Avaya Aura® Call Center Elite Multichannel Call Routing Server User Guide

Release 6.3
December 2013

© 2013 Avaya Inc. All Rights Reserved.

Notice

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer

"Documentation" means information published by Avaya in varying mediums which may include product information, operating instructions and performance specifications that Avaya generally makes available to users of its products. Documentation does not include marketing materials. Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of documentation unless such modifications, additions, or deletions were performed by Avaya. End User agrees to indemnify and hold harmless Avaya, Avaya's agents, servants, and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

Link disclaimer

Avaya is not responsible for the contents or reliability of any linked websites referenced within this site or documentation provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty

Avaya provides a limited warranty on its hardware and Software ("Product(s)"). Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this Product while under warranty is available to Avaya customers and other parties through the Avaya Support website:

http://support.avaya.com

Please note that if you acquired the Product(s) from an authorized Avaya Channel Partner outside of the United States and Canada, the warranty is provided to you by said Avaya Channel Partner and not by Avaya. "Software" means computer programs in object code, provided by Avaya or an Avaya Channel Partner, whether as stand-alone products or pre-installed on hardware products, and any upgrades, updates, bug fixes, or modified versions.

Licenses

THE SOFTWARE LICENSE TERMS AVAILABLEON THE AVAYA WEBSITE, HTTP://SUPPORT.AVAYA.COM/LICENSEINFO ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AUTHORIZED AVAYA CHANNEL PARTNER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AUTHORIZED AVAYA CHANNEL PARTNER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA AUTHORIZED AVAYA CHANNEL PARTNER; AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING ORSELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS "YOU" AND "END USER", AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE "AVAYA".

Avaya grants you a license within the scope of the license types described below. Where the order documentation does not expressly identify a license type, the applicable license will be a Designated System License. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of licenses or units of capacity is specified in the documentation or other materials available to you. "Designated Processor" means a single stand-alone computing device. "Server" means a Designated Processor that hosts a software application to be accessed by multiple users.

License type(s)

Designated System(s) License (DS). End User may install and use each copy of the Software only on a number of Designated Processors up to the number indicated in the order. Avaya may require the Designated Processor(s) to be identified in the order by type, serial number, feature key, location or other specific designation, or to be provided by End User to Avaya through electronic means established by Avaya specifically for this purpose.

Concurrent User License (CU). End User may install and use the Software on multiple Designated Processors or one or more servers, so long as only the licensed number of Units are accessing and using the Software at any given time. A "Unit" means the unit on which Avaya, at its sole discretion, bases the pricing of its licenses and can be, without limitation, an agent, port or user, an e-mail or voice mail account in the name of a person or corporate function (e.g., webmaster or helpdesk), or a directory entry in the administrative database utilized by the Software that permits one user to interface with the Software. Units may be linked to a specific, identified Server.

Copyright

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation, Software, or hardware provided by Avaya. All content on this site, the documentation and the Product provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software unless expressly authorized by Avaya. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil offense under the applicable law.

Third-party components

"Third Party Components" mean certain software programs or portions thereof included in the Software that may contain software (including open source software) distributed under third party agreements ("Third Party Components"), which contain terms regarding the rights to use certain portions of the Software ("Third Party Terms"). Information regarding distributed Linux OS source code (for those Products that have distributed Linux OS source code) and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply is available in the Documentation or on Avaya's website at: http://support.avaya.com/Copyright. You agree to the Third Party Terms for any such Third Party Components.

Preventing Toll Fraud

"Toll Fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of Toll Fraud associated with your system and that, if Toll Fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Toll Fraud intervention

If you suspect that you are being victimized by Toll Fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Support website: http://support.avaya.com.Suspected security vulnerabilities with Avaya products should be reported to Avaya by sending mail to: securityalerts@avaya.com.

Trademarks

The trademarks, logos and service marks ("Marks") displayed in this site, the Documentation and Product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the Documentation and Product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party.

Avaya is a registered trademark of Avaya Inc.

All non-Avaya trademarks are the property of their respective owners, and "Linux" is a registered trademark of Linus Torvalds in the U.S. and other countries.

Downloading documents

For the most current versions of documentation, see the Avaya Support website:

http://support.avaya.com

Contact Avaya Support

See the Avaya Support website: http://support.avaya.com for product notices and articles, or to report a problem with your Avaya product.

For a list of support telephone numbers and contact addresses, go to the Avaya Support website: http://support.avaya.com, scroll to the bottom of the page, and select Contact Avaya Support.

Contents

Chapter 1: Introduction 3
Purpose 3
Intended audience 3
Reasons for reissue 3
Document changes since last issue 3
Related resources 3
Documentation 4
Training 5
Avaya Mentor videos 5
Support 6

Chapter 2: Overview 7
VDN registration 7
Calls received to routing VDN 8
Calls received to a monitored VDN 8
SQL Plug-in 8
Call flow 9
Licensing 10
Error logging 10
Diagnostic testing error logging 11
System requirements 11

Chapter 3: Installation 13

Chapter 4: Administration 15
Configuring Call Routing Server 15
[Error Logging] 15
[Telephony Services] 16
[Call Routing Server] 16
[AS Client Extensions] 17
Starting Call Center Elite Multichannel services 24

Appendix A: Plug-ins 27
Rules Plug-in 27
SOAP Plug-in 27
SOAP 28
Script Plug-in 29
SQL Plug-in 31
Plug-in installation 32

Plug-in configuration 32
[Sections]. 33
Result Parameters 36
Transfer Parameters 37
Fixed Value Parameters 37
Error log file 39
Events sent from Call Routing Server to Plug-in 39
CallDelivered 40
CallEstablished 41
ConnectionCleared 42
CallCleared 42
RouteRequest 43
RouteEnd 43
RouteUsed 44
TServerLinkStatus 44
Events sent from Plug-in to Call Routing Server 44
RouteSelect 44
RouteEnd 45
ClearCall 45
ClearConnection 45
Script COM Interface 46
Script COM Object - Parameter Set 46
Script COM - Properties 47
Script COM - Methods 47

Index 49

Chapter 1: Introduction

Purpose

This guide provides information about Call Routing Server that enables intelligent call routing for inbound calls in Call Center Elite Multichannel.

Intended audience

This guide is intended for customers who use Call Center Elite Multichannel. You can use this guide as a source of information for Call Routing Server that is provided in Avaya Aura® Call Center Elite Multichannel.

Reasons for reissue

The UUI parameter from Appendix A: Plug-ins is removed from the list of parameters which pass with the ConnectionCleared event.

Document changes since last issue

In *Avaya Aura® Call Center Elite Multichannel Call Routing Server User Guide*, the following guides are merged and added to Appendix A: Plug-ins:

- SOAP Plug-in User Guide
- Script Plug-in User Guide
- SQL Plug-in User Guide

Related resources

Documentation

Title	Description
Avaya Aura® Call Center Elite Multichannel Configuration Client Developer Guide	Provides information about the Configuration Client Developer application, which is a control that allows an application to transparently access the configuration information, regardless of its location.
Administering Avaya Aura® Call Center Elite Multichannel	Provides product overview, supported products, installation, configuration, and licensing requirements for Avaya Aura® Call Center Elite Multichannel.
Installing Avaya Aura® Call Center Elite Multichannel	Provides product overview, supported products, installation, configuration, and licensing requirements for Avaya Aura® Call Center Elite Multichannel.
Avaya Aura® Call Center Elite Multichannel Overview	Provides an overview of the Call Center Elite Multichannel features.
Avaya Aura® Call Center Elite Multichannel Desktop User Guide	Provides information about the Desktop application, and how to use the application to receive, view, and respond to voice and multimedia work items.
Avaya Aura® Call Center Elite Multichannel Application Management Service User Guide	Provides installation and administration information about the application management service in Call Center Elite Multichannel.
Avaya Aura® Call Center Elite Multichannel Upgrade and Migration Guide	Provides information about how to upgrade or migrate Avaya Aura® Call Center Elite Multichannel from Release 6.2.x to Release 6.3. The Upgrade Sequence section in the respective upgrading chapters gives a high-level overview of the process involved.
Avaya Aura® Call Center Elite Multichannel Reporting User Guide	Provides information about the reports for Agents, Customers, Interaction, Program and Schedule, Skills, and VDN. This guide also provides information about historical reports and real-time reports.

Title	Description
Avaya Aura® Call Center Elite Multichannel TTrace Console User Guide	Provides a brief overview of the TTrace Console application.
Installing TTrace Console for Avaya Aura® Call Center Elite Multichannel	Provides information necessary for installing and configuring TTrace.

Training

The following courses are available on http://avaya-learning.com. Enter the course code in the **Search** field and click **Go** to search for the course.

