User Guide

IVR Server

Contact Center Express

Release 2.1.1 - Issue 0

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.

Contents

CHAPTER 1

Preface

This chapter provides information that will help you use this document.

In This Chapter

Document Conventions

Convention	Description
Initial Capital Letters	Names of windows and dialog boxes. For example, the Add VDN dialog box appears.
[key] or [button]	The name of a button or keyboard key. For example, click the [OK] button.
Key+key	Hot key combinations you press down simultaneously to make the computer perform a function. For example, the Ctrl+S hot key combination saves your document.
Italicized text	Reference documents.
Click and double-click	The action of pressing the left or right mouse button once or twice. Always click the left button unless the right button is specified.

Related Documents

For information on how to install the IVR Server DIP component on an Avaya IVR, refer to the *DIP for Avaya IVR User Guide.*

For information on how to develop client applications using IVR Client, refer to the *IVR Client Developer Guide*.

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

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

Product Name Changes

The Avaya Interactive Voice Response (Avaya IVR) referred to in this document was previously known as the Conversant System for Interactive Voice Response (Conversant IVR).

In addition, Avaya Computer Telephony (Avaya CT) was previously known as CentreVu Computer Telephony (CentreVu CT).

Knowledge Base

For information on any errors and updates relating to this document, visit the Avaya Contact Center Express Knowledge Base via the ***website*** (http://www.AvayaContactCenterExpress.com).

CHAPTER 2

Introduction

This chapter introduces the capabilities and benefits of IVR Server and explains how the service works.

In This Chapter

What is IVR Server?

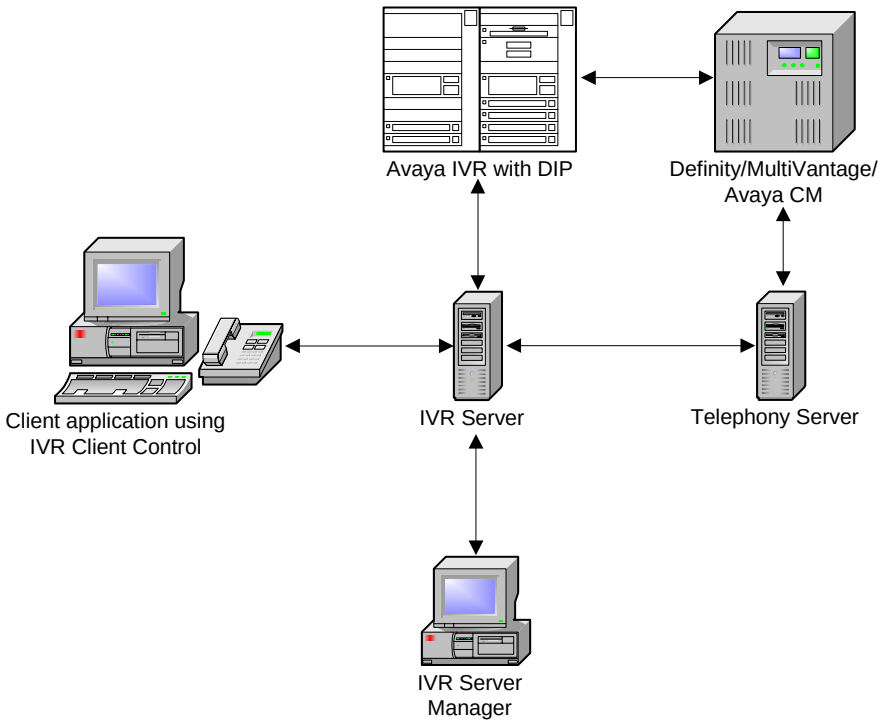
The IVR Server acts as a programming conduit between an IVR and a client application. Running on a Windows NT 4.0 or higher server, it monitors the VDNs used to distribute calls to IVR ports and monitors the individual IVR ports.

The IVR Server uses Avaya Computer Telephony software on the Telephony Server to give IVR scripts immediate access to call-related data and to invoke CTI-based call transfers.

The IVR Server Data Interface Process (DIP) acts as a bridge between the IVR Server and the Avaya IVR (formerly Conversant IVR). It retrieves data from the server, runs scripts, implements requests and returns events and messages to the server.

The server is stopped and started using IVR Server Manager. This application also allows you to specify the source of the server's configuration data and view the status of all monitored IVR ports.

The IVR Client control, which comes as part of the Developer toolkit, provides the interface developers need to create applications that interact with IVR scripts.



