Installer Guide

DIP for Avaya IVR

Contact Center Express

Release 2.0 - Issue 0

Notice

Every effort was made to ensure that the information in this manual was complete and accurate at the time of printing. However, information is subject to change.

Your Responsibility for Your System's Security

Toll fraud is the unauthorized use of your telecommunications system by an unauthorized party, for example, persons other than your company's employees, agents, subcontractors, or persons working on your company's behalf. Note that there may be a risk of toll fraud associated with your telecommunications system and, if toll fraud occurs, it can result in substantial additional charges for your telecommunications services.

You and your system manager are responsible for the security of your system, such as programming and configuring your equipment to prevent unauthorized use. The system manager is also responsible for reading all installation, instruction, and system administration documents provided with this product in order to fully understand the features that can introduce risk of toll fraud and the steps that can be taken to reduce that risk. Avaya Inc. does not warrant that this product is immune from or will prevent unauthorized use of common-carrier telecommunication services or facilities accessed through or connected to it. Avaya Inc. will not be responsible for any charges that result from such unauthorized use.

Avaya Fraud Intervention

If you suspect that you are being victimized by toll fraud and you need technical support or assistance, call the Technical Service Center Toll Fraud Intervention Hot-line at 1-800-643-2353.

Trademarks

- Avaya Computer Telephony is a registered trademark of Avaya Inc.
- Avaya CallMaster is a registered trademark of Avaya Inc.
- Definity is a registered trademark of Avaya Inc.
- MultiVantage is a registered trademark of Avaya Inc.
- INTEL and Pentium are registered trademarks of Intel Corporation.
- Microsoft, MS, MS-DOS, and Windows are registered trademarks of Microsoft Corp.

All other product names mentioned herein are the trademarks of their respective owners.

Avaya National Customer Care Center

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. For technical support, customers outside the United States should call their Avaya representative or distributor.

European Union Declaration of Conformity

Avaya Inc. Business Communications Systems declares that the equipment specified in this document conforms to the referenced European Union (EU) Directives and Harmonized Standards listed below:

- EMC Directive 89/336/EEC
- Low Voltage Directive 73/23/EEC

The CE” mark affixed to the equipment means that it conforms to the above Directives.

Website

For more information on all Avaya Contact Center Express products, refer to the company ***website*** (http://www.AvayaContactCenterExpress.com).

Software License Agreement

Definitions

Term	Definition
Avaya	Avaya Inc.
You, your or licensee	The person or business entity who purchased this license to use this client software or for whom such license was purchased.
Client software	A software application that operates on a computer system.
Documentation	The manual and any other printed material provided by Avaya for the client software.
License	The license purchased and granted pursuant to this agreement.

License and Protection

License Grant. Avaya grants to you, subject to the following terms and conditions, a nonexclusive, nontransferable right to use the client software on one or more single-user devices. The maximum simultaneous users of the client software being limited to the number of single-user licenses purchased and owned by you. Avaya reserves all rights not expressly granted to you.

Protection of Software. You agree to take all reasonable steps to protect the client software and documentation from unauthorized copy or use. The client software source code represents and embodies trade secrets of Avaya and/or its licensors. The source code and embodied trade secrets are not licensed to you and any modification, addition, or deletion is strictly prohibited. You agree not to disassemble, decompile, or otherwise reverse engineer the client software in order to discover the source code and/or the trade secrets contained in the source code or for any other reason. To the extent that the client software is located in a Member State of the European Community and you need information about the client software in order to achieve interoperability of an independently created software program with the client software, you shall first request such information from Avaya. Unless Avaya refuses to make such information available, you shall not take any steps, such as reverse assembly or reverse compilation, to derive a source code equivalent to the client software. Avaya may charge you a reasonable fee for the provision of such information.

Copies. You may make multiple copies of the client software for your own use with Avaya contact center agent digital voice terminals, provided you do not violate the License Grant in paragraph 1, and you do not receive any payment, commercial benefit, or other consideration for reproduction or use. You may not copy documentation unless it carries a statement that copying is permitted. All proprietary rights notices must be faithfully reproduced and included on all copies.

