
	Implementation of Business Logic for Advice of Charge	131
Chapter 7	SCXML Application Development.....	135
	Creating SCXML-Based Applications	135
	Deployment Procedures	136
	Deploying Voice SCXML Applications	136
	Manually Deploying a Voice SCXML Application.....	139
	Debugging SCXML Applications with Composer	139
	Samples.....	140

Chapter 8	Installing Orchestration Server	141
	Installation Package Location	142
	Installing on Windows Operating Systems	142
	Installing on UNIX-Based Platforms	146
	Installing Orchestration Server on a UNIX-Based Platform	147
Chapter 9	Starting and Stopping Procedures	149
	Starting Orchestration Server	149
	Starting Orchestration Server Manually	151
	When ORS is started, a window opens and messages are sent regarding its status. ORS also establishes connections to all servers listed in the <code>Connections</code> tab of the Orchestration Server Application object.	152
	Stopping.....	152
	Non-Stop Operation.....	153
	Version Identification	154
Chapter 10	Uninstalling Orchestration Server	155
	Removing Orchestration Server with Genesys Administrator	156
	Removing Orchestration Server Manually.....	156
Appendix	Installing and Configuring Apache Cassandra Server	159
	Recommendations/Requirements	159
	Cassandra Overview	160
	Cassandra Data Model	160
	Installing, Configuring, Starting, and Testing a Cassandra Node	161
	Prerequisites	161
	Cassandra Installation	162
	Downloading Cassandra and Commons Daemon.....	162
	Editing Configuration Files	164
	Starting Up Cassandra.....	167
	Loading the Cassandra Schema	168
	Using Nodetool	168
	Troubleshooting	169
	Configuring Cassandra Server	173
Supplements	Related Documentation Resources	175
	Document Conventions	178
Index	181



List of Procedures

Cluster/ORS Node Configuration	49
Creating/Configuring the ORS Application object.	65
Configuring a Cluster	67
Manually loading an SCXML application on a DN	69
Manually loading an SCXML application on an Enhanced Routing Script	69
Configuring the ApplicationParms section of an Enhanced Routing Script object	70
Configuring eServices Application with Type T-Server	72
Deploying a Voice SCXML Application in Composer	136
Manually Deploying a Voice SCXML Application	139
Installing Orchestration Server on Windows using the Installation Wizard.	142
Installing Orchestration Server on a UNIX platform	147
Starting Orchestration Server with Solution Control Interface	150
Starting Orchestration Server on UNIX-Based Platforms.	151
Stopping Orchestration Server using Solution Control Interface	152
Removing the Orchestration Server component with Genesys Administrator	156
Manually removing Orchestration Server on Windows.	156
Manually removing Orchestration Server on UNIX-Based Platforms.	157
Downloading Cassandra and Commons Daemon	163
Editing Configuration Files	164
Edit the cassandra-env.sh and cassandra.bat configuration files	166
Edit log4j-server.properties.	166
Set up the Cassandra service (Windows)	167
Starting Up Cassandra on Linux or Windows.	167
Loading the Cassandra Schema	168
Using Nodetool.	169



Preface

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

This document is valid for the 8.1.3 and later releases of this product.

Note: For versions of this document created for other releases of this product, visit the Genesys Customer Care website, or request the Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.com.

This preface contains the following sections:

- [About Orchestration Server, page 9](#)
- [Intended Audience, page 12](#)
- [Making Comments on This Document, page 13](#)
- [Contacting Genesys Customer Care, page 13](#)
- [“Document Change History” on page 13](#)

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on [page 175](#).

About Orchestration Server

Orchestration Server takes 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.

The following example explains some of the ORS capabilities.

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. The agent fills in an application form for the customer and asks him to fax some supporting documents. 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. Within the next day or two, the customer receives an e-mail congratulating him on the approval of his application. This example, involving voice, IVR, fax, SMS, and e-mail channels, shows how ORS is able to treat all the various transactions as a single service.

Orchestration Server 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.

Orchestration Server in conjunction with Universal Routing Server (URS) can direct interactions from a wide variety of platforms, such as toll-free carrier switches, premise PBXs or CDs, IVRs, IP PBXs, e-mail servers, web servers, and workflow servers.

It 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.

For more information, see “Orchestration Platform Capabilities” on [page 21](#).

CIM Platform

ORS and URS are a part of the Genesys Customer Interaction Management (CIM) Platform, which provides the core interaction management functionality.

CIM Components

The Genesys Customer Interaction Management (CIM) 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

Figure 1 graphically depicts the CIM Platform.

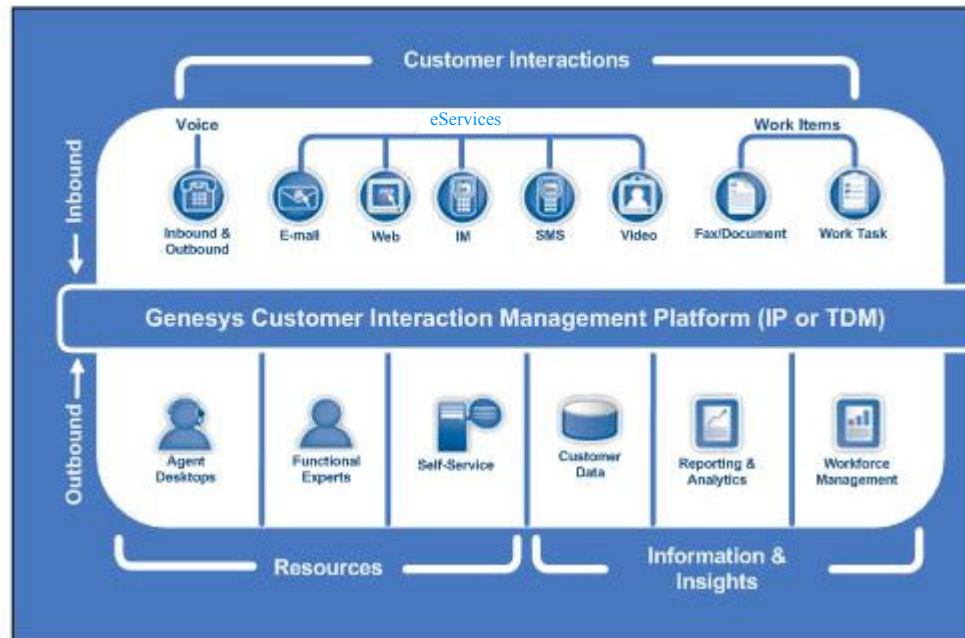


Figure 1: Customer Interaction Management Platform

Orchestration Server 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 in Figure 1, 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 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. Figure 2 provides an example.

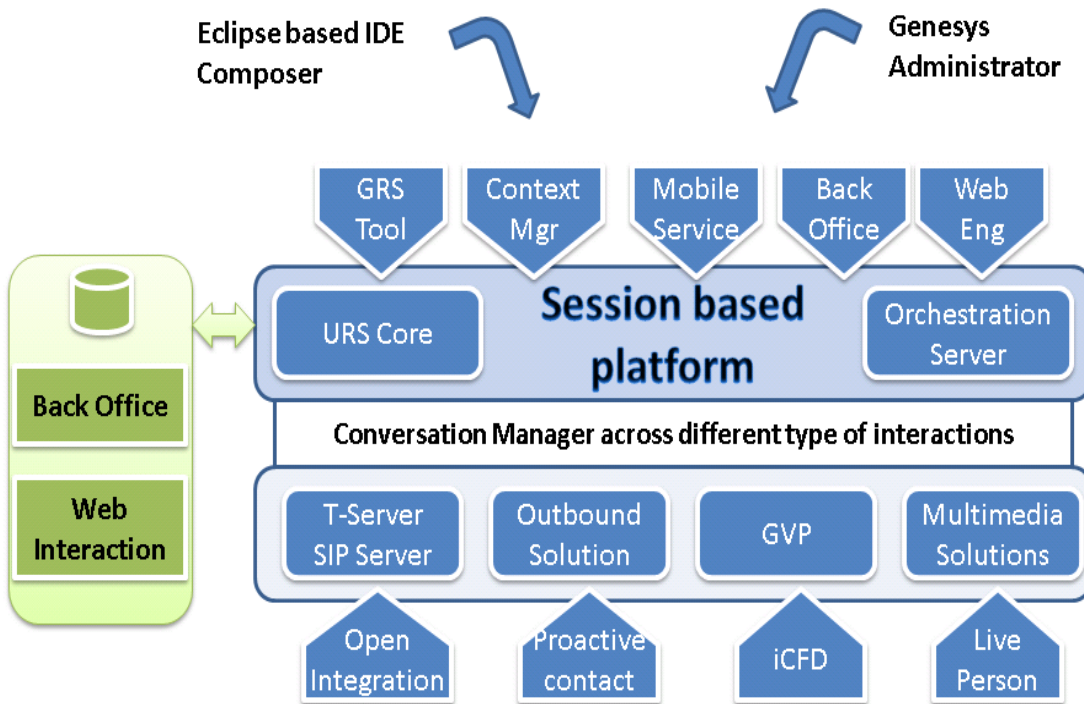


Figure 2: Example Orchestration Solution

An Orchestration solution provides the ability to create multiple application types, such as routing strategies, session logic (business processes), service logic, combinations, and more.

Intended Audience

This document is primarily intended for those involved in deploying ORS 8.1.3 with Genesys Universal Routing 8.1.3. It is written with the assumption that you have a basic understanding of:

- Computer-telephony integration (CTI) concepts, processes, terminology, and applications
- Web concepts such as “web services”
- Network design and operation
- Your own network configurations

You should also be familiar with Genesys Framework architecture and functions, eServices, and the Genesys Administrator interface and object-mapping operations.

Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to Techpubs.webadmin@genesyslab.com.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this document. Please limit your comments to the scope of this document only and to the way in which the information is presented. Contact your Genesys Account Representative or Genesys Customer Care if you have suggestions about the product itself.

When you send us comments, you grant Genesys a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

Contacting Genesys Customer Care

If you have purchased support directly from Genesys, please contact [Genesys Customer Care](#).

Before contacting Customer Care, please refer to the [Genesys Care Program Guide](#) for complete contact information and procedures.

Document Change History

ORS 8.1.3 contains architectural differences from ORS 8.1.2 (see “Release 8.1.3” on [page 23](#)). As a result, The [Orchestration Server Wiki](#) contains both an *Orchestration Server 8.1.2 Deployment Guide* and this guide, *Orchestration Server 8.1.3 Deployment Guide*.

This section lists topics that are new or have changed significantly since the 8.1.2 release of this document.

New in Document Version 8.1.301.00

Significant updates/additions from the 8.1.2 version of this guide are listed below.

- Added “MIME Types” on [page 21](#).
- Added new features in “Release 8.1.3” on [page 23](#).
- Added support for a new Cassandra version on [page 23](#).
- Removed support for Red Hat 4 in “Installing Orchestration Server on a UNIX-Based Platform” on [page 147](#).
- Updated Table 3 on [page 26](#) for client-side port definition support.

- Updated architecture diagram in Figure 3 on [page 28](#).
- Updated “Deploying Persistent Storage” on [page 44](#).
- Updated “Configuring Persistent Storage” on [page 45](#).
- Updated “High Availability/Redundancy” on [page 46](#).
- Added “HA Deployment and Session Creation” on [page 46](#).
- Added “ORS Cluster Architecture” on [page 48](#).
- Added “Support for Multiple Data Centers” on [page 51](#) to [Chapter 3](#).
- Added “Deployment Models” on [page 53](#).
- Added Table 5, “Prerequisites for ORS Deployment,” on [page 60](#).
- Transferred information to Chapter 4, “ORS General Deployment,” on [page 59](#), which was previously contained in 8.1.2 Chapter 5.
- Chapter 5 on [page 75](#) is renamed “[Configuration Options](#)”. The 8.1.2 chapter name was previously “Configuring Orchestration Server.”
- Removed general Configuration Manager procedures, which were previously contained in 8.1.2 Chapter 5 to [Task Summary: ORS Deployment Tasks](#), on [page 62](#).
- Updated [Procedure: Configuring a Cluster](#), on [page 67](#).
- Chapter 5, “[Configuration Options](#),” on [page 75](#) groups and alphabetizes options based on the following categories:
 - “Application Level Options” on [page 84](#)
 - “Common Log Options” on [page 110](#)
 - “Switch Object, gts Section” on [page 122](#)
 - “DN-Level Options” on [page 122](#)
 - “Enhanced Routing Script Options” on [page 124](#)
- Added [Procedure: Deploying a Voice SCXML Application in Composer](#), on [page 136](#).
- Updated [Procedure: Manually Deploying a Voice SCXML Application](#), on [page 139](#).



Chapter

1

Orchestration Server Overview

This chapter gives a general overview of the Orchestration Platform and Orchestration Server 8.1.3. It contains the following topics:

- [Orchestration Server and Routing, page 15](#)
- [Orchestrating Customer Service, page 16](#)
- [SCXML and Orchestration Applications, page 16](#)
- [Functional Modules, page 19](#)
- [Supported Application Servers, page 20](#)
- [Orchestration Platform Capabilities, page 21](#)
- [New Features, page 23](#)
- [Security, page 26](#)

Orchestration Server and Routing

Orchestration Server (ORS) is the session-based Genesys Routing component that takes Genesys' core capability of routing and extends, generalizes, and integrates it with other Genesys products to provide additional features and capabilities.

ORS executes routing applications that are written in SCXML (State Chart Extensible Markup Language). These SCXML-based applications can be created in Genesys Composer, are hosted on an Application Server/Web Server, and are provisioned through Genesys Administrator by defining the HTTP definition of the path and parameters.

Universal Routing Server (URS) within the Orchestration Platform, provides ORS with support for routing functions.

The ORS documentation set contains information you need in order to deploy and work with ORS and its SCXML engine.

For more information, see the [Orchestration Server](#) documentation set on the [Genesys Documentation Wiki](#).

Please refer to the [Universal Routing 8.1 Deployment Guide](#) for a thorough overview of Universal Routing capabilities, features, licensing, security, and migration issues.

Orchestrating Customer Service

Orchestration is a suite of capability using a logic integration platform. Orchestration provides the ability to process (“orchestrate”) customer service applications:

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

Orchestration provides the ability to work with multiple application types, such as routing strategies, session logic, service logic, and Service State logic of the interaction.

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

- Previous to ORS, URS executed routing strategies that were created by using the proprietary Genesys Interaction Routing Language (IRL).
- In contrast, ORS can execute routing strategies and applications that are written in a non-proprietary, open language: SCXML.

SCXML and Orchestration Applications

SCXML is an XML-based markup language which is based on Harel statecharts. The language supports parallel states, sub-state, entry/exit functions, transition conditions, and other state machine mechanisms.

It is well-suited to event-driven solutions, and can also be used for sequential workflow solutions. As such, SCXML provides the backbone logic for Orchestration applications.

The SCXML application utilizes Functional Modules (see [page 19](#)) which exist in URS, Interaction Server, and other components, that are exposed for use within the application that is driving the business logic.

Customer service occurs in a highly event-driven, asynchronous environment. From a routing-application standpoint, SCXML is ideal for this type of environment for several reasons.

- SCXML can accommodate various customer service states and the transitions between them. While relatively new as a notation/language, SCXML is well-proven for building state-based models, and facilitates the process of orchestrating customer service solutions.
- When fully implemented, the SCXML-support feature allows integration of your existing Genesys routing with other operational systems in your enterprise.

Composer and SCXML

Composer provides a rich, graphically oriented approach to creating SCXML documents which can run on the Orchestration Platform.

- It provides an integrated development environment based on [eclipse](#) for creating workflows (routing strategies) which can then be converted to SCXML.
- It has a built-in text editor for creating or modifying SCXML documents.
- Starting with Composer 8.1, you can migrate routing strategies created with Interaction Routing Designer (IRD) 8.0+ into Composer Projects as SCXML-based workflow diagrams.
- Composer-created applications can be deployed to an Application Server, which then is responsible for providing the SCXML document to ORS during execution. For testing purposes, Composer supports automated deployment of routing applications to a bundled Tomcat server or to a local IIS server.
- Composer provides real-time debugging capabilities for stepping through an SCXML document/workflow. (See “Debugging SCXML Applications with Composer” on [page 139](#).)

For more information, see:

- *Composer 8.1 Deployment Guide* and *Composer 8.1.3 Help* on the [Composer Documentation Wiki](#).
- Chapter 7, “SCXML Application Development,” on [page 135](#) in this guide.
- The *Orchestration Server Developer's Guide* on the [Orchestration Server wiki](#) for more information about SCXML.

SCXML + ECMAScript

Orchestration Server 8.1 with Universal Routing 8.1 supports SCXML plus ECMAScript as a routing language. ECMAScript is functionally equivalent to JavaScript and helps customers build applications in a natural-language construct. ORS supports SCXML in accordance with the World Wide Web Consortium (W3C) Working Draft (see [Table 1](#)).

The core SCXML provides state chart functionality. Orchestration-specific instructions are specified in the executable content of SCXML in the form of SCXML extensions (action elements) and/or ECMA script extensions (properties of special ECMA script objects).

ORS 8.1 extensions to the SCXML executable content are grouped by Functional Module as identified in the *SCXML Language Reference* available in the *Orchestration Server Developer's Guide* on the [Orchestration Server Wiki](#) (see Table 2 on [page 18](#)).

Public Reference Documents

[Table 1](#) lists publicly available reference documents for the SCXML-based routing-applications at the time of this guide's publication:

Table 1: External Reference Documents

Document Title	File Name
State Chart XML (SCXML): State Machine Notation for Control Abstraction, May 2009	http://www.w3.org/TR/2009/WD-scxml-20090507/
ECMAScript Language Specification	http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf
Standard ECMA-327	http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-327.pdf

Genesys Reference Documents

[Table 2](#) lists Genesys-supplied resource documents for the SCXML-based routing-application language:

Table 2: Genesys Reference Documents

Document Title
Genesys 8.1 SCXML Technical Reference
Genesys 8.1 SCXML Samples

Functional Modules

All Orchestration-related functionality is categorized by Functional Module as described in the *Genesys 8.1 SCXML Technical Reference* document that is listed in [Table 2](#).

The example below uses the Queue Functional Module. The namespace definition of the Queue Functional Module is the following:

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

The Queue Functional 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 Functional Modules listed below. For more information, see [Orchestration Extensions](#) and [Core Extensions](#) in the *Orchestration Developer's Guide*.

- **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 Functional Module action items as custom action elements inside SCXML executable content. All Functional 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`)

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

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

Supported Application Servers

Genesys supports the following types of Application Server software:

- Microsoft Internet Information Services (IIS), which was formerly called Internet Information Server). Genesys supports IIS 6.0–7.0.
- JBoss Application Server (or JBoss AS). This free software/open source Java EE-based Application Server is usable on any operating system that Java supports. Genesys supports version JBoss 4.0–6.0.
- IBM's Websphere Application Server (WAS). This software Application Server, built using open standards (such as Java EE, XML, and Web Services) works with a number of Web servers. Genesys supports IBM Websphere Application Server 5.1–7.0.

Note: Servlet Engines: Composer bundles Tomcat and deploys it as a Windows Service. Apache 2.0 and Tomcat 6.0 are supported.

MIME Types

MIME (Multipurpose Internet Mail Extensions) refers to a common method for transmitting non-text files via Internet e-mail. By default the SCXML MIME type is already configured in the Tomcat server bundled with Composer.

If you are using the Internet Information Services (IIS) Application Server to deploy SCXML-based applications, add the following file extensions with MIME type through the IIS Manager of your webserver:

.json	text/json
.scxml	text/plain
.xml	text/xml

Orchestration Platform Capabilities

Orchestration enables an enterprise to model customer service processes and dialogues in order to:

- Deliver a consistent and tailored customer experience across interaction channels, whenever and however the customer chooses to transact with the enterprise, and with the capability of the transaction cycle to span multiple interactions over a period of time.
- Empower the enterprise to create differentiated services and products using flexible and adaptive business rules and process flows.

Orchestration provides four main areas of capability:

- Modeling and managing the customer service process with agility.
- Coordinating customer interactions and dialog in the service process.
- “Orchestrating” resources and product services spanning the Genesys suite in real time.
- Service history binding of customer-enterprise dialogues to service-process work steps.

The following sections describe each of these capabilities.

Modeling/Managing the Customer Service Process

The Orchestration platform:

- Provides an integrated development tool, Composer, which is used to define a cohesive service process. Composer is available separately from the ORS component and is part of the Genesys CIM component.
 - The service process spans contact center and back-office touch points (IVR, e-mail, SMS, web, fax, agents, and knowledge experts in branch office) as a customer transacts with the enterprise.

- The service process provides openness for the customer to use a commercially available language to describe the service process.
- Interacts and collaborates with an ecosystem of business processes (such as CRM or BPM) in the enterprise. Data exchange and events notification trigger the next-best action to serve the customer.
- Authors and embeds business rules in the model to promote business agility and adaptability in the service process flow.
- Provides users with access to a real-time view of sessions within an orchestrated environment. Users can employ existing reporting functionality with different media.

Coordinating Customer Interactions and Dialogues

The Orchestration platform tailors customer-enterprise dialogues and corresponding message content based on contextual knowledge of the customer's service history that binds cross-channel interactions and service process work steps:

- Picks up and continues previous dialogues when the customer transacts on the same channel or switches channels in the middle of a long, live transaction.
- Proactively invokes timely and relevant notification to the customer in addition to invitations for assistance on any channel.
- Presents relevant up-sell engagement conditioning based on the customer's web behavior, qualification for marketing and customer retention programs, and recency of acceptance/rejection of a previous up-sell engagement.

Orchestrating Resources and Product Services

The Orchestration layer executes the service process flow. It coordinates customer interactions and dialogues (such as response, proactive engagement for upsell, notifications, and agent assistance) by invoking Genesys product services such as the following:

- Genesys Voice Platform (GVP) services
- Proactive contact notification services (Outbound solution)
- eServices (e-mail, SMS, and web)
- Media channel services (TDM and IP voice)

There are two kinds of proactive engagements: transition of self service to assisted service (and vice-versa) and upsell engagement.

Service History Binding

Orchestration provides Context Management Services, through Universal Contact Server, for customer-enterprise dialogues.

A new service history (as an operational data store) contextually links interactions and dialogues to service process work steps:

- to communicate to the customer regarding its service progress.
- to enhance the agent's insights into the customer's history in the context of service progress, enabling intelligent interaction with the customer.
- so that service logic can assess the service state of the customer and use that information to determine the next best steps in the service process for the customer.

Next best steps might include:

- Sending a notification to the customer.
- Scheduling an appointment to follow up.
- Offering the customer a personalized interaction dialogue on the Voice self-service channel.

New Features

This section contains a brief description of the new features in Genesys Orchestration Server 8.1.x releases.

Release 8.1.3

- Starting with release 8.1.3 ORS can operate in a high-availability (HA) environment. The move to a HA architecture implies the existence of redundant ORS applications: a Primary and a Backup (i.e., nodes). If the Primary application fails, the Backup can take over its operations without significant loss of data or impact to business operations. See Chapter 3, “Persistence; High Availability; Load Balancing; Multiple Data Centers,” on [page 41](#).
- Multiple Data Center architecture. ORS now supports a Data Center architecture where each Data Center is served by cluster of ORS nodes. ORS supports Data Center failover. See Chapter 3, “Recovery of Failed Sessions,” on [page 47](#).
- A single Cassandra instance across multiple Data Centers. The Apache Cassandra open source solution packaged with Genesys Orchestration Solution now supports a single Cassandra instance across multiple Data Centers. See “Support for Multiple Data Centers” on [page 51](#).
- Support of a new version of Cassandra. ORS 8.1.3 supports Apache Cassandra 1.1.x, beginning with Version 1.1.12.

- Enhanced voice interaction action. The `<createcall>` action with type “predictive” was extended by providing Call Progress Detection (CPD) results for voice interactions in Interaction interface events. The CPD result shows the outcome, either positive (live voice), semi-positive (answering machine, fax or silence) or negative (busy, no answer e.t.c.) of an attempt to reach an intended party. This allows ORS to determine the next actions for this interaction, such as redialing at the later time for negative call results or connecting an established outbound call to pre-recorded message instead of an agent for ‘answering machine’ call result. Refer to the *Orchestration Developer’s Guide* for more information.
- A change has been implemented in ORS behavior called *infinite loop prevention*. This modification prevents ORS from infinite loops between states or endless number of times a state is entered as a direct result of a transition element when processing an SCXML application. See the following new options:
 - “max-state-entry-count” on [page 86](#)
 - “max-microstep-count” on [page 85](#)
 - “max-pending-events” on [page 104](#)
 - “process-event-timeout” on [page 107](#)
- Enhanced multi-site support. Simplification of processing multi-site interactions by separating interactions objects from different sites between different SCXML strategies. See “Enhanced Orchestration Multi-Site Support” on [page 37](#).

Release 8.1.2

- ORS provides the possibility to apply voice treatments (announcement, collect digits, and so on) to an interaction by sending a T-LIB request to T-Server/SIP Server directly. Previously, ORS instructed URS to apply a voice treatment to an interaction.
- ORS now has features which enhance multi-site routing from an ORS cluster.
- ORS now supports a fallback URI as a failover mechanism to provision the SCXML application from a Web Server. See “alternate-url” on [page 124](#).
- This release of ORS supports multiple views of an Interaction queue. This provides users with the flexibility to branch out earlier, based on conditions up-front, before kick-starting a work flow.
- ORS supports real-time debugging capabilities for stepping through an SCXML document with Composer. See “Debugging SCXML Applications with Composer” on [page 139](#).

Release 8.1.1

- ORS supports the `PrivateServiceRequest` method of `TLib`, which enables support for the SIP Server Advice of Charge (AoC) feature. AoC enables the creation of applications that implement the business logic to insert charge messages.
- ORS now encrypts the security password by using the Secure Socket Layer (SSL) protocol.
- ORS supports Cassandra 0.7x and later versions. (Cassandra 0.6x is no longer supported in ORS 8.1.1 and later releases.) ORS 8.1 and later releases require persistence storage; the supported method is Cassandra NoSQL.
- ORS supports Service Level routing rules.
- ORS supports user name, password, and headers in the `session:fetch` action.