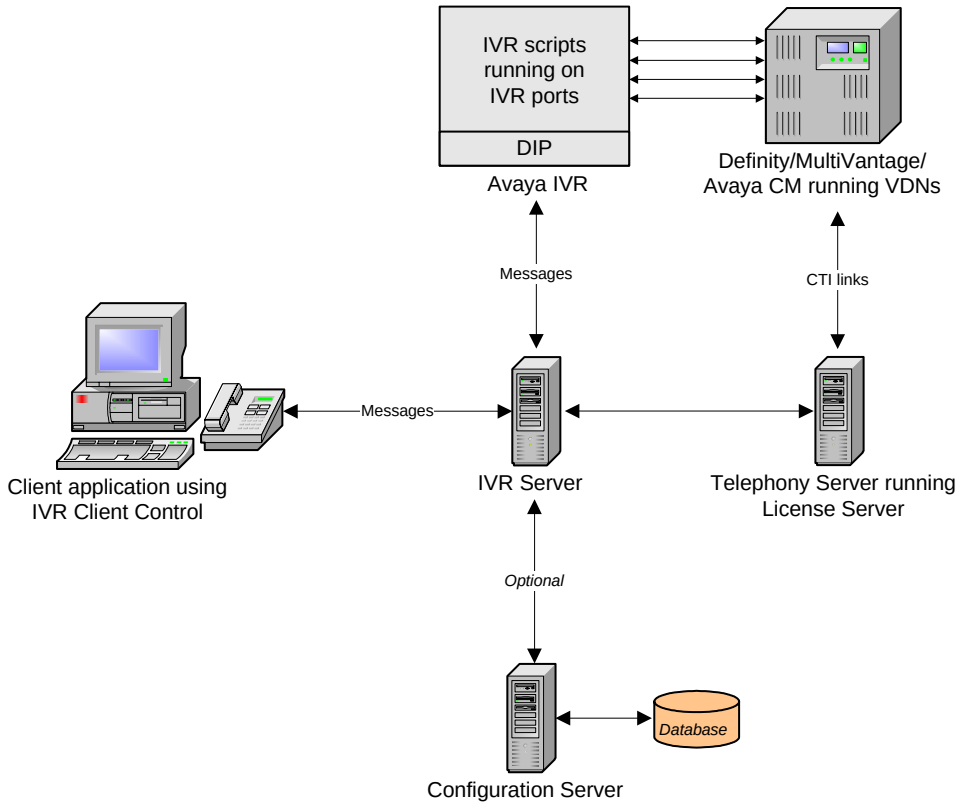
How Does IVR Server Work?

When IVR Server starts up, it retrieves all configuration data from its local configuration file.

Using this configuration information, IVR Server connects to each IVR via TCP/IP. It then connects to the Telephony Server, which runs License Director, and specifies how many IVR ports it requires run-time licenses for.

Once the licenses are received from License Director, IVR Server starts monitoring all specified IVR ports and VDNs.

The IVR Server, DIP interface, IVR Server Manager and IVR Client start to communicate using a common message protocol.



Licensing

The IVR Server requires a runtime license for each IVR port it is configured to monitor. It requests each license from the Telephony Server responsible for the port's call control.

The Telephony Server runs the transport software (Avaya Computer Telephony) that License Director uses to issue, audit and release licenses.

If the number of licenses available is more than the number requested, the IVR Server licenses (enables) all the ports it is configured to monitor.

The IVR server adjusts available ports based on the number free licenses. All modifications of the available runtime licenses are logged to the IVR Server error log for reference.

Messages for unlicensed ports

Messages from the IVR or IVR Client for ports with no runtime license are responded to with an error message. Call-related events from the Telephony Server for ports with no runtime license are ignored.

For more information on the License Director, refer to the *License Director User Guide*.

Error Logging

The IVR Server logs error information relating to its own operation and that of the IVR and IVR Server Manager to a series of log files.

A new log file is created for each day of the week. Each error log is overwritten on a weekly cycle. The name of the error log file records the day of the week and clearly identifies the file, for example, AIVRSMon.log.

The type of errors logged by the error log are determined by the logging level retrieved from its configuration data. Levels of error logging are:

- ERROR_LEVEL_NONE. No error logging takes place.
- ERROR_LEVEL_INFORMATION. Logs fatal, major, minor and trace information.
- ERROR_LEVEL_MINOR. Logs fatal, major and minor errors.
- ERROR_LEVEL_MAJOR. Logs fatal and major errors.
- ERROR_LEVEL_FATAL. 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.

All log files are placed in a folder named AIVRSLogFiles.

Error format

The format of an error in an error log starts with the date and time the error occurred, followed by the error log level, error description and location. For example:

[Wed Dec 11 15:45:38.205 2002]info, VDN 1234 is monitored[CStartMonitor]

CHAPTER 3

Installation

In This Chapter

Install Application

For full instructions on how to install this application, refer to the Contact Center Express Installation Guide.pdf.

The Contact Center Express Installation Guide is on the Contact Center Express CD (Overview and Miscellaneous folder) or can be downloaded from the ***Avaya Contact Center Express website*** (http://www.AvayaContactCenterExpress.com/Public_Documentation.htm).

CHAPTER 4

Configuration

This chapter explains how to configure and start the IVR Server.

In This Chapter

Configurable Parameters

The following configuration data is required by the IVR Server for operation.

[AIVR Server Data]

ServerIP. The IVR Server's IP address.

ServerPort. The IP port number the IVR Server uses to accept connections from clients. The default is 29093.

License Application Name. The type of license that IVR Server will require from the License Director. The default is: CCE IVR.

Present Last Monitored VDN To IVR. A value that determines if the last monitored VDN (True) or the first monitored VDN (False) will be presented to the IVR through the *CallDelivered* event. The default value is True.

[License Director]

License Director IP. The License Director IP through which this application will request and release licenses.

License Director Port. The License Director port through which this application will request and release licenses. The default is 29095.

[AIVRS Error Log Data]

ErrLogLevel.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.

MaxLogFileLen. The maximum amount of information, in kilobytes, that will be stored in an error log file before it is archived and a new file is created.

[IVR1]

IPAddr. The IVR's IP address.

IPPort. The IP port number the IVR Server uses to make outbound connections to different IVRs. The default is 29094.