Course code	Course title
ASC00206WEN	Contact Center Intelligence
ASC00207WEN	Contact Center Automated
ATA00781VEN	Avaya Aura® Call Center Elite Multichannel Solutions Basic Implementation
AVA001002H00	Avaya Aura® Call Center Elite Multichannel Solution - Configuration
AVA00989H00	Avaya Aura® Call Center Elite Multichannel Solutions - Basic System Supervisor
AVA01013WEN	Avaya Aura® Call Center Elite Multichannel Solutions Overview
AVA01043H00	Avaya Aura® Call Center Elite Multichannel Solutions - Basic Installation
AVA00990H00	Avaya Aura® Call Center Elite Multichannel Solutions Advanced System Supervisor

Avaya Mentor videos

Avaya Mentor videos are available to provide technical content on how to install, configure and troubleshoot Avaya products. Videos are available on the Avaya support site, listed under the video document type, and on the Avaya-run channel on YouTube.

To find videos on the Avaya supported site, select the product name, and check the videos checkbox to see a list of available videos.

Note:
Videos are not available for all products.

To find the Avaya Mentor videos on YouTube, go to http://www.youtube.com/AvayaMentor and perform one of the following:

1. Enter a key word or key words in the Search channel to search for a specific product or topic.
2. Scroll down Playlists, and click the name of the topic to see the available list of videos posted on the site.

Support

Visit the Avaya Support website at http://support.avaya.com for the most up-to-date documentation, product notices, and knowledge articles. You can also search for, release notes, downloads, and resolutions to issues. Use the online service request system to create a service request. Chat with live agents to get answers to questions, or request an agent to connect you to a support team if an issue requires additional expertise.

Chapter 2: Overview

With Call Routing Server, you can use intelligent call routing for inbound calls. The routing depends on received call data that matches with customer information, contact center statistics, or agent availability.

Call Routing Server:

- Manages the Avaya Computer Telephony connection
- Monitors VDNs
- Manages VDN licensing
- Registers for routing services
- Receives call events
- Issues routing instructions
- Loads or manages generic plug-ins, such as SQL Plug-in that gives the server access to SQL Server databases

Call Routing Server supports a single connection to Avaya Communication Manager.

VDN registration

On startup, Call Routing Server requires configuration to run through Configuration Client. Call Routing Server extracts the configuration from a network Configuration Server or the `ActiveCallRoutingServer.ini` file. The configuration allows Call Routing Server to open a connection with Telephony Server. In addition, the configuration provides a secondary telephony link name which is used when the primary telephony link is unavailable or fails during operation.

After having a connection with Telephony Server, Call Routing Server sends monitoring and routing registration requests for the VDNs listed in the configuration and loads the listed generic plug-ins.

Calls received to routing VDN

When routing requests are received from a controlled VDN, Call Routing Server passes the requests to correct generic plug-in. The passed event contains complete information required for the return routing decision. Call Routing Server contains an internal list of outstanding routing requests. When Telephony Server receives response for a routing request from a plug-in or the RouteEnd event, then that routing request is removed from the internal list.

Calls received to a monitored VDN

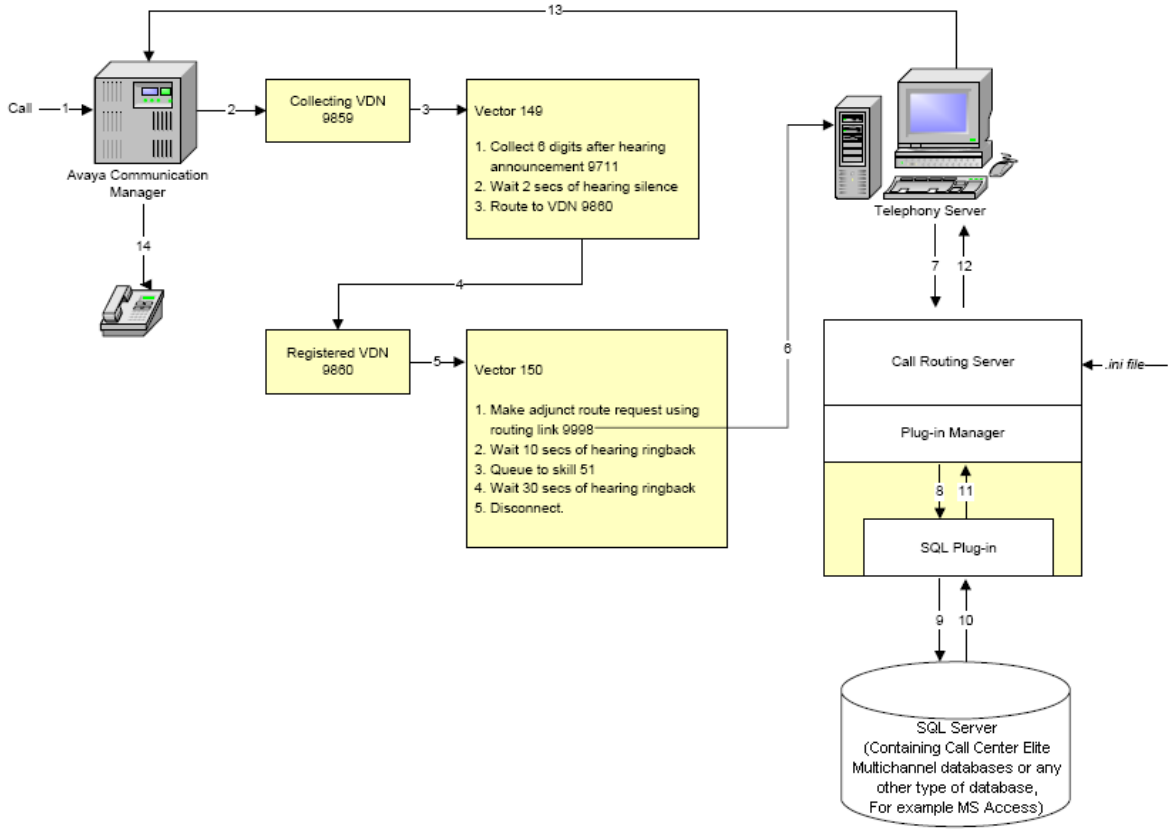
When a monitored VDN processes a call, a sequence of call events is generated. Call Routing Server passes these events to connected generic plug-ins. Call Routing Server does not track these calls.

SQL Plug-in

Call Routing Server uses SQL Plug-in to access information stored in an SQL server database. The database can be a Call Center Elite Multichannel database or any other type, for example, Microsoft Access database.

Call flow

The following call flow shows how Call Routing Server handles routing requests.



A vector sends an adjunct routing request to Telephony Server using a routing link extension. Telephony Server then informs Call Routing Server that has registered the associated VDN. Call Routing Server handles the request for information related to the collected digits.

The database passes back the extension number to which the call must be sent. The extension number can be a skill, split, agent DN, DDI, or international number. Call Routing Server processes the extension number and a route selection request passes to Telephony Server and onto the switch.

If the switch does not receive the call destination within the wait-time specified in the vector, it processes the next command in the list.

Note:

Only one Call Routing Server can register an adjunct routing VDN at a time.

Licensing

To run Call Routing Server, you do not need to buy any additional Call Center Elite Multichannel licenses.

Error logging

Agent logs the error information related to its own operation to a series of log files.

A new log file is created for each day of the week. The name of the error log file clearly identifies the application and the day of the week, for example: `MonACRS.log`.

When a log file reaches the size limit specified in the configuration, then that log file is archived and a new log file is created.

Note:
The archive stores only one log file. If the second error log reaches the specified maximum size, it overrides the previously archived log file.

The types of errors logged by the error log are determined by the logging level retrieved from the application's configuration data.

Levels of error logging are:

- 0. No error logging takes place.
- 1. Logs fatal, major, and minor errors and trace information.
- 2. Logs fatal, major, and minor errors.
- 4. Logs fatal and major errors.
- 8. Logs fatal errors only.

Each file records the selected logging level as well as the date, time, location, and description of every error that occurs.

Error log files are automatically saved into the current working folder of the application, which is the same folder as the application executable.

Diagnostic testing error logging

Diagnostic testing error logging is another error log level. Diagnostic testing error logging allows you to create log files that do not override each other every time the maximum log file size limit is reached. This logging level is for diagnostics only and is achieved by adding 128 to one of the logging level values specified above. For example, if you specify Error Log Level=129, new error log files are continuously created, and contain fatal, major, and minor errors and trace information.

Each new file has a unique name based on the date (year, month, and day) and time (hours, minutes, and seconds) the file is created.

For example: `YYYYMMDDHHMMSSASMediaDirector.log`. Where,

YYYY = Year	MM = Month	DD = Day
HH = Hour	MM = Minute	SS = Second

WARNING:
To prevent a multitude of log files affecting your available disk space, this error logging level must only be used for short periods of time while carrying out diagnostic testing.

System requirements

For information about system requirements for installing Call Center Elite Multichannel and its components, see *Installing Avaya Aura® Call Center Elite Multichannel*.

Chapter 3: Installation

For full instructions on how to install Call Routing Server, see *Installing Avaya Aura® Call Center Elite Multichannel*.

In addition to the standard installation steps, this document explains:

- How to use command line parameters during installation to dictate where configuration information is retrieved.
- How to change the configuration source from the `local .inc` file to the Configuration Server if a program software is already installed.
- How to automatically and silently install a program software using a set of predefined selection options.

You can download *Installing Avaya Aura® Call Center Elite Multichannel* from the Avaya Support Web site: http://support.avaya.com.