Release 8.1.0

- ORS supports interaction persistence through the use of Cassandra. This functionality provides uninterrupted processing of sessions regardless of ORS instance failure. Other applications within the ORS cluster will pick up the pending task as stipulated in persistence database.

Note: Cassandra is a highly scalable second generation distributed database and is part of the Apache project. Cassandra services can also be implemented in an N+1 architecture similar to ORS and is bundled on the Universal Routing CD. For more information on Cassandra, please refer to <http://cassandra.apache.org>.

- ORS now facilitates a real-time view of sessions through Genesys Administrator. Users can see all current sessions listed by cluster, Orchestration service, and so on, as well as activity for up to the previous 24 hours.
- ORS now allows the use of Statistical Functional Module functions without subscribing the required statistics beforehand within the SCXML application.
- ORS is now supported natively on the Red Hat Enterprise Linux 5.5 32-bit and 64-bit operating systems, in addition to Windows Server 2003/2008. For additional information, refer to the [Genesys Supported Operating Environment Guide](#) on the Genesys Documentation website.
- ORS now supports TCP/IP v6. TCP/IP v6 improves network security and addresses allocation and efficiency in routing traffic (not supported for Windows 2003).

Security

As described in the *Genesys 8.1 Security Deployment Guide*, 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.

Note: 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 8.0 External Authentication Reference Manual*.

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.

Table 3: Component Support for Client-Side Port Security

Clients	Config. Server/Config. Server Proxy	T-Server	URS
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*.



Chapter

2

Orchestration Server Architecture and Interaction Flows

This chapter describes the architecture and interaction flows of Orchestration Server (ORS) 8.1.3. It also describes enhancements to Orchestration Server multi-site support.

This chapter includes the following section:

- [Orchestration Platform Architecture, page 27](#)
- [Voice Interaction Flow, page 29](#)
- [eServices Interaction Flow, page 30](#)
- [Enhanced Orchestration Multi-Site Support, page 37](#)

Orchestration Platform 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 ORS services for routing.

Composer

Within the Orchestration Platform, Composer (see “Composer and SCXML” on [page 17](#)) serves as the Integrated Development Environment to create and test SCXML-based routing applications. An application server is utilized to provision SCXML routing applications to ORS. For more information, see “Composer and SCXML” on [page 17](#).

Cassandra

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

High Level Architecture

Figure 3 illustrates the high-level architecture of ORS 8.1.3.

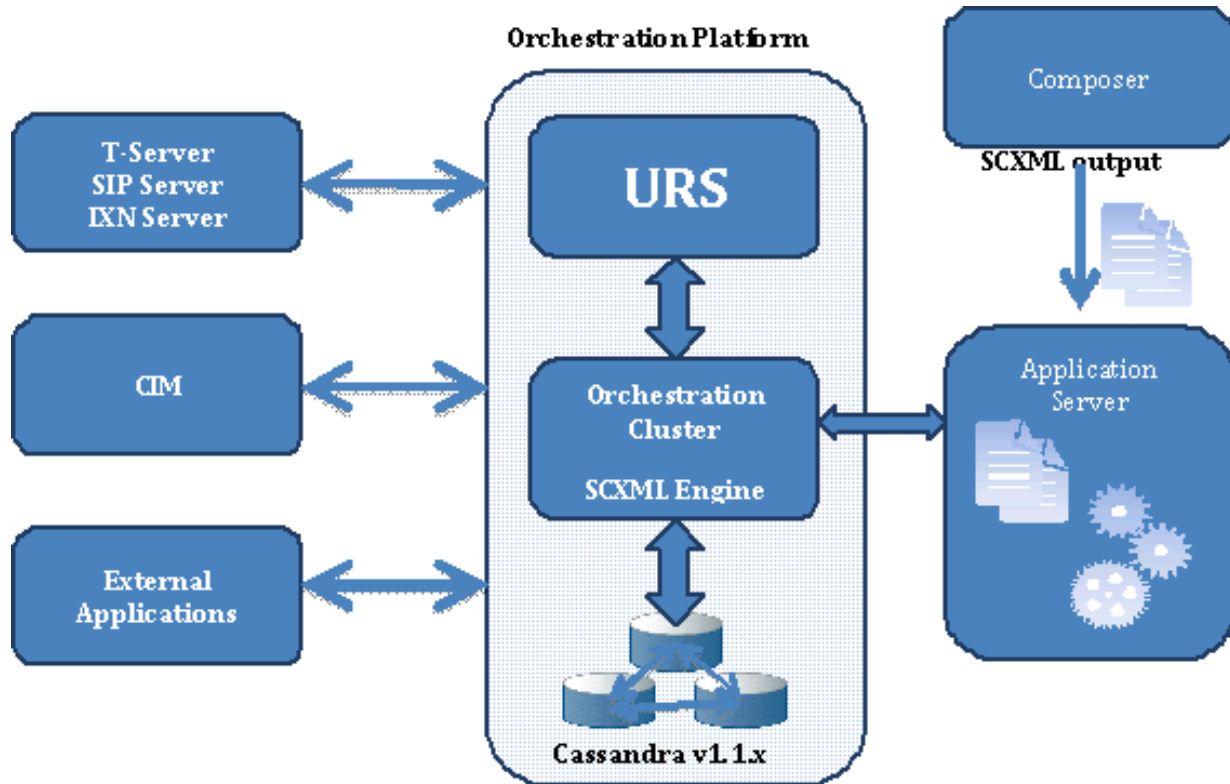


Figure 3: Orchestration Server Architecture

Voice Interaction Flow

Figure 4 shows a sample voice interaction flow that is based on the architecture diagram in Figure 3 on [page 28](#).

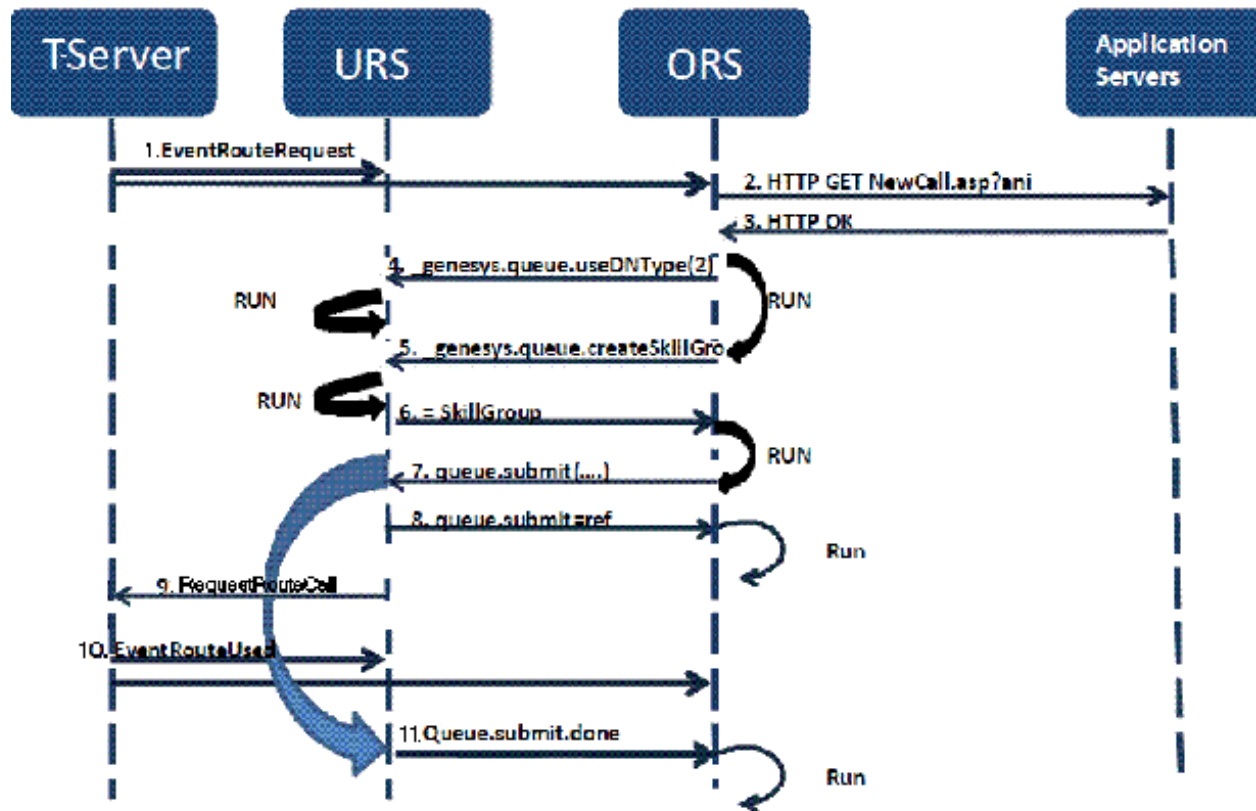


Figure 4: Sample Interaction Flow

The following explanation describes the sample interaction flow in [Figure 4](#).

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

- Item 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`).
- Item 8.** URS performs the required action(s), shown here as RUN, and reports the results to ORS.
- Item 9.** If needed, URS sends a request to T-Server (`RequestRouteCall`).
- Item10.** 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.

Figure 5 shows an eServices-specific architecture diagram for an ORS deployment.

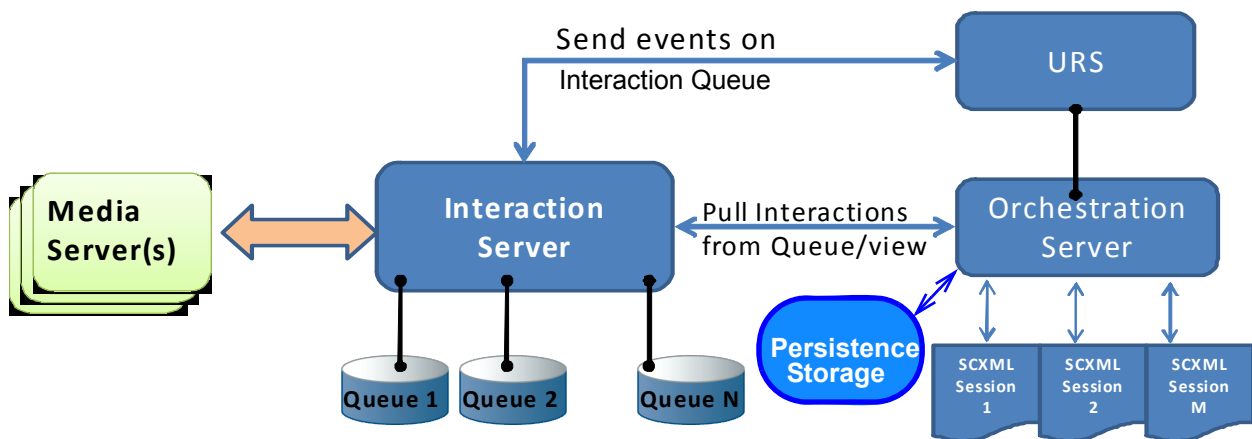


Figure 5: eServices-Specific ORS Architecture

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 may require a connection to an additional Interaction Server. Interaction Server can be configured based either on an Interaction Server template or on a T-Server template. If the Interaction Server is configured based on an Interaction Server template, ORS needs to be connected as a *client* to one or more Interaction Server(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. See Chapter 4, “Configuring ORS to Interact with eServices,” on [page 71](#)

- 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` on [page 84](#), 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 does not require this connection in order to operate.

Basic Multimedia Interaction Routing

The following sequence diagram (Figure 6) illustrates a scenario for basic multimedia (eServices) interaction routing for ORS 8.1.3.

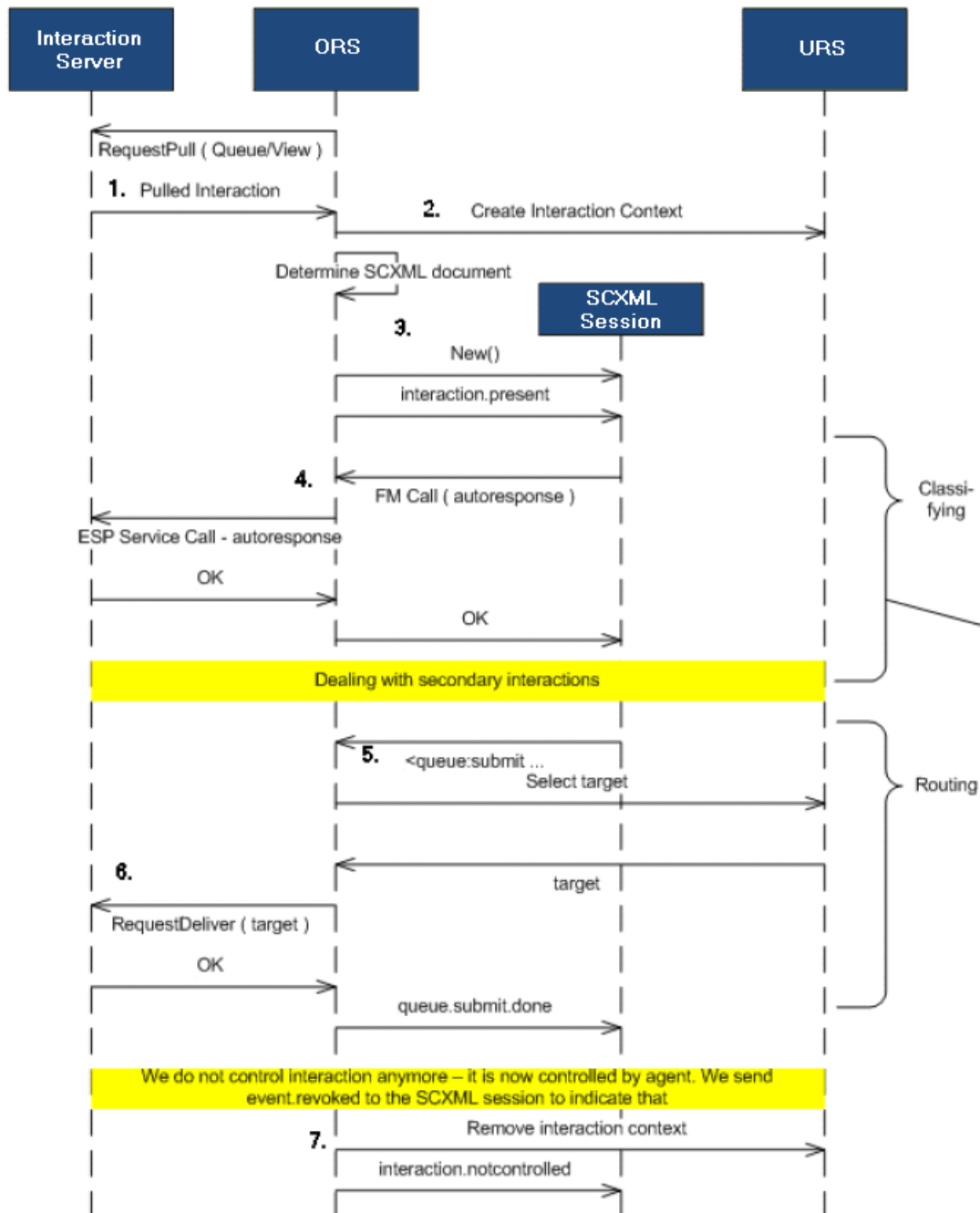


Figure 6: Basic Multimedia Interaction Flow Scenario

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

- Item 1.** ORS pulls the next multimedia interaction from a queue.
- Item 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).
- Item 3.** A new session starts, and an `interaction.present` event is fired into the session, which allows the interaction to be processed.
- Item 4.** The SCXML application submits a request to Interaction Server to auto respond to the interaction.
- Item 5.** After this, a `queue.submit` action is invoked which locates the appropriate resource to process the interaction.
- Item 6.** When an available resource is found, the interaction is delivered to this resource.
- Item 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.

How ORS Processes eServices Interactions

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

ORS processes the interactions by pulling them from the Interaction Server. A `RequestPull` request is used to retrieve a subset of interactions from the specified Queue/View combination (every `Queue` object must have at least one `View` associated with it). The Queues from which ORS pulls the interactions must be explicitly associated with that specific ORS `Application`. ORS checks the `Queue` section in the `Annex` tab of the Interaction Queue `Application` for the `Orchestration` section and pulls interactions from these Queues only. See the `Annex` tab in [Figure 7](#).

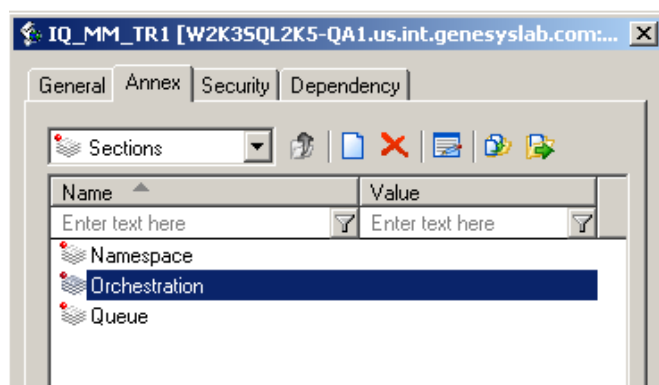


Figure 7: Queue and Orchestration Sections in Interaction Queue

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

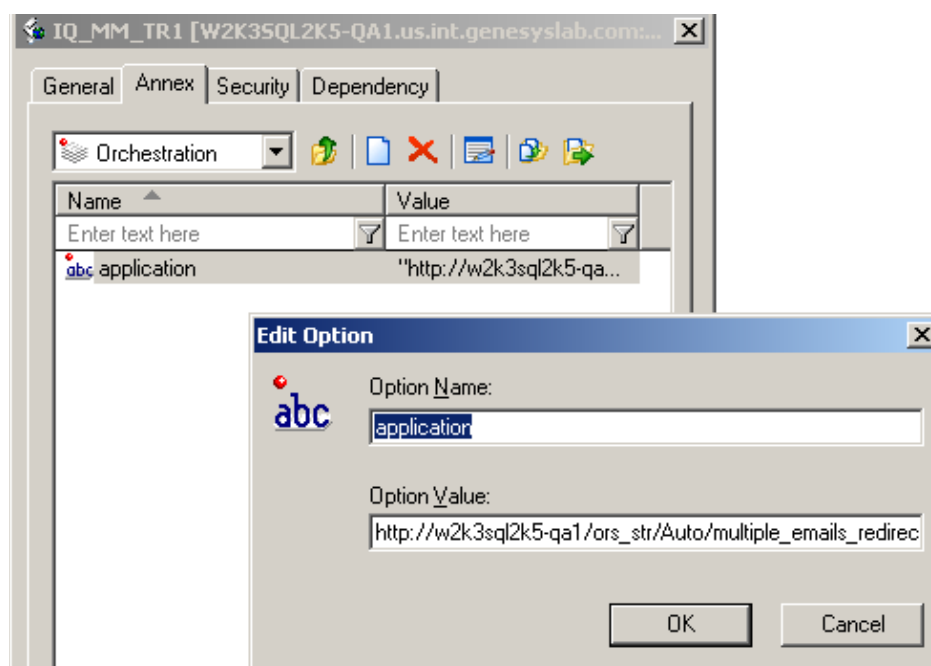


Figure 8: Annex Tab—Orchestration Configuration Option

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

1. Go to Provisioning > Routing/eServices > Interaction Queues.
2. Navigate to the properties of a particular Interaction Queue.

3. In the Application field of Orchestration section, select the strategy (SCXML script) that will be assigned to that Queue and save the changes.

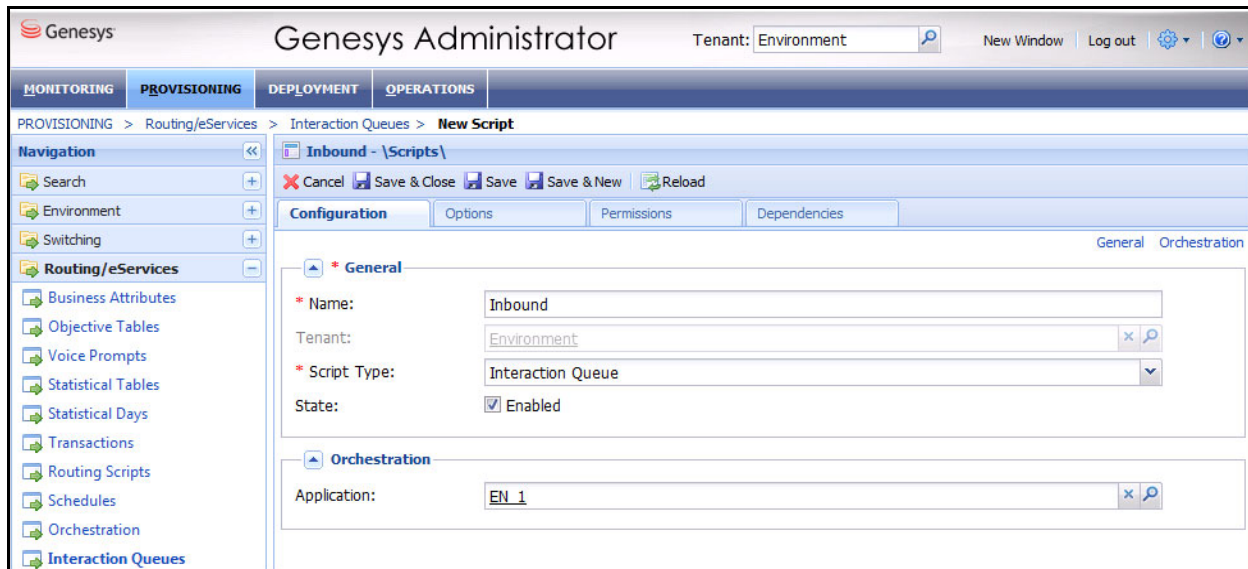


Figure 9: Interaction Queue in Genesys Administrator

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 8.0 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 Interactions

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” on [page 121](#)
- “om-max-in-memory” on [page 121](#)
- “om-delete-from-memory” on [page 122](#)

Enhanced Orchestration 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*). In these deployments prior to ORS 8.1.3, 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

[Figure 10](#) 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.

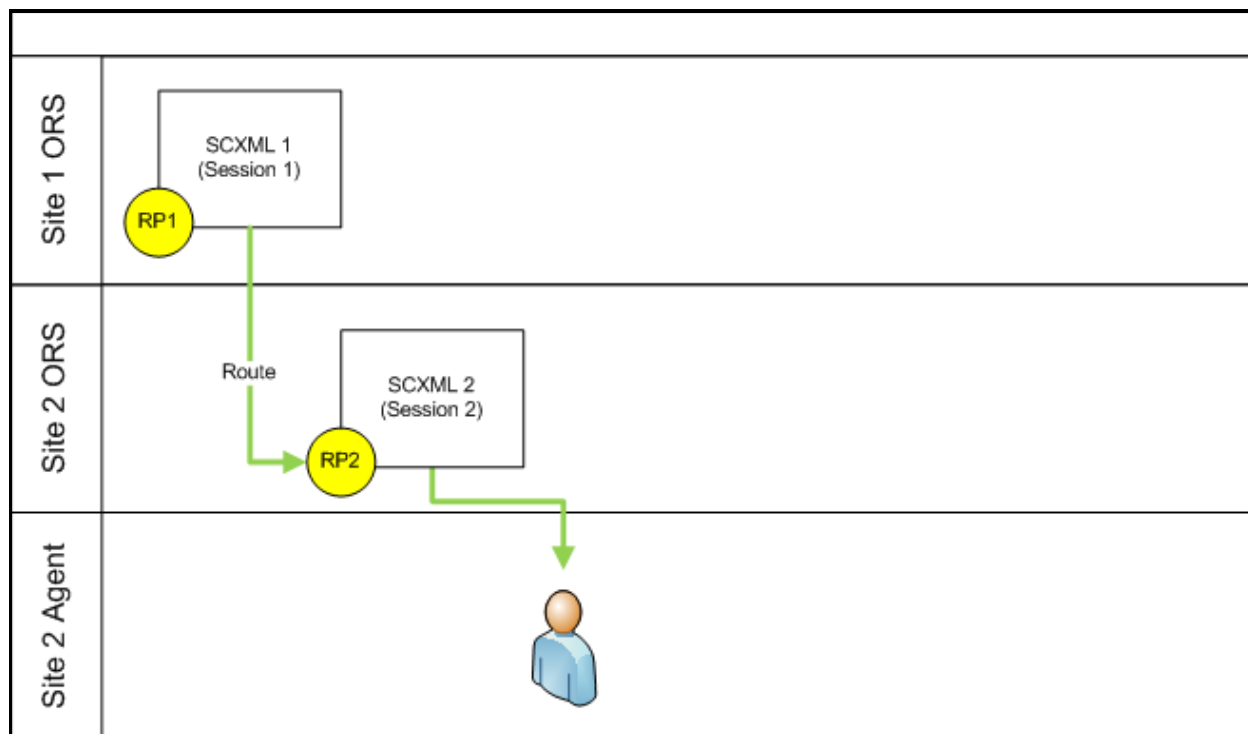


Figure 10: SCXML Session

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 association 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.

3

Persistence; High Availability; Load Balancing; Multiple Data Centers

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

This chapter contains the following topics:

- [Persistence and SCXML Sessions, page 41](#)
- [High Availability/Redundancy, page 46](#)
- [ORS Cluster Architecture, page 48](#)
- [Support for Multiple Data Centers, page 51](#)
- [Deployment Models, page 53](#)

Persistence and SCXML Sessions

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

Whenever required, these stored sessions or documents which 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).

Overview of Persistence Design and Configuration

This section provides an overview of persistence design and configuration.

The key function of persistence is to support the retrieval of the 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.

ORS Persistence 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 does not require a connection to Cassandra. If a connection is not present, an alarm is raised.

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

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

Persistent Storage Operation

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

Document Persistence

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

1. 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`.
2. 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, in [Task Summary: ORS Deployment Tasks](#), on [page 62](#).

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: Note that 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” on [page 48](#) for a description of multiple ORS instances configured as clusters. All ORS instances are active, and all work with persistent storage.