This is also where you list monitored IVR ports. IVR port information displays in the format *x=y*, where *x* is a port number assigned by the IVR and *y* is the extension number assigned to the port in the Definity/MultiVantage/Avaya CM server.

For example:

0=4501
1=4402

[IVR2]

IP address and IVR port information specific to a second IVR. Identical format to above.

[IVR3]

IP address and IVR port information specific to a third IVR.

[AvayaCT Server Data]

ApplicationVersion.The version of TSAPI (Telephony Services Application Programming Interface) on the Avaya CT Server. TS1=TSAPI Version 1, TS2=TSAPI Version 2.

AvayaCTUserName. A valid user name on the Avaya CT Server (as entered in the Avaya CT security database).

AvayaCTUserPasswd. The password associated with above user name. By default, the Contact Center Express application will encrypt this data. For more information, see the *Contact Center Express Installation Guide* (Configuration Commands).

AvayaCTServerName. The telephony link (T-Link) name of the Avaya CT Server (Telephony Server) this server or application will connect to for information.

[IVR VDN Data]

A list of VDNs monitored via the Telephony Server. VDN information displays in the format *x=y*, where *x* is the name of the VDN (this could be a VDN that collects digits) and *y* is the number of the VDN. Note: The VDN name can includes spaces.

For example:

Sales=4486
Customer Services=4402

[AIDServerData]

AIDSEnabled. True=enabled, False=disabled.

AIDSIP. The IP address of the Interaction Data Server.

AIDSPort. The IP port number the Interaction Data Server uses to accept connections from clients and other servers. The default is 29090.

AIDSServerName. The user-friendly name of the Interaction Data Server. Reserved for future use. Leave blank.

AIDSServerType. Leave the default value: WindowsService. Reserved for future use. Leave blank.

AIDSUserName. The user name used to open a connection to the Interaction Data Server. Reserved for future use. Leave blank.

AIDSUserPasswd. The password associated with above user name. Reserved for future use. Leave blank.

AIDSClientAppName. The name of the client application that interacts with IVR scripts. Reserved for future use. Leave blank.

AIDSVersionNumber. Reserved for future use. Leave blank.

AIDSCurrentLicInfo. Reserved for future use. Leave blank.

[ASMClientToBeLoaded]

Reserved for future use.

[AS Client Extensions]

This section lists all loadable generic plug-ins. Each entry has the format "Friendly name=Plug-in section name". The plug-in section name points to (and is the same as) the section in the file that contains configuration data for that plug-in. Note: The friendly name can contain spaces.

For example:

SQL Plug-in=ASGSimpleSQL
Rules Plug-in=ASGRules

[ASGSimpleSQL]

For SQL Plug-in parameter definitions and syntax information, refer to the *SQL Plug-in User Guide*.

For examples of those events supported specifically by the IVR Server, refer to the following sample.

This sample demonstrates how you can use IVR Server with the SQL 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 EventName, 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 Account Balance) if it receives a RequestCustomerData event from an IVR script named SampleTest.

Once the match is made, the server passes the event to the SQL Plug-in, which executes an SQL query on the database identified in the Connection String parameter (sample.mdb). The plug-in uses the specified file path.

If the SQL query matches the caller digits received with the RequestCustomerData event with digits in the CallerDigits column of the database table, the SQL Plug-in will return the number 0 and the string in the AccountBalance column of the table. It returns these values via the RequestCustomerDataReturn event.

	CallerDigits	AccountBalance
	1285	7786
	2435	5678
▶		

The plug-in will also return the received MsgSessionID, MsgInvokeID and IVRPort parameters to identify the data request and the IVR port running the SampleTest script.

If the SQL query fails to match the caller digits, the SQL Plug-in will return the number 101 and string "NoMatching".

On receiving the RequestCustomerDataReturn event, the IVR Server, will pass it to the SampleTest script, which will read the values (number and string) to the caller.

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.

Client Enabled=True

Client Library Name=ASGSimpleSQL.dll

Display Name=Link to Database

Display ICON=someicon.ico

Event Type=Native

First Section

Configuration Section Name=Check Account Balance

Connection String=Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program Files\Avaya\Active Telephony\Server\Call Routing Server\Samples\SQL plug-in\sample.mdb;Persist Security Info=False

SQL Server Name=

SQL Data Base Name=

SQL User Name=

SQL User Password=

SQL Query=SELECT * FROM tblChecking WHERE CallerDigits=cstr(%Key1%)

SQL Stored Procedure Name=

SQL Stored Procedure Parameter Sequence=

Event Name=RequestCustomerData

Event Filter Name=ScriptName

Event Filter Value=SampleTest

Return Event Name=RequestCustomerDataReturn

Return Event Parameter Sequence=%MsgSessionID%, %MsgInvokeID%, %IVRPort%, Value1=0, Value2=$AccountBalance$

Return Event Allow Multiple Events=False

Return Event Total Record Count Name=

No Record Event Name=

No Record Event Parameter Sequence=%MsgSessionID%, %MsgInvokeID%, %IVRPort%, Value1=101, Value2="NoMatching"

Thread Pool Size=2

[ASGRules]

For Rules Plug-in parameter definitions and syntax information, refer to the *Rules Plug-in User Guide*.

For examples of those events supported specifically by the IVR Server, refer to the following sample.

This sample demonstrates how you can use IVR Server with the Rules Plug-in to retrieve data.

