Notice

While reasonable efforts were made to ensure that the information in this document was complete and accurate at the time of printing, Avaya Inc. can assume no liability for any errors. Changes and corrections to the information in this document may be incorporated in future releases.

Documentation disclaimer

Avaya Inc. is not responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya.

Link disclaimer

Avaya Inc. is not responsible for the contents or reliability of any linked Web sites referenced elsewhere within this Documentation, and Avaya does not necessarily endorse the products, services, or information described or offered within them. We cannot guarantee that these links will work all of the time and we have no control over the availability of the linked pages.

License

USE OR INSTALLATION OF THE PRODUCT INDICATES THE END USER'S ACCEPTANCE OF THE TERMS SET FORTH HEREIN AND THE GENERAL LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE AT http://support.avaya.com/LicenseInfo/ ("GENERAL LICENSE TERMS"). IF YOU DO NOT WISH TO BE BOUND BY THESE TERMS, YOU MUST RETURN THE PRODUCT(S) TO THE POINT OF PURCHASE WITHIN TEN (10) DAYS OF DELIVERY FOR A REFUND OR CREDIT.

Avaya grants End User a license within the scope of the license types described below. 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 End User. "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. "Software" means the computer programs in object code, originally licensed by Avaya and ultimately utilized by End User, whether as stand-alone Products or pre-installed on Hardware. "Hardware" means the standard hardware Products, originally sold by Avaya and ultimately utilized by End User.

License type(s)

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 Product that permits one user to interface with the Software. Units may be linked to a specific, identified Server.

Copyright

Except where expressly stated otherwise, the Product is protected by copyright and other laws respecting proprietary rights. Unauthorized reproduction, transfer, and or use can be a criminal, as well as a civil, offense under the applicable law.

Third-party components
Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the Product ("Third Party Terms"). Information identifying Third Party Components and the Third Party Terms that apply to them is available on Avaya’s web site at: http://support.avaya.com/ThirdPartyLicense/.

**Avaya 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. Suspected security vulnerabilities with Avaya Products should be reported to Avaya by sending mail to: securityalerts@avaya.com.

For additional support telephone numbers, see the Avaya Web site: http://www.avaya.com/support.

**Trademarks**

Avaya is a trademark of Avaya Inc.

All non-Avaya trademarks are the property of their respective owners.

**Avaya support**

Avaya provides a telephone number for you to use to report problems or to ask questions about your contact center. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Web site: http://www.avaya.com/support.

For the most current versions of documentation, go to the Avaya support Web site: http://www.avaya.com/support.
**Preface**

**Contents**

This following few pages provides information that will help you use this document.

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>iv</td>
</tr>
<tr>
<td>Related Documents</td>
<td>vi</td>
</tr>
<tr>
<td>Knowledge Base</td>
<td>vi</td>
</tr>
<tr>
<td>Chapter 1 Introduction</td>
<td>7</td>
</tr>
<tr>
<td>What is the Script Plug-in?</td>
<td>8</td>
</tr>
<tr>
<td>Error Logging</td>
<td>9</td>
</tr>
<tr>
<td>Installation</td>
<td>9</td>
</tr>
<tr>
<td>Configure Script Plug-in</td>
<td>11</td>
</tr>
<tr>
<td>Sample Configuration for Call Routing Server</td>
<td>12</td>
</tr>
<tr>
<td>Sample Configuration for IVR Server</td>
<td>15</td>
</tr>
<tr>
<td>Appendix 1 Appendix: Call Routing Server</td>
<td>19</td>
</tr>
<tr>
<td>Call Routing Server to Plug-in</td>
<td>20</td>
</tr>
<tr>
<td>Plug-in to Call Routing Server</td>
<td>24</td>
</tr>
<tr>
<td>Script COM Interface</td>
<td>26</td>
</tr>
<tr>
<td>Properties</td>
<td>26</td>
</tr>
<tr>
<td>Methods</td>
<td>27</td>
</tr>
</tbody>
</table>
### Appendix 2 Appendix: IVR Server

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVR Server to Plug-in</td>
<td>29</td>
</tr>
<tr>
<td>Plug-in to IVR Server</td>
<td>31</td>
</tr>
</tbody>
</table>

### Glossary of Terms

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32</td>
</tr>
</tbody>
</table>

### Index

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>33</td>
</tr>
</tbody>
</table>
Related Documents

For more information on how the SQL Plug-in interacts with the Call Routing Server, refer to the *Call Routing Server User Guide*.