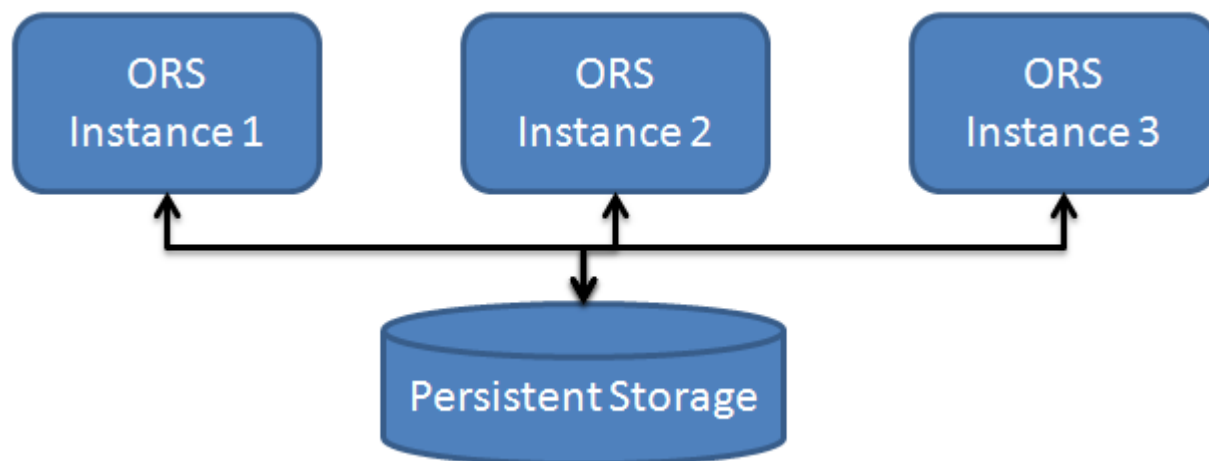


Figure 11: Multiple Orchestration Server Instances Running

Configuring Persistent Storage

Table details the configuration needed to configure persistent storage in the ORS Application object, Options tab, persistence section.

Table 4: Persistent Storage Configuration Options

Option Name	Value
<code>cassandra-listenport</code>	Set to the value of the Cassandra port.
<code>cassandra-nodes</code>	Enter a semicolon-separated list of host names of the Cassandra nodes in the cluster.
<code>cassandra-keyspace-name</code>	Specify the name of the Cassandra Keyspace.
<code>cassandra-schema-version</code>	Enter the 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 a Data Center Cassandra cluster deployment.
<code>cassandra-strategy-options</code>	Set the replication factor for a given Keyspace.

Examples

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

Enabling Persistence in the SCXML Application

Enabling persistence for an SCXML application is controlled with the `<scxml>` tag's attribute `_persist`, which has values of `true` and `false`.

Starting with release 8.1.2, the default value for this attribute is `false`, therefore persistence for an SCXML application is disabled by default. A value of `true` activates persistence for an SCXML application.

Also see “persistence-default” on [page 107](#).

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

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

High Availability/Redundancy

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: 8.1.3 ORS does not support Super Nodes and Master Super Nodes in ORS Cluster architecture

HA Deployment and Session Creation

The ORS node (Primary/Backup pair) that is responsible for a given interaction processing is named *assigned node*.

Within the same cluster, an alternative node (Primary/Backup pair), named *reserved node*, can be another node in single Data Center or another node in another Data Center.

- When receiving an appropriate event on a DN loaded with a routing strategy, the Primary server/application of the assigned node starts call processing by trying to create a session.
- A Backup application and a reserved node are watching the Primary application during the timeout specified in the `orchestration/call-watching-timeout` option. This continues until the session is created by Primary application (assigned node) or until the timeout expired.

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

Session Creation Sequence

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

Recovery of Failed Sessions

The new 8.1.3 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 `_recoveryEnabled`, 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 v 8.1.3 allows `_recoveryEnabled` attributes to be added to the root `<scxml>` element via Extensions of interaction process diagram properties.

Limitations for ORS HA Deployment

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

ORS Cluster Architecture

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. 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 URS in his connections.

Note: ORS 8.1.3 cluster deployment allows each ORS node to have its own URS.

Figure 12 on [page 49](#) illustrates a simple two node pair in a cluster.

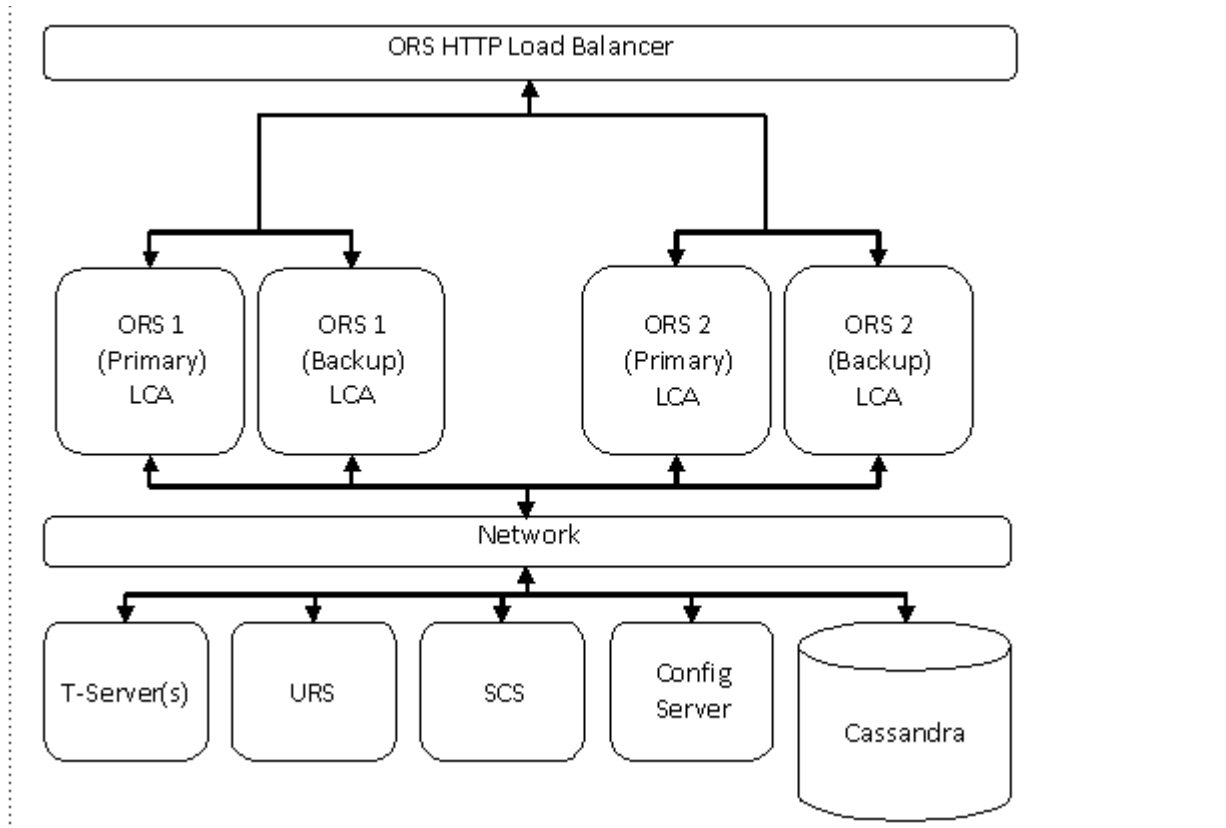


Figure 12: Simple Two Node ORS Cluster

Figure 12 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

This section contains the procedure for cluster/ORS node configuration.

Procedure: Cluster/ORS Node Configuration

Start of procedure

1. Under the Environment tenant (for a multi-tenant deployment) or under the Resources tenant (for a single tenant deployment), configure a Transaction object of type List.
2. Name it ORS.

3. On the Annex tab of the ORS Transaction List create a new section with the name of your cluster and add all Primary ORS nodes participating in cluster.
4. If ORS cluster deployment is logically distributed between several Data Centers (DC), specify a value parameter (alphanumeric string) to link ORS nodes to the particular DC.

End of procedure

Notes: All ORS nodes with the DC set to an empty string will belong to one “nameless” Data Center. Refer to Chapter 4, “Configuring an ORS Cluster,” beginning on [page 67](#) for details.

ORS 8.1.3 requires creating an ORS Transaction List even the deployment has only one ORS node.

Load Balancing Among ORS Nodes

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 ORS cluster and provides load balancing services for HTTP requests.

ORS exposes the RESTful Web Services interface. This interface is based on HTTP. ORS may accept and execute a set of HTTP requests.

This interface is based on standard HTTP and 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 to the correct node.

For RESTful interface, you must set up the http port definition under Server Info during ORS installation, as described in “Configure ORS Application Object” on [page 65](#).

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).

Support for Multiple Data Centers

ORS 8.1.3 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.

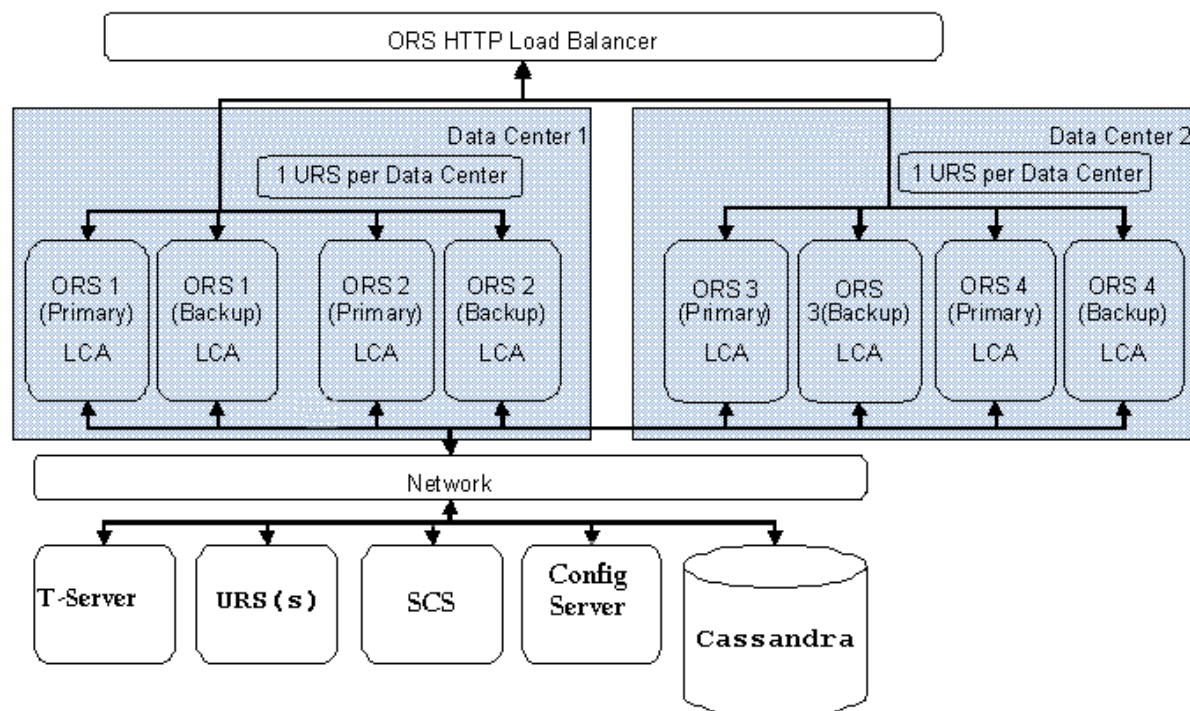


Figure 13: Multiple Data Centers

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 lists 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 (see [Figure 14 on page 53](#)).

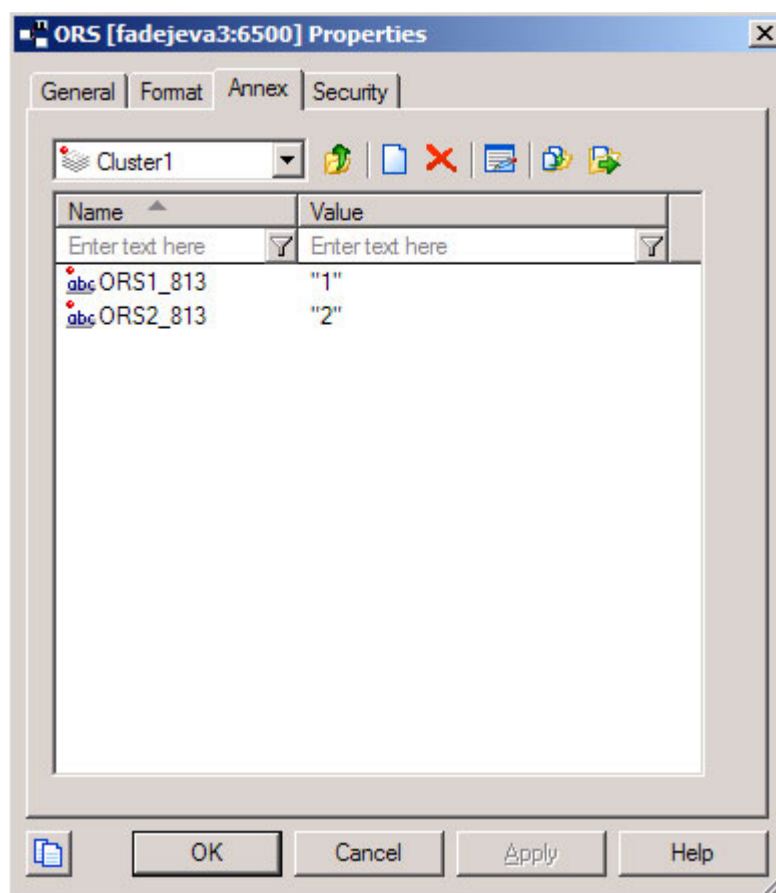


Figure 14: Primary ORS Application

Deployment Models

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

Single-Site Deployment Model

The single-site deployment example assumes that all components are planned to reside within a single box. This deployment does not provide high availability, redundancy, or failure handling. [Figure 15](#) shows a single site deployment example.

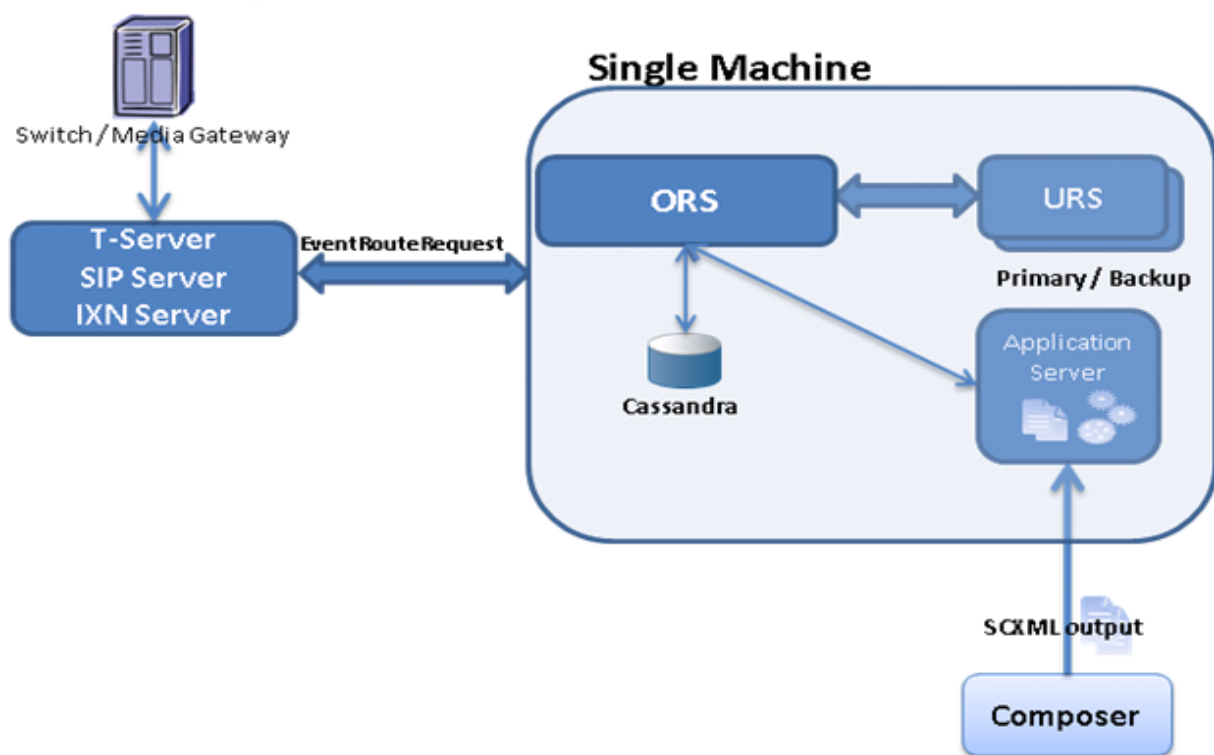


Figure 15: Model of Single Site Deployment

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.

Figure 16 shows a single site with redundancy deployment example.

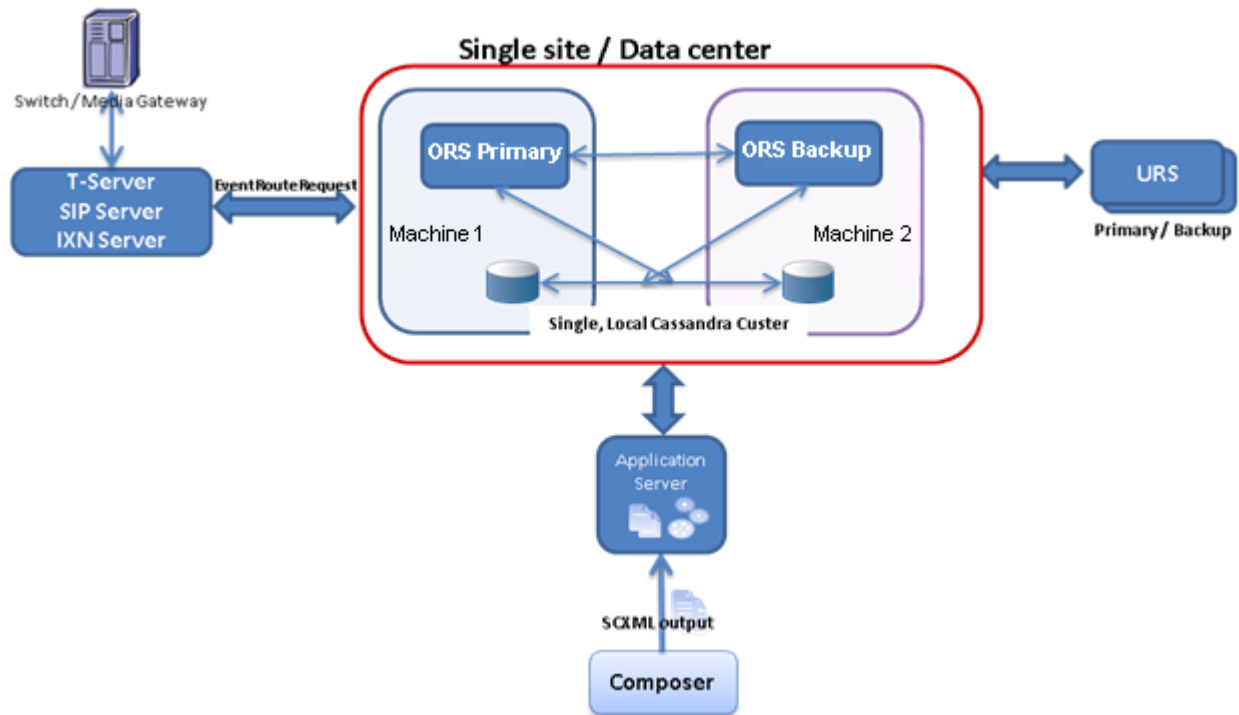


Figure 16: Model of Single Site with Redundancy Deployment

Multiple-Site Deployment Model

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

Figure 17 shows a multiple-site deployment example.

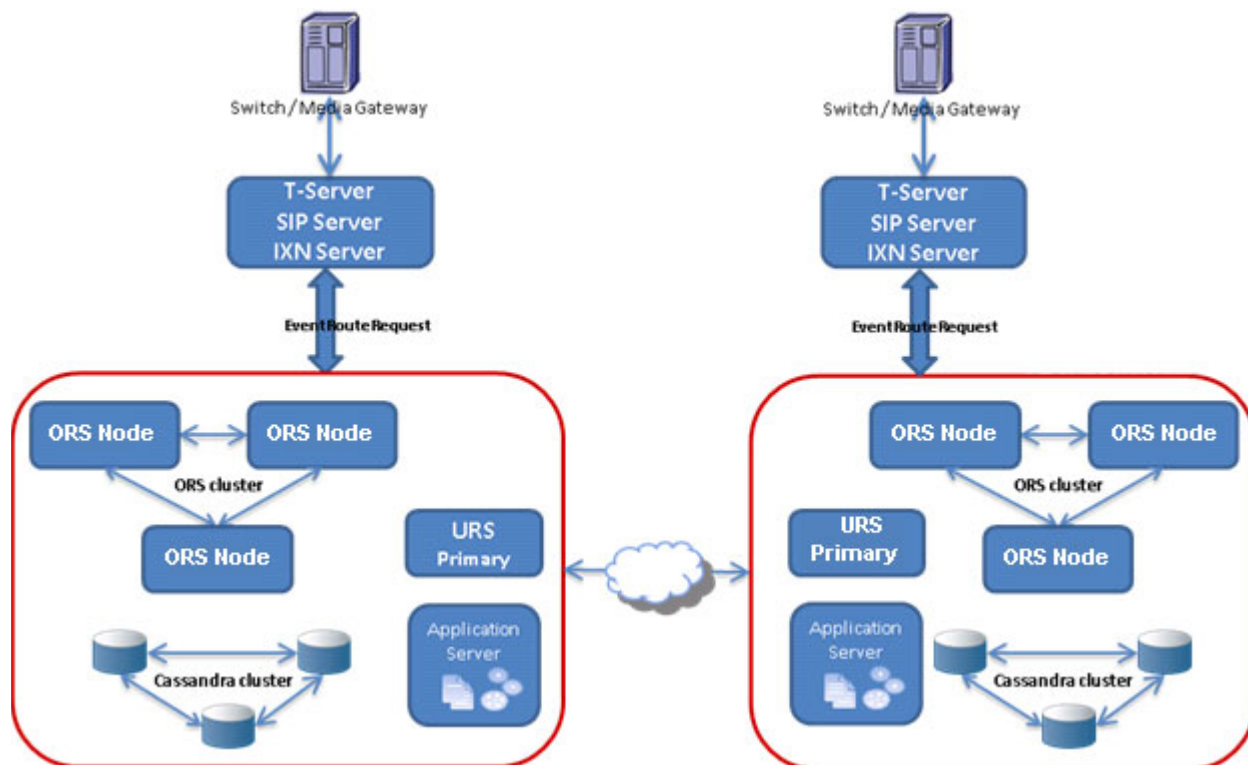


Figure 17: Model of Multiple-site Deployment

Single Cassandra Cluster across Multiple ORS Data Centers Deployment Model

Figure 18 shows a Single Cassandra Cluster across Multiple-site Deployment example.

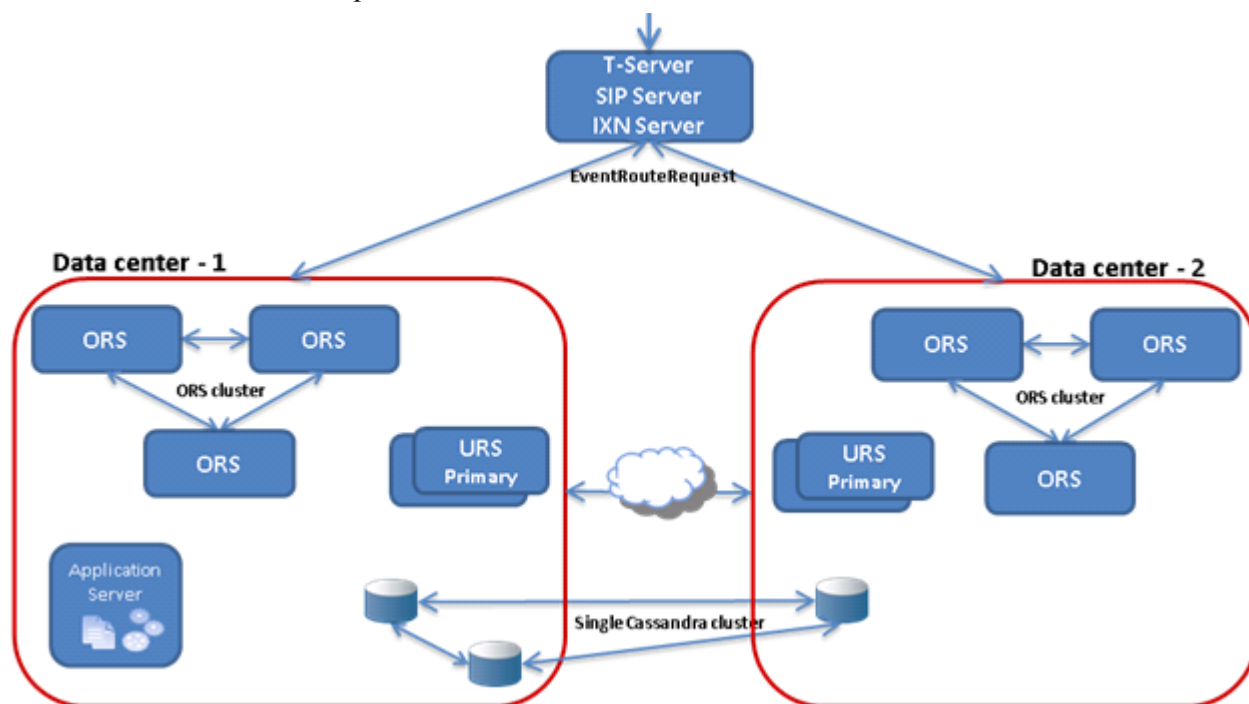


Figure 18: Model of a Single Cassandra Cluster across Multiple-site Deployment

4

ORS General Deployment

This chapter 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.

This chapter contains the following topics:

- [Prerequisites, page 59](#)
- [ORS Deployment Tasks, page 62](#)
- [Configure ORS Application Object, page 65](#)
- [Configuring an ORS Cluster, page 67](#)
- [Manually Loading an SCXML application, page 68](#)
- [Configuring ORS to Interact with eServices, page 71](#)

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 Genesys Framework. 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.