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 EventName, 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 if it receives a RequestCustomerData event from an IVR script named SampleTest.

Once the match is made, the server passes the event to the Rules Plug-in, which executes the configured rule(s).

In this case, the rule will run if the received event is RequestCustomerData and it carries a key value containing collected digits. If these rule requirements are met, the plug-in will return the number 0 and string 'Matched' via the RequestCustomerDataReturn event. This event also returns the received MsgSessionID, MsgInvokeID and IVRPort parameters to identify the data request and the IVR port running the SampleTest script.

On receiving the RequestCustomerDataReturn event, the IVR Server, will pass it to the SampleTest script, which will read the values (number and string) to the caller.

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.

Client Enabled=True

Client Library Name=ASGRules.dll

Display Name=Rules Engine

Display ICON=someicon.ico

Event Type=Native

First Section

Event Name=RequestCustomerData

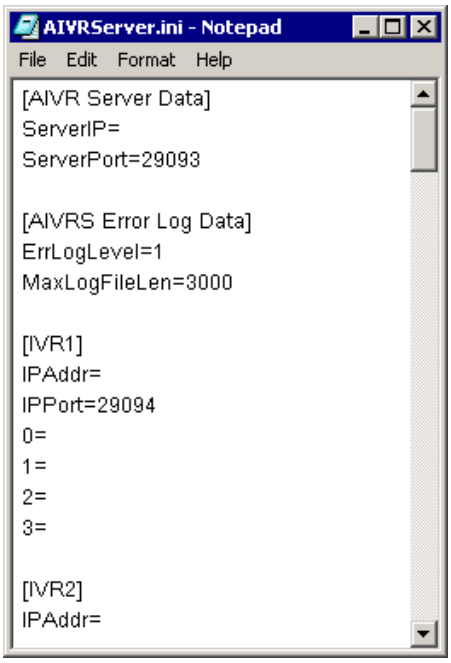
Event Filter Name=ScriptName

Event Filter Value=SampleTest

Rule1=When RequestCustomerData And Key1>Nothing Do ReturnEvent RequestCustomerDataReturn , %MsgSessionID%, %MsgInvokeID%, %IVRPort%,Value1=0,Value2="Matched" Then Stop

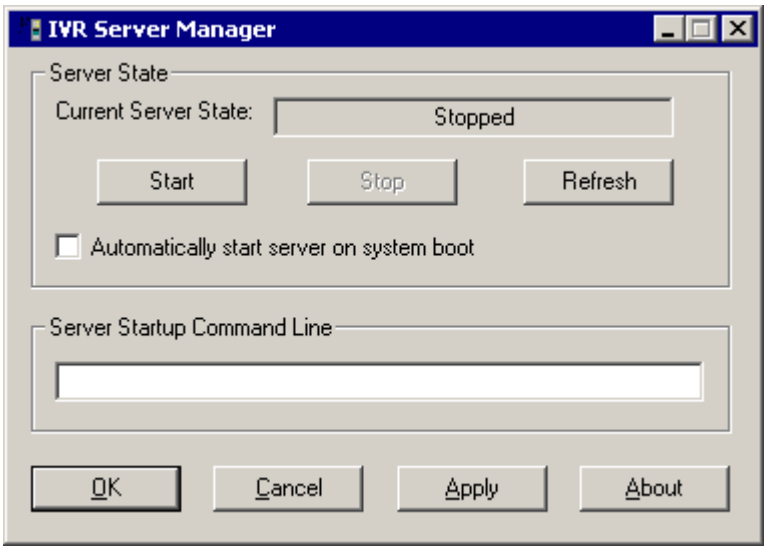
Configure IVR Server via Ini File

1. Click the [Start] button on the Windows Taskbar and select Programs > Avaya Contact Center Express > Server > IVR Server > Edit AIVRServer.ini from the pop-up menu.
2. Add configuration information as necessary. For detailed parameter information, refer to ***Configurable Parameters*** (on page 16). Note: Do not change section names, key names and, if possible, the default IP port numbers.



Start IVR Server

1 To open the IVR Server Manager, click the [Start] button on the Windows Taskbar and select Programs > Avaya Contact Center Express > Server > IVR Server > IVR Server Manager from the pop-up menu.



2 If you want IVR Server to retrieve its configuration data from:

- the .ini file in the default AIVRSConfig folder, do not alter the Command line.
- an.ini file stored on a shared network or location other than the default install folder, type /f *newfilepath*\AIVRServer.ini /z AIVRServer in the Command line text box.

3 To start the server, click the [Start] button.

CHAPTER 5

Appendix: Plug-ins

This appendix outlines the events that pass between the IVR Server and any of the Contact Center Express plug-ins.

In This Chapter

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 *Programmer's Guide for Definity Enterprise Communications Server (Defprog.pdf)* on your Avaya Computer Telephony CD-ROM.

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 Se*tCustomerData* 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 *Programmer's Guide for Definity Enterprise Communications Server (Defprog.pdf)* on your Avaya Computer Telephony CD-ROM.

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)

CHAPTER 6

Java Interface for IVR Server

In This Chapter

What is the Java Interface for IVR Server?

Java Interface for IVR Server is a java application which serves as an interface between the IVR Server and Java client applications.