For more information on how the Script Plug-in interacts with the IVR Server, refer to the *IVR Server User Guide*.

Knowledge Base

For information on any errors and updates relating to this document, visit the *Avaya Contact Center Express Knowledge Base* (http://support.avayacce.com).
CHAPTER 1

Introduction

In This Chapter

What is the Script Plug-in? .................................................. 8
What is the Script Plug-in?

Script Plug-in is a simple plug-in mechanism that takes advantage of scripting engines developed by Microsoft and others, to allow simple scripts written in VB Script or Java Script to be executed and provide services to the Contact Center Express suite.

This plug-in can be plugged in to any Contact Center Express server application that supports the Plug-in Manager, such as IVR Server and the Call Routing Server.

All scripting engines that conform to the Microsoft standard implement the IActiveScript interface. This common interface allows all scripting engines to be consumed by parent applications in an identical manner regardless of the scripting language they implement.

The two most commonly available scripting engines are VB Script and Jscript from Microsoft. These are components installed with Internet Explorer.
Error Logging

Contact Center Express plug-ins log error information relating to their operation into the error log file of their controlling application.

For Call Routing Server log files, refer to the directory: C:\Program Files\Avaya\Contact Center Express\Server\Call Routing Server\ACR Server\ACRSLogFiles.

Installation

For IVR Server log files, refer to the directory: C:\Program Files\Avaya\Contact Center Express\Server\IVR Server\AIVR Server\AIVRSLogFiles.

The configuration and monitoring of this Contact Center Express service is accomplished by the Application Management Service installation of which is a necessary pre-requisite for the configuration and setup of Virtual Agent. For detailed information on how to install Application Management Service refer to the about the Contact Center Express Installation Guide.pdf.

One component of this service is the Contact Center Express Control Panel, a user interface that allows you to add and change configuration and operation information about currently monitored servers. For detailed information on how to configure this service, refer to the about the Contact Center Express Control Panel User Guide.pdf.
Install Plug-in

Contact Center Express plug-ins are not installed as separate components, but as part of the installation for applications that support their integration.

For full instructions on how to install these supporting applications, refer to the Contact Center Express Installation Guide.pdf.

The Contact Center Express Installation Guide is on the Contact Center Express DVD (Overview and Miscellaneous folder) or can be downloaded from the Avaya Contact Center Express website (http://www.avayacontactcenterexpress.com).
CHAPTER 2

Administration

In This Chapter

Configure Script Plug-in .................................................. 12
Sample Configuration for Call Routing Server .................. 15
Sample Configuration for IVR Server .............................. 17
Configure Script Plug-in

**Enable Client.** A value that determines if the controlling application will enable this plug-in on startup. True=enabled, False=disabled.

**Client Library Name.** The name of the generic plug-in: ASGScriptHost.dll.

**Display Name.** The text that displays on the client's user interface.

**Display Icon.** An icon that displays on the client's user interface to represent its running process.

**Event Type.** Indicates to the server loading the plug-in if it wishes to do maintenance tasks or real work. Use Event Type=Native to tell the server that the plug-in wishes to interface for native server functionality.

**[Sections]**

The following data must appear as one section but can be repeated to create more sections. A section will be used if the received event and filter values match the configured event and filter parameters.

**Configuration Section Name.** A unique name to identify the configuration set. If the value is empty, the plug-in will populate it with the value “UnNamed” plus a number indicating the section's position in the configuration. For example, if the fourth configuration set is not named, it will be given a default name of “UnNamed4”.