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

Orchestration Server and Local Control Agent

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

Persistent Storage

Determine whether you will use persistent storage (Apache Cassandra) (see “Persistence and SCXML Sessions” on [page 41](#)). If you chose to do so, then installing Cassandra should be done as the first 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.genesyslabs.com.

Task Summary: Prerequisites for ORS deployment

Below is a table containing summarizing prerequisites for deployment.

Table 5: Prerequisites for ORS Deployment

Objective	Related Procedures and Actions
Deploy Configuration Layer and ensure that Configuration Manager is running.	See the Framework 8.1 Deployment Guide for details.
Deploy Network objects (such as Host objects).	See the Framework 8.1 Deployment Guide for details.

Table 5: Prerequisites for ORS Deployment

Objective	Related Procedures and Actions
Deploy the Management Layer.	See the Framework 8.1 Deployment Guide for details. Also see the Framework 8.1 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 Framework 8.1 Management Layer User's Guide .
Deploy persistent storage.	See the Cassandra Installation/Configuration Guide . Also see “Persistence and SCXML Sessions” on page 41 .

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 relevant Wizard pages or 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 located in Chapter 5, “Configuration Options,” on [page 75](#) of this book.

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

ORS Deployment Tasks

You can configure ORS entirely in Configuration Manager or in Genesys Administrator. This chapter describes ORS configuration using Configuration Manager. [Table Task Summary](#): presents a high-level summary of ORS deployment tasks.

Task Summary: ORS Deployment Tasks

Task	Related Action or Procedure
<p>In Configuration Manager, import the appropriate Application Template for ORS.</p> <ul style="list-style-type: none"> The file name is <code>OR_Server_813.apd</code> if you are using Configuration Server 8.0.3 or later. The file name is <code>OR_Server_Genesys_Server.apd</code> if you are using a Configuration Server earlier than release 8.0.3. 	<p>If you need help with this task, consult the <i>Configuration Manager Help</i> available on the Management Framework Wiki. See Configuration Database Objects > Environment > Application Templates.</p>
Create the ORS Application object.	<p>Configure Server Info tab:</p> <ul style="list-style-type: none"> Host: Select name of host where ORS application will be running Port: Enter available listening port of ORS Configure http port: Port ID: http Communication port: Enter any available port Connection protocol: select http <p>See Procedure: Creating/Configuring the ORS Application object, on page 65.</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) 	<p>Use the Connection tab of the Application object in Configuration Manager to set up connections to other servers.</p> <p>See Procedure: Creating/Configuring the ORS Application object, on page 65</p>

Task Summary: ORS Deployment Tasks (Continued)

Task	Related Action or Procedure
Configure ORS application to work with persistence storage.	<p>Go to Orchestration Server Application object > Options tab > persistence section:</p> <ul style="list-style-type: none"> • <code>cassandra-listenport</code> (page 90). Set to the value of Cassandra port. • <code>cassandra-nodes</code> (page 90). Enter semi-colon separated list of host names of Cassandra nodes in cluster. • <code>cassandra-keyspace-name</code> (page 89). Specify the name of Cassandra keyspace • <code>cassandra-schema-version</code> (page 91). Enter Cassandra schema version. • <code>cassandra-strategy-class</code> (page 91). Set to <code>SimpleStrategy</code> if Cassandra is deployed as a single cluster, Set to <code>NetworkTopologyStrategy</code> in case of Data Centers Cassandra cluster deployment. • <code>cassandra-strategy-options</code> (page 92). 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>
Configure ORS cluster.	See Procedure: Configuring a Cluster , on page 67.
Manually loading an SCXML application on objects other than ORS.	<p>See the following:</p> <ul style="list-style-type: none"> • Procedure: Manually loading an SCXML application on a DN, on page 69. • Procedure: Manually loading an SCXML application on an Enhanced Routing Script, on page 69. • Procedure: Configuring the ApplicationParms section of an Enhanced Routing Script object, on page 70.

Task Summary: ORS Deployment Tasks (Continued)

Task	Related Action or Procedure
Configure ORS application to work with multimedia interactions (if needed).	Go to Orchestration Server Application object > Options tab > orchestration section: mcr-pull-by-this-node (page 84). Set into true if application should work with multimedia interactions Also see: Procedure: Configuring eServices Application with Type T-Server , on page 72 .
Set a Redundancy type (if needed).	In Server Info tab: Primary server: Redundancy type field, set to Warm Standby Backup server: select backup server application Backup server: Redundancy type field, set to Warm Standby See Chapter 3, “Persistence; High Availability; Load Balancing; Multiple Data Centers,” on page 41 .
Add ORS application into the ORS cluster.	See “Configuring an ORS Cluster” on page 67 .
Install Orchestration Server.	See Chapter 8, “Installing Orchestration Server,” on page 141 .
Test your ORS deployment.	See Chapter 9, “Starting and Stopping Procedures,” on page 149 .

Note: [Task Summary: 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).

Configure ORS Application Object

This section describes how to create/configure the ORS Application object.

Procedure: Creating/Configuring the ORS Application object

Start of procedure

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.

Note: 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).

Note: The **Application** name should not contain spaces.

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**—the name of the host on which ORS resides.
 - **Ports**—the listening port.
9. 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.
10. Enter an unused port number for the **Communication Port**.

Note: For information on this dialog box, see the Port Info Tab topic in the [Framework 8.1 Configuration Manager Help](#).

11. For web service access to this ORS application, configure the HTTP port:
 - a. In the New Port Info dialog box, in Port ID enter http.
 - b. For Communication Port, enter an unused port number.
 - c. For Connection Protocol, select http from the drop-down menu list.
 - d. Click OK.
12. Select the Start Info tab and specify the following:
 - Working Directory—the Application location (example: C:/GCTI/or_server)
 - Command Line—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—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—the time interval the server waits until restart if the server fails.
 - Shutdown time—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).
13. Select the Connections tab and specify all the servers to which ORS must connect
 - T-Server
 - Interaction Server
 - Universal Routing Server

End of procedure

Configuring an ORS Cluster

ORS provides a new configuration transaction of the type `List`, called `ORS` in the tenant environment, to determine the 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.

Adding 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 to a specific Orchestration application to a Data Center. Follow the procedure below to add an ORS Application to a cluster.

Procedure: Configuring a Cluster

Start of procedure

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.

An example is shown in [Figure 19](#).

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.

4. Double-click on the cluster name.
5. Right-click inside the `Section` window and select `New` from the shortcut menu.
6. In the `Option Name` field, enter the name of an Orchestration application configured as `Primary`.
7. In the `Option Value` field, enter the name of the Data Center associated with the Orchestration Node.

8. Click OK to save.
9. Repeat Steps 7 - 10 for all Orchestration Nodes that belong to this cluster.
10. Click on Up One Level.
11. Repeat Steps 5 - 12 for all clusters.
12. Click OK to save and exit. An example is shown in [Figure 19](#).

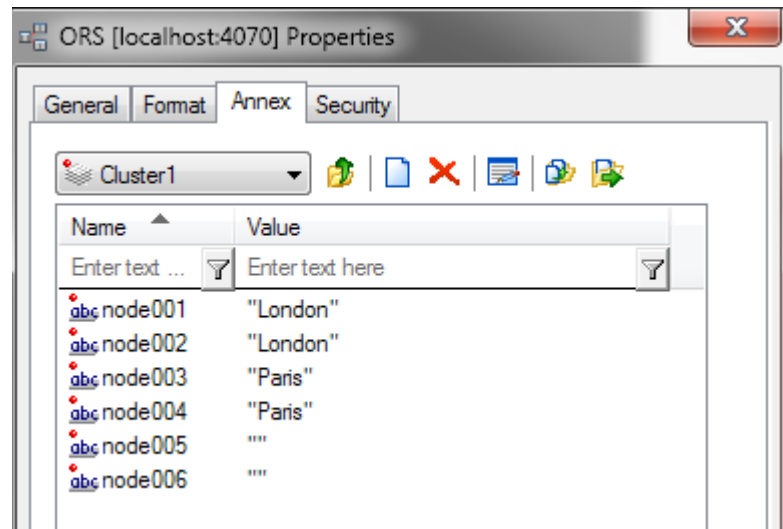


Figure 19: Orchestration Server Nodes Inside Cluster

End of procedure

In the example in [Figure 19](#), 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, work allocation happens automatically, based on the configuration of the cluster described above.

Manually Loading an SCXML application

This section describes configuring objects other than ORS.

Procedure:
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” on [page 122](#).

Start of procedure

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. Refer to the option description for “application” on [page 122](#) for a full description of this configuration option its valid values.
8. Click OK to save

End of procedure

Procedure:
Manually loading an SCXML application on an Enhanced Routing Script**Purpose:**

See “Enhanced Routing Script Options” on [page 124](#).

Start of procedure

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.
Refer to the “url” on [page 128](#) option description for option `url` in the `Application` section for a full description of this configuration option its valid values.
7. Click **OK** to save

End of procedure

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.

Procedure: Configuring the `ApplicationParms` section of an Enhanced Routing Script object

Purpose: See “Enhanced Routing Script Options” on [page 124](#).

Start of procedure

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.

Refer to the option description for “{Parameter Name}” on [page 129](#) for a full description of this configuration option its valid values. Table 12, “Parameter Elements for `ApplicationParms`,” on [page 129](#) provides useful information about parameters that can be added. [Figure 20](#) shows an example of the use of the `ApplicationParms` section.

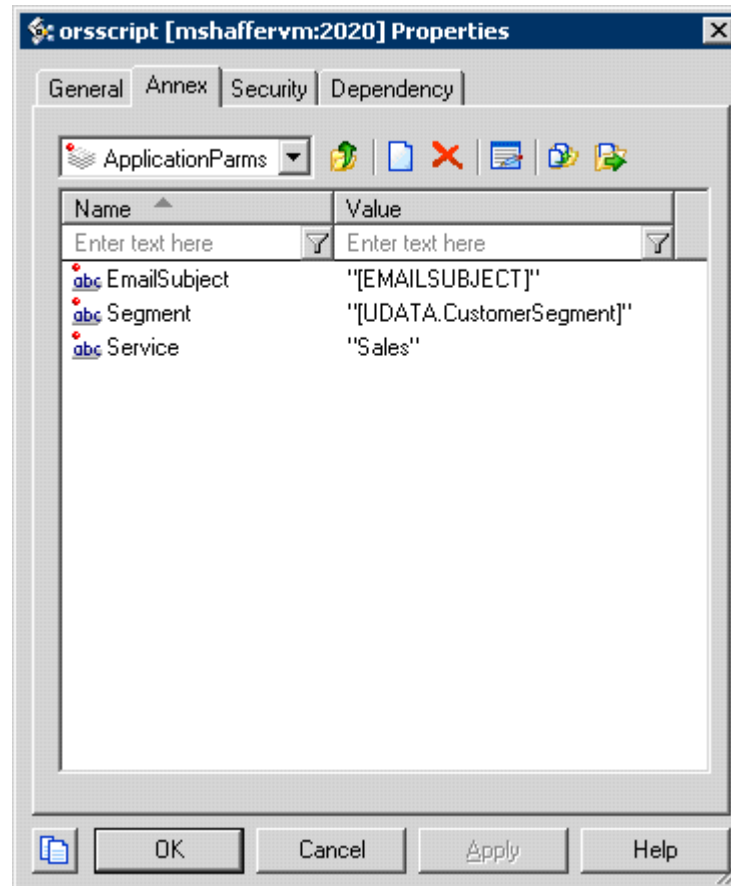


Figure 20: Script Object Annex Tab for ApplicationParms

8. Click OK to save.
9. Repeat from [Step 5](#) to add another option in this section.

End of procedure

Configuring ORS to Interact with eServices

This section describes how to configure ORS to interact with eServices.

Connection to eServices Application

ORS, acting as a client, must have two separate connections to every instance of Interaction Server operating in the environment. Each instance of Interaction Server is represented by two `Application` objects in the ORS `Connections` tab (see [Figure 21](#))

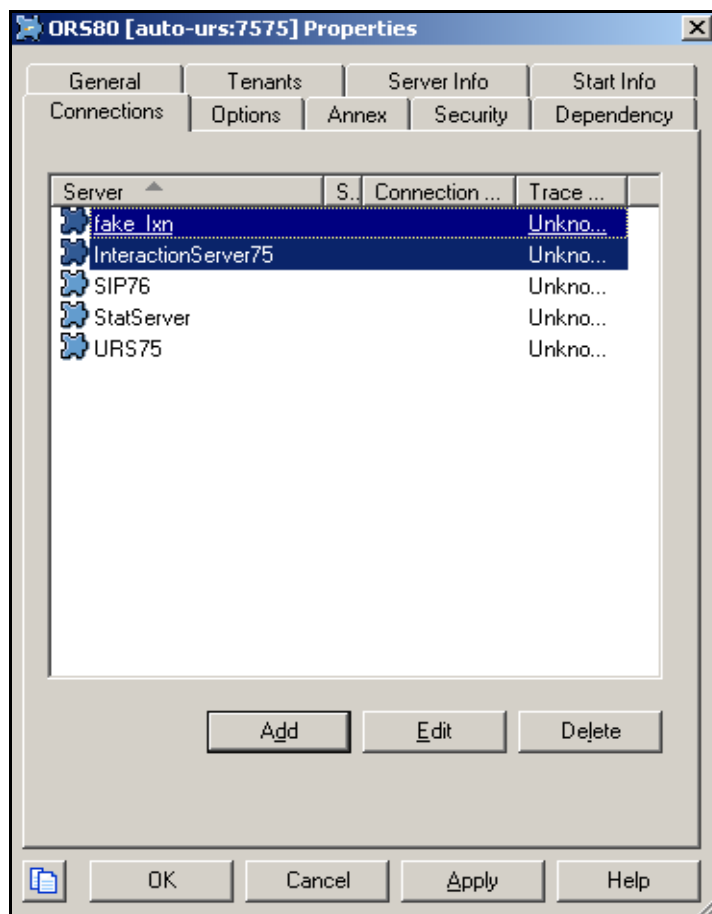


Figure 21: Interaction Server Instances in ORS Application Properties

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

Procedure: Configuring eServices Application with Type T-Server

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

Start of procedure

1. Import two templates to create the Interaction Server Application objects.
 - a. Use an Interaction Server Application template for the Application object that you are using for the actual installation.

- b. Use a T-Server Application template (with its default configuration) for the second Application object. See [Figure 22](#).

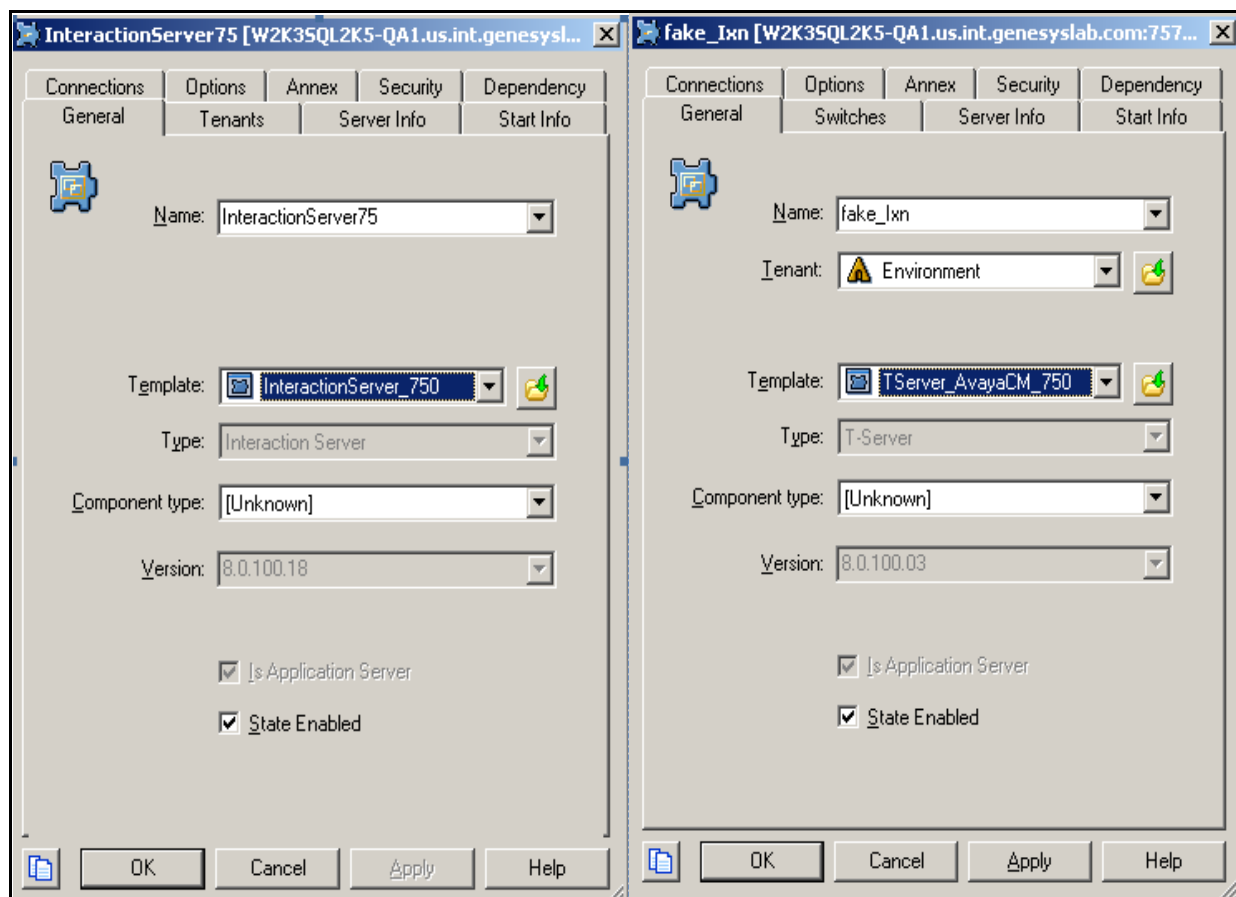


Figure 22: Interaction Server and T-Server Application Templates

2. Create the Interaction Server Application objects.
Ensure that both of the Interaction Server Application objects have different Application object names. (see [Figure 21](#))
3. In the second Application object (based on the T-Server template) configure a connection to the same multi-media switch that is used for the first Application object (based on the Interaction Server template).

Note: The first Interaction Server Application (created by using the Interaction Server template) has no association in its properties with a multi-media switch. ORS determines this association, based on the tenant that is specified in General tab of the ORS Application. Alternatively, the second the Interaction Server Application (based on T-Server template) must have a multi-media switch configured in its properties, and it must be the same one that is used by the first Interaction Server Application.

- On the **Server Info** tab of each Interaction Server Application, configure the same port number and ensure both Applications exist on the same host. See [Figure 23](#).

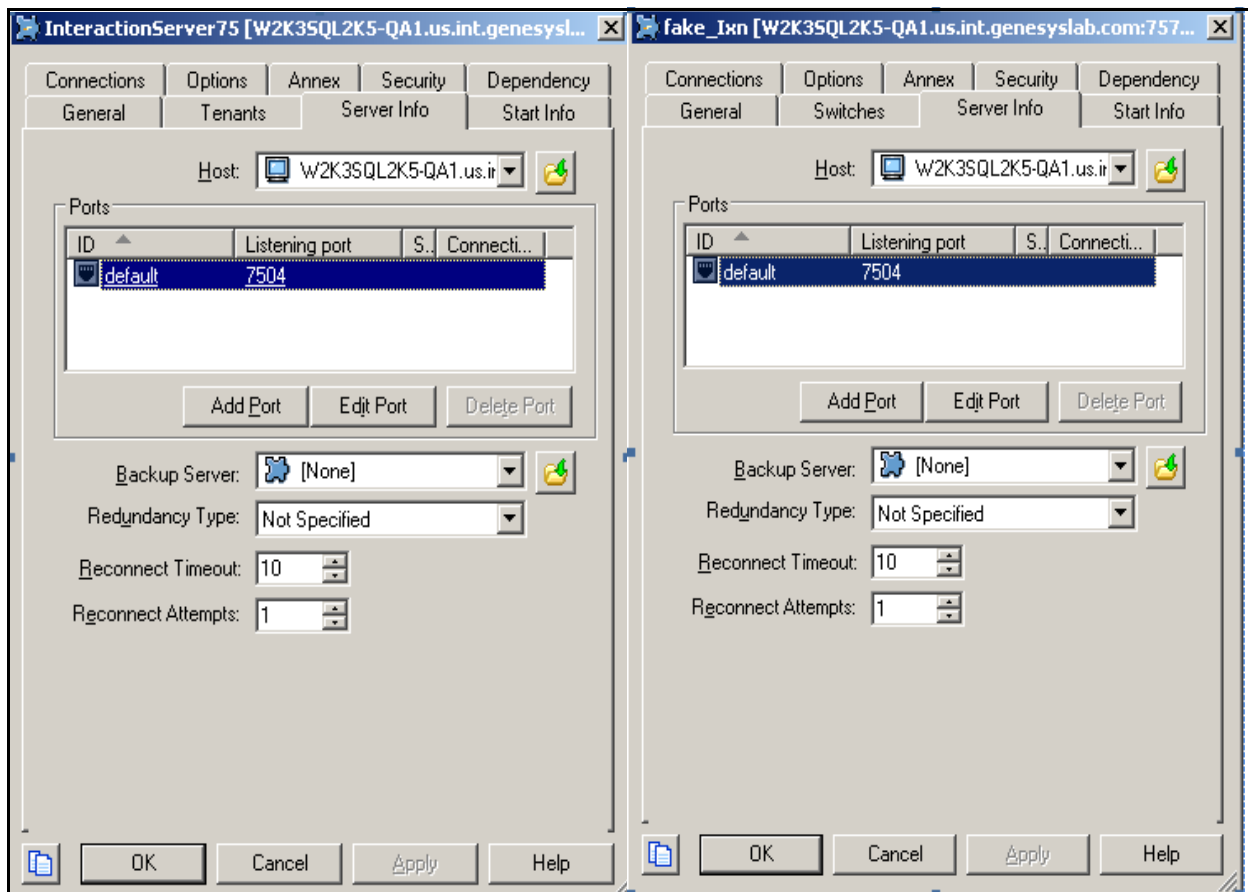


Figure 23: Interaction Server Application Objects on the Same Host

- Save the configuration.

End of procedure

For additional information, see Chapter 2, “How ORS Processes eServices Interactions,” on [page 34](#).



Chapter

5

Configuration Options

This chapter describes configuration options for Orchestration Server (ORS) and includes the following topics:

- [Options Tab, page 75](#)
- [Application Level Options, page 84](#)
- [Common Log Options, page 110](#)
- [Switch Object, gts Section, page 122](#)
- [DN-Level Options, page 122](#)
- [Enhanced Routing Script Options, page 124](#)

Options Tab

Most options described in this section are specified on the `Options` tab of the Properties window of the `Application` object. Figure 24 on [page 76](#) shows an example of the properties window of the ORS `Application` object in Configuration Manager with the `scxml` section visible.

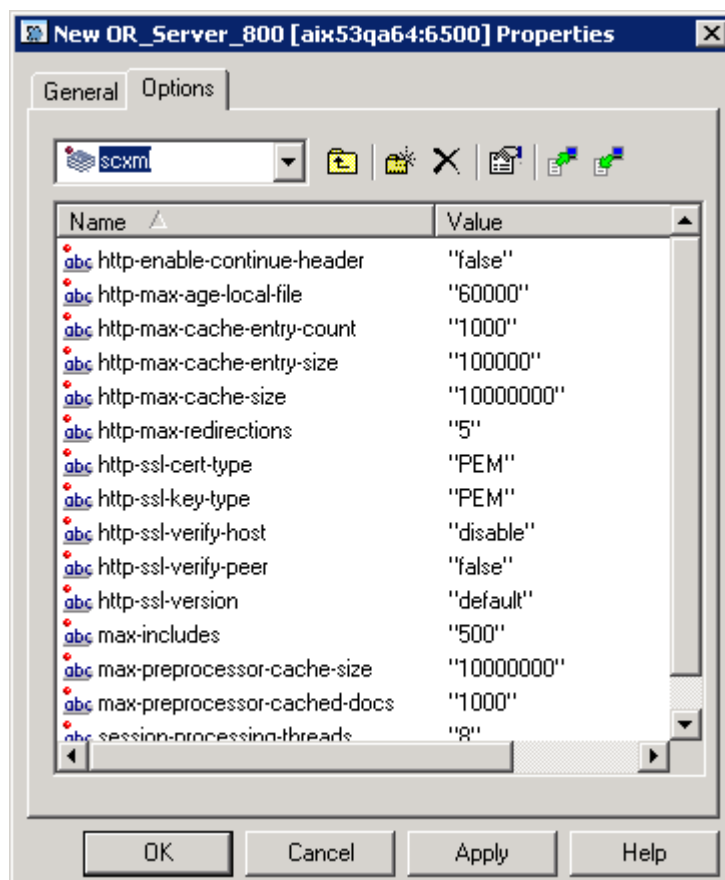


Figure 24: Properties Dialog Box, Options Tab, scxml Section

ORS Application Sections

ORS options are placed in the following option folders:

- For the ORS Application object—In the orchestration, persistence, scxml, log, and mcr sections of the Options tab of the ORS Properties dialog box.
- For DN (Extension or Route Point), or Interaction Queue objects—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.

Option List for ORS Application

[Table 6](#) lists the options that you can configure in the `Application` object for ORS, (as specified in the full description of the option), a brief option description, and the page number on which the option information can be found. [Table 6](#) lists ORS options by *section*, and then in alphabetical order.