Chapter 4: Administration

Configuring Call Routing Server

To configure Call Routing Server, you must configure the properties in the `Edit ActiveCallRoutingServer.ini` file.

To configure properties in the `Edit ActiveCallRoutingServer.ini` file:

1. Click **Start** > **All Programs** > **Avaya Aura CC Elite Multichannel** > **Server** > **Call Routing Server**.

   The `Edit ActiveCallRoutingServer.ini` file displays.
2. Right-click the `Edit ActiveCallRoutingServer.ini` file and click **Open**.
3. Configure the following sections in the file by entering appropriate values:
   - [Error Logging]
   - [Telephony Services]
   - [Call Routing Server]
   - [AS Client Extensions]

Note:
Do not change section names or parameter names.

[Error Logging]

- **Error Log Level:** The value which determines the level of error detail that the log file saves. For a list of error logging levels, see Error logging on page 10.

  For information about diagnostic testing log level, see Diagnostic testing error logging on page 11.
- **Error Log File Path:** The folder path for saving error log files. By default, this parameter is left blank. If this parameter is left blank, the path of the error log file automatically sets to the current working folder of the application. The current working folder of the application is the same folder as the application executable.
- **Error Log File Extension:** The extension of error log files for error logging application. Extension refers to part of the file name and the file type extension, such as .log. The application automatically precedes the default extension with the day of the week when it creates the error logs.

- **Maximum Error Log File Size KB:** The maximum amount of information, in KB, that an error log file stores before the system archives the log file and creates a new file. The default is 1000. The minimum level that the error log file can be set to is 100. If a second error log reaches the specified maximum size, it overrides the previously archived file. If, however, you select the diagnostic testing error log level in Error Log Level, the system creates a new file with a new name every time the log file reaches the maximum limit.

Note:
The archive only stores one log file.

[Telephony Services]

- **Primary Link Name:** The name of the primary link that the Telephony Services uses to connect to Avaya Telephony Server and switch.
- **Primary User Name:** The user name required for the primary link.
- **Primary User Password:** The password for the primary user name. By default, Call Center Elite Multichannel encrypts this data. For more information about configuration commands, see *Installing Avaya Aura® Call Center Elite Multichannel*.
- **Secondary Link Name:** The name of the secondary link that the Telephony Services uses to connect to the Avaya Telephony Server and switch.
- **Secondary User Name:** The user name required for the secondary link.
- **Secondary User Password:** The password for the secondary user name. By default, Call Center Elite Multichannel encrypts this data. For more information about configuration commands, see *Installing Avaya Aura® Call Center Elite Multichannel*.

[Call Routing Server]

- **Routing VDN List:** The VDNs for which Call Routing Server issues RouteRequest events. Ensure that you separate VDNs with commas and use a hyphen to specify a range. For example: Routing VDN List=9885,9888,9910-9916.
- **Monitored VDN List:** The VDNs for which Call Routing Server issues monitor request commands. VDNs listed with this configuration setting generates the CallDelivered, CallEstablished, and CallCleared events. Ensure that you separate VDNs with commas and use a hyphen to specify a range. For example: Monitored VDN List=4401-4424,4637.

[AS Client Extensions]

This section lists all loadable generic plug-ins. Each item in this section has the format **Friendly name=Plug-in section name**. For example, SQL Plug-in=ASG SQL, Rules Plug-in=ASG Rules, Script Plug-in=ASG Script, and SOAP Plug-in=ASG SOAP.

The **Plug-in section name** points to the section in the file that contains configuration data for the plug-in. Note that the friendly name can have spaces.

This following sections contain configuration data for the generic plug-ins:

- [ASG Simple SQL]
- [ASG Rules]
- [ASG Script]
- [ASG SOAP]

[ASG Simple SQL]

For information about definitions and syntax of the SQL Plug-in parameter, see Appendix A: Plug-ins on page 27.

The following example shows how to use Call Routing Server with SQL Plug-in to route incoming calls to the appropriate agent based on the digits enter by the caller.

Call Routing Server receives an event from Telephony Server and searches each section of the configuration file to find a match for Event Name, Filter Name, and Filter Value. If Call Routing Server does not find a match in one section, it moves to the next section.

If Call Routing Server receives a RouteRequest event from VDN 9888*, it stops searching at Check Caller Digits section.

After Call Routing Server finds a match, it passes the event to the SQL Plug-in. The SQL Plug-in executes an SQL query on the database identified in the Connection String parameter (sample.mdb). The plug-in uses the specified file path.

The SQL query matches the caller digits received with the RouteRequest event with digits in the CallerDigits column of the database table. If the match is found, the SQL Plug-in returns the string in the Agent column of the table through the RouteSelect event. The SQL Plug-in also returns the received RouteRegisterReqID and RoutingCrossRefID parameters to identify the route request.

	CallerDigits	Agent
	1234	8765
	5678	8892
▶		

If the match is not found, the RouteSelect event requests for routing the call to extension 8574.

If an error occurs that prevents the execution of the SQL query, the RouteSelect event requests for routing the call to extension 8553.

On receiving the RouteSelect event, Call Routing Server passes the event to Telephony Server, which in turn passes the event to Avaya Communication Manager for routing to the agent.

The register * VDN 9888 as a routing VDN can receive RouteRequest events from VDN. For more information about routing VDN lists, see the Routing VDN list in [Call Routing Server] on page 16.

Note:

Ensure that you type the event parameters accurately. Also, ensure that you use uppercase letters exactly as mentioned in parameter names and without spaces between words. To confirm a parameter name, see Appendix A: Plug-ins.

The following are various fields of the SQL Plug-in and their appropriate values:

- Enable Client=True
- Client Library Name=ASGSimpleSQL.dll
- Display Name=Link to SQL Database
- Display Icon=someicon.ico
- Event Type=Native

The following parameters related to the SQL Plug-in appears as one section:

- Configuration Section Name=Check Caller Digits
- Connection String=Provider=Microsoft.Jet.OLEDB.4.0;Data Source=*<CCEM_INSTALL_DIR>*\Avaya Aura CC Elite Multichannel\Server\Call Routing Server\Samples\SQL Plug-in\sample.mdb;Persist Security Info=False
- Database Server Name=
- Database Name=
- Database User Name=
- Database User Password=
- SQL Query=SELECT * FROM tblChecking WHERE CallerDigits=cstr(%UserEnteredCode%)
- SQL Stored Procedure Name=
- SQL Stored Procedure Parameter Sequence=
- Event Name=RouteRequest
- Event Filter Name=VDN
- Event Filter Value=9888
- Return Event Name=RouteSelect
- Return Event Parameter Sequence=%RouteRegisterReqID%, %RoutingCrossRefID%, RouteSelected=$Agent$

- Return Event Allow Multiple Events=False
- Return Event Total Record Count Name=RecordCount
- Return Event Current Record Count Name=CurrentRecord
- No Record Event Name=RouteSelect
- No Record Event Parameter Sequence=%RouteRegisterReqID%, %RoutingCrossRefID%, RouteSelected=8574
- Error Event Name= RouteSelect
- Error Event Parameter Sequence=%RouteRegisterReqID%, %RoutingCrossRefID%, RouteSelected=8553
- Thread Pool Size= 2
- Maximum Queued Requests=10

[ASG Rules]

For information about the Rules Plug-in parameter definitions and syntax, see *Avaya Aura® Call Center Elite Multichannel Desktop User Guide*

The following example shows how to use Call Routing Server with the Rules Plug-in to route incoming calls to a particular agent based on the digits enter by the caller.

Call Routing Server receives an event from Telephony Server and searches each section of the configuration file to match the received information with the Event Name, Filter Name, and Filter Value. If Call Routing Server does not find a match in one section, it moves to the next section.

If Call Routing Server receives a RouteRequest event from VDN 9885*, it stops searching at Check Caller Digits section.

After Call Routing Server finds a match, it passes the event to the Rules Plug-in, which executes the configured rules.

The first rule runs if the received event is RouteRequest and this event carries digits collected with the call. If the rule requirements are met, the plug-in returns a RouteSelect event to Call Routing Server, instructing it to:

- route the call to the extension entered by the user and,
- display that extension number to the agent as user-to-user information (UUI).

The Rules Plug-in also returns the received RouteRegisterReqID and RoutingCrossRefID parameters to identify the route request.

If the first rule requirements are not met, then there are no collected digits with the call. In this case, the second rule is used and the RouteSelect event requests for routing the call to extension 8888.

On receiving the RouteSelect event, Call Routing Server passes the event to Telephony Server, which in turn passes the event to Avaya Communication Manager for routing to the agent.

The register * VDN 9885 as a routing VDN can receive RouteRequest events from VDN. For more information about routing VDN lists, see the Routing VDN list in [Call Routing Server] on page 16.

Note:
Ensure that you type the event parameters accurately. Also, ensure that you use uppercase letters exactly as mentioned in parameter names and without spaces between words. To confirm a parameter name, see Appendix A: Plug-ins.