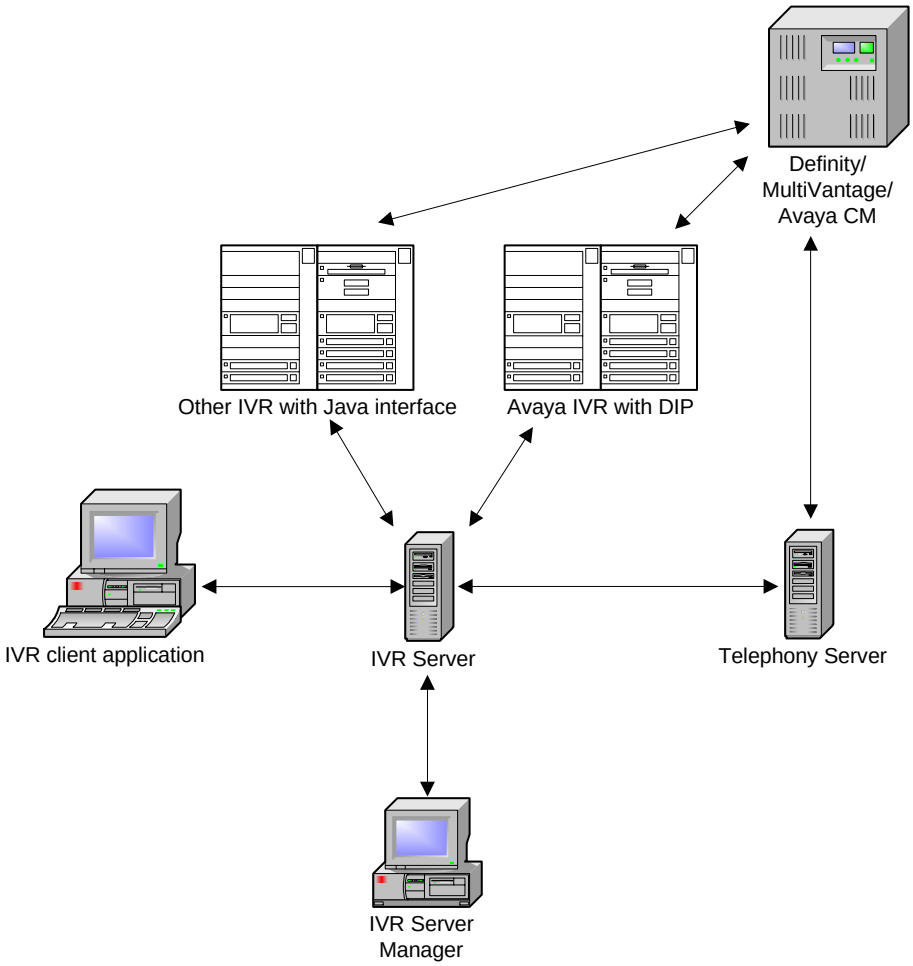
Each client opens one TCP/IP connection to the Java Interface for IVR Server.

All the messages flowing between the Java Interface for IVR Server and its clients are XML format.

The Java Interface for IVR Server contains two servers: one with a specified port or the default port 29095 to which IVR Server will open one permanent connection and the other with a specified port or the default port 29023 to which each Java Interface for IVR Server client will connect.

The default IP ports for the Java Interface for IVR Server to start up can be retrieved from aivrsjava.configdata file, but if the file does not exist or fails, Java Interface for IVR Server starts with its default IP ports.

The Java Interface for IVR Server is started through a command line.



Error Logging

The Java Interface for IVR Server logs error information relating 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 records the day of the week and clearly identifies the file, for example, aivrsjava_20030211.log is an error log file for 11 February 2003.

The type of errors logged by the error log are determined by the logging level retrieved from its configuration data. Levels of error logging are:

LOG_LEVEL_DEBUG	Logs all information including debug messages.
LOG_LEVEL_INFO	Logs all information except debug messages.
LOG_LEVEL_APPERR	Logs all messages except debug and information.
LOG_LEVEL_EXCEPTION	Logs exception and basic messages.
LOG_LEVEL_BASIC	Logs only basic messages.
LOG_LEVEL_NONE	No error logging takes place.

All log files are placed in a folder named ErrorLog in the directory. C:\Program Files\Avaya\Contact Center Express\Server\IVR Server\AIVRS Java Interface

Configure Java Interface for IVR Server

The configuration data file, aivrsjava.configdata, must be configured properly before starting Java Interface for IVR Server. It must be placed in a folder called ConfigData in the AIVRS Java Interface directory.

1. Open Windows Explorer and double-click the aivrsjava.configdata file located in: C:\Program Files\Avaya\Contact Center Express\Server\IVR Server\AIVRS Java Interface\ConfigData.
2. Change the IP port settings or error log level if necessary.

[Default IP/Port Settings]

IPPortForAIVRServer. The IP port for the IVR Server. The default is 29094.

IPPortForAIVRSJClient. The IP port for the Java Interface for IVR Server client application. The default is 29023.

[Error Log Settings]

AIVRSJavaLogLevel. Error Log Level. Debug = 0, Info = 1, Apperr = 2, Exception = 4, Basic = 8, None = 16.

Start Java Interface for IVR Server

There are three choices for the command line:

-version or version

When Java Interface for IVR Server is started with this command line, it displays version information and exits.

none

When Java Interface for IVR Server is started without a command line, it starts with the default IP ports specified in the aivrsjava.configdata file or its inner fixed IP ports if no aivrsjava.configdata file exists.

port1 port2

When Java Interface for IVR Server is started with two specified IP ports, port1 is for the IVR Server connection and port2 is for Java Interface for IVR Server client connections. It is possible to start up more than one Java Interface for IVR Server on a single machine as long as they have different IP ports.