Table 6: ORS Application Configuration Options

Name	Brief Description	Detailed Description
orchestration section		
mcr-pull-by-this-node	Specifies that the pulling of eServices interactions is allowed to be performed by a node if set to <code>true</code> .	84
mcr-pull-interval	Specifies the number of milliseconds between attempts to pull interactions from the pairs of queues/views managed by the ORS.	85
mcr-pull-limit	Specifies 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.	85
scxml-log-filter-level	Depending on how this option is configured, ORS generates corresponding log events of the Standard level.	86
max-microstep-count	Specifies the maximum number of microsteps allowed to be taken between the processing of a new event before a session is terminated by the system.	85
parse-start-params	This option specifies whether to enable or disable serialization/deserialization of session parameters.	86
session-hung-timeout	Specifies the time (in seconds) that a hung session has to complete processing and exit gracefully before being terminated by the system.	87
switch-multi-links-enabled	Determines whether ORS supports multi-link switch configurations (load sharing Network T-Servers or IVR Servers).	87

Table 6: ORS Application Configuration Options (Continued)

Name	Brief Description	Detailed Description
webfm-event-hold-response	Enhances the behavior of WebFM Event responses.	85
persistence section		
cassandra-connect-attempt-timeout	Specifies the maximum time allowed for a connection to the Cassandra cluster to complete, in milliseconds.	89
cassandra-keyspace-name	Specifies the name of the Cassandra keyspace to use for an ORS cluster.	90
cassandra-listenport	Specifies the Cassandra client connection port when using Cassandra-based persistence method.	90
cassandra-max-latency	Specifies the maximum Cassandra request latency time allowed before logging a message indicating that this maximum has been exceeded.	90
cassandra-nodes	Semi-colon-separated list of host names or IP addresses when using Cassandra-based persistence method.	90
cassandra-schema-version	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.	91
cassandra-strategy-class	Specifies the strategy class that Cassandra uses for the cluster, either SimpleStrategy or NetworkTopologyStrategy	91
cassandra-strategy-options	Specifies the replication_factor to be configured for the given Orchestration keyspace.	93
cassandra-thread-count	Sets the number of threads in the persistence worker thread pool.	93
max-cache-count	Defines the number of entries allowed within the internal Orchestration persistence cache.	93
max-cache-size	Sets the maximum size of the amount of memory (in bytes) that the internal Orchestration persistence cache can use.	93

Table 6: ORS Application Configuration Options (Continued)

Name	Brief Description	Detailed Description
scxml section		
debug-enabled	Specifies whether debugging is enabled in the system. If it is enabled, you must also specify the <code>debug-port</code> .	93
debug-port	Specifies the port to be used by the debug client to interact with the SCXML Engine during a debug session.	93
default-encoding	Activates Unicode support for <code>List</code> objects.	93
fips_enable	Enables FIPS compliance in the SCXML library.	94
http-client-side-port-range	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.	94
http-enable-continue-header	Specifies whether to enable or disable the <code>100-continue</code> header in the HTTP 1.1 post request.	95
http-max-age-local-file	The maximum age of a local file, in milliseconds.	95
http-max-cache-entry-count	The maximum number of entries that can be stored in the cache.	95
http-max-cache-entry-size	The maximum size of each cache entry, in bytes.	95
http-max-cache-size	The maximum size of the HTTP cache, in bytes.	96
http-max-redirections	The maximum number of times to follow the <code>Location</code> header in the HTTP response.	96
http-no-cache-urls	A comma-delimited list of substrings to prevent a response from being cached.	96
http-proxy	The HTTP proxy server (if applicable).	97
http-ssl-ca-info	The file name holding one or more CA certificates with which to verify the peer.	97
http-ssl-ca-path	The path holding one or more CA certificates with which to verify the peer.	97
http-ssl-cert	The file name of the SSL certificate.	98

Table 6: ORS Application Configuration Options (Continued)

Name	Brief Description	Detailed Description
http-ssl-cert-type	The SSL Certificate type (PEM or DER).	98
http-ssl-cipher-list	The cipher list as defined by OpenSSL.	98
http-ssl-key	The path or file name of the SSL private key.	99
http-ssl-key-type	The SSL Key type (PEM or DER).	99
http-ssl-random-file	The file which is read from in order to see the random engine for SSL.	99
http-ssl-verify-host	Specifies how the common name from the peer certificate should be verified during the SSL handshake.	100
http-ssl-verify-peer	Specifies whether or not the system should verify the peer's certificate.	100
http-ssl-version	The SSL version to be used.	100
https-proxy	The HTTPS proxy server (if applicable).	101
js-preload-files	Provides a list of JavaScript files that are pre-loaded at ORS startup.	101
max-age	The maximum age of a cached resource, in seconds.	96
max-age-local-file	Sets the maximum age of the local file in milliseconds.	102
max-compiler-cache-size	The maximum amount of memory (in bytes) allowed for the <code><xsi:include></code> compiler cache.	103
max-compiler-cached-docs	The maximum number of items that the <code><xsi:include></code> compiler cache can have.	103
max-compiler-cached-doc-size	The maximum size, in bytes, allowed to cache the results of the compiled <code><xsi:include></code> document.	103
max-debug-sessions	The maximum number of simultaneous debug sessions within the current SCXML Engine instance.	105
max-includes	The maximum number of documents that may be included using <code><xsi:include></code> .	104

Table 6: ORS Application Configuration Options (Continued)

Name	Brief Description	Detailed Description
max-pending-events	The maximum number of events allowed to be queued to a session (inclusive of internal, external, delayed and undelivered events).	104
max-preprocessor-cache-size	The amount of memory, in bytes, that the <code><x i : include></code> pre-processor cache can use.	105
max-preprocessor-cached-docs	The maximum number of items that the <code><x i : include></code> pre-processor cache can have.	105
max-preprocessor-cached-doc-size	The maximum size, in bytes, of the cached results of the preprocessed <code><x i : include></code> documents.	105
max-session-age	The maximum age in seconds that an ORS session should exist.	105
max-stale	The number of seconds to extend the life of a cached resource.	106
max-state-entry-count	Specifies the maximum number of entries into a target state resulting from a transition before a session is terminated by the system.	86
password	The SSL key password.	106
persistence-default	Allows to suppress all persistence capabilities with the SCXML engine.	107
persistence-max-active	Allows setup of a maximum number of active sessions that the SCXML engine will keep in memory.	107
process-event-timeout	Specifies the maximum time allotted for the processing of the event queue.	107
session-processing-threads	The number of threads in the thread pool.	108
stuck-thread-timeout	Specifies waiting time in milliseconds before declaring a session processing thread as being stuck.	108
system-id	Allows setup of the ORS system ID in order to have unique session IDs across ORS instances.	108

Table 6: ORS Application Configuration Options (Continued)

Name	Brief Description	Detailed Description
log section		
all buffering expire segment verbose	These are standard Genesys log options.	n/a
x-server-trace-level	The level of tracing to be enabled for the ORS.	109
x-server-gcti-trace-level	The level of GCTI tracing to be enabled for the ORS. Controls how much detail should be in the logs for GCTI-related events, such as those from T-server.	109
x-server-config-trace-level	The level of configuration tracing to be enabled for the ORS. Controls how much detail should be in the logs for configuration-related events.	109
x-print-attached-data	Specifies whether or not attached data should be formatted and printed in the logs.	110
mcr section		
om-memory-optimization	Specifies whether memory optimization is in effect.	121
om-max-in-memory	The maximum number of interactions that will be stored in memory cache before any interactions will be removed (when om-memory-optimization is set to true).	121
om-delete-from-memory	The number of calls that will be deleted when the value of om-max-in-memory has been attained.	122
http-version	The HTTP version to use when fetching documents from the application server.	126
max-age	Tells the ORS how long an application script can be cached.	127
max-stale	The number of seconds to extend the life of a cached file.	127
url	Specifies the URL of the SCXML document to load.	128

Option List for DNs

[Table 7](#) lists the options that you can configure in the DN object with types *Extension*, *ACD Queue*, and *Routing Point* as well as *Interaction Queue* objects (as specified in the full description of the option), a brief option description, and the page number on which the option information can be found. [Table 7](#) lists options by *section*, and then in alphabetical order.

Table 7: DN-Level Options

Name	Brief Description	Detailed Description
Orchestration section		
application	Specifies the URL of the SCXML document to load.	122
send-retries	The option specifies the number of times ORS tries to resend an event if the first attempt does not succeed.	124

Option List for Enhanced Routing Script Objects

[Table 8](#) lists the options that you can configure in the Enhanced Routing Script object (as specified in the full description of the option), a brief option description, and the page number on which the option information can be found. [Table 8](#) lists options by *section*, and then in alphabetical order.

Table 8: Enhanced Routing Script Options

Name	Brief Description	Detailed Description
Application section		
alternate-url	This option specifies the URL of the SCXML document to load.	124
fetch-timeout	Specifies the time (in milliseconds) before a document fetch is abandoned.	126
http-useragent	The string to use in the HTTP header field <i>User Agent</i> , which identifies the application to the web server.	126
http-version	The HTTP version to use when fetching documents from the application server.	126
max-age	Tells the ORS how long an application script can be cached.	127

Table 8: Enhanced Routing Script Options

Name	Brief Description	Detailed Description
max-stale	The number of seconds to extend the life of a cached file.	127
url	Specifies the URL of the SCXML document to load.	128
alternate-url	This option specifies the URL of the SCXML document to load.	124
fetch-timeout	Specifies the time (in milliseconds) before a document fetch is abandoned.	126
http-useragent	The string to use in the HTTP header field User Agent, which identifies the application to the web server.	126
ApplicationParms section		
{Parameter Name}	Specifies a string that represents a parameter value to be passed to the application.	{Parameter Name} 129

Application Level Options

The next section details the ORS options.

Orchestration Section

This section describes orchestration section options.

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 this also allows more than one node to pull interactions.

For example:

```
orchestration/ mcr-pull-by-this-node = true
```

mcr-pull-interval

Option section: `orchestration`

Configuration object: `ORS Application` object

Default value: `1000`

Valid values: Any positive integer.

Value changes: in setting the next timer interval

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

For example:

```
orchestration/mcr-pull-interval = 5000
```

mcr-pull-limit

Option section: `orchestration`

Configuration object: `ORS Application` object

Default value: `1000`

Valid values: Any positive integer from 0 to 5000

Value changes: in setting the next timer interval

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

For example:

```
orchestration/mcr-pull-limit = 5000
```

max-microstep-count

Option section: `Orchestration`

Configuration object: `ORS Application` object

Default value: `1000`

Valid values: any integer (no maximum)

Value changes: `Immediately`

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

```
<scxml _microStepLimit="1000" · />
<scxml _stateEntryLimit="100" · />
```

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

This option specifies the maximum number of times in which transitions may be taken during a session without the processing of a new event. Once this value has been exceeded, the session is exited gracefully before being

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

max-state-entry-count

Option section: `Orchestration`

Configuration object: `ORS Application` object

Default value: `100`

Valid values: any integer (no maximum)

Value changes: `Immediately`

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

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

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

```
<scxml _stateEntryLimit="100" · />
```

This option specifies the maximum number of times that a transition may be taken to a target state. Every state element within an SCXML document keeps an individual count of the number of times entered via a targeted transition. Once this value has been exceeded, the session is exited gracefully before being terminated by the system. A value of 0 indicates that the session is not to be forcibly terminated.

parse-start-params

Option section: `Orchestration`

Configuration object: `ORS Application` object

Default value: `false`

Valid values: `true/false`

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

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

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.

scxml-log-filter-level

Option section: `Orchestration`

Configuration object: `ORS Application` object

Default value: `empty string`

Valid values: any combination of the digits 1,2,3,4,5 (minimum of 1 and maximum of 555555555)

Value changes: Immediately

If this option is configured and the level parameter of the `__Log` function or level attribute of the `<log>` element in the SCXML application equals to the option value, ORS generates corresponding log events of the Standard level.

For example, if `scxml-log-filter-level = 45`:

```
<log expr="'This is level 45...'" label="23011" level="45" />
```

or

```
__Log('This is level 45...', 23011, 45);
```

then the following log is generated:

```
Std 23011 METRIC <log sid='MGRMMP0T1L1JN5U4TN7N5M0FPS000002'
expr='This is level 45...' label='23011' level='45' />
```

session-hung-timeout

Option section: orchestration

Configuration object: ORS Application object

Default value: 0

Valid values: Any integer greater than or equal to 0

Value changes: Immediately

This option specifies the time (in seconds) that a hung session has to complete processing and exit gracefully before being terminated by the system. A value of 0 indicates that the session will not be forcibly terminated.

The ORS can detect the following conditions that represent a hung session:

- a session spends too long executing a script element.
- a session executes too many iterations of a loop (stuck in a loop) in a single ECMAScript script. This applies to `<script>`, `expr` or `cond`.

For example:

```
orchestration/ session-hung-timeout = 0 (no timeout)
```

```
orchestration/ session-hung-timeout = 3600 (1 hour)
```

```
orchestration/ session-hung-timeout = 604800 (1 week)
```

switch-multi-links-enabled

Option section: Orchestration

Configuration object: ORS Application object

Default value: false

Valid values: true or false.

Value changes: after restart

This option determines whether ORS supports multi-link switch configurations (load sharing Network T-Servers or IVR Servers). When set to true, ORS

enables support of multi-link switch configuration. When set to `false`, ORS does not enable multi-link switch support.

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

Load Sharing Switch Object

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

switch-multi-links-enabled

Default Value: `0`

Valid Values:

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+.

webfm-event-hold-response

Option section: `Orchestration`

Configuration object: `ORS Application` object

Default value: `true`

Valid values: `true`, `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` and `Reason-Phrase: OK`

If no transition has been selected, response contains headers `Status-Code: 204` and `Reason-Phrase: No Content`

If an error has occurred while processing the event, response contains headers `Status-Code: 400` and `Reason-Phrase: Bad Request`

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

Persistence Section

This section describes options in the `persistence` section.

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
```

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.

Note: Note that 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.

This is a semi-colon separated list of host names or IP addresses.

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: 100

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 Cassandra request latency time allowed before logging a message indicating that this maximum has been exceeded. If a subsequent request is within the maximum, a second message is logged indicating the condition has cleared.

For example:

```
persistence/cassandra-max-latency = 500
```

cassandra-nodes

Option section: persistence

Configuration object: ORS Application object

Default value: <empty string>

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

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

This option provides the Cassandra client connection port when using the Cassandra-based persistence method.

This is a semi-colon separated list of host names or IP addresses.

For example:

```
persistence/cassandra-nodes = DWS;mpswin;test47
```

cassandra-schema-version

Option section: persistence

Configuration object: ORS Application object

Default value: ORS813000

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 ORSdddddd, 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 = ORS813000
```

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/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
```

max-cache-count

Option section: persistence

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.

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

For example:

```
persistence/max-cache-count = 5000
```

max-cache-size

Option section: `persistence`

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 amount of memory (in bytes) that the internal Orchestration persistence cache can use.

For example:

```
persistence/max-cache-size = 100000
```

scxml Section

This section describes `scxml` section options.

debug-enabled

Option section: `scxml`

Configuration object: `ORS Application` object

Default value: `false`

Valid values: `Boolean`

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

This option allows you to specify whether debugging 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 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 activates Unicode support for List Objects. When this option is set to the valid name of a converter, ORS correctly processes the return value of the `getListItemValue` function if the return value contains non-English

fips_enable

Option section: `scxml`

Configuration object: ORS Application object

Default value: `false`

Valid values: `true`, `false`

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

Enables FIPS compliance in the SCXML library.

For example:

```
scxml/fips_enable = false
```

http-client-side-port-range

Option section: `scxml`

Configuration object: ORS Application object

Default value: `<empty string>`

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

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

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

For example:

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

http-enable-continue-header

Option section: `scxml`

Configuration object: `ORS Application object`

Default value: `false`

Valid values: `true`, `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-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 (100K)`

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 (10 MB)

Valid values: Any integer greater than or equal to 0

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

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

For example:

```
scxml/http-max-cache-size = 12500000
```

http-max-redirections

Option section: `scxml`

Configuration object: ORS Application 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-redirections = 10
```

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 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
```

Note: 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
```

Note: 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`, `false`

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

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

For example:

```
scxml/http-ssl-verify-peer = true
```

http-ssl-version

Option section: `scxml`

Configuration object: ORS Application object

Default value: `default`

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

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

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

For example:

```
scxml/http-ssl-version = SSLv2
```

https-proxy

Option section: scxml

Configuration object: ORS Application object

Default value: <empty string>

Valid values: Valid IP Address or URL: Port

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

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

For example:

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

js-preload-files

Option section: scxml

Configuration object: ORS Application object

Default value: No value

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

Value changes: On restart

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

max-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.

This option is an HTTP Cache-Control value which tells the ORS how long an application script can be cached. When `max-age` is provided in the request, it indicates that the client is willing to accept a response whose age is no greater than the specified time in seconds. Unless `http-max-stale` is also included, the client is not willing to accept a stale response.

If another interaction causes the same URL to be fetched, and it is within the `http-max-age` value specified, the cached version will be used instead of fetching a new version from the Application server.

Note: The application can only be cached if the URL is the same for a particular cached version. If application parameters are specified which are unique for each invocation (for example, an `i=[ANI]` parameter), the application will be fetched each time regardless of this value.

For example:

Application/ `max-age = 0` (no caching)

Application/ `max-age = 10` (application contents can be cached for 10 seconds)

Application/ `max-age = 60` (application contents can be cached for 1 minute)

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`

max-cache-entry-count

Option section: `scxml`

Configuration object: ORS Application object

Default value: `1000`

Valid values: Any integer greater than or equal to 0

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

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

For example:

```
scxml/http-max-cache-entry-count = 1250
```

max-cache-entry-size

Option section: `scxml`

Configuration object: ORS Application object

Default value: `100000` (100K)

Valid values: Any integer greater than or equal to 0

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

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

For example:

```
scxml/http-max-cache-entry-size = 125000
```

max-compiler-cache-size

Option section: `scxml`

Configuration object: ORS Application object

Default value: `10000000`

Valid values: non-negative integer

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

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

max-compiler-cached-docs

Option section: `scxml`

Configuration object: ORS Application object

Default value: `1000`

Valid values: non-negative integer

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

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

max-compiler-cached-doc-size

Option section: `scxml`

Configuration object: ORS Application object

Default value: `10000000`

Valid values: non-negative integer

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

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

max-debug-sessions

Option section: `scxml`

Configuration object: ORS Application object

Default value: 0 (maximum value of 128)

Valid values: an integer

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

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-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.

Example: `scxml/max-pending-events = 1000`

max-preprocessor-cache-size

Option section: `scxml`

Configuration object: ORS Application object

Default value: `10000000`

Valid values: Any integer greater than or equal to 0

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

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

For example:

`scxml/max-preprocessor-cache-size = 20000000`

max-preprocessor-cached-docs

Option section: `scxml`

Configuration object: ORS Application object

Default value: `1000`

Valid values: Any integer greater than or equal to 0

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

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

For example:

`scxml/max-preprocessor-cache-docs = 2000`

max-preprocessor-cached-doc-size

Option section: `scxml`

Configuration object: ORS Application object

Default value: `10000000`

Valid values: non-negative integer

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

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

max-session-age

Option section: `scxml`

Configuration object: ORS Application object

Default value: 500

Valid values: 0 to 10000

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.

For example:

```
scxml/max-includes = 200
```

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

max-stale

Option section: `scxml`

Configuration object: ORS Application object

Default value: 60 seconds

Valid values: Any integer greater than or equal to 0

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

This option is an HTTP Cache-Control value which enables caching for requests made via the Web FM. It is similar to `max-stale` option, and determines the number of seconds to extend the life of a cached file. The option `http-max-stale` indicates that the client is willing to accept a response that has exceeded its expiration time. If `http-max-stale` has a value, the client is willing to accept a stale response that has exceeded its expiration time by no more than the specified number of seconds. (without server re-validation).

For example, if the cached file would have expired 120 seconds ago but `http-max-stale` is set to 300, the local cached file will be sent back to the platform without first verifying the status of the file from the Application server.

If `http-max-stale` is 0 or is not specified, a response that has expired will not be accepted and server re-validation will be performed.

For example:

```
scxml/http-max-age = 0 (cached file that has exceeded expiration time is not accepted)
```

```
scxml/http-max-stale = 10 (cached file can exceed expiration time by 10 seconds before verifying status from the Application server)
```

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, false

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

When set to false, all persistence capabilities with the SCXML engine are suppressed.

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
```

stuck-thread-timeout

Option section: `scxml`

Configuration object: `ORS Application` object

Default value: 60000

Valid values: Any positive integer

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

This option specifies waiting time in milliseconds before declaring a session processing thread as being stuck.

For example:

```
scxml/stuck-thread-timeout = 30000 (equivalent to 30 seconds)
```

system-id

Option section: `scxml`

Configuration object: `ORS Application` object

Default value: -1

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

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

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

For example:

```
scxml/system-id = 11
```

log Section

This section describes log options specific to ORS.

x-server-trace-level

Option section: log

Configuration object: ORS Application object

Default value: 0

Valid values: 0 - 3

Value changes: as soon as committed to Configuration Server

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

For example:

```
log/x-server-trace-level = 2
```

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
```

Common Log Options

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

log Section

This section must be called log.

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

buffering

Default Value: true

Valid Values:

true	Enables buffering.
false	Disables buffering.

Changes Take Effect: Immediately

Turns on/off operating system file buffering. The option is applicable only to the `stderr` and `stdout` output (see [page 117](#)). Setting this option to `true` increases the output performance.

Note: When buffering is enabled, there might be a delay before log messages appear at the console.

check-point

Default Value: 1

Valid Values: 0–24

Changes Take Effect: Immediately

Specifies, in hours, how often the application generates a check point log event, to divide the log into sections of equal time. By default, the application generates this log event every hour. Setting the option to 0 prevents the generation of check-point events.

compatible-output-priority

Default Value: `false`

Valid Values:

<code>true</code>	The log of the level specified by “ Log Output Options ” is sent to the specified output.
<code>false</code>	The log of the level specified by “ Log Output Options ” and higher levels is sent to the specified output.

Changes Take Effect: Immediately

Specifies whether the application uses 6.x output logic. For example, you configure the following options in the `log` section for a 6.x application and for a 7.x application:

[log]

`verbose = all`

`debug = file1`

`standard = file2`

The log file content of a 6.x application is as follows:

- `file1` contains Debug messages only.
- `file2` contains Standard messages only.

The log file content of a 7.x application is as follows:

- `file1` contains Debug, Trace, Interaction, and Standard messages.
- `file2` contains Standard messages only.

If you set `compatible-output-priority` to `true` in the 7.x application, its log file content will be the same as for the 6.x application.

Warning! Genesys does not recommend changing the default value of this option unless you have specific reasons to use the 6.x log output logic—that is, to mimic the output priority as implemented in releases 6.x. Setting this option to `true` affects log consistency.

expire

Default Value: `false`

Valid Values:

<code>false</code>	No expiration; all generated segments are stored.
<code><number> file</code> or <code><number></code>	Sets the maximum number of log files to store. Specify a number from 1–1000.
<code><number> day</code>	Sets the maximum number of days before log files are deleted. Specify a number from 1–100.

Changes Take Effect: Immediately

Determines whether log files expire. If they do, sets the measurement for determining when they expire, along with the maximum number of files (segments) or days before the files are removed. This option is ignored if log output is not configured to be sent to a log file.

Note: If an option's value is set incorrectly—out of the range of valid values— it will be automatically reset to 10.

keep-startup-file

Default Value: `false`

Valid Values:

<code>false</code>	No startup segment of the log is kept.
<code>true</code>	A startup segment of the log is kept. The size of the segment equals the value of the <code>segment</code> option.
<code><number> KB</code>	Sets the maximum size, in kilobytes, for a startup segment of the log.
<code><number> MB</code>	Sets the maximum size, in megabytes, for a startup segment of the log.

Changes Take Effect: After restart

Specifies whether a startup segment of the log, containing the initial configuration options, is to be kept. If it is, this option can be set to `true` or to a specific size. If set to `true`, the size of the initial segment will be equal to the size of the regular log segment defined by the `segment` option. The value of this option will be ignored if segmentation is turned off (that is, if the `segment` option is set to `false`).

memory

Default Value: No default value

Valid Values: <string> (memory file name)

Changes Take Effect: Immediately

Specifies the name of the file to which the application regularly prints a snapshot of the memory output, if it is configured to do this (see “Log Output Options” on [page 117](#)). The new snapshot overwrites the previously written data. If the application terminates abnormally, this file will contain the latest log messages. Memory output is not recommended for processors with a CPU frequency lower than 600 MHz.

Note: If the file specified as the memory file is located on a network drive, an application does not create a snapshot file (with the extension *.memory.log). Log output to a file at a network location is not recommended and could cause performance degradation.

memory-storage-size

Default Value: 2 MB

Valid Values:

<number> KB or <number> The size of the memory output, in kilobytes.
The minimum value is 128 KB.

<number> MB The size of the memory output, in megabytes.
The maximum value is 64 MB.

Changes Take Effect: When memory output is created

Specifies the buffer size for log output to the memory, if configured. See also “Log Output Options” on [page 117](#).

message_format

Default Value: short

Valid Values:

short An application uses compressed headers when writing log records in its log file.

full An application uses complete headers when writing log records in its log file.

Changes Take Effect: Immediately

Specifies the format of log record headers that an application uses when writing logs in the log file. Using compressed log record headers improves application performance and reduces the log file’s size.

With the value set to short:

- A header of the log file or the log file segment contains information about the application (such as the application name, application type, host type, and time zone), whereas single log records within the file or segment omit this information.
- A log message priority is abbreviated to Std, Int, Trc, or Dbg, for Standard, Interaction, Trace, or Debug messages, respectively.
- The message ID does not contain the prefix GCTI or the application type ID.

A log record in the full format looks like this:

```
2002-05-07T18:11:38.196 Standard localhost cfg_dbserver GCTI-00-05060
Application started
```

A log record in the short format looks like this:

```
2002-05-07T18:15:33.952 Std 05060 Application started
```

Note: Whether the full or short format is used, time is printed in the format specified by the `time_format` option.

messagefile

Default Value: As specified by a particular application

Valid Values: <string>.lms (message file name)

Changes Take Effect: Immediately, if an application cannot find its *.lms file at startup

Specifies the file name for application-specific log events. The name must be valid for the operating system on which the application is running. The option value can also contain the absolute path to the application-specific *.lms file. Otherwise, an application looks for the file in its working directory.

Warning! An application that does not find its *.lms file at startup cannot generate application-specific log events and send them to Message Server.

print-attributes

Default Value: false

Valid Values:

true Attaches extended attributes, if any exist, to a log event sent to log output.

false Does not attach extended attributes to a log event sent to log output.