The following are various fields of the Rules Plug-in and their appropriate values:

- Enable Client=True
- Client Library Name=ASGRules.dll
- Display Name=Rules Engine
- Display Icon=someicon.ico
- Event Type=Native

The following parameters related to the Rules Plug-in appears as one section:

- Event Name=RouteRequest
- Event Filter Name=VDN
- Event Filter Value=9885
- Rule1=When RouteRequest And UserEnteredCode>Nothing Do ReturnEvent RouteSelect, %RouteRegisterReqID%, %RoutingCrossRefID%, RouteSelected=%UserEnteredCode%, UUI=%UserEnteredCode% Then Stop
- Rule2=When RouteRequest Always Do ReturnEvent RouteSelect, %RouteRegisterReqID%, %RoutingCrossRefID%, RouteSelected=8888

[ASG Script]

For information about the Script Plug-in parameter definitions and syntax, see Appendix A: Plug-ins on page 27

The following example shows how to use Call Routing Server with the Script Plug-in to route incoming calls to a particular agent based on the digits enter by the caller.

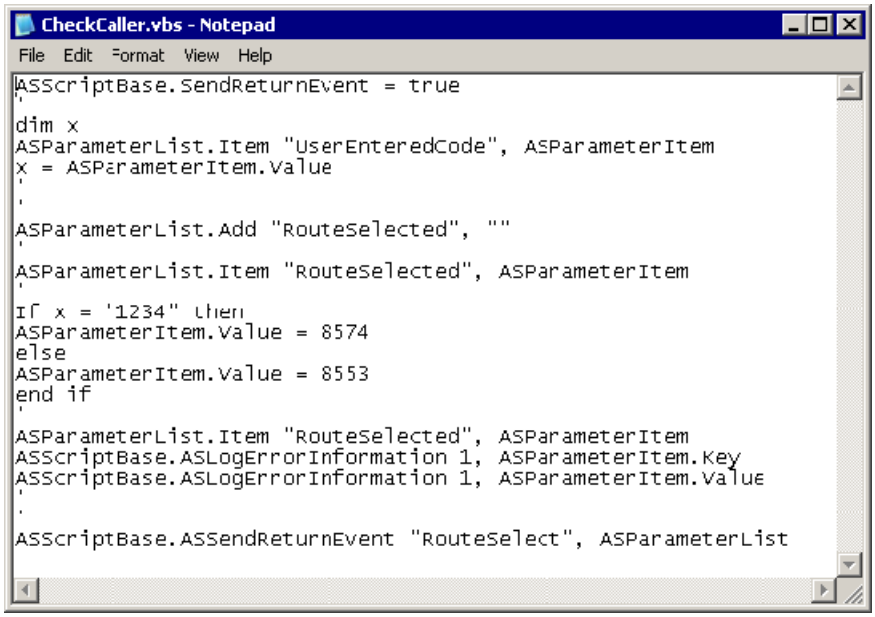
Call Routing Server receives an event from Telephony Server and searches each section of the configuration to match the received information with the Event Name, Filter Name, and Filter Value. If Call Routing Server does not find a match in one section, it moves to the next section.

If Call Routing Server receives a RouteRequest event from VDN 9888*, it stops searching at Check Caller CLI section.

After Call Routing Server finds a match, it passes the event to the Script Plug-in. The Script Plug-in runs the script identified in the Script File Name parameter (CheckCaller.vbs). The Script Plug-in looks for the script in the same folder that contains the Call Routing Server executable (*<CCEM_INSTALL_DIR>*\Avaya Aura CC Elite Multichannel\Server\Call Routing Server\ACR Server).

The following script matches the caller-entered digits received with the RouteRequest event (UserEnteredCode parameter) with the digits 1234. If the match is successful, the Script Plug-in returns a RouteSelect event to Call Routing Server, instructing it to route the call to extension 8574. If the script does not match the caller-entered digits with 1234, the Script Plug-in instructs the server to route the call to extension 8553.

The Script Plug-in sends a RouteSelect event and returns the value of the extension to Call Routing Server as the RouteSelected parameter. The Script Plug-in also returns the received RouteRegisterReqID and RoutingCrossRefID parameters to identify the route request.



If an error occurs that prevents the execution of the script, the RouteSelect event requests the call to be routed to extension 8887.

On receiving the RouteSelect event, Call Routing Server passes it to Telephony Server. Telephony Server in turn passes it to the Avaya switch for routing to the agent.

The register * VDN 9888 as a routing can receive RouteRequest events from VDN. For more information about Routing VDN Lists, see the Routing VDN list in [Call Routing Server] on page 16.

Note:

Ensure that you type the event parameters accurately. Also, ensure that you use uppercase letters exactly as mentioned in parameter names and without spaces between words. To confirm a parameter name, see Appendix A: Plug-ins.

The following are various fields of the Script Plug-in and their appropriate values:

- Enable Client=True
- Client Library Name=ASGScriptHost.dll
- Display Name=Link to SQL Database
- Display Icon=someicon.ico
- Event Type=Native

The following parameters related to the Script Plug-in appears as one section:

- Configuration Section Name=Check Caller CLI
- Script Language=VBScript
- Script File Name=CheckCaller.vbs
- Script Start Function=main
- Event Name=RouteRequest
- Event Filter Name=VDN
- Event Filter Value=9888
- Return Event Name=RouteSelect
- Return Event Parameter Sequence=%RouteRegisterReqID%,%RoutingCrossRefID%
- Error Event Name=RouteEnd
- Error Event Parameter Sequence=%RouteRegisterReqID%,%RoutingCrossRefID%,%ErrorValue%,%VDN%
- Thread Pool Size=2
- Maximum Queued Requests=10

[ASG SOAP]

For information about the SOAP Plug-in parameter definitions and syntax, see Appendix A: Plug-ins on page 27.

The following example shows how to use Call Routing Server with the SOAP Plug-in to route incoming calls to the appropriate agent based on the digits enter by the caller.

Call Routing Server receives an event from Telephony Server and searches each section of the configuration to match the received information with the Event Name, Filter Name, and Filter Value. If Call Routing Server does not find a match in one section, it moves to the next section.

If Call Routing Server receives a RouteRequest event from VDN 9888*, it stops searching at Check Customer Details section

After Call Routing Server finds a match, it passes the event to the SOAP Plug-in. The SOAP Plug-in then connects to the Web service identified in the SOAP URL parameter (http://fitzgers/SOAPSample/Service1.asmx?WSDL).

The SOAP connection passes the Collected Digits and UUI variables with the RouteRequest event. For a select case of collected digits, the service returns a routing destination and account information within the SOAP return.

```
[WebMethod]
        public string CallRoute( string CollectedDigits, string CallerUUI, out
string RouteSelected, out string UUI)
        {

                switch (CollectedDigits)
                {
                    case "12345678":
                    {
                        RouteSelected = "3999";
                        UUI = "A/C 12345678, Balance $200.55, Priority Gold";
                        break;
                    }

                    case "56789012":
                    {
                        RouteSelected = "4999";
                        UUI = "A/C 56789012, Balance $4200.90, Priority Bronze";
                        break;
                    }
                    default:
                    {
                        RouteSelected = "5999";
                        UUI = "No Account information found";
                        break;
                    }

                }


               if (CallerUUI != "")
               {
                   RouteSelected = CallerUUI;
                   return "Routed using call UUI";
               }

               return "No UUI Received";


            }
```

To identify the route request, the SOAP Plug-in sends the following mandatory parameters to Call Routing Server:

- RouteRegisterReqID
- Routing CrossRefID

Followed by the following parameters from the SOAP return:

- RouteSelected
- UUI parameters

On receiving the RouteSelect event, Call Routing Server passes the event to Telephony Server, which in turn passes the event to the Avaya switch for routing to the agent.

The register * VDN 9888 as a routing VDN can receive RouteRequest events from VDN. For more information about Routing VDN Lists, see the Routing VDN list in [Call Routing Server] on page 16.

Note:
Ensure that you type the event parameters accurately. Also, ensure that you use uppercase letters exactly as mentioned in parameter names and without spaces between words. To confirm a parameter name, see Appendix A: Plug-ins.