Note: The platform, on which Java Interface for IVR Server will run, must have Java Virtual Machine installed.

- *To start the Java Interface for IVR Server, run the following command line:*

java -classpath *ABSOLUTEPATH*AIVRSJava.jar
com.aivrsjava.main.AIVRSJava %1 %2

Where *ABSOLUTEPATH* may be *ABSOLUTEPATH* = c:\Agile\AIVRServer\ on a Windows platform, or *ABSOLUTEPATH* = /ivr/aivrserver/ on a Unix platform.

Note: The above command line can be made to a batch file.

XML Message Format

All XML messages between the Java Interface for IVR Server and its clients must conform with the following format:

```
<Msg>

      <MID>Message ID</MID> //e.g. TransferCall,
TransferReturn, etc.

      <IID>Invoke ID</IID>

      <Port>IVR port number</Port>

      <Data>

            <Script>Script name</Script>

            <Target>1234</Target>

            <UUI>AIVRSJava</UUI>

            --

      </Data>

</Msg>
```

Note: All the parameters must be placed within the <Data> tag. Each message is a case-sensitive string. Only the message tags and message IDs defined in this chapter can be used.

Message IDs

The following list covers the message IDs that flow between the Java Interface for IVR Server and its clients.

Clients to Java Interface for IVR Server

- Initialise
- GetCallData
- GetCustomerData
- SetCustomerData
- ExecuteCustomerFunction
- SetLoggingData
- ClearCall
- TransferCall
- MakeCall
- HoldCall
- RetrieveCall
- QueryACDState

- ConferenceStart
- ConferenceComplete
- ConferenceAbort
- GetCallState
- WaitCallState

Java Interface for IVR Server to clients

- GetCallDataReturn
- GetCustomerDataReturn
- SetCustomerDataReturn
- ExecuteCustomerFunctionReturn
- SetLoggingDataReturn
- CallCleared
- TransferReturn
- MakeCallReturn
- HoldCallReturn
- RetrieveCallReturn
- QueryACDStateReturn
- ConferenceStartReturn
- ConferenceCompleteReturn
- ConferenceAbortReturn
- GetCallStateReturn
- WaitCallStateReturn

Error Messages

- AIVRSDown (IVR Server is down)
- Unlicensed (The specified IVR port is not licensed)
- Error (An error has occurred)
- Trace (Trace messages)

Clients to Java Interface for IVR Server

Initialise

```
<Msg>
      <MID>Initialise</MID>
      <IID>10000</IID>
      <Port>2</Port>
      <Data>
            <Script>TestApp</Script>
      </Data>
</Msg>
```

GetCallData

```
<Msg>
      <MID>GetCallData</MID>
      <IID>10001</IID>
      <Port>2</Port>
</Msg>
```

GetCustomerData

```
<Msg>
      <MID>GetCustomerData</MID>
      <IID>10002</IID>
      <Port>2</Port>
      <Data>
            <Key1>abc1</Key1>
            <Key2>abc2</Key2>
            <Key3>abc3</Key3>
```

```
            <Key4>abc4</Key4>
            <Key5>abc5</Key5>
        </Data>
    </Msg>
```

SetCustomerData

```
    <Msg>
        <MID>SetCustomerData</MID>
        <IID>10003</IID>
        <Port>2</Port>
        <Data>
            <Key1>abc1</Key1>
            <Val1>value1</Val1>
            <Key2>abc2</Key2>
            <Val2>value2</Val2>
        </Data>
    </Msg>
```

ExecuteCustomerFunction

```
    <Msg>
        <MID>ExecuteCustomerFunction</MID>
        <IID>10004</IID>
        <Port>2</Port>
        <Data>
            <FuncID>functionID</FuncID>
            <Val1>value1</Val1>
            <Val2>value2</Val2>
            <Val3>value3</Val3>
            <Val4>value2</Val4>
            <Val5>value3</Val5>
```

```
      </Data>
</Msg>
```

SetLoggingData

```
<Msg>
      <MID>SetLoggingData</MID>
      <IID>10005</IID>
      <Port>2</Port>
      <Data>
            <Key>key name</Key>
            <Val>key value</Val>
      </Data>
</Msg>
```

ClearCall

```
<Msg>
      <MID>ClearCall</MID>
      <IID>10006</IID>
      <Port>2</Port>
      <Data>
            <UUI>AIVRSJava</UUI>
      </Data>
</Msg>
```

TransferCall

```
<Msg>
      <MID>TransferCall</MID>
      <IID>10007</IID>
      <Port>2</Port>
      <Data>
```

```
            <Target>2233</Target>
            <UUI>AIVRSJava</UUI>
      </Data>
</Msg>
```

MakeCall

```
<Msg>
      <MID>MakeCall</MID>
      <IID>10008</IID>
      <Port>2</Port>
      <Data>
            <Target>2233</Target>
            <UUI>AIVRSJava</UUI>
      </Data>
</Msg>
```

HoldCall

```
<Msg>
      <MID>HoldCall</MID>
      <IID>10009</IID>
      <Port>2</Port>
      <Data>
            <Target>2233</Target>
      </Data>
</Msg>
```