Changes Take Effect: Immediately

Specifies whether the application attaches extended attributes, if any exist, to a log event that it sends to log output. Typically, log events of the Interaction log level and Audit-related log events contain extended attributes. Setting this option to true enables audit capabilities, but negatively affects performance. Genesys recommends enabling this option for Solution Control Server and

Configuration Server when using audit tracking. For other applications, refer to *Genesys Combined Log Events Help* to find out whether an application generates Interaction-level and Audit-related log events; if it does, enable the option only when testing new interaction scenarios.

segment

Default Value: `false`

Valid Values:

<code>false</code>	No segmentation is allowed.
<code><number> KB</code> or <code><number></code>	Sets the maximum segment size, in kilobytes. The minimum segment size is 100 KB.
<code><number> MB</code>	Sets the maximum segment size, in megabytes.
<code><number> hr</code>	Sets the number of hours for the segment to stay open. The minimum number is 1 hour.

Changes Take Effect: Immediately

Specifies whether there is a segmentation limit for a log file. If there is, sets the mode of measurement, along with the maximum size. If the current log segment exceeds the size set by this option, the file is closed and a new one is created. This option is ignored if log output is not configured to be sent to a log file.

spool

Default Value: The application's working directory

Valid Values: `<path>` (the folder, with the full path to it)

Changes Take Effect: Immediately

Specifies the folder, including full path to it, in which an application creates temporary files related to network log output. If you change the option value while the application is running, the change does not affect the currently open network output.

time_convert

Default Value: `Local`

Valid Values:

<code>local</code>	The time of log record generation is expressed as a local time, based on the time zone and any seasonal adjustments. Time zone information of the application's host computer is used.
<code>utc</code>	The time of log record generation is expressed as Coordinated Universal Time (UTC).

Changes Take Effect: Immediately

Specifies the system in which an application calculates the log record time when generating a log file. The time is converted from the time in seconds since the Epoch (00:00:00 UTC, January 1, 1970).

time_format

Default Value: `time`

Valid Values:

<code>time</code>	The time string is formatted according to the HH:MM:SS.sss (hours, minutes, seconds, and milliseconds) format.
<code>locale</code>	The time string is formatted according to the system's locale.
<code>ISO8601</code>	The date in the time string is formatted according to the ISO 8601 format. Fractional seconds are given in milliseconds.

Changes Take Effect: Immediately

Specifies how to represent, in a log file, the time when an application generates log records.

A log record's time field in the ISO 8601 format looks like this:

`2001-07-24T04:58:10.123`

verbose

Default Value: `all`

Valid Values:

<code>all</code>	All log events (that is, log events of the Standard, Trace, Interaction, and Debug levels) are generated.
<code>debug</code>	The same as <code>all</code> .
<code>trace</code>	Log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels) are generated, but log events of the Debug level are not generated.
<code>interaction</code>	Log events of the Interaction level and higher (that is, log events of the Standard and Interaction levels) are generated, but log events of the Trace and Debug levels are not generated.
<code>standard</code>	Log events of the Standard level are generated, but log events of the Interaction, Trace, and Debug levels are not generated.
<code>none</code>	No output is produced.

Changes Take Effect: Immediately

Determines whether a log output is created. If it is, specifies the minimum level of log events generated. The log events levels, starting with the highest priority level, are Standard, Interaction, Trace, and Debug. See also "Log Output Options" on [page 117](#).

Note: For definitions of the Standard, Interaction, Trace, and Debug log levels, refer to the *Framework Management Layer User's Guide*, *Framework Genesys Administrator Help*, or to *Framework Solution Control Interface Help*.

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.

Warnings!

- 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.

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

all

Default Value: No default value

Valid Values (log output types):

<code>stdout</code>	Log events are sent to the Standard output (<code>stdout</code>).
<code>stderr</code>	Log events are sent to the Standard error output (<code>stderr</code>).
<code>network</code>	Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. Setting the <code>all</code> log level option to the <code>network</code> output enables an application to send log events of the <code>Standard</code> , <code>Interaction</code> , and <code>Trace</code> levels to Message Server. <code>Debug</code> -level log events are neither sent to Message Server nor stored in the Log Database.
<code>memory</code>	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
<code>[filename]</code>	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which an application sends all log events. The log output types must be separated by a comma when more than one output is configured. For example:

```
all = stdout, logfile
```

Note: To ease the troubleshooting process, consider using unique names for log files that different applications generate.

alarm

Default Value: No default value

Valid Values (log output types):

stdout	Log events are sent to the Standard output (stdout).
stderr	Log events are sent to the Standard error output (stderr).
network	Log events are sent to Message Server, which resides anywhere on the network, and Message Server stores the log events in the Log Database.
memory	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
[filename]	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which an application sends the log events of the Alarm level. The log output types must be separated by a comma when more than one output is configured. For example:

```
standard = stderr, network
```

standard

Default Value: No default value

Valid Values (log output types):

stdout	Log events are sent to the Standard output (stdout).
stderr	Log events are sent to the Standard error output (stderr).
network	Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database.
memory	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
[filename]	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which an application sends the log events of the Standard level. The log output types must be separated by a comma when more than one output is configured. For example:

```
standard = stderr, network
```

interaction

Default Value: No default value

Valid Values (log output types):

<code>stdout</code>	Log events are sent to the Standard output (<code>stdout</code>).
<code>stderr</code>	Log events are sent to the Standard error output (<code>stderr</code>).
<code>network</code>	Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database.
<code>memory</code>	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
<code>[filename]</code>	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which an application sends the log events of the Interaction level and higher (that is, log events of the Standard and Interaction levels). The log outputs must be separated by a comma when more than one output is configured. For example:

```
interaction = stderr, network
```

trace

Default Value: No default value

Valid Values (log output types):

<code>stdout</code>	Log events are sent to the Standard output (<code>stdout</code>).
<code>stderr</code>	Log events are sent to the Standard error output (<code>stderr</code>).
<code>network</code>	Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database.
<code>memory</code>	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
<code>[filename]</code>	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which an application sends the log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace

levels). The log outputs must be separated by a comma when more than one output is configured. For example:

```
trace = stderr, network
```

debug

Default Value: No default value

Valid Values (log output types):

<code>stdout</code>	Log events are sent to the Standard output (<code>stdout</code>).
<code>stderr</code>	Log events are sent to the Standard error output (<code>stderr</code>).
<code>memory</code>	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
<code>[filename]</code>	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which an application sends the log events of the Debug level and higher (that is, log events of the Standard, Interaction, Trace, and Debug levels). The log output types must be separated by a comma when more than one output is configured—for example:

```
debug = stderr, /usr/local/genesys/logfile
```

Note: Debug-level log events are never sent to Message Server or stored in the Log Database.

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 Technical Support 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`.

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`, `false`

Value changes: take effect immediately.

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

For example:

```
mcr/om-memory-optimization = true
```

om-max-in-memory

Option section: mcr

Configuration object: ORS Application object

Default value: `100`

Valid values: `1` to `2000`

Value changes: take effect immediately.

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

For example:

```
mcr/om-max-in-memory = 100
```

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 calls should be deleted when the value of om-max-in-memory has been attained.

For example:

```
mcr/om-delete-from-memory = 1
```

Switch Object, gts Section

See “Limitation <createcall> and <consult>” on [page 140](#) for information on the following Switch options:

valid-digits

max-digits

DN-Level Options

This section describes options to be configured on the DN level.

Orchestration section

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

application

Option section: Orchestration

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

Default value: none (this is a required option)

Valid values: any valid URL

Value changes: For the next interaction that hits this resource

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

- file:<path>
- http://<url>
- script:<name of script object >

The use of `script:` allows an indirect reference to a script object of type `CfgEnhanced Routing`, which can contain the application URL, parameters, and other configuration values. The URL can also contain *parameters* which will be passed to the Application server. The values shown in Table 9 on [page 123](#) are substituted at run-time based on the information in the interaction.

Table 9: URL Parameter Elements for application option

Formatting Element	Description
[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:

```
application = http://xserver.genesyslab.com:80/NewCallReq.asp
application=
http://xserver.genesyslab.com:80/NewCallReq.asp?ani=[ANI]&dnis=[DNIS]&s
ervicetype=[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
```

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

send-retries

Option section: `Orchestration`

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

Default value: `2`

Valid values: `0-10`

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

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

Example: `orchestration / send-retries = 1`

Enhanced Routing Script Options

Application Section

This section describes Enhanced Routing Script object Application section options.

alternate-url

Option section: `Application`

Configuration object: `Script object (CfgEnhancedRouting)`

Default value: `none` (This option is not required)

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 in [Table 11](#) are substituted at run-time based on the information in the interaction.

Table 10: URL Parameter Elements for alternate-url option

Formatting Element	Description
[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 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]

fetch-timeout

Option section: Application

Configuration object: Script object (CfgEnhancedRouting)

Default value: 5000 [5000 ms is 5 seconds]

Valid values: Any integer greater than or equal to 0

Value changes: For the next document fetch

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

The actual fetch waiting time can be up to 10 ms more than specified. If fetch-timeout is 0 or is not specified, the system will wait indefinitely for the document to be fetched.

For example:

Application/ fetch-timeout = 0 (no timeout)

Application/ fetch-timeout = 500 (wait up to 500 ms to fetch the document)

Application/ fetch-timeout = 60000 (wait up to 1 minute)

http-useragent

Option section: Application

Configuration object: Script object (CfgEnhancedRouting)

Default value: GOES/8.1

Valid values: Any string

Value changes: For the next document fetch

This option specifies the string to use in the HTTP header field User Agent, which identifies the application to the web server.

For example:

http-useragent = 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 documents from the application server.

For example:

http-version = 1.1

http-max-age

Option section: Application

Configuration object: Script object (CfgEnhancedRouting)

Default value: The value specified in ORS configuration for `scxml/http-max-age`.

Valid values: Any integer greater than or equal to 0

Value changes: For the next document fetch

This option is an HTTP Cache-Control value which tells the ORS how long an application script can be cached. If another interaction causes the same URL to be fetched, and it is within the `http-max-age` value specified, the cached version will be used instead of fetching a new version from the Application server.

Note: The application can only be cached if the URL is the same for a particular cached version. If application parameters are specified which are unique for each invocation (for example, `ani=[ANI]` parameter), the application will be fetched each time regardless of this value.

The value of `max-age` is the time to cache *in seconds*. If `max-age` is 0 or is not specified, the system will not cache this application script.

For example:

Application/ `max-age` = 0 (no caching)

Application/ `max-age` = 1000 (application contents can be cached for 1 second)

Application/ `max-age` = 60000 (application contents can be cached for 1 minute)

http-max-stale

Option section: Application

Configuration object: Script object (CfgEnhancedRouting)

Default value: The value specified in ORS configuration for `scxml/http-max-stale`.

Valid values: Any integer greater than or equal to 0

Value changes: For the next document fetch

This option determines the number of seconds to extend the life of a cached file. If the cached file would have expired 120 seconds ago, but `max-stale` is set to 300, the local cached file will be sent back to the platform without first verifying the status of the file from the Application server.

If `max-stale` is 0 or is not specified, the system will not cache this application.

For example:

Application/ `max-stale` = 0 (no caching)

Application/ max-stale = 1000 (application contents can be 1 second old before verifying the status from the Application server)

url

Option section: Application

Configuration object: Script object (CfgEnhancedRouting)

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 [Table 11](#) are substituted at run-time based on the information in the interaction.

Table 11: URL Parameter Elements for url option

Formatting Element	Description
[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&·

Table 11: URL Parameter Elements for url option (Continued)

Formatting Element	Description
[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]

{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, as specified in Table 12 on [page 129](#), for example:

Segment = [UDATA.CustomerSegment]

Table 12: Parameter Elements for ApplicationParms

Formatting Element	Description
[DNIS]	The DNIS attribute of the interaction
[ANI]	The ANI attribute of the interaction

Table 12: Parameter Elements for ApplicationParms (Continued)

Formatting Element	Description
[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: [UDATA.ServiceType] could resolve to: "CreditCards"

For example:

ApplicationParms =

Service = Sales

EmailSubject = [EMAILSUBJECT]

Segment = [UDATA.CustomerSegment]

AppServer = http://myappserver:8080

6

Business Logic for Advice of Charge

This chapter describes how Orchestration Server supports the Business Logic for Advice of Charge (AoC) and includes the following topic:

- [Implementation of Business Logic for Advice of Charge, page 131](#)

Implementation of Business Logic for Advice of Charge

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.

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

[Table 13](#) contains the `<privateservice>` action attributes and their descriptions. There are no default values for any of these attributes.

Table 13: Private Service Request Attributes

Attribute name	Type	Valid Values	Description
<code>requestid</code> (not required)	Location expression	Any valid location expression	Represents the location of the request ID that is returned in the request. Any data model expression evaluating to a data model location. The location's value is configured as an internally generated unique string identifier that is associated with the action that is sent. If this attribute is not specified, the event identifier is dropped. This identifier can be tested by the completion event handler to distinguish between multiple outstanding requests. If this attribute is not specified, the identifier can be acquired from the fetch completion event. Every request must receive a unique identifier.
<code>serviceid</code> (required)	Value expression	Any value expression that returns a valid integer	A value expression that returns an integer to indicate the type of information that is passed or the service that is requested. It is specific to the T-Server that is handling the call. Before you configure this value, check the T-Server documentation for your switch.
<code>interactionid</code> (required)	Value expression	Any value expression that returns a valid integer	A value expression that returns the <code>_genesys.FMname.interactions[x].g_guid</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.
<code>resource</code> (not required)	Value expression	Any valid string or resource object	A value expression that returns the DN of the controlling agent or route point for which the information is provided. This attribute corresponds to the <code>thisDN</code> parameter within the <code>TLibTPrivateService</code> method. Before you configure this value, check the T-Server documentation for your switch. The <code><ixn:privateservice></code> resource attribute is mandatory in this release.
<code>udata</code> (not required)	ECMAScript object	Any valid ECMAScript object	An ECMAScript object that contains the list of key/value pairs are attached to the call in question.
<code>reasons</code> (not required)	ECMAScript object	Any valid ECMAScript object	An ECMAScript object that contains the list of key/value pairs that provide additional information associated with this Private Service request, and is intended to specify reasons for and results of actions taken by the user.

Table 13: Private Service Request Attributes (Continued)

Attribute name	Type	Valid Values	Description
extensions (not required)	ECMAScript object	Any valid ECMAScript object	An ECMAScript object that contains the list of key/value pairs that provide an additional data structure that is intended to take into account the switch-specific features that cannot be described by other parameters, or the list of key/value pairs in the original structure of the user data that is associated with this Private Service request.
hints (not required)	Value expression	Any valid ECMAScript object	A value expression that returns the ECMAScript object that contains information that might be used by the implementing functional module while performing this action. This information might consist of protocol-specific parameters, protocol selection guidelines, or other related data. The meaning of these hints is specific to the implementing functional module.
Note: For more information about the expressions and data values in this table, see the <i>State Chart XML (SCXML): State Machine Notation for Control Abstraction, W3C Working Draft 7 May 2009</i> W3C Specification.			

Determining the Target T-Server

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

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

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

```
<state id="do_private_service">
  <datamodel>
    <data id="reqid"/>
```

```

</datamodel>
<onentry>

    <script>
        var myuserdata = {details : {name: "Smith, John", age
: 45} };

        var myreasons = {code: "New Update"};
        var myextensions = { keyname : "Its value"};
        var myhints = {handle_responses : "false"};
    </script>

    <ixn:privateservice requestid="_data.reqid"
        serviceid="1234"
        interactionid="_genesys.ixn.interactions[0].g_uid"
        resource="'9000'"
        udate = "myuserdata"
        reasons = "myreasons"
        extensions = "myextensions"/>

</onentry>
<transition event="voice.privateservice.done" target="statex"/>
<transition event="error.voice.privateservice" target="statey"/>
</state>

```

Children

There are no child objects.

Events

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

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



Chapter

7

SCXML Application Development

This chapter provides an overview of SCXML application development and contains the following topics:

- [Creating SCXML-Based Applications, page 135](#)
- [Deployment Procedures, page 136](#)
- [Debugging SCXML Applications with Composer, page 139](#)
- [Samples, page 140](#)

Creating SCXML-Based Applications

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.
- The Genesys Composer GUI, 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 *Genesys Composer 8.1.x Help*. Also see the *Genesys Composer 8.1 Deployment Guide*. Both can be accessed from the [Composer Documentation Wiki](#).

Deployment Procedures

Task Summary: SCXML Application Development

Objective	Related Procedure
Deploy a voice application in Composer	Procedure: Deploying a Voice SCXML Application in Composer, on page 136
Manually deploy a voice application	Procedure: Manually Deploying a Voice SCXML Application, on page 139

Deploying Voice SCXML Applications

This section describes how to deploy a voice SCXML application created in Composer.

Procedure: Deploying a Voice SCXML Application in Composer

Start of procedure

1. Within Composer, click toolbar icon to create a new Java Composer Project.
2. Specify the Route type. Specifying Route indicates the development ORS/URS routing strategies.
3. Build the interaction process diagram, which was created automatically in step 1 above (default name: `default.ixnprocess`).
4. In the interaction process diagram (`default.ixnprocess`), locate the Resource property, and point to the workflow (`Workflows/default.workflow`), which can be developed later.
5. Within Composer, connect to Configuration Server so you can view/select objects, such as agents, places, and so on.
6. Develop your workflow diagram (routing strategy) using the diagram-building blocks from the palette, such as from the Flow Control and Routing block categories.
7. Save the workflow and interaction process diagrams.
8. Validate the workflow and interaction process diagrams.
9. View any warnings or problem messages and correct.

10. Generate the code. As a result, two SCXML files are generated and can be viewed in an SCXML editor.
11. Publish the interaction process diagram to Configuration Server. An Enhanced Routing Script object will be created. [Figure 25](#) shows an example.

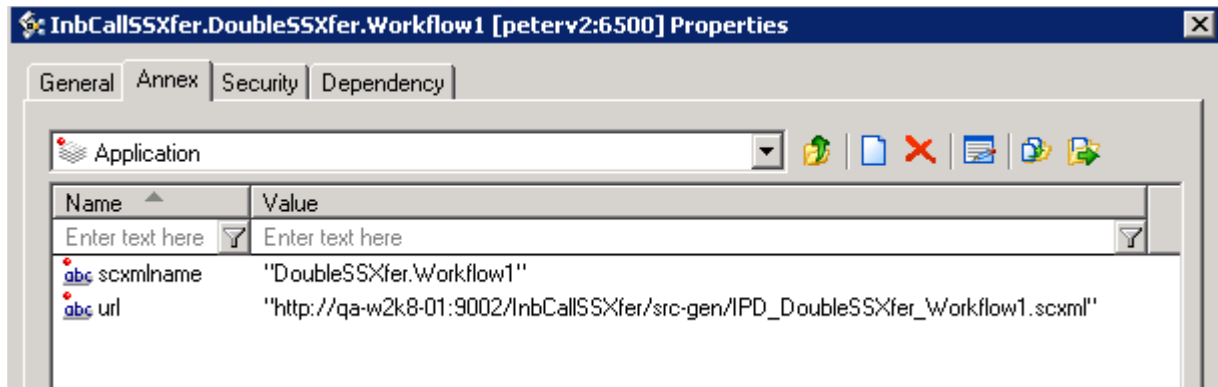


Figure 25: Annex Tab, Enhanced Routing Script Object

[Figure 25](#) 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. [Figure 26](#) shows an example.

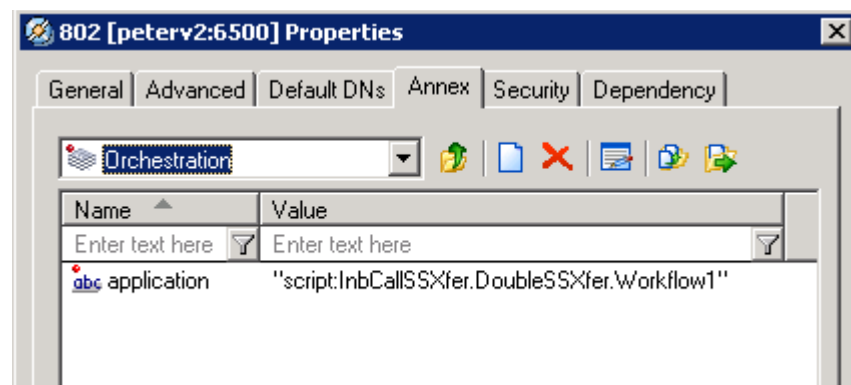


Figure 26: Route Application Loaded on a Routing Point

[Figure 26](#) shows the Routing Point with the `application` option set to the name of the Enhanced Routing Script object created in [Step 11](#) on [page 137](#) above.

- a. Log into Genesys Administrator (see [Figure 27](#)).

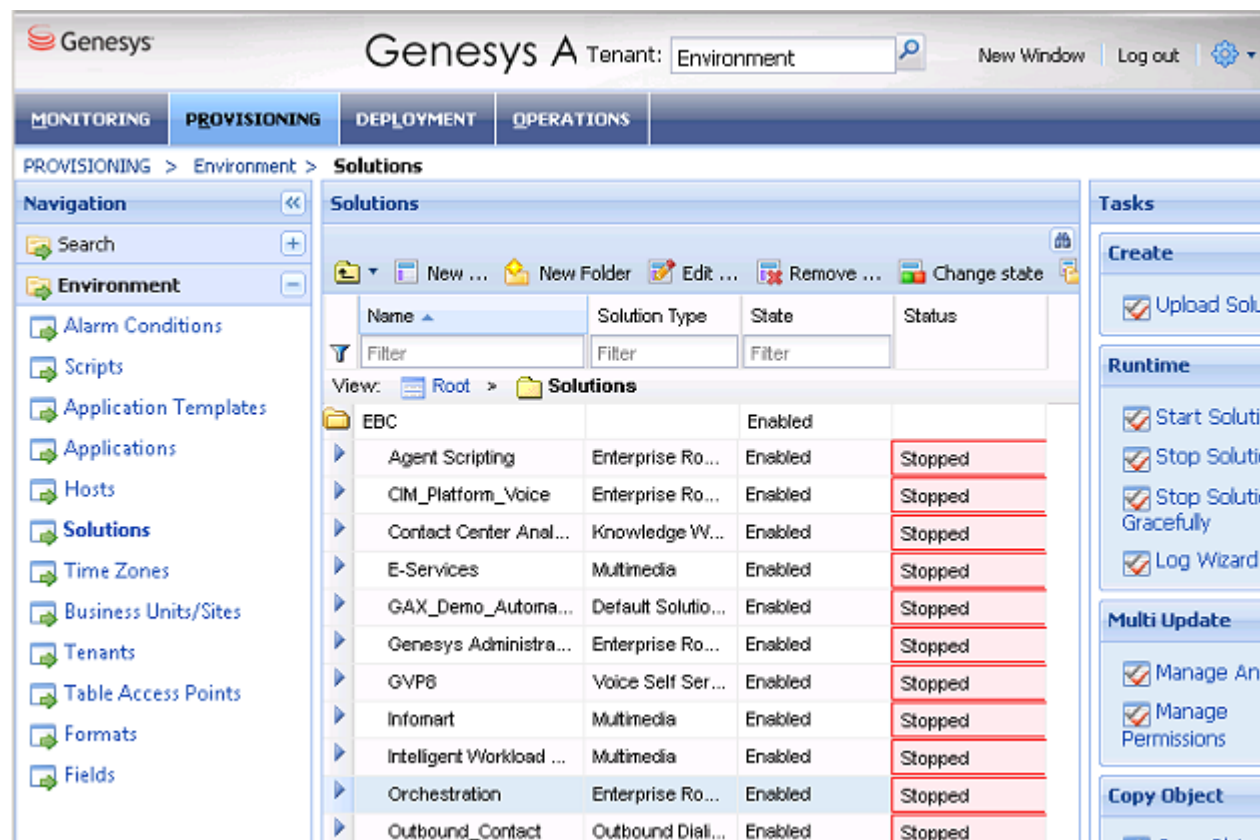


Figure 27: Genesys Administrator

- b. Navigate to Provisioning>Routing/eServices >Orchestration
 - c. Locate the Enhanced Routing Script object.
 - d. Click on DNS tab.
 - e. Add your DN (as in example Routing Point).
 - f. Load this application to this Routing Point (see [Figure 26](#)).
13. Test that the application is successfully provisioned and deployed. Make a test call.

End of procedure

Manually Deploying a Voice SCXML Application

This section describes how to manually deploy a voice SCXML application.

Procedure: Manually Deploying a Voice SCXML Application

Start of procedure

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. See [Procedure: Manually loading an SCXML application on a DN](#), on [page 69](#).

End of procedure

Debugging SCXML Applications with Composer

The SCXML Real Time Debugging (RTD) feature provides possibility for the 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:

- “debug-port” on [page 93](#)
- “debug-enabled” on [page 93](#)
- “max-debug-sessions” on [page 104](#)

The debugging SCXML applications functionality is supported in Composer starting from v8.1.2. For more information, see [Composer 8.1 Deployment Guide](#).

Limitation <createcall> and <consult>

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



Chapter

8

Installing Orchestration Server

This chapter includes the following topics:

- [Installation Package Location, page 142](#)
- [Installing on Windows Operating Systems, page 142](#)
- [Installing on UNIX-Based Platforms, page 146](#)

Note: 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](https://docs.genesyslabs.com) at docs.genesyslabs.com.

Warning! Genesys does not recommend installation of its components via a Microsoft Remote Desktop connection. The installation should be performed locally.

Task Summary: Installing Orchestration Server

Objective	Related Procedures and Activities
Locate the installation package	See “Installation Package Location” on page 142
Install components on Windows	Procedure: Installing Orchestration Server on Windows using the Installation Wizard, on page 142
Install components on Linux	Procedure: Installing Orchestration Server on a UNIX platform, on page 147

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.

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 Operating Systems

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 Multi-media/eServices components) are installed as services in automatic startup mode.