The following are various fields of the SOAP Plug-in and their appropriate values:

- Enable Client=True
- Client Library Name=ASGSOAP.dll
- Display Name=Link to SOAP Services
- Display Icon=someicon.ico
- Event Type=Native

The following parameters related to the SOAP Plug-in appears as one section:

- Configuration Section Name=Check Customer Details
- SOAP URL=http://fitzgers/SOAPSample/Service1.asmx?WSDL
- SOAP Service=Service1
- SOAP Port=Service1Soap
- SOAP Method=ActiveCallRoute
- SOAP Method Parameter Sequence=CollectedDigits = %UserEnteredCode%, %UUI%, %VDN%, %UCID%
- Event Name=RouteRequest
- Event Filter Name=VDN
- Event Filter Value=9888
- Return Event Name=RouteSelect
- Return Event Parameter Sequence=%RouteRegisterReqID%, %RoutingCrossRefID%, $*$
- Thread Pool Size=2
- Maximum Queued Requests=10

Starting Call Center Elite Multichannel services

To start Call Center Elite Multichannel services:

1. From the Windows **Start** menu, select **Control Panel**.
2. Click **Administrative Tools**.
3. Click **Services**.

4. Right-click the following Call Center Elite Multichannel services and select **Start**.
   - Avaya Call Recording Config Service
   - Active Configuration Server
   - Avaya Experience Portal Config Service
   - AS AOL-ICQ Instant Messenger Gateway
   - AS Application Management Director
   - AS Call Routing Server
   - AS Communicator gateway
   - AS Email Media Store
   - AS Interaction Data Server-MultiMedia
   - AS Interaction Data Server-View
   - AS Interaction Data Server-Voice and Presence
   - AS License Director
   - AS Media Director
   - AS Media Proxy Service
   - AS MSN Messenger Gateway
   - AS Preview Contact Media Store
   - AS Short Message Service Gateway
   - AS Simple Messaging Media Store
   - AS Web Chat Gateway
   - AS XML Server

Appendix A: Plug-ins

Rules Plug-in

For information about the Rules Plug-in parameter definitions and syntax, see *Avaya Aura® Call Center Elite Multichannel Desktop User Guide*.

SOAP Plug-in

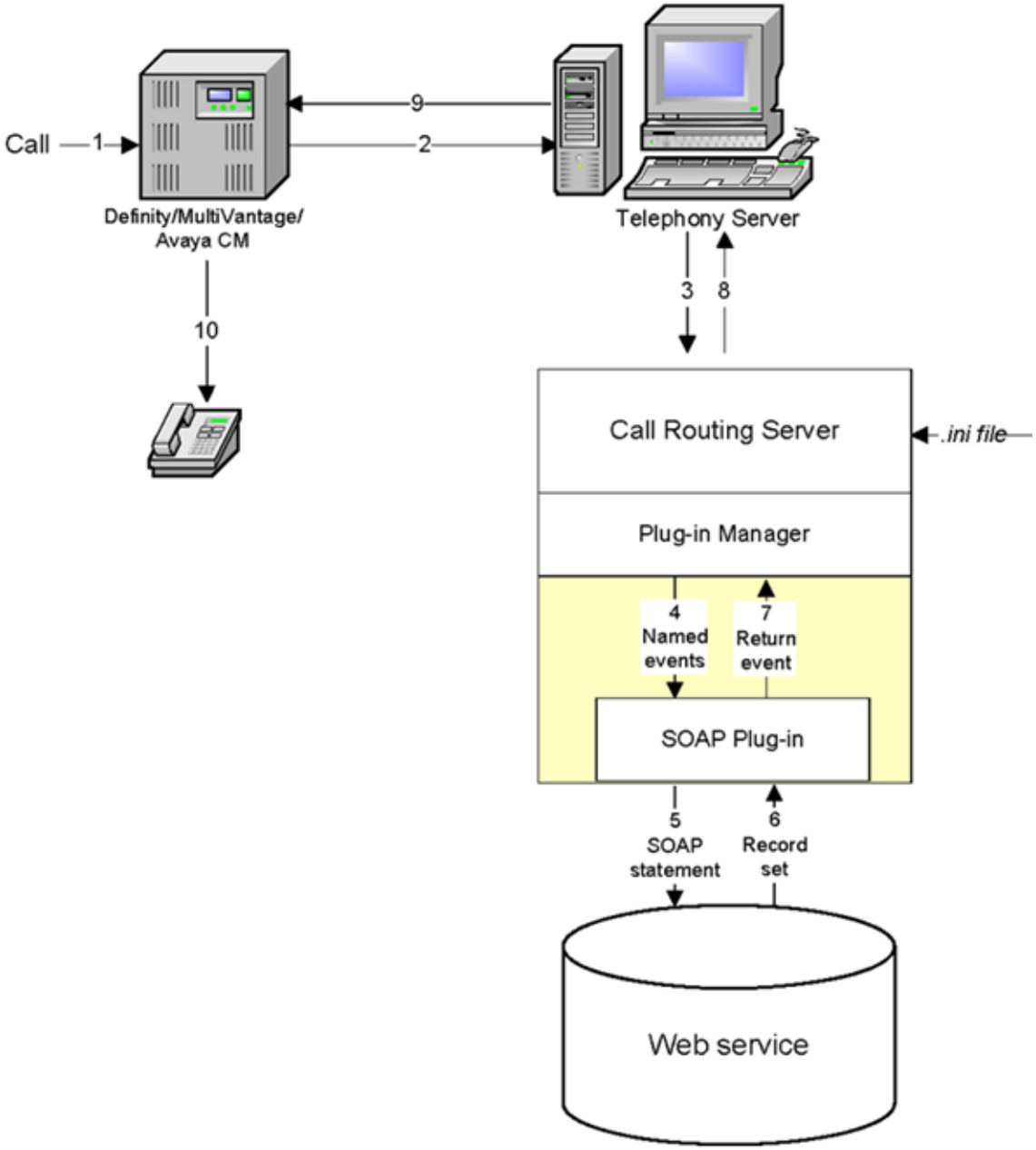
The SOAP Plug-in is a simple plug-in that allows you to integrate Call Center Elite Multichannel server applications with any Web service or SOAP service. You can integrate the SOAP Plug-in on an Intranet or the Internet, without the need for new development on the server.

You can plug in the SOAP Plug-in to any Call Center Elite Multichannel server application that supports the Plug-in Manager, such as Call Routing Server.

SOAP Plug-in uses the Microsoft Simple Object Access Protocol (SOAP) to connect to the Web service and allows Web-based information to be available to the controlling application.

SOAP

SOAP is a message-based protocol based on XML for accessing services on the Web. Initiated by Microsoft, IBM, and others, it uses XML syntax to send text commands across the Internet using HTTP.



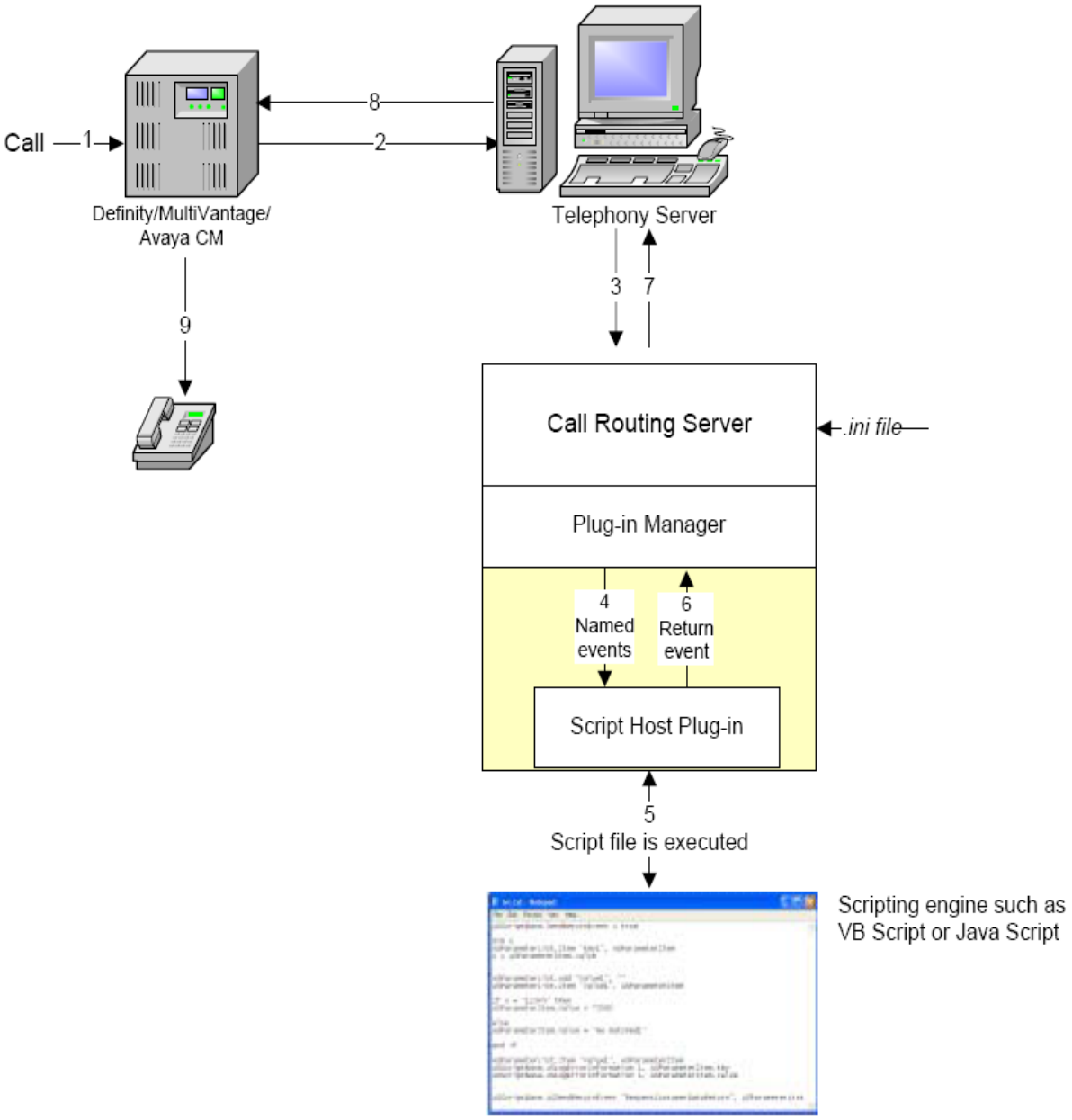
Script Plug-in

The Script Plug-in takes advantage of scripting engines developed by Microsoft and others. The Script Plug-in allows the execution of simple scripts written in VB Script or Java Script and provides services to Call Center Elite Multichannel.