RetrieveCall

```
<Msg>
      <MID>RetrieveCall</MID>
      <IID>100010</IID>
```

```
        <Port>2</Port>
        <Data>
            <Target>2233</Target>
        </Data>
    </Msg>
```

QueryACDState

```
    <Msg>
        <MID>QueryACDState</MID>
        <IID>100011</IID>
        <Port>2</Port>
        <Data>
            <Target>9988</Target>
        </Data>
    </Msg>
```

ConferenceStart

```
    <Msg>
        <MID>ConferenceStart</MID>
        <IID>100012</IID>
        <Port>2</Port>
        <Data>
            <Target>2233</Target>
            <UUI>AIVRSJava</UUI>
        </Data>
    </Msg>
```

ConferenceComplete

```
    <Msg>
        <MID>ConferenceComplete</MID>
```

```
      <IID>100013</IID>
      <Port>2</Port>
      <Data>
            <Target>2233</Target>
      </Data>
</Msg>
```

ConferenceAbort

```
<Msg>
      <MID>ConferenceAbort</MID>
      <IID>100014</IID>
      <Port>2</Port>
      <Data>
            <Target>2233</Target>
      </Data>
</Msg>
```

GetCallState

```
<Msg>
      <MID>GetCallState</MID>
      <IID>100015</IID>
      <Port>2</Port>
      <Data>
      </Data>
</Msg>
```

Please refer to ***Call States*** (on page 50).

WaitCallState

```
<Msg>
      <MID>WaitCallState</MID>
```

```
        <IID>100016</IID>

        <Port>2</Port>

        <Data>

              <CallState>2002</CallState>

              <WaitTime>2</WaitTime>

        </Data>

</Msg>
```

Note: <WaitTime>2</WaitTime> refers to a wait time of 2 seconds.

Please refer to ***Call States*** (on page 50).

Java Interface for IVR Server to Clients

GetCallDataReturn

```
<Msg>
      <MID>GetCallDataReturn</MID>
      <IID>10001</IID>
      <Port>2</Port>
      <Data>
            <Aparty>8823</Aparty>
            <Bparty>8888</Bparty>
            <VDN>9100</VDN>
            <CollectedDigits>123456</CollectedDigits>
            <UCID>9988776655</UCID>
            <UUI>Hello</UUI>
      </Data>
</Msg>
```

GetCustomerDataReturn

```
<Msg>
      <MID>GetCustomerDataReturn</MID>
      <IID>10002</IID>
      <Port>2</Port>
      <Data>
            <ReturnCode>0</ReturnCode>
            <Val1>value1</Val1>
            <Val2>value2</Val2>
            <Val3>value3</Val3>
```

```
            <Val4>value4</Val4>
            <Val5>value5</Val5>
        </Data>
    </Msg>
```

SetCustomerDataReturn

```
    <Msg>
        <MID>SetCustomerDataReturn</MID>
        <IID>10003</IID>
        <Port>2</Port>
        <Data>
            <ReturnCode>0</ReturnCode>
        </Data>
    </Msg>
```

ExecuteCustomerFunctionReturn

```
    <Msg>
        <MID>ExecuteCustomerFunctionReturn</MID>
        <IID>10004</IID>
        <Port>2</Port>
        <Data>
            <ReturnCode>0</ReturnCode>
            <Val1>value1</Val1>
            <Val2>value2</Val2>
            <Val3>value3</Val3>
            <Val4>value4</Val4>
        </Data>
    </Msg>
```

SetLoggingDataReturn

```
<Msg>
      <MID>SetLoggingDataReturn</MID>
      <IID>10005</IID>
      <Port>2</Port>
      <Data>
            <ReturnCode>0</ReturnCode>
      </Data>
</Msg>
```

CallCleared

```
The call is cleared by the client.
<Msg>
      <MID>CallCleared</MID>
      <IID>10006</IID>
      <Port>2</Port>
</Msg>
The call is released by the others. IID is 0.
<Msg>
      <MID>CallCleared</MID>
      <IID>0</IID>
      <Port>2</Port>
</Msg>
```

TransferReturn

```
<Msg>
      <MID>TransferReturn</MID>
      <IID>10007</IID>
      <Port>2</Port>
```

```
        <Data>
            <ReturnCode>0</ReturnCode>
        </Data>
    </Msg>
```

MakeCallReturn

```
    <Msg>
        <MID>MakeCallReturn</MID>
        <IID>10008</IID>
        <Port>2</Port>
        <Data>
            <ReturnCode>0</ReturnCode>
        </Data>
    </Msg>
```

HoldCallReturn

```
    <Msg>
        <MID>HoldCallReturn</MID>
        <IID>10009</IID>
        <Port>2</Port>
        <Data>
            <ReturnCode>0</ReturnCode>
            <Party>2233</Party>
        </Data>
    </Msg>
```

RetrieveCallReturn

```
    <Msg>
        <MID>RetrieveCallReturn</MID>
        <IID>100010</IID>
```

```
        <Port>2</Port>
        <Data>
            <ReturnCode>0</ReturnCode>
            <Party>2233</Party>
        </Data>
    </Msg>
```

CallRetrieved

```
    <Msg>
        <MID>CallRetrieved</MID>
        <IID>0</IID>
        <Port>2</Port>
    </Msg>
```