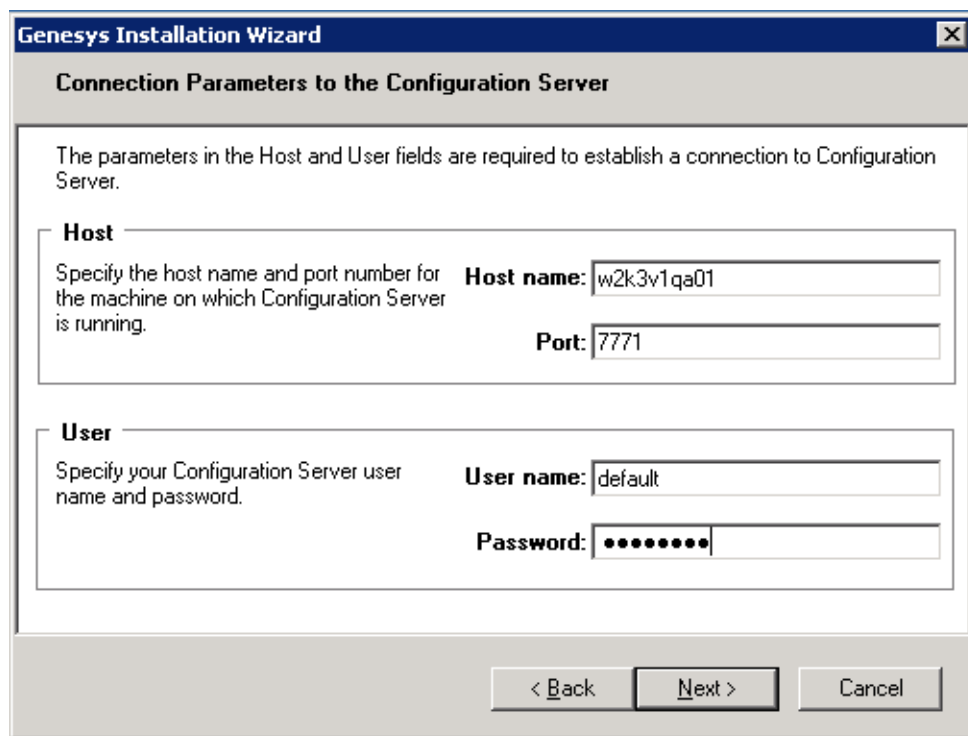
Procedure: Installing Orchestration Server on Windows using the Installation Wizard

Start of procedure

1. Double-click `setup.exe`.
 - If Orchestration Server was downloaded from an FTP site, the file is located in the download directory.

The Install Shield opens the Welcome screen.

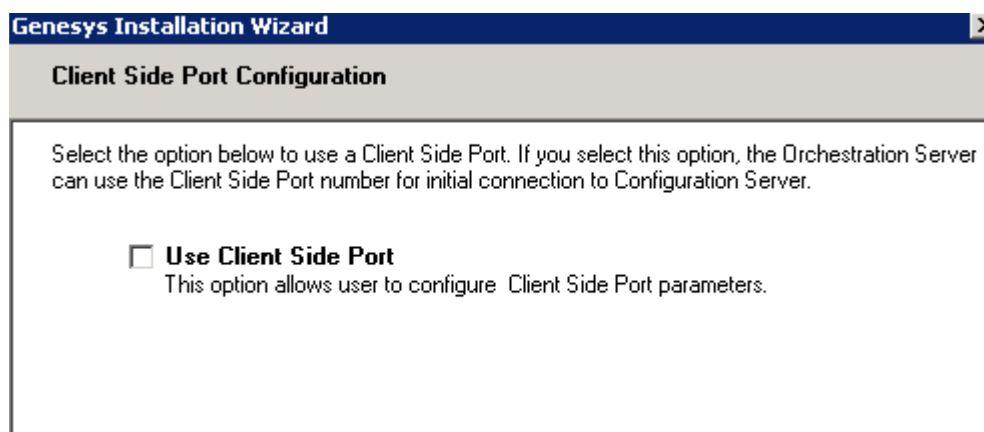
2. Click Next. The Connection Parameters to the Configuration Server screen appears (see [Figure 28](#)).



The screenshot shows a window titled "Genesys Installation Wizard" with a subtitle "Connection Parameters to the Configuration Server". Below the subtitle, a message states: "The parameters in the Host and User fields are required to establish a connection to Configuration Server." There are two main sections: "Host" and "User". The "Host" section has a text box for "Host name" containing "w2k3v1qa01" and a text box for "Port" containing "7771". The "User" section has a text box for "User name" containing "default" and a text box for "Password" containing eight dots. At the bottom right, there are three buttons: "< Back", "Next >", and "Cancel".

Figure 28: Connection Parameters to the Genesys Configuration Server

3. 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.
4. Under User, enter the user name and password used for logging on to Configuration Server.
5. Click Next to open the Client Side Port Configuration screen. (see [Figure 29](#)).



The screenshot shows a window titled "Genesys Installation Wizard" with a subtitle "Client Side Port Configuration". Below the subtitle, a message states: "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." There is a checkbox labeled "Use Client Side Port" which is currently unchecked. Below the checkbox, a text box contains the text: "This option allows user to configure Client Side Port parameters."

Figure 29: Client-Side Port Configuration

6. 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. See [Figure 30](#).

Genesys Installation Wizard

Client Side Port Configuration

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

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

Configuration Options

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

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

< Back Next > Cancel

Figure 30: Client-Side Port Configuration Screen, Configuration Options

7. Specify the following parameters:
 - **Port**—Enter any free port number (this is *not* the Listening port in the `Server Info` tab of the Orchestration Server Application object).
 - **IP Address**—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.

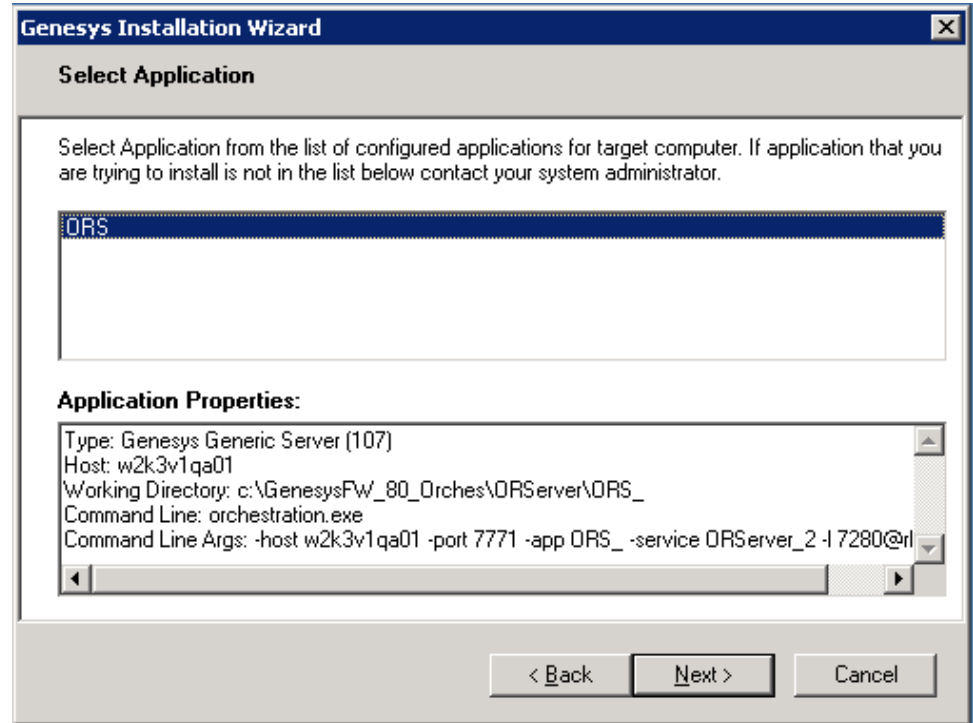


Figure 31: Select Application

8. Click Next. The Select Application screen appears. [Figure 31](#) shows example entries.
9. On the Select Application screen, 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.
10. Click Next. The Choose Destination Location screen appears (see [Figure 32](#)).

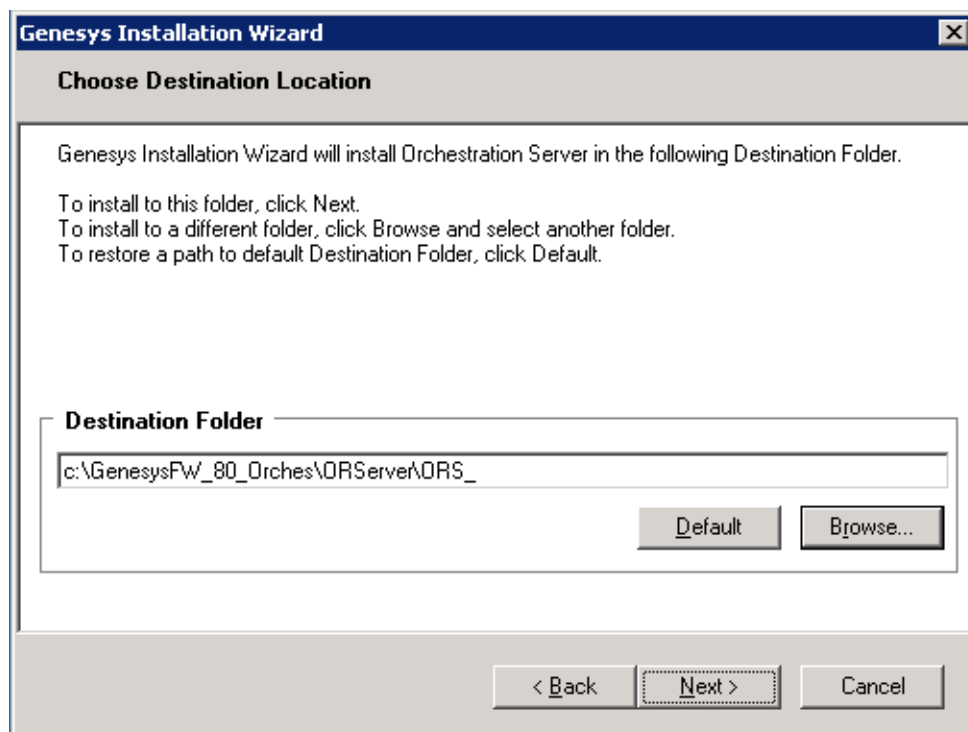


Figure 32: Choose Destination Location

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

End of procedure

Installing on UNIX-Based Platforms

Warning! 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.

Installing Orchestration Server on a UNIX-Based Platform

Complete the procedures in this section to install Orchestration Server on a UNIX platform. This release of Orchestration Server supports:

- Red Hat Enterprise Linux AS 5

Procedure: Installing Orchestration Server on a UNIX platform

Start of procedure

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` (see “Installation Package Location” on [page 142](#)).
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
6. 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)`
7. When prompted, type either Y for yes or N for no. The instructions below assume you typed Y.
8. 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:`
9. 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>

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

Please enter full path of the destination directory for installation:

After you enter the destination directory, the installation continues. A message appears that starts with `xtracting tarfile: d`

11. 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 type that you use.

End of procedure

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.

9

Starting and Stopping Procedures

This chapter provides instructions for starting and stopping Orchestration Server (ORS) either with the Solution Control Interface (SCI), or manually.

This chapter includes the following topics:

- [Starting Orchestration Server, page 149](#)
- [Stopping, page 152](#)
- [Non-Stop Operation, page 153](#)
- [Version Identification, page 154](#)

Starting Orchestration Server

Task Summary: Starting Orchestration Server

Objective	Related Procedures and Actions
Start Orchestration Server using Solution Control Interface	Procedure: Starting Orchestration Server with Solution Control Interface, on page 150
Start Orchestration Server on UNIX-Based Platforms	Procedure: Starting Orchestration Server on UNIX-Based Platforms, on page 151

Start Orchestration Server from the Solution Control Interface (SCI). [Figure 33](#) shows an example that could include Orchestration Server as well as other server applications.

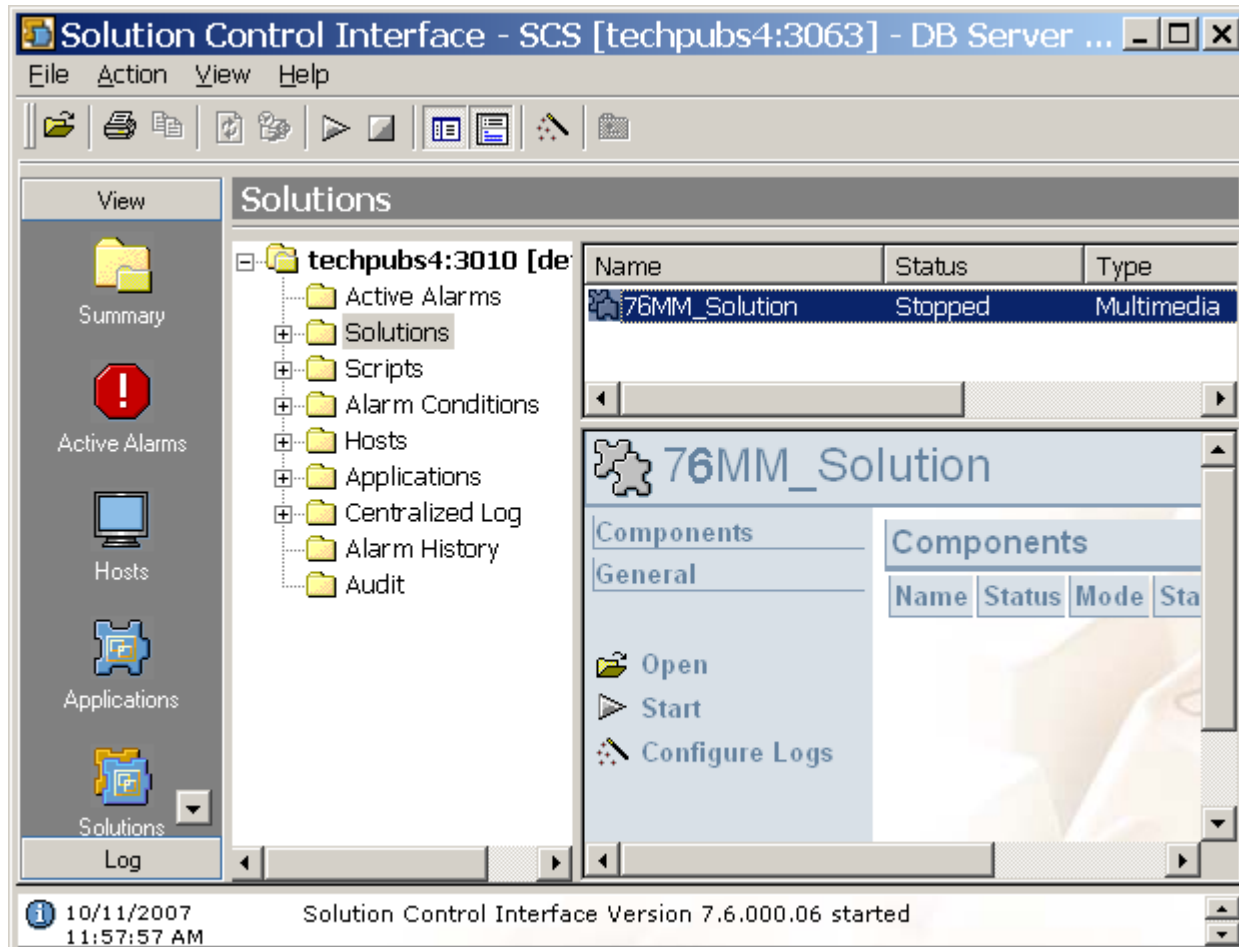


Figure 33: Solution Control Interface

Procedure: Starting Orchestration Server with Solution Control Interface

Start of procedure

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

End of procedure

Starting Orchestration Server Manually

This section describes how to manually start Orchestration Server (ORS). 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.

Note: The above 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".

Procedure: Starting Orchestration Server on UNIX-Based Platforms

Installation of ORS creates a `run.sh` file.

You can start Orchestration Server on UNIX platforms by just running this file which contains:

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

Start of procedure

1. Open a terminal window.
2. Log in.
3. Choose the appropriate directory.

4. Run the `run.sh` file.

End of procedure

Note: The text in angle brackets (<text>) above indicates the variables you enter that are unique to your environment and are required. Your information should replace the text and the brackets. See the example below for clarification.

The following is an example of a `run.sh` file:

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

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

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

Stopping

This section describes how to stop Orchestration Server by using the Solution Control Interface

Task Summary: Stopping Orchestration Server

Objective	Related Procedures and Activities
Stop Orchestration Server using Solution Control Interface	Procedure: Stopping Orchestration Server using Solution Control Interface, on page 152

Procedure: Stopping Orchestration Server using Solution Control Interface

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

Start of procedure

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
- Select the desired application and choose `Action > Stop` on the menu bar.

End of procedure

The command to stop the application is sent to Solution Control Server, which uses Local Control Agent to terminate the application.

SCI reports a successful stop of the application. When stopped, Orchestration Server's status changes from `Started` to `Stopped`.

Non-Stop Operation

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

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

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

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

`-nco xcount/xthreshold`

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

`-nco`

Starts NCO with default parameters (six faults in 10 seconds)

Examples:

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

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

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

Version Identification

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

10

Uninstalling Orchestration Server

This chapter describes how to uninstall Orchestration Server. It includes the following topics:

- [Removing Orchestration Server with Genesys Administrator, page 156](#)
- [Removing Orchestration Server Manually, page 156](#)

Task Summary: Uninstalling Orchestration Server

Objective	Related Procedures and Activities
Uninstall Orchestration Server in Genesys Administrator	Procedure: Removing the Orchestration Server component with Genesys Administrator, on page 156
Uninstall Orchestration Server manually	Procedure: Manually removing Orchestration Server on Windows, on page 156 Procedure: Manually removing Orchestration Server on UNIX-Based Platforms, on page 157

Removing Orchestration Server with Genesys Administrator

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

Procedure: Removing the Orchestration Server component with Genesys Administrator

Purpose:

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

Start of procedure

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

End of procedure

Removing Orchestration Server Manually

This section describes how to remove the Orchestration Server manually.

Procedure: Manually removing Orchestration Server on Windows

Start of procedure

Do the following on each machine that hosts Orchestration Server:

1. From the Windows Start menu, open the Control Panel (Start > 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.

End of procedure

Procedure:

Manually removing Orchestration Server on UNIX-Based Platforms

Start of procedure

- For the directory in which each Orchestration Server component is installed, run the following command:

```
rm -R
```

Note: If an instance of the component is running, the command will fail.

End of procedure



Appendix

Installing and Configuring Apache Cassandra Server

This Appendix describes how to install and configure Apache Cassandra Server in a single-node cluster or a multi-node cluster to prepare it for use with Orchestration Server 8.1.3.

For the most current information on Orchestration Server support of Apache Cassandra, see the [Cassandra Installation/Configuration Guide](#) on the Orchestration Server page of the Genesys Documentation website at www.docs.genesyslab.com.

This Appendix includes the following sections:

- [Recommendations/Requirements, page 159](#)
- [Cassandra Overview, page 160](#)
- [Installing, Configuring, Starting, and Testing a Cassandra Node, page 161](#)
- [Configuring Cassandra Server, page 173](#)

Recommendations/Requirements

Genesys recommends that all nodes in a Cassandra cluster be deployed on the same platform type (all Windows-based or all Linux-based). The operating system version is not a factor in this consideration, except for the minimum operating system version that Cassandra requires.

- ORS supports the latest 1.1.x version of Cassandra. To download Apache Cassandra, see cassandra.apache.org.

Orchestration Server 8.1 and later releases require persistence storage; the supported method is Cassandra. ORS 8.1.3 supports Apache Cassandra Version 1.1.x, beginning with Version 1.1.12. Consult your Genesys representative for the appropriate version for your deployment.

- Orchestration Server 8.1.3 does not require a connection to Cassandra. In ORS 8.1.3, if a connection is lost, ORS does not exit, instead, an alarm is raised. A session failover is not available.

Orchestration Server 8.1.2 and earlier releases require connectivity to at least one configured Cassandra node. When this is not possible, Orchestration Server exits.

- For the most current Apache Cassandra Server installation and configuration information, see the [Cassandra Installation/Configuration Guide](#). For Linux installations, the UNIX style paths (and other UNIX conventions) should be used. Orchestration Server supports Red Hat Enterprise Linux 5.

Cassandra Overview

Apache Cassandra is an open source, eventually consistent, distributed database system. It is designed to be highly scalable and decentralized (such that there is no single point of failure). See [Installing, Configuring, Starting, and Testing a Cassandra Node, page 161](#) for step-by-step instructions for:

- Obtaining the Cassandra installation package
- Deployment of the package to a cluster of Cassandra nodes
- Configuration of the cluster
- Loading a sample schema (Orchestration)

Cassandra Data Model

The Cassandra data model is divided into four basic elements:

- Columns
- Column Families
- Rows
- Keyspaces

These elements are defined as follows:

Table 14: Cassandra Data Model Element Definitions

Element	Description
Columns:	A column is the smallest element of the Cassandra data structure. It is a tuple which consists of a name, value, and timestamp.
Column Families:	A column family is to Cassandra what a table is to SQL (RDBMS). Column families store rows, and every row is referenced by a key.

Table 14: Cassandra Data Model Element Definitions

Element	Description
Rows:	A row groups related columns together. Every row is given a key and the key determines which Cassandra node the data will be stored to.
Keyspaces:	A keyspace is roughly the equivalent of a schema or a database in SQL (RDBMS). Keyspaces contain collections of column families. The name of the keyspace is a first dimension of a Cassandra hash.

The following provides an example of Cassandra data:

```
{
  "Genesys" : { // Keyspace
    "Services" : { // Column Family
      "ORS" : { // Row
        "Name" : "Orchestration", // Column
        "Version" : "8.1.2" }, // Column
      "GVP" : {
        "Name" : "Genesys Voice Platform",
        "Version" : "8.1.1" },
      "URS" : {
        "Name" : "Universal Routing Server",
        "Version" : "8.1.2" }
    }
  }
}
```

Installing, Configuring, Starting, and Testing a Cassandra Node

This section provides the prerequisites, and the procedures for installation, configuring, starting, and testing a Cassandra node.

Prerequisites

Before installing Cassandra, you should ensure that the following prerequisites are installed:

- Download the latest binary tarball from <http://cassandra.apache.org/>.
- Download Commons Daemon from:
<http://www.apache.org/dist/commons/daemon/binaries/>. This tool allows Cassandra to be run as a service/daemon.
- If running Windows, you must download the Windows archive. The archive should contain prunsrv.exe.
- Install JDK/JRE 6 version 1.6.0_19 or 1.6.0_21, or later.

Cassandra Installation

The following is an overview of the steps involved in the installation and configuration of Cassandra.

1. Download Cassandra and Commons Daemon.
2. Edit configuration files.
3. Start Cassandra.
4. Load the Schema.
5. Use nodetool to test Cassandra clusters.

Task Summary: Installing/Configuring a Cassandra Node

Objective	Related Procedures and Actions
Download Cassandra and Commons Daemon.	Procedure: Downloading Cassandra and Commons Daemon, on page 163
Edit configuration files.	Procedure: Editing Configuration Files, on page 164
Start Cassandra.	Procedure: Starting Up Cassandra, on page 167
Load the Schema.	Procedure: Loading the Cassandra Schema, on page 168
Use nodetool, test the Cassandra clusters.	Procedure: Using Nodetool, on page 168

Downloading Cassandra and Commons Daemon

This section describes how to retrieve the Cassandra Server and Commons Daemon from the Apache website, download it to a local drive, and extract the contents of the compressed file.

Procedure: Downloading Cassandra and Commons Daemon

Start of procedure

1. Download Cassandra and Commons Daemon.
 - Download the latest stable release of Cassandra from <http://cassandra.apache.org/>.
 - To be able to install Cassandra as a service/daemon, download Commons Daemon from <http://www.apache.org/dist/commons/daemon/binaries/> Copy the install image file to a computer that will serve as the Cassandra node—for example: C:\CassandraServerNodeOne.
2. Copy the Cassandra archive (and Commons Daemon archive) to each node in the Cassandra cluster, to the desired installation directory. In this guide, we will use the following:
 - Windows: C:\Cassandra
 - Linux: /cassandra
3. Extract the contents of the Cassandra (and Commons Daemon) archive(s) on each node. On Windows, use WinZip “Extract here...” or equivalent. On Linux, use `gunzip` or `tar -xvf`. If following the example, the directories should be placed as such:
 - Windows: C:\Cassandra\apache-cassandra-1.x.x
 - Linux: /cassandra/apache-cassandra-1.x.xExtract the contents of the Commons Daemon archive to:
 - `apache-cassandra-1.x.x\daemon`
4. Set the installation directory path to the environment variable `CASSANDRA_HOME`, for example:
 - `set CASSANDRA_HOME=C:\Cassandra\apache-cassandra-1.x.x`or
 - `export CASSANDRA_HOME=/cassandra/apache-cassandra-1.x.x`
5. Set the `JAVA_HOME` environment variable to the Java JRE/JDK root, for example:
 - `set JAVA_HOME=C:\Program Files (x86)\Java\jre6`or
 - `export JAVA_HOME=/usr/java/jdk1.x.x_x`
6. For Linux-like installs, edit `JAVA_HOME` in `%CASSANDRA_HOME%\bin\cassandra-in.sh`
7. Extract Commons Daemon and rename the extracted folder from `commons-daemon-1.x.x` to `daemon`. The `daemon` folder should be placed within the `apache-cassandra-1.x.x` directory.

8. Optionally, create the following subdirectories in the installation directory. These are required Cassandra installation directories for the database commitlog, log files, cache, and keyspace filestore.

- commitlog
- logs
- saved_caches
- data
- data\system
- data\Orchestration

End of procedure

Editing Configuration Files

This section describes how to edit a number of configuration files that come with the Cassandra distribution

Procedure: Editing Configuration Files

Purpose: To edit the Cassandra configuration files which are located in the %CASSANDRA_HOME%\conf directory.

Start of procedure

1. The `cassandra.yaml` contains default configurations for the Cassandra cluster. It is recommended to fill in the initial tokens for nodes in the cluster by editing `initial_token`.
 - For the first node, the initial token can be left blank or set to 0. (Optionally, you may also let the other nodes automatically obtain an initial token if `AutoBootstrap` is enabled.)