You can plug in the Script Plug-in to any Call Center Elite Multichannel server application that supports the Plug-in Manager, such as Virtual Agent and Call Routing Server.

All scripting engines that conform to the Microsoft standard implement the ActiveScript interface. This common interface allows parent applications to use all scripting engines in an identical manner regardless of the scripting language.

The two most commonly available scripting engines are Visual Basic script and Java script, which are installed with Internet Explorer.

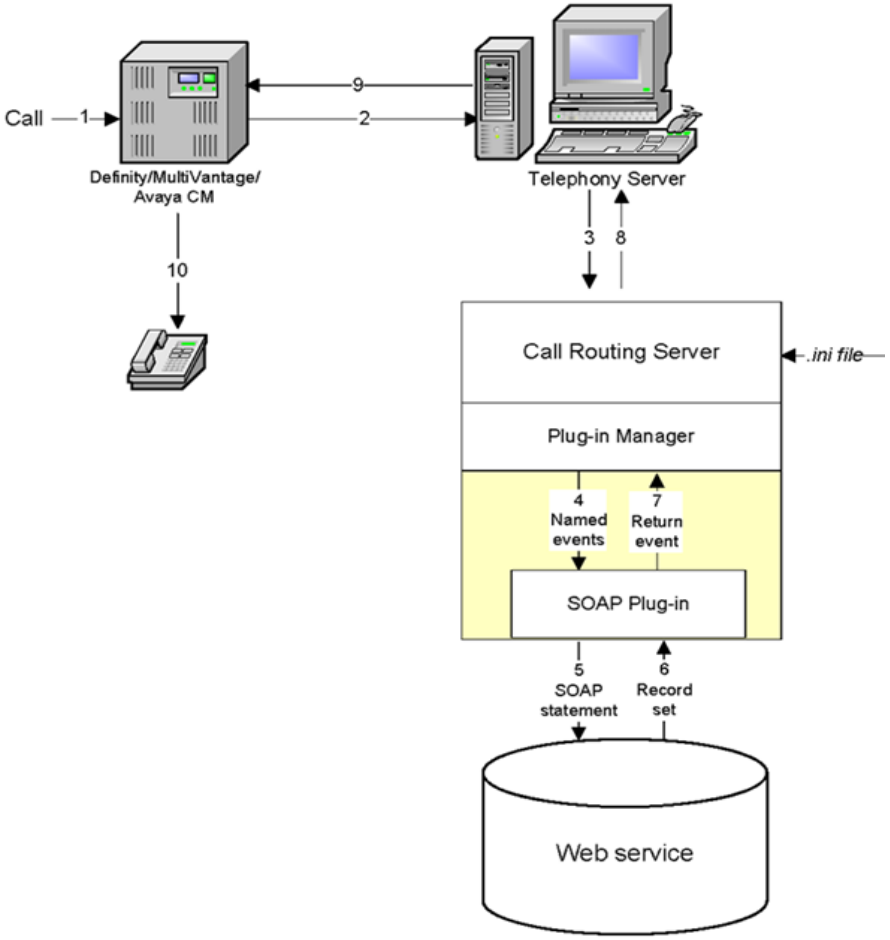


SQL Plug-in

The SQL Plug-in is a simple plug-in mechanism that allows you to integrate Call Center Elite Multichannel server applications with any SQL Server database. The SQL Plug-in does not need any development on the server.

You can plug in the SQL Plug-in to any Call Center Elite Multichannel server application that supports the Plug-in Manager, such as Call Routing Server.

SQL Plug-in uses Microsoft ADO to connect to a database and allows simple SQL functionality to be available to the controlling application. The detailed configuration set of the plug-in allows named events to be received from the controlling application. The events and associated parameters convert to a direct SQL statement which then passes to the database for processing. Returned results are extracted from the returned record set and passed back to the controlling application through an associated return event.



Plug-in installation

Install the Call Center Elite Multichannel plug-ins as part of the installation for applications that support the Plug-in integration and not as separate components.

For full instructions on how to install the supporting applications, see *Installing Avaya Aura® Call Center Elite Multichannel*.

In addition to the standard installation steps, this document explains:

- How to use command line parameters during installation to dictate where configuration information is retrieved.
- How to change the configuration source from the local .inc file to the Configuration Server if an application is already installed.
- How to automatically and silently install an application following a set of pre-defined selection options.

You can download *Installing Avaya Aura® Call Center Elite Multichannel* from the Avaya Support Web site: http://support.avaya.com.

Plug-in configuration

After you install the Call Center Elite Multichannel plug-ins, you must configure the plug-ins.

To configure Rules Plug-in, see *Avaya Aura® Call Center Elite Multichannel Desktop User Guide*.

To configure SOAP Plug-in, Scripts Plug-in, and SQL Plug-in, you must enter appropriate values in the following fields:

- **Enable Client**: A value that determines whether the controlling application enables the plug-in on startup. To enable the plug-in on startup, you must set this value to True.
- **Client Library Name**: The name of the generic plug-in. For example, ASGSOAP.dll.
- **Display Name**: The text that displays on the user interface of the client.
- **Display Icon**: An icon that displays on the user interface of the client. This icon represents the running processes of plug-ins.
- **Event Type**: A value that specifies whether the plug-in loaded by the server wants to do maintenance tasks or real work. If you set this value to Native, then it specifies that the plug-in wants to interface for native server functionality.

[Sections]

All parameters related to a plug-in must appear as one section and repeat the data to create more sections. A section is used if the received event and filter values match the configured event and filter parameters.

Parameters	Applicable for Plug-in
Configuration Section Name	SOAP Plug-in, Script Plug-in, SQL Plug-in
SOAP URL	SOAP Plug-in
SOAP Service	SOAP Plug-in
SOAP Port	SOAP Plug-in
SOAP Method	SOAP Plug-in
Event Name	SOAP Plug-in, Script Plug-in, SQL Plug-in
Event Filter Name	SOAP Plug-in, Script Plug-in, SQL Plug-in
Event Filter Value	SOAP Plug-in, Script Plug-in, SQL Plug-in
Return Event Name	SOAP Plug-in, Script Plug-in, SQL Plug-in
Return Event Parameter Sequence	SOAP Plug-in, Script Plug-in, SQL Plug-in
Script Language	Script Plug-in
Script File Name	Script Plug-in
Script Start Function	Script Plug-in
Connection String	SQL Plug-in
Database Server Name	SQL Plug-in
Database Name	SQL Plug-in
Database User Name	SQL Plug-in
Database User Password	SQL Plug-in
SQL Query	SQL Plug-in
SQL Stored Procedure Name	SQL Plug-in
SQL Stored Procedure Parameter Sequence	SQL Plug-in

- **Configuration Section Name**: A unique name to identify the configuration set. If the value is empty, the plug-in populates it with the value “UnNamed” plus a number indicating the section's position in the configuration. For example, if the name for the fourth configuration set not specified, then the plug-in populates it with a default name of “UnNamed4”.
- **SOAP URL**: The URL of the SOAP server. This URL allows the SOAP toolkit to fully load the Web Service Description Language (WSDL) file that describes the SOAP services offered.

  For example:

  `SOAP URL=`www.AvayaActiveEnterprise.com/TestService/TestSoap?WSDL
- **SOAP Service**: The service in the WSDL file that contains the operation specified in the Simple Object Access Protocol (SOAP) request message.
- **SOAP Port**: The name of the port in the WSDL file that contains the operation specified in the SOAP request message.
- **SOAP Method**: The name of the method that is executed on this SOAP service.

  SOAP Method Parameter Sequence is a comma-delimited sequence of parameters that passes to the SOAP method when run. This sequence can contain both static information and information received with the event from the controlling application. Parameters pass to the SOAP toolkit in the order in which they are listed.

  – **Static Information:** Static information takes the form NAME=VALUE.

  – **Received Information:** Information received from the controlling application that passes to the SOAP method. This information is denoted by the enclosing “%” character. An empty string is used where a parameter that is not present in the received parameter sequence is specified.

  For example:

  SOAP Method Parameter Sequence`=CallingNumber = %CallingNumber%, ID=27`
- **Event Name**: The name of the event sent by the controlling application that runs this configuration set. When combined with the **Event Filter Name** and **Event Filter Value** parameters, this name represents a unique combination. If filtering is not used, that is, the **Event Filter Name** parameter is empty, then the **Event Name** parameter must be unique. If the **Event Name** parameter is not unique, the first entry in the configuration sequence is used.
- **Event Filter Name**: A name expected in the parameter list sent with the event from the controlling application. This filter allows multiple configuration sets to have the same Event Name**,** but to be identified by a received parameter value. If this parameter is empty, filtering does not happen.