**Script Language.** The script language this script will run. This must match the name registered by the script engine when it is installed.

**Script File Name.** The name of the script file that will run when the event is received. If a file path is specified, the plug-in runs the script from there. Otherwise, it searches for the script in the working directory (the same folder that contains the executable of the controlling application).

**Script Start Function.** Optionally allows the entry point in the script to be specified. If this parameter is empty, it is assumed the script is self describing and has a known entry point.

**Event Name.** The name of the event sent by the controlling application that will invoke this configuration set. When combined with the Event Filter Name and Event Filter Value parameters, this represents a unique combination. If filtering is not used (by the Event Filter Name parameter being empty), the Event Name parameter is expected to be unique. If it is not, 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 be identified by a received parameter value. If this parameter is empty, filtering will not be used.

**Event Filter Value.** The value the plug-in will try and match with the received event and event filter. If there is a match, this configuration set is deemed the correct sequence. 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 will return any results obtained by this configuration set to the controlling application. If this parameter is empty, the plug-in will not send a return event. Note: With the COM interface (see "Script COM Interface" on page 26) available from the script, a return event can be passed directly from the script code. If this functionality is used, this parameter can be ignored.

Return Event Parameter Sequence. A list of parameters the plug-in will send back to the controlling application with the return event. If this parameter is empty, no parameters are returned. The parameters, which are separated in a list by commas, come from the following sources:

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.

Example: If the configuration set contained the entry Return Event Parameter Sequence=%RouteRegisterReqID%, %RoutingCrossRefID%, the return event parameter list will contain the entries "RouteRegisterReqID" and "RoutingCrossRefID" with the values copied from the parameter list received from the controlling application.

If you need to transfer every parameter from the received event to the returned event, you can use the parameter %**%. This parameter set may have been modified by the script.

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

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

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 will be passed with the error event. This sequence takes the same format as Return Event Parameter Sequence.

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 will receive multiple simultaneous events from the controlling application and will need to process these requests in parallel.) Requests received when there are no threads immediately available will be queued for the next available thread. The thread pool size should range 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.
Sample Configuration for Call Routing Server

This sample demonstrates how you can use Call Routing Server with the Script Plug-in to route incoming calls to the appropriate agent based on caller-entered digits.

Upon receiving an event from the Telephony Server, the Call Routing Server searches each section of its configuration, attempting to match the received information with the Event Name, Filter Name and Filter Value values. If the server doesn't make a match with one section, it moves to the next.

The Call Routing Server would stop at the following section (named Check Caller CLI) if it receives a RouteRequest event from VDN 9888*.

Once the match is made, the server passes the event to the Script Plug-in, which runs the script identified in the Script File Name parameter (CheckCaller.vbs). The plug-in looks for the script in the same folder that contains the Call Routing Server executable (C:\Program Files\Avaya\Contact Center Express\Server\Call Routing Server\ACR Server).

If the script (see below) matches the caller-entered digits received with the RouteRequest event (UserEnteredCode parameter) with the digits 1234, the plug-in will return a RouteSelect event to the 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 plug-in will instruct the server to route the call to extension 8553.