AutoBootstrap: The `AutoBootstrap` option is available in Cassandra 0.5 and above. When enabled, new nodes will automatically bootstrap themselves on startup and will pick an initial token such that it will obtain half the keys of the node with the most disk space used.
 - Nodes specified as seeds will not `AutoBootstrap`. However, even with `AutoBootstrap`, it is still recommended that you specify `initial_token` as the picking of an initial token will almost certainly result in an unbalanced ring.

2. Ensure that the following options are pointing to the correct paths. Paths specified below are examples:

`data_file_directories:`

- `C:\Cassandra\apache-cassandra-1.x.x\data`

`commitlog_directory:` `C:\Cassandra\apache-cassandra-1.x.x\commitlog`

`saved_caches_directory:`

`C:\Cassandra\apache-cassandra-1.x.x\saved_caches`

3. Also in `cassandra.yaml`, configure the `cluster_name`, `seeds`, `listen_address` and `rpc_address` as needed:

- `cluster_name`

The `cluster_name` must be identical for all nodes within a Cassandra cluster. `seeds` must be provided as a comma-delimited list of IP addresses which new nodes will be able to contact for information about the Cassandra cluster.

- `seeds`

It is recommended that all nodes have the same list of `seeds` specified, and that all nodes be specified as seed nodes.

- `listen_address`

`listen_address` is the IP address that other Cassandra nodes use to connect to this node, and the

- `rpc_address`

`rpc_address` is the listen address for remote procedure calls (and clients, such as the `cassandra-cli`). They are defaulted to `localhost`.

- `storage_port` and `rpc_port`

Also verify that `storage_port` and `rpc_port` do not conflict with other configured services. The `storage_port`, which defaults to 7000, is the port used by Cassandra nodes for inter-node communication. The `rpc_port`, which defaults to 9160, is used for remote procedure calls (e.g. `cassandra-cli`) and the Thrift service. This is the port to use when building clients for the Cassandra API.

End of procedure

Next Steps

- [Procedure: Edit the `cassandra-env.sh` and `cassandra.bat` configuration files](#)

Procedure:**Edit the `cassandra-env.sh` and `cassandra.bat` configuration files**

1. To configure Cassandra's JVM (Java Virtual Machine) and the JMX (Java Management Extensions) interface, edit `conf/cassandra-env.sh` for Linux, or `bin/cassandra.bat` for Windows.
2. The JMX port is used for management connections (such as `nodetool`). If necessary, edit the following line and ensure that there are no port conflicts with existing services:
 - For `cassandra-env.sh`:
Specifies the default port over which Cassandra will be available for JMX connections. `JMX_PORT="7199"`
 - For `cassandra.bat`, also set `SERVICE_JVM` to the desired Windows Service name, `PATH_PRUNSRV` to the daemon directory, and `PR_LOGPATH` to the logs directory.

```
Dcom.sun.management.jmxremote.port=7199
...
:doInstallOperation
set SERVICE_JVM="cassandra_gre_01"
rem location of Prunsrv

set PATH_PRUNSRV=%CASSANDRA_HOME%\daemon\
set PR_LOGPATH=%CASSANDRA_HOME%\logs
```

End of procedure**Next Steps**

- [Procedure: Edit `log4j-server.properties`](#)

Procedure:**Edit `log4j-server.properties`****Start of procedure**

1. The logging options can be found in `conf/log4j-server.properties`. Update the path for the log file and ensure that the specified path exists:
`log4j.appender.R.File=/var/log/cassandra/system.log`

2. You may find it helpful for testing to change the logging level from INFO to DEBUG:
`log4j.rootLogger=INFO, stdout, R`

End of procedure

Next Steps

- [Procedure: Set up the Cassandra service \(Windows\)](#)

Procedure: Set up the Cassandra service (Windows)

Start of procedure

1. Install the Cassandra service:
 - To install: `'bin\cassandra.bat INSTALL '`
 - To uninstall: `'bin\cassandra.bat UNINSTALL '`
 - Once installed, you will be able to find and start up the Cassandra service from the Windows Services GUI. The name of the service depends on the value of `SERVICE_JVM`.

End of procedure

Starting Up Cassandra

This section contains information for starting up the Cassandra Server on Linux or Windows and describes the following procedure:

- [“Starting Up Cassandra”](#)

Procedure: Starting Up Cassandra on Linux or Windows

Start of procedure

1. Starting up on Linux:
 - Start up Cassandra by invoking `bin/cassandra -f`. It will start up in the foreground and will log to std-out. If you do not see any error or fatal log messages or Java stack traces, then chances are you've succeeded.
 - Or, press "Control-C" to stop Cassandra. If you start up Cassandra without `-f` option, it will run in the background, so you need to kill the process to stop.

2. Starting up on Windows:

To start the service from Windows, there are two options:

- Use the Windows Services GUI:
- From commandline, in the daemon dir (as set above):
 - start: `prunsrv.exe start <Cassandra Service Name>`
 - stop: `prunsrv.exe stop <Cassandra Service Name>`

End of procedure

Loading the Cassandra Schema

This section contains information for loading the Cassandra schema and describes the following procedure.

- [“Loading the Cassandra Schema”](#)

Note: The manual loading of the schema is optional with ORS 8.1.3, since this operation can be provided by ORS on startup of the first Orchestration application.

Procedure: Loading the Cassandra Schema

Start of procedure

1. If you have a Cassandra schema prepared, load the schema. From %CASSANDRA_HOME%, execute:

```
bin\cassandra-cli -host 'host-address' --file
'path\to\schema-filename.txt'
```

This requires a Cassandra node to be running at the specified host-address. For Orchestration, see the Sample Schema in the [Cassandra Installation/Configuration Guide](#).

2. In the event that `cassandra-cli` does not behave as expected (`java.lang.ClassNotFoundException`), refer to the Troubleshooting section of the [Cassandra Installation/Configuration Guide](#).

End of procedure

Using Nodetool

Once all Cassandra nodes have been started, you can check the status of the Cassandra cluster using Nodetool. For more information about this tool, see the [Cassandra Installation/Configuration Guide](#).

This section describes how to use Nodetool and contains the following procedure:

Procedure: Using Nodetool

Start of procedure

1. In one of the Cassandra nodes, from %CASSANDRA_HOME%, run
`/bin/nodetool -h <listen_address> -p <jmx_port> ring`
2. The output of this command should be similar to the output example found in the Useful Tools section in the [Cassandra Installation/Configuration Guide](#). There should be as many addresses in the list as the number of Cassandra nodes configured.
3. If there are fewer nodes than expected, check and make sure that all nodes have unique initial tokens.

End of procedure

Troubleshooting

cassandra-cli fails to run! 'java.lang.ClassNotFoundException'

In the event that `cassandra-cli` displays a `java.lang.ClassNotFoundException` when run, ensure that it is being run from %CASSANDRA_HOME% as opposed to %CASSANDRA_HOME%\bin.

If `cassandra-cli` still fails, consider replacing the shell scripts/batch files with the following, specifically for Windows or Unix, below:

cassandra-cli.bat (Windows)

```
@REM
@REM Licensed to the Apache Software Foundation (ASF) under one or
@REM more
@REM contributor license agreements. See the NOTICE file distributed with
@REM this work for additional information regarding copyright ownership.
@REM The ASF licenses this file to You under the Apache License, Version
@REM 2.0
@REM (the "License"); you may not use this file except in compliance with
@REM the License. You may obtain a copy of the License at
@REM
@REM http://www.apache.org/licenses/LICENSE-2.0
```

```

@REM
@REM Unless required by applicable law or agreed to in writing, software
@REM distributed under the License is distributed on an "AS IS" BASIS,
@REM WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
@REM either express or implied.
@REM See the License for the specific language governing permissions and
@REM limitations under the License.

@echo off if "%OS%" == "Windows_NT" setlocal

if NOT DEFINED CASSANDRA_HOME set CASSANDRA_HOME=%~dp0..
if NOT DEFINED JAVA_HOME goto :err

REM Ensure that any user defined CLASSPATH variables are not used on
startup
set CLASSPATH=

REM For each jar in the CASSANDRA_HOME lib directory call append to
build the CLASSPATH variable.
for %i in ("%CASSANDRA_HOME%\lib\*.jar") do call :append "%i"
goto okClasspath

:append
set CLASSPATH=%CLASSPATH%;%1
goto :eof

:okClasspath
REM Include the build\classes\main directory so it works in development
set
CASSANDRA_CLASSPATH=%CLASSPATH%; "%CASSANDRA_HOME%\build\classes\main"; "
%CASSANDRA_HOME%\build\classes\thrift"
goto runCli

:runCli
echo Starting Cassandra Client
"%JAVA_HOME%\bin\java" -cp %CASSANDRA_CLASSPATH%
org.apache.cassandra.cli.CliMain %*
goto finally

:err
echo The JAVA_HOME environment variable must be set to run this
program!
pause

```

```
:finally
```

```
ENDLOCAL
```

```
cassandra-cli (UNIX)
```

```
#!/bin/sh
```

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.

```
if [ "x$CASSANDRA_INCLUDE" = "x" ]; then
    for include in /usr/share/cassandra/cassandra.in.sh \
                  /usr/local/share/cassandra/cassandra.in.sh \
                  /opt/cassandra/cassandra.in.sh \
                  ~/.cassandra.in.sh \
                  `dirname $0`/cassandra.in.sh; do
        if [ -r $include ]; then
            . $include
            break
        fi
    done
elif [ -r $CASSANDRA_INCLUDE ]; then
    . $CASSANDRA_INCLUDE
fi
```

```

# Use JAVA_HOME if set, otherwise look for java in PATH
if [ -x $JAVA_HOME/bin/java ]; then
    JAVA=$JAVA_HOME/bin/java
else
    JAVA=`which java`
fi

#if [ -z $CASSANDRA_CONF -o -z $CLASSPATH ]; then
#     echo "You must set the CASSANDRA_CONF and CLASSPATH vars" >&2
#     exit 1
#fi
#
## Special-case path variables.
#case ".uname." in
#    CYGWIN*)
#        CLASSPATH=`cygpath -p -w "$CLASSPATH"`
#        CASSANDRA_CONF=`cygpath -p -w "$CASSANDRA_CONF"`
#        ;;
#esac

if [ -z $CLASSPATH ]; then
    echo "You must set the CLASSPATH var" >&2
    exit 1
fi

cassandra_classpath=''

for lib in lib/*.jar
do
    cassandra_classpath=";${lib}${cassandra_classpath}"
done
CLASSPATH="${CLASSPATH}${cassandra_classpath}"

$JAVA -ea -cp $CLASSPATH \
    -Xmx256M \
    -Dlog4j.configuration=log4j-tools.properties \
    org.apache.cassandra.cli.CliMain "$@"

# vi:ai sw=4 ts=4 tw=0 et

```

Configuring Cassandra Server

To configure Cassandra Server, please see the [Cassandra Installation/Configuration Guide](#) on the Genesys Documentation web site at www.docs.genesyslab.com.



Supplements

Related Documentation Resources

The following resources provide additional information that is relevant to this software. Consult these additional resources as necessary.

Universal Routing

- *Universal Routing 8.1 Deployment Guide*, which describes installation, configuration, and deployment of the Universal Routing Server and Custom Server components that work with the Orchestration Server.
- *Universal Routing 8.1 Reference Manual*, which describes and defines routing strategies, IRD objects, Universal Routing Server and other server functions and options, number translation, pegs, and statistics used for routing.
- *Universal Routing 8.1 Business Process User's Guide*. This guide contains step-by-step instructions for creating interaction workflows (business processes), which direct incoming customer interactions through various processing objects. The goal is to generate an appropriate response for the customer.
- *Universal Routing 8.1 Strategy Samples*, which simplifies strategy configuration for first-time users of the strategy development tool, Interaction Routing Designer. To achieve this goal, this document supplies examples of simple voice and e-mail routing strategies that can be used as general guides during the design stage.
- *Universal Routing 8.0 Routing Application Configuration Guide* (previously *Universal Routing 7.0 Routing Solutions Guide*), which contains information on the various types of routing that can be implemented, including skills-based routing, business-priority routing, and share agent by service level agreement routing. It also summarizes cost-based routing, proactive routing, and a SIP/instant message solution.

- *Universal Routing 8.1 Interaction Routing Designer Help*, which describes how to use Interaction Routing Designer to create routing strategies. It also describes Interaction Design view where you create business processes that route incoming interactions through various processing objects with the goal of generating an appropriate response for the customer.

eServices and Other

- *eServices 8.1 Deployment Guide*, which includes a high-level overview of features and functions of Genesys eServices together with architecture information and deployment-planning materials. It also introduces you to some of the basic concepts and terminology used in this product.
- *eServices 8.1 User's Guide*, which provides overall information and recommendations on the use and operation of Genesys eServices.
- *eServices 8.1 Open Media Interaction Models Reference Manual*, which presents a set of basic interaction models, showing the components involved and the messaging (requests, events) among them.
- “Universal Routing and Multi-media/eServices Log Events” in *Framework 8.1 Combined Log Events Help*, which is a comprehensive list and description of all events that may be recorded in Management Layer logs.

Genesys

- *Genesys 7.6 Proactive Routing Solution Guide*, which documents a solution that enables you to proactively route outbound_preview interactions to Genesys Agent Desktop, as well as to completely process Calling List and Do Not Call List records solely from the logic of a routing strategy without agent intervention.
- *Genesys 8.x Proactive Contact Solution Guide*, which provides an overview of the Proactive Contact solution using Outbound Contact 8.0 and Genesys Voice Platform (GVP) 8.1.
- *Genesys Events and Models Reference Manual*, which provides information on most of the published Genesys events and their attributes, and an extensive collection of models describing core interaction processing in Genesys environments.
- *Genesys Technical Publications Glossary*, which ships on the Genesys Documentation Library DVD and which provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms used in this document.
- *Genesys Migration Guide*, which ships on the Genesys Documentation Library DVD, and which provides documented migration strategies for Genesys product releases. Contact Genesys Customer Care for more information.

- Release Notes and Product Advisories for this product, which are available on the Genesys Customer Care website at <http://genesyslab.com/support>.

Information about supported hardware and third-party software is available on the Genesys Customer Care website in the following documents:

- *[Genesys Supported Operating Environment Reference Guide](#)*
- *[Genesys Supported Media Interfaces Reference Guide](#)*

Consult these additional resources as necessary:

- *[Genesys Hardware Sizing Guide](#)*, which provides information about Genesys hardware sizing guidelines for Genesys releases.
- *[Genesys Interoperability Guide](#)*, which provides information on the compatibility of Genesys products with various Configuration Layer Environments; Interoperability of Reporting Templates and Solutions; and Gplus Adapters Interoperability.
- *[Genesys Licensing Guide](#)*, which introduces you to the concepts, terminology, and procedures relevant to the Genesys licensing system.
- *[Genesys Database Sizing Estimator 8.0 Worksheets](#)*, which provides a range of expected database sizes for various Genesys products.

For additional system-wide planning tools and information, see the [System Level Documents](#) on the Genesys Documentation website.

Genesys product documentation is available on the:

- Genesys Customer Care website at <http://genesyslab.com/support>.
- Genesys Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.com.

Document Conventions

This document uses certain stylistic and typographical conventions—introduced here—that serve as shorthands for particular kinds of information.

Document Version Number

A version number appears at the bottom of the inside front cover of this document. Version numbers change as new information is added to this document. Here is a sample version number:

8'fr_ref_04-2011_v8.1.001.00

You will need this number when you are talking with Genesys Customer Care about this product.

Screen Captures Used in This Document

Screen captures from the product graphical user interface (GUI), as used in this document, may sometimes contain minor spelling, capitalization, or grammatical errors. The text accompanying and explaining the screen captures corrects such errors *except* when such a correction would prevent you from installing, configuring, or successfully using the product. For example, if the name of an option contains a usage error, the name would be presented exactly as it appears in the product GUI; the error would not be corrected in any accompanying text.

Type Styles

[Table 15](#) describes and illustrates the type conventions that are used in this document.

Table 15: Type Styles

Type Style	Used For	Examples
Italic	<ul style="list-style-type: none"> Document titles Emphasis Definitions of (or first references to) unfamiliar terms Mathematical variables <p>Also used to indicate placeholder text within code samples or commands, in the special case where angle brackets are a required part of the syntax (see the note about angle brackets on page 179).</p>	<p>Please consult the <i>Genesys Migration Guide</i> for more information.</p> <p>Do <i>not</i> use this value for this option.</p> <p>A <i>customary and usual</i> practice is one that is widely accepted and used within a particular industry or profession.</p> <p>The formula, $x + 1 = 7$ where x stands for. . .</p>

Table 15: Type Styles (Continued)

Type Style	Used For	Examples
Monospace font (Looks like teletype or typewriter text)	<p>All programming identifiers and GUI elements. This convention includes:</p> <ul style="list-style-type: none"> The <i>names</i> of directories, files, folders, configuration objects, paths, scripts, dialog boxes, options, fields, text and list boxes, operational modes, all buttons (including radio buttons), check boxes, commands, tabs, CTI events, and error messages. The values of options. Logical arguments and command syntax. Code samples. <p>Also used for any text that users must manually enter during a configuration or installation procedure, or on a command line.</p>	<p>Select the Show variables on screen check box.</p> <p>In the Operand text box, enter your formula.</p> <p>Click OK to exit the Properties dialog box.</p> <p>T-Server distributes the error messages in EventError events.</p> <p>If you select true for the inbound-bsns-calls option, all established inbound calls on a local agent are considered business calls.</p> <p>Enter exit on the command line.</p>
Square brackets ([])	A particular parameter or value that is optional within a logical argument, a command, or some programming syntax. That is, the presence of the parameter or value is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information.	smcp_server -host [/flags]
Angle brackets (< >)	<p>A placeholder for a value that the user must specify. This might be a DN or a port number specific to your enterprise.</p> <p>Note: In some cases, angle brackets are required characters in code syntax (for example, in XML schemas). In these cases, italic text is used for placeholder values.</p>	smcp_server -host <confighost>

Index

Symbols

[] (square brackets)	179
< > (angle brackets)	179

A

action elements	20
alarm	
common log option	118
all	
common log option	117
angle brackets	179
Apache	159
Apache Cassandra	23
application	
SCXML-based	15
Application Servers	
supported in ORS 8.1	20
audience, for document	12
auto-restart	
ORS	66

B

brackets	
angle	179
square	179
buffering	
common log option	110
business-priority routing	175

C

Cassandra	
Cassandra Data Model Element Definitions	160
deployment	159
download Cassandra and Commons Daemon	163

edit the Cassandra configuration files	164
Installation	162
installing	161
installing/Configuring a Cassandra Node	162
prerequisites	161
start up Cassandra on Linux or Windows	167
Cassandra Server	159
Cassandra service (Windows)	167
cassandra.bat	166
cassandra-env.sh	166
check-point	
common log option	111
CIM (Customer Interaction Management)	
Platform	10
Classification module	19
client side port definition	26, 147
ORS	143
clustering	
configuration	67
clusters	
failure recovery	44
sessions/requests	44
Command Line	
ORS	66
commenting on this document	13
common configuration options	110
log section	110
common log options	110
alarm	118
all	117
buffering	110
check-point	111
compatible-output-priority	111
debug	120
expire	112
interaction	119
keep-startup-file	112
log section	110
memory	113
memory-storage-size	113
message_format	113

messagefile	114
print-attributes	114
segment	115
spool	115
standard	118
time_convert	115
time_format	116
trace	119
verbose	116
common options	
common log options	110
compatible-output-priority	
common log option	111
configuration option	
webfm-event-hold-response	88
Configuration options	
mcr-pull-by-this-node	84
configuration options	
{Parameter Name}, ApplicationParms section	129
alternate-url	122, 124
application	122
cassandra-connect-attempt-timeout	89
cassandra-keyspace-name	89
cassandra-listenport	90
cassandra-max-latency	90
cassandra-nodes	90
cassandra-schema-version	91
cassandra-strategy-class	91
cassandra-strategy-options	92
cassandra-thread-count	92
common log options	110
common options	110
debug-enabled	93
debug-port	93
default-encoding	94
fetch-timeout	126
fips_enable	94
http-client-side-port-range	94
http-enable-continue-header	95
http-max-age	127
http-max-age-local-file	95, 102, 104
http-max-cache-entry-count	92, 95, 102
http-max-cache-entry-size	95
http-max-cache-size	96
http-max-redirects	96
http-max-stale	127
http-no-cache-urls	96
http-proxy	97
https-proxy	101
http-ssl-ca-info	97
http-ssl-ca-path	97
http-ssl-cert	98
http-ssl-cert-type	98
http-ssl-cipher-list	98
http-ssl-key	99
http-ssl-key-type	99
http-ssl-random-file	99
http-ssl-verify-host	100
http-ssl-verify-peer	100
http-ssl-version	100
http-useragent	126
http-version	126
js-preload-files	101
max-age	96, 127
max-age-local-file	102
max-cache-entry-count	92
max-cache-entry-size	103
max-cache-size	93
max-compiler-cached-docs	103
max-compiler-cached-doc-size	103
max-compiler-cache-size	103
max-debug-sessions	104, 105
max-includes	104, 105
max-microstep-count	85
max-pending-events	104
max-preprocessor-cached-docs	105
max-preprocessor-cached-doc-size	105
max-preprocessor-cache-size	105
max-session-age	105
max-stale	106, 127
max-state-entry-count	86
mcr-pull-interval	85
mcr-pull-limit	85
om-delete-from-memory	122
om-max-in-memory	121
om-memory-optimization	121
parse-start-params	86
password	106
persistence-default	107
persistence-max-active	107
primary-webdav-url	87
process-event-timeout	107
scxml-log-filter-level	86
send-retries	124
session-hung-timeout	87
session-processing-threads	108
spool	110
stuck-thread-timeout	108
switch-multi-links-enabled	87
system-id	108
url	128
x-print-attached-data	110
x-server-config-trace-level	109
x-server-gcti-trace-level	109
x-server-trace-level	109
Configuration Server Proxy	66
connection to Persistence Storage	32
Connections tab	
ORS	66
conventions	
in document	178

type styles	178
coordinating customer interactions and dialogues	22
cost-based routing	175
createcall action	24

D

Data Center	51
debug	
common log option	120
debugging capabilities	17
Dialog module	19
document	
audience.	12
change history.	13
conventions	178
errors, commenting on	13
version number	178
Documentation	
Universal Routing 8.1 Interaction Routing Designer Help	176
documentation	
eServices User's Guide	176
Framework 8.1 Combined Log Events Help	176
Log Events Help	176
Universal Routing Business Process User's Guide	175
Universal Routing Deployment Guide	175
Universal Routing Reference Manual	175
Universal Routing Routing Application Configuration Guide	175
Universal Routing Strategy Samples	175

E

ECMAScript	17
eServices Open Media Interaction Models.	176
expire	
common log option	112

F

Failed Sessions	47
fallback URI functionality	124
font styles	
italic	178
monospace	179
Functional Module	19
Functional Modules	19

G

General tab	
ORS.	65
Genesys Voice Platform	22
GVP	22

H

high availability.	48
http.	51

I

IIS	20
infinite loop prevention.	24
installation package directories	142
installing ORS	
Linux	147
Windows	142
intended audience	12
Interaction	31
interaction	
common log option	119
Interaction module	19
Internet Information Services	20
italics.	178

J

JBoss Application Server	20
------------------------------------	----

K

keep-startup-file	
common log option	112

L

Linux	
Orchestration Server installation	147
load balancing	
RESTful web services	50
log configuration options.	110, 116
log section	
common log options	110
log4j-server.properties	166
logs	
Management Layer	176

M

Management Layer logging	176
------------------------------------	-----

max-digits	140
media channels	11
memory	
common log option	113
memory optimization	
configuration	36
memory-storage-size	
common log option	113
message_format	
common log option	113
messagefile	
common log option	114
MIME Types	21
modeling/managing customer service process	21
monospace font	179
Multiple Data Center architecture	23

N

Non-Stop Operation	153
--------------------	-----

O

options	
options list	77, 83
orchestrating resources and product services	22
Orchestration	21
definition	16
eServices interaction routing	33
platform capabilities	21
Orchestration Server	
About	9
architecture	27
auto-restart	66
clustering	
configuring	67
Configuration Server Proxy	66
eServices interaction routing	33
eServices interactions	30
high availability	48
Linux installation	147
memory optimization	
configuring	36
persistent storage	
configuring	45
deploying	44
operation	43
Primary setting	66
shutdown time	66
startup time	66
voice interactions	29
web service access	65
Windows installation	142
Orchestration Server options, table	77, 83
ORS	24

ORS cluster configuration	67
ORS memory optimization configuration	36
Outbound solution	22

P

page	31
Persistent Storage	
configuration	45
deployment	44
operation of	43
port info for ORS	65
pre-start information	149
print-attributes	
common log option	114
Proactive contact	22
pro-active routing	175

Q

Queue module	19
--------------	----

R

Recovery of failed sessions	47
Resource module	20
routing	
strategy samples	175

S

Samples	175
SCXML Actions	38
SCXML application support	15
SCXML session startup	51
Secure port configuration	65
Secure Sockets Layer	26
security	26
segment	
common log option	115
Server Info tab	
ORS	65
service history binding	23
Session module	20
Share Agents by Service Level Agreement	175
Solution Control Interface	150
spool	
common log option	115
square brackets	179
SSL	26
standard	
common log option	118
standard event processing	50
Start Info tab	

ORS 66
 starting
 Orchestration Server 151
 Orchestration Server in Linux 151
 Orchestration Server using SCI 150
 Solution Control Interface 150
 Statistics module 19
 stopping
 Enterprise Routing solution 152
 strategy
 samples 175
 supported Application Servers 20

T

TDM 22
 Tenants tab
 ORS 65
 time_convert
 common log option 115
 time_format
 common log option 116
 TLS 26
 Tomcat 20
 trace
 common log option 119
 type styles
 conventions 178
 italic 178
 monospace 179
 typographical styles 178

U

uninstalling ORS
 Linux 157
 Windows 156
 with Genesys Administrator 156

V

valid-digits 140
 verbose
 common log option 116
 version identification 154
 version numbering, document 178

W

Web Services module 19
 Websphere Application Server 20
 Windows
 Orchestration Server installation 142
 Working Directory

ORS 66

X

XML 15