- **Event Filter Value**: The value the plug-in tries to match with the received event and event filter. If the received event and event filter match, this configuration set is in the correct sequence.

  For example:

  With a configuration set of "Event Filter Name=VDN" and "Event Filter Value=9830", any event received with a matching Event Name must also contain, as part of the event's parameter sequence, a set VDN/9830.

- **Return Event Name**: The name of the event that returns the results obtained by the configuration set to the controlling application. If this parameter is empty, the plug-in does not send a return event.
- **Return Event Parameter Sequence**: A list of parameters that the plug-in returns to the controlling application with the return event. If Return Event Parameter Sequence is empty, no parameters are returned. The parameters that are separated in a list by commas, come from the following sources:
  - **Script Language:** The language that this script runs. The script language must match with the name registered by the script engine during installation.
  - **Script File Name:** The name of the script file that runs when the event is received. If you specify a file path, then the plug-in runs the script from that path. Otherwise, the plug-in searches for the script in the working folder, which is the same folder that contains the executable of the controlling application.
  - **Script Start Function:** Optionally allows you to specify an entry point in the script. If this parameter is empty, it is assumed that the script is self describing and has a known entry point.
- **Connection String**: An ADO connection string that is used for database connectivity. If this parameter is empty, the **Database Server Name**, **Database Name**, and **Database User Name** parameters are used to form the connection string. Encrypt this string if it contains a password by using the %%ENCRYPT command. For more information, see Configuration Commands in *Installing Avaya Aura® Call Center Elite Multichannel*.
- **Database Server Name**: The name of the server where the database is located. If the **Connection String** parameter contains data, this entry is ignored.
- **Database Name**: The name of the database. If the **Connection String** parameter contains data, this entry is ignored.
- **Database User Name**: The user name required to gain access to the database. If this parameter is left blank, integrated security is assumed. If the **Connection String** parameter contains data, this entry is ignored.
- **Database User Password**: The password for the database user name. By default the password is "CCEUser0" before encryption. For encryption information, see Configuration Commands in *Installing Avaya Aura® Call Center Elite Multichannel*. If the **Connection String** parameter contains data, this entry is ignored.

- **SQL Query**: The SQL statement that runs when the event is received. This statement contains placeholder characters that allow the values received with the event to be inserted into the string before being sent forward to the database. A placeholder is the name of the variable received from the controlling application with a leading and trailing "%" character.

  For example:

  SQL Query=Select * from tblCustomers where CustomerNumber=%CallingNumber% or CustomerNumber=%UUI%. If an event is received with a parameter CallingNumber equal to 92996015 and a UUI equal to 111, the resultant SQL statement sent forward is Select * from tblCustomers where CustomerNumber=92996015 or CustomerNumber=111. If an event is received that does not contain the correct parameters, empty strings replace the values.

  The **SQL Query** and the **SQL Stored Procedure Name** parameters are mutually exclusive. If both parameters are present, **SQL Query** takes precedence.

- **SQL Stored Procedure Name**: The name of the stored procedure that is executed when the plug-in receives the event. The **SQL Query** and the **SQL Stored Procedure Name** parameters are mutually exclusive. If both parameters are present, **SQL Query** takes precedence.

- **SQL Stored Procedure Parameter Sequence**: A comma-delimited sequence of parameters that passes to the stored procedure. This sequence consists of parameters that are extracted from the received event parameters and fixed value parameters.

  – **Extracted Parameters:** If the sequence parameter contains a leading and trailing "%" character, such as %UUI%, the value associated is extracted from the parameters received with the event.

  – **Fixed Value Parameters:** Fixed value parameters takes the form "NAME=VALUE".

  For example:

  SQL Stored Procedure Parameter Sequence=%UUI%,%AParty%,CollectedDigits=44. This example sends three parameters to the stored procedure called UUI, AParty, and CollectedDigits. The parameters UUI and AParty contain the values extracted from the received event parameters. The CollectedDigits parameter have the fixed value "44".

Result Parameters

Result parameters allow you to send results generated from the SOAP service to the controlling application. Result parameters are enclosed in the "$" character. For example, if the configuration set contains the entry **Return Event Parameter Sequence**=$Destination$, the returned parameter is from the "Destination" field in the returned record set. The parameter set key is "Destination".

The returned parameter set also allows you to change the key value, using the format “KEYNAME=$SQL VALUE$”. For example, if the configuration set contains the entry **Return Event Parameter Sequence**=Destination=VDN, the returned parameter is from the “VDN” field in the returned record set. The parameter set key is “Destination”.

To return all fields in the resultant record set, you can use the parameter $*$. When this value is used, the key/value pairs in the event parameter list contain keys and values extracted from the record set.

For SQL Plug-in, you can also specify the SQL result field by the numeric position in the returned record set. You can specify the SQL result field using the format “$_NUMERIC LOCATION_$”.

For example:

If the configuration set contained the entry **Return Event Parameter Sequence=Destination**=$_1_$, the returned parameter is from the first field in the returned record set. The parameter set key is “Destination”.

Transfer Parameters

Transfer parameters allow you to transfer important sequence and tracking information received with the original event back to the controlling application. This information allows the controlling application to match the original event with the return event. Transfer parameters are enclosed in the “%” character.

For example:

If the configuration set contains the entry **Return Event Parameter Sequence**=%RouteRegisterReqID%, %RoutingCrossRefID%, then the return event parameter list contains the entries "RouteRegisterReqID" and "RoutingCrossRefID". The values are copied from the parameter list received from the controlling application.

To transfer every parameter from the received event to the returned event, you can use the parameter %*%.

Fixed Value Parameters

Fixed value parameters allow you to include fixed key/value data in the parameter list returned to the controlling application. Fixed value parameters take the form “NAME=VALUE".

For example:

If the configuration set contains the entry **Return Event Parameter Sequence**=Destination=8579, then the return event parameter list contains a key entry “Destination” with the fixed value “8579”.

Parameter types can appear in any order in the parameter sequence. The order in which the parameter types appear dictates the order in which they are placed into the parameter list. Example: If the configuration set contains the entry **Return Event Parameter Sequence**=Destination=8579, %RouteRegisterReqID%, then the return event parameter list has a key entry “Destination” which has a fixed value “8579”, a key of “RouteRegisterReqID” with a value copied from the original received event.

Parameters	Applicable for Plug-in
Return Event Allow Multiple Events	SOAP Plug-in, SQL Plug-in
Thread Pool Size	SOAP Plug-in, Script Plug-in, SQL Plug-in
Maximum Queued Requests	SOAP Plug-in, Script Plug-in, SQL Plug-in
Error Event Name	Script Plug-in, SQL Plug-in
Error Event Parameter Sequence	Script Plug-in, SQL Plug-in
Return Event Total Record Count Name	SQL Plug-in
Return Event Current Record Count Name	SQL Plug-in
No Record Event Name	SQL Plug-in
No Record Event Parameter Sequence	SQL Plug-in

- **Return Event Allow Multiple Events**: If set to True, this parameter allows the controlling application to receive multiple return events. If set to False, only the first record in the record set is returned.
- **Thread Pool Size**: The number of threads this configuration section can use to dispatch event requests for processing. A number of threads are required because each configuration section receive multiple simultaneous events from the controlling application and need to process these requests in parallel. Requests received when there are no threads immediately available is queued for the next available thread. The thread pool size should range between 1 and 10. The default is 2.
- **Maximum Queued Requests**: The maximum number of requests that can be queued to the thread pool.
- **Error Event Name:** The name of the event passed to the controlling application when any error occurs that prevents the execution of the specified stored procedure or SQL statement. This includes all ADO errors, including failure to open the connection to the specified database. If this parameter is empty, the event specified in the Return Event Name is used. If the Return Event Name is used, the parameter sequence from the Error Event Parameter Sequence is used, if available. If not, the parameter sequence from the Return Event Parameter Sequence is used.
- **Error Event Parameter Sequence:** The sequence of parameters that passes with the error event. This sequence takes the same format as Return Event Parameter Sequence.

- **Return Event Total Record Count Name**: This parameter allows you to alter the name of the event parameter that contains the number of records received with the SQL query. If this parameter is blank, no information about the number of records received returns.

  For example:

  Return Event Total Record Count Name=NumberOfRecords. With this example, if the query of the database returned 10 records then, when the return event passes to the controlling application, it contains a parameter set of NumberOfRecords=10.

- **Return Event Current Record Count Name**: The name of the event parameter that contains the number of the current record in a multi-record set to be altered. If this parameter is blank, no information about the current record number returns.

  For example:

  Return Event Current Record Count Name=CurrentRecord. With this example, if the query of the database returned 10 records then, when the return event passes for the fourth record to the controlling application, it contains a parameter set of CurrentRecord=4.

  When the current record count and total record count parameters are equal in an event, then this is the last event.

Error log file

The Call Center Elite Multichannel plug-ins log error information related to plug-ins operation into the error log file of their controlling application.