The plug-in sends a RouteSelect event and returns the extension's value to the Call Routing Server as the RouteSelected parameter. It will also return the received RouteRegisterReqID and RoutingCrossRefID parameters to identify the route request.

```vbs
Option Explicit
Dim x
Dim ParameterList, Item "UserEnteredCode", ParameterItem
x = ParameterItem.Value

ParameterList.Add "RouteSelected", ""
ParameterList.Item "RouteSelected", ParameterItem
If x = "1234" Then
    ParameterItem.Value = 8574
Else
    ParameterItem.Value = 8553
End If

ParameterList.Item "RouteSelected", ParameterItem
AScriptBase.ASLogError, Information 1, ParameterItem.Key
AScriptBase.ASLogError, Information 1, ParameterItem.Value

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

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

* VDN 9888 must be registered as a routing VDN before Call Routing Server can receive RouteRequest events from it (see the Call Routing Server parameter **Routing VDN List**).

Note: It is essential that event parameters are typed accurately, with uppercase letters in the right place and without spaces between words. To confirm a parameter, refer to the Appendix at the end of this document.

**Enable Client**=True

**Client Library Name**=ASGScriptHost.dll

**Display Name**=Link to SQL Database

**Display Icon**=someicon.ico

**Event Type**=Native

**First 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
Sample Configuration for IVR Server

This sample demonstrates how you can use IVR Server with the Script Plug-in to check a customer's account balance based on caller digits.

Upon receiving an event from the Telephony Server, the IVR Server searches each section of its configuration, attempting to match the received information with the Event Name, Filter Name and Filter Value values. If the server doesn't make a match with one section, it moves to the next.

The IVR Server would stop at the following section (named Check Caller CLI) if it receives a RequestCustomerData event from an IVR script named PlugInTest.

Once the match is made, the server passes the event to the Script Plug-in, which runs the script identified in the Script File Name parameter (CheckCallerivr.vbs). The plug-in uses the specified file path.

```vbs
AsscriptBase.SendReturnEvent = true

Dim x
ASPParameterList.Item "Key1", ASPParameterValue
x = ASPParameterValue.Value

ASPParameterList.Add "Value1", ""
ASPParameterList.Item "Value1", ASPParameterValue

If x = "12345" then
    ASPParameterValue.Value = 73502
Else
    ASPParameterValue.Value = "No matched1"
End If

ASPParameterList.Item "Value1", ASPParameterValue
AsscriptBase.ASLogErrorInformation 1, ASPParameterValue.Key
AsscriptBase.ASLogErrorInformation 1, ASPParameterValue.Value