QueryACDStateReturn

```
    <Msg>
        <MID>QueryACDStateReturn</MID>
        <IID>100011</IID>
        <Port>2</Port>
        <Data>
            <ReturnCode>0</ReturnCode>
            <ACD>9988</ACD>
            <AvailableAgents>3</ AvailableAgents>
            <StaffedAgents>5</StaffedAgents>
            <CallsInQueue>2</CallsInQueue>
        </Data>
    </Msg>
```

ConferenceStartReturn

```
    <Msg>
```

```
<MID>ConferenceStartReturn</MID>
<IID>100012</IID>
<Port>2</Port>
<Data>
<ReturnCode>0</ReturnCode>
<Party>2233</Party>
</Data>
</Msg>
```

ConferenceCompleteReturn

```
<Msg>
<MID>ConferenceCompleteReturn</MID>
<IID>100013</IID>
<Port>2</Port>
<Data>
<ReturnCode>0</ReturnCode>
<Party>2233</Party>
</Data>
</Msg>
```

ConferenceAbortReturn

```
<Msg>
<MID>ConferenceAbortReturn</MID>
<IID>100014</IID>
<Port>2</Port>
<Data>
<ReturnCode>0</ReturnCode>
<Party>2233</Party>
</Data>
</Msg>
```

Note: A non-zero return code (<ReturnCode>) indicates that an error has occured. In this case, the message from the Java Interface for IVR Server to clients will contain a tag of <Text> to describe the error very briefly. For example:

```
<Msg>
	<MID>TransferReturn</MID>
	<IID>10007</IID>
	<Port>2</Port>
	<Data>
		<ReturnCode>2</ReturnCode>
		<Text>DestinationIsBusy</Text>
	</Data>
</Msg>
```

GetCallStateReturn

```
<Msg>
	<MID>GetCallStateReturn</MID>
	<IID>100015</IID>
	<Port>2</Port>
	<Data>
		<ReturnCode>0</ReturnCode>
		<CallState>2002</CallState>
	</Data>
</Msg>
```

Please refer to ***Call States*** (on page 50).

WaitCallStateReturn

```
<Msg>
	<MID>WaitCallStateReturn</MID>
	<IID>100016</IID>
	<Port>2</Port>
	<Data>
```

```
            <ReturnCode>0</ReturnCode>
            <CallState>2002</CallState>
        </Data>
</Msg>
```

Please refer to ***Call States*** (on page 50).

Call States

The following IVR port states are supported by the IVR Server.

IVR_PORT_OOS = 2000	The port is out of service.
IVR_PORT_IDLE = 2001	There is no call on the port.
IVR_PORT_ACTIVE = 2002	The call on the port is active (connected).
IVR_PORT_IDLE_PENDING = 2003	The call on the port is idle pending.
IVR_PORT_HOLD_PENDING = 2004	The call on the port is held pending.
IVR_PORT_HELD = 2005	The call on the port is held.
IVR_PORT_TRANSFERRED_PENDING = 2006	The call on the port has been transferred pending.
IVR_PORT_TRANSFERRED = 2007	The call on the port has been transferred.
IVR_PORT_CONFERENCED_PENDING = 2008	The call on the port has been conferenced pending.
IVR_PORT_CONFERENCED = 2009	The call on the port has been conferenced.
IVR_PORT_UNKNOWN = 2010	The state of the call on the port is unknown.

Error Messages

This chapter outlines the error and failure messages from the Java Interface for IVR Server to its clients.

There are four types of errors that can occur:

- AIVRSDown (indicates the IVR Server is down)
- Unlicensed (indicates the specified IVR port is not licensed)
- Error (indicates an error has occurred)
- Trace (indicates a trace message)

AIVRSDown

```
<Msg>
      <MID>AIVRSDown</MID>
      <IID>10007</IID>
      <Port>2</Port>
      <Data>
            <Text>AIVRServer is not connected</Text>
      </Data>
</Msg>
```

Unlicensed

```
<Msg>
      <MID>Unlicensed</MID>
      <IID>10007</IID>
      <Port>2</Port>
      <Data>
            <Text>The IVR port is not licensed</Text>
      </Data>
</Msg>
```

Error

```
<Msg>
      <MID>Error</MID>
      <IID>10007</IID>
      <Port>2</Port>
      <Data>
            <Text>DeviceIsInvalid</Text>
      </Data>
</Msg>
```

Trace

```
<Msg>
      <MID>Trace</MID>
      <IID>10007</IID>
      <Port>2</Port>
      <Data>
            <Text>AIVRServer is working</Text>
      </Data>
</Msg>
```

Developer's Sample Code

The following Java code is a sample from a client application that communicates with the Java Interface for IVR Server. It illustrates how you can send a message to the Java Interface for IVR Server and the structure of information when it is returned.

The protocol between the Java Interface for IVR Server and its clients consists of two parts: the length of the XML-formatted message and the XML-formatted message body.

The client sending a message to the Java Interface for IVR Server:

```
// xmlMsg is a string object containing the XML-formatted
// message to be sent out.
int len = xmlMsg.length();

// First, the length of the XML-formatted message
// (out is an instance of DataOutputStream).
out.writeInt(len);

// Then, the message body.
out.writeBytes(xmlMsg);

out.flush();
```

The client receiving a message from the Java Interface for IVR Server.

```
// Read the length of the message (in is an instance
// of DataInputStream).
int len = in.readInt();

byte[] data = new byte[len];

// Read the message body.
in.readFully(data);
```

Index

U

W

X