For Call Routing Server log files, refer to the directory: `<CCEM_INSTALL_DIR>\Avaya Aura CC Elite Multichannel\Server\Call Routing Server\ACR Server\ACRSLogFiles`.

Events sent from Call Routing Server to Plug-in

The following events are sent from Call Routing Server to the plug-in.

For parameter descriptions, see the *Avaya Application Enablement Services TSAPI for Avaya Communication Manager Programmer's Reference Guide*.

CallDelivered

This event starts when the call is delivered to a monitored VDN. This event starts when Call Routing Server receives the cstaCallDelivered event from Telephony Server.

The following parameters pass with the CallDelivered event:

- VDN
- CallID
- CallingDevice
- CalledDevice
- AlertingDevice
- LastRedirectionDevice
- Cause
- DeliveredType
- Split
- UserEnteredCode
- UUI
- TrunkGroup
- TrunkGroupMember
- UCID
- LAIType
- LAIPriority
- LAIHours
- LAIMinutes
- LAISeconds
- LAISourceVDN
- OCICallingDevice
- OCICalledDevice
- OCITrunkMember
- OCITrunkGroup
- OCIUserEnteredCode
- OCIUUI
- OCIUCID

- OCILAIType
- OCILAIPriority
- OCILAIHours
- OCILAIMinutes
- OCILAISeconds
- OCILAISourceVDN

CallEstablished

This event starts when the call is answered and is triggered by receiving the cstaEstablished event from Telephony Server.

The following parameters pass with the CallEstabished event:

- VDN
- CallID
- AnsweringDevice
- CallingDevice
- CalledDevice
- Cause
- LastRedirectionDevice
- Split
- UserEnteredCode
- UUI
- TrunkGroup
- TrunkGroupMember
- UCID
- LAIType
- LAIPriority
- LAIHours
- LAIMinutes
- LAISeconds
- LAISourceVDN
- OCICallingDevice

- OCICalledDevice
- OCITrunkMember
- OCITrunkGroup
- OCIUserEnteredCode
- OCIUUI
- OCIUCID
- OCILAIType
- OCILAIPriority
- OCILAIHours
- OCILAIMinutes
- OCILAISeconds
- OCILAISourceVDN

ConnectionCleared

This event is mapped directly from the cstaConnectionCleared event received from Telephony Server.

The following parameters pass with the ConnectionCleared event:

- VDN
- CallID
- ReleasingDevice
- Cause

CallCleared

This event is mapped directly from the cstaCallCleared event received from Telephony Server.

The following parameters pass with the CallCleared event:

- VDN
- CallID
- Cause
- Reason

RouteRequest

This event is mapped directly from the cstaRouteRequest event received from Telephony Server.

The following parameters pass with the RouteRequest event:

- RouteRegisterReqID
- RouteCrossRefID
- CurrentRoute
- CallID
- CallingDevice
- TrunkGroup
- TrunkGroupMember
- UserEnteredCode
- LAIType
- LAIPriority
- LAIHours
- LAIMinutes
- LAISeconds
- LAISourceVDN
- UUI
- UCID
- VDN

RouteEnd

This event is mapped directly from the cstaRouteEnded event received from Telephony Server.

The following parameters pass with the RouteEnd event:

- RouteRegisterReqID
- RoutingCrossRefID
- ErrorValue
- VDN

RouteUsed

This event is mapped directly from the event received from Telephony Server.

The following parameters pass with the RouteUsedEvent:

- EventName
- Comma-Separated Event Arguments

TServerLinkStatus

This event is broadcasted to all extensions when Telephony Server link status changes.

The following parameters pass with the TServerLinkStatus event:

- **LinkStatus:** Two values are valid: “Up” and “Down”.
- **IsPrimaryLink:** “True” or “False”. This parameter indicates whether this link is the primary link or the secondary link.

Events sent from Plug-in to Call Routing Server

The plug-in instructs Call Routing Server to perform tasks by use of a fixed command set.

The following commands contain a mixture of command parameters that are mandatory (M) and command parameters that are optional (O). If a mandatory parameter is missing, the command request is ignored.

For parameter descriptions, see the *Avaya Application Enablement Services TSAPI for Avaya Communication Manager Programmer's Reference*.

RouteSelect

Received from the Call Center Elite Multichannel plug-in, this event instructs Call Routing Server to direct a call to the specified destination. This event expects the following parameters:

- RouteRegisterReqID (M)
- RoutingCrossRefID (M)
- RouteSelected (M)
- RouteUsedReq (O)

- CallingDevice (O)
- DirectAgentCallSplit (O)
- UserProvidedCode (O)
- UUI (O)

RouteEnd

Received from the Call Center Elite Multichannel plug-in, this event instructs Call Routing Server to end routing association on a specific call. This event expects the following parameters:

- RouteRegisterReqID (M)
- RoutingCrossRefID (M)
- ErrorValue (M)

ClearCall

Received from the Call Center Elite Multichannel plug-in, this instructs Call Routing Server to clear the call. The following parameter is received with the request:

- **CallID (M):** The Avaya Communication Manager that represents the call to be cleared.

ClearConnection

Received from the Call Center Elite Multichannel plug-in, this instructs Call Routing Server to clear a party from the call. The following parameters are received with the request:

- **CallID (M):** The Avaya Communication Manager that represents the call.
- **Device (M):** The device to be dropped from the call.
- **UUI (O):** Any user-to-user information that should be inserted with the clear connection request when it is sent to the Avaya Telephony Server.

Script COM Interface

The scripting interface IActiveScript allows custom COM objects to be exposed to the script. This gives the script access to host parameters and methods. The Script Plug-in exposes several COM interfaces to allow the script to operate effectively. Most script interfaces expect COM interfaces to be dual, and parameters passed to be variants. The Script Plug-in exposes interfaces that conform to this requirement.

Script COM Object - Parameter Set

Parameters received from the controlling application are made available to the script. This is achieved by creating a COM object that can be accessed from the script. This COM object is created and populated with the parameters received with the ASHandleClientEvent. It must also be added to the script site using the AddNamedItem function. The COM object is named in the script as ASParameterList.

COM Object Definition - ParameterItemList

The COM object exposed to the script has the following properties and methods:

- **Count:** This property returns a count for the number of items in the parameter set.
- **Item:** This method takes a single parameter and returns the matching ParameterItem object from the list. The supplied parameter is either the index in the parameter list or the key of the parameter item.
- **Add:** This method takes two parameters and adds a new value to the parameter list. The parameters that are passed include the parameter name and parameter value.

COM Object Definition – Parameter Item

The event data passed with the event is a listing of ParameterItem objects. Each object represents a key/value pair. These properties are implemented in the COM object as:

- **Key:** Returns the name of the parameter item.
- **Value:** Returns the value of the parameter item.

Script COM - Properties

SendReturnEvent

Syntax:

```
SendReturnEvent As Boolean
```

Description:

The script sets this property to suppress sending of a return event automatically by the plug-in.

When False, the plug-in automatically sends the return event specified in the configuration when the script ends.

Read-only. The default value for this property is False.

CurrentEventName

Syntax:

```
CurrentEventName As String
```

Description:

This property allows the script to determine the name of the event received from the controlling application that caused the script instance to run.

Read-only

Script COM - Methods

ASSendReturnEvent

Syntax:

```
ASSendReturnEvent(EventName, ParameterSet)
```

Description:

This method allows the script to call a named event from the script code.

Parameters:

- EventName

- ParameterSet

ASLogErrorInformation

Syntax:

```
ASLogErrorInformation(ErrorLevel, ErrorText)
```

Description:

This method allows the script to log execution details in the error log of the controlling application.

Parameters:

- **ErrorLevel:** The value that determines what level of error detail is saved in the error log. 0=None, 1=Information, 2=Minor, 4=Major, 8=Fatal.
- **ErrorText:** Free form text that is added to the error log at the specified level.

Index

C

Call Routing Server
  administration 15
Call Routing Server ini file
  AS client extensions 17
  ASG rules 19
  ASG script 20
  ASG SOAP 22
  ASG SQL 17
  call routing server 16
  error logging 15
  telephony services 16
Call Routing Server to plug-in 39
calls received to monitored VDN 8
calls received to routing VDN 8
configuring Call Routing Server 15

E

Error Logging 39
error logging 10
  diagnostic testing 11
  logging levels 10
events from Call Routing Server
  CallCleared 42
  CallDelivered 40
  CallEstablished 41
  ConnectionCleared 42
  RouteEnd 43
  RouteRequest 43
  RouteUsed 44
  TServerLinkStatus 44
events from Call Routing Server to the plug-in 39
events from the plug-in
  ClearCall 45
  ClearConnection 45
  RouteEnd 45
  RouteSelect 44
events from the plug-in to Call Routing Server 44

I

installation 13

L

licensing 10

P

plug-in to Call Routing Server 44

S

SQL plug-in
  accessing information from SQL server database . . 8
starting the CCEM service 24

W

what is call routing server? 7
what is the Script plug-in? 29
what is the SOAP Plug-in? 27
what is the SOAP? 28
what is the SQL Plug-in? 31