AsscriptBase.ASSendReturnEvent "RequestCustomerDataReturn", ASPParameterValue
```
If the script matches the caller digits with 12345, the Script Plug-in writes the value 73502 to its own error log file.

If the script does not match the caller digits with 12345, the Script Plug-in writes the value 'No matched1' to the error log.

Note: It is essential that event parameters are typed accurately, with uppercase letters in the right place and without spaces between words. To confirm a parameter, refer to the Appendix at the end of this document.

**Enable Client**=True

**Client Library Name**=ASGScriptHost.dll

**Display Name**=Link to SQL Database

**Display Icon**=someicon.ico

**Event Type**=Native

First Section

**Configuration Section Name**=Check Caller CLI

**Script Language**=VBScript

**Script File Name**=c:\CheckCallerivr.vbs

**Event Name**=RequestCustomerData

**Event Filter Name**=ScriptName

**Event Filter Value**=PlugInTest

**Thread Pool Size**=2

**Maximum Queued Requests**=10

**Note**: The script file used in the above sample (CheckCallerivr.vbs) can be found in C:\Program Files\Avaya\Contact Center Express\Server\IVR Server\Samples\SQL Plug-in.
Appendix: Call Routing Server

This appendix outlines the events that pass between the Call Routing Server and the Rules Plug-in.

In This Appendix

Call Routing Server to Plug-in.................................20
Plug-in to Call Routing Server.................................24
Script COM Interface.............................................26
Call Routing Server to Plug-in

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

For parameter descriptions, refer to the *Avaya MultiVantage Application Enablement Services TSAPI for Avaya Communication Manager Programmer's Reference Release 3.1.0* (aes_tsapi-cm.pdf).

**CallDelivered**

This event is fired when the call is delivered to a monitored VDN. This event will fire whenever the Call Routing Server receives the `cstaCallDelivered` event from the Telephony Server.

The following parameters will be passed 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
Appendix: Call Routing Server

- OCILAIPriority
- OCILAIHours
- OCILAIMinutes
- OCILAISeconds
- OCILAISourceVDN

**CallEstablished**

This event is fired when the call is answered. This will be triggered by receiving the *cstaEstablished* event from the Telephony Server.

The following parameters will be passed with the *CallEstablished* 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 the Telephony Server.

The following parameters will be passed with the `ConnectionCleared` event.

- VDN
- CallID
- ReleasingDevice
- Cause
- UUI

**CallCleared**

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

The following parameters will be passed with the `CallCleared` event.

- VDN
- CallID
- Cause
- Reason

**RouteRequest**

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

The following parameters will be passed 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 \textit{cstaRouteEnded} event received from the Telephony Server.

The following parameters will be passed with the \textit{RouteEnd} event.

- RouteRegisterReqID
- RoutingCrossRefID
- ErrorValue
- VDN

**RouteUsed**

This event is mapped directly from the \textit{cstaRouteUsed} event received from the Telephony Server.

The following parameters will be passed with the \textit{RouteUsedEvent}.

- RouteRegisterReqID
- RoutingCrossRefID
- RouteUsed
- CallingDevice
- DestRoute

**TServerLinkStatus**

This event will be broadcasted to all extensions when the Telephony Server link status is changed.

The following parameters will be passed with the \textit{TServerLinkStatus} event.

- \textit{LinkStatus}: Two values are valid: “Up” and “Down”
- \textit{IsPrimaryLink}: “True” or “False”. This parameter indicates whether this link is the primary link or the secondary link.
Plug-in to Call Routing Server

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

In the following commands there is a mixture of command parameters that are mandatory (M) and those that are optional (O). If a mandatory parameter is missing, the command request will be ignored.

For parameter descriptions, refer to the *Avaya MultiVantage Application Enablement Services TSAPI for Avaya Communication Manager Programmer's Reference Release 3.1.0* (aes_tsapi-cm.pdf).

**RouteSelect**

Received from the Contact Center Express plug-in, this event instructs the Call Routing Server to direct a call to the specified destination. This event will expect the following parameters:

- RouteRegisterReqID (M)
- RoutingCrossRefID (M)
- RouteSelected (M)
- RouteUsedReq (O)
- CallingDevice (O)
- DirectAgentCallSplit (O)
- UserProvidedCode (O)
- UUI (O)

**RouteEnd**

Received from the Contact Center Express plug-in, this event instructs the Call Routing Server to end routing association on a specific call. This event will expect the following parameters:

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

**ClearCall**

Received from the Contact Center Express plug-in, this instructs the Call Routing Server to clear the call. The following parameters will be received with the request:

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

**ClearConnection**

Received from the Contact Center Express plug-in, this instructs the Call Routing Server to clear a party from the call. The following parameters will be 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 will be 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 will be 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 of the number of items that are in the parameter set.
- **Item.** The item method takes a single parameter and returns the matching ParameterItem from the list. The supplied parameter is either the index in the parameter list or the key of the parameter item.
- **Add.** The add method takes two parameters and adds a new value to the parameter list. The parameters passed will be the parameter name and the parameter value.

COM Object Definition – ParameterItem

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.

Properties

SendReturnEvent

Syntax: \[SendReturnEvent As Boolean\]

Description: This property can be set by the script to suppress the automatic sending (by the plug-in) of a return event.

When False, the plug-in will automatically send 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

Methods

**ASSendReturnEvent**

Syntax: \texttt{ASSendReturnEvent(EventName, ParameterSet)}

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

**EventName**

**ParameterSet**

**ASLogErrorInformation**

Syntax: \texttt{ASLogErrorInformation(ErrorLevel, ErrorText)}

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

**ErrorLevel** The value that determines what level of error detail will be saved in the error log. 0=None, 1=Information, 2=Minor, 4=Major, 8=Fatal.

**ErrorText** Free form text that will be added to the error log at the specified level.
Appendix: IVR Server

This appendix outlines the events that pass between the IVR Server and the SQL Plug-in.

In This Appendix

IVR Server to Plug-in..........................................................29
Plug-in to IVR Server..........................................................31
IVR Server to Plug-in

The following events will be sent from the IVR Server to the plug-in.

For parameter descriptions, refer to the *Avaya MultiVantage Application Enablement Services TSAPI for Avaya Communication Manager Programmer's Reference Release 3.1.0* (aes_tsapi-cm.pdf).

**CallDelivered**

The following parameters will be passed with the *CallDelivered* event.

- `MsgSessionID`
- `MsgInvokeID`
- `IVRPort`
- `AParty`
- `BParty`
- `DeliveringDN`
- `CollectedDigits`
- `UUI`
- `UCID`

**CallCleared**

The following parameters will be passed with the *CallCleared* event.

- `MsgSessionID`
- `MsgInvokeID`
- `IVRPort`

**RequestCustomerData**

The following parameters will be passed with the *RequestCustomerData* event.

- `MsgSessionID`
- `MsgInvokeID`
- `IVRPort`
- `ScriptName`
- `Key1`
- `Key2`
- `Key3`
- `Key4`
- `Key5`

**SetCustomerData**

The following parameters will be passed with the *SetCustomerData* event.
- MsgSessionID
- MsgInvokeID
- IVRPort
- ScriptName
- Key1
- Value1
- Key2
- Value2

**ExecuteCustomerFunction**

The following parameters will be passed with the `ExecuteCustomerFunction` event.

- MsgSessionID
- MsgInvokeID
- IVRPort
- ScriptName
- FunctionID
- Value1
- Value2
- Value3
- Value4
- Value5
Plug-in to IVR Server

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

In the following commands there is a mixture of command parameters that are mandatory (M) and those that are optional (O). If a mandatory parameter is missing, the command request will be ignored.

For parameter descriptions, refer to the Avaya MultiVantage Application Enablement Services TSAPI for Avaya Communication Manager Programmer's Reference Release 3.1.0 (aes_tsapi-cm.pdf).

RequestCustomerDataReturn
- MsgSessionID (M)
- MsgInvokeID (M)
- IVRPort (M)
- Value1 (O)
- Value2 (O)
- Value3 (O)
- Value4 (O)
- Value5 (O)

SetCustomerDataReturn
- MsgSessionID (M)
- MsgInvokeID (M)
- IVRPort (M)
- ReturnCode (O)

ExecuteCustomerFunctionReturn
- MsgSessionID (M)
- MsgInvokeID (M)
- IVRPort (M)
- ReturnCode (O)
- Value1 (O)
- Value2 (O)
- Value3 (O)
- Value4 (O)
Glossary of Terms

J

JScript
Microsoft's implementation of JavaScript. It is similar to JavaScript, but has extensions specifically for the Windows environment. Internet Explorer natively supports JScript and VBScript, and both scripting languages have their strengths and weaknesses for Web page programming.

S

Script
A program written in a general-purpose programming language. Typically less comprehensive than full-blown compiled languages.

Scripting host
The facility within a program that runs another program. For example, a Web browser is a scripting host that can execute instructions in languages such as Java and JavaScript.

Scripting language
A high-level programming, or command, language that is interpreted (translated on the fly) rather than compiled ahead of time. A scripting, or script, language may be a general-purpose programming language or it may be limited to specific functions used to augment the running of an application or system program.

V

VBScript
(Visual Basic SCRIPT) A scripting language from Microsoft. A subset of Visual Basic, VBScript is widely used on the Web for both client processing within a Web page and server-side processing in Active Server Pages (ASPs). As an ActiveX scripting language, VBScript is also used with the Windows Script Host to perform functions locally on a Windows machine.
Index

A
Administration • 11
Appendix
   Call Routing Server • 19
   IVR Server • 28

C
Call Routing Server to Plug-in • 20
Configure Script Plug-in • 12

E
Error Logging • 9

I
Install Plug-in • 10
Installation • 9
Introduction • 7
IVR Server to Plug-in • 29

J
JScript • 32

K
Knowledge Base • vi

M
Methods • 27

P
Plug-in to Call Routing Server • 24
Plug-in to IVR Server • 31
Preface • iv
Properties • 26

R
Related Documents • vi

S
Sample Configuration for Call Routing Server • 15
Sample Configuration for IVR Server • 17
Script • 32
Script COM Interface • 13, 26
Scripting host • 32
Scripting language • 32