Ownership. Ownership of, and title to, the client software and documentation (including any adaptations or copies) remains with Avaya and/or its licensors.

Restrictions. You agree not to rent, lease, sublicense, modify or time share the client software or documentation.

Termination. This agreement shall automatically terminate if you breach any of the terms or conditions of this agreement. You agree to destroy the original and all copies of the client software and documentation, or to return them to Avaya, upon termination of this license.

Limited Warranty and Limited Liability

Compatibility. The client software is only compatible with certain computers and operating systems. The software is not warranted for noncompatible systems.

Software. Avaya warrants that if the client software fails to substantially conform to the specifications in the documentation and if the client software is returned to the place from which it was purchased within one (1) year from the date purchased, then Avaya will either replace the client software or offer to refund the license fee to you upon return of all copies of the client software and documentation to Avaya. In the event of a refund, the license shall terminate.

Disclaimer of Warranties. Avaya makes no warranty, representation or promise not expressly set forth in this agreement. Avaya disclaims and excludes any and all implied warranties of merchantability or fitness for a particular purpose. Avaya does not warrant that the client software or documentation will satisfy your requirements or that the client software or documentation are without defect or error or that the operation of the software will be uninterrupted. Some states or countries do not allow the exclusion of implied warranties or limitations on how long an implied warranty lasts, so the above limitation may not apply to you. This warranty gives you specific legal rights which vary from state to state.

Exclusive Remedy. Except for bodily injury caused by Avaya's negligence, Avaya's entire liability arising from or relating to this agreement or the client software or documentation and your exclusive remedy is limited to direct damages in an amount not to exceed $10,000. Avaya shall not in any case be liable for any special incidental, consequential, indirect or punitive damages even if Avaya has been advised of the possibility of such damages. Avaya is not responsible for lost profits or revenue, loss of use of the client software, loss of data, costs of recreating lost data, the cost of any substitute equipment or program, or claims by any party other than you. Some states or countries do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

General Conditions

Governing Law. This agreement shall be governed by, and interpreted in accordance with, the substantive laws of the State of New Jersey of the United States of America.

Entire Agreement. This agreement sets forth the entire understanding and agreement between you and Avaya and may be amended only in a writing or writings signed by you and Avaya. No vendor, distributor, dealer, retailer, sales person or other person is authorized to modify this agreement or to make any warranty, representation or promise which is different than, or in addition to, the representations or promises of this agreement about the software.

Export. Licensee hereby agrees that it will not knowingly, directly or indirectly, without prior written consent, if required, of the Office of Export Licensing of the U.S. Department of Commerce, Washington D.C. 20230, export or transmit any of the Products to any group Q, S, W, Y, or Z country specified in the Export Administration Regulations issued by the U.S. Department of Commerce or to any country which such transmission is restricted by applicable regulations or statues.

U.S. Government Restricted Rights. Use, duplication, or disclosure by the United States Government is subject to restrictions as set forth in FAR 52.227-14 (June 1987) Alternate III (g)(3) (June 1987), FAR 52.227-19 (June 1987), or DFARS 52.227-7013 (c)(1)(ii) (June 1988), as applicable Contractor/Manufacturer is Avaya Inc. 11900 North Pecos Street, Westminster, Colorado 80234.

Assignment. Avaya may without your consent or notice to you, assign this agreement to an entity to which it transfers ownership of the client software. Upon the effective date of such assignment, you agree that Avaya shall be released and discharged from all obligations and liabilities under this agreement.

Contents

CHAPTER 1

Preface

Chapter Overview

This chapter covers the conventions used in this document.

In This Chapter

Conventions Used in this Document

Convention	Description
Initial Capital Letters	Names of windows, dialog boxes, text boxes and drop-down list boxes. For example, the Add VDN dialog box appears.
[key] or [button]	The name of a button or keyboard key. For example, click the [Enter] button or press the [F5] key.
`Courier text`	Unix operating system commands

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).

The Avaya Interactive Voice Response Designer (Avaya IVR Designer) application was previously known as Voice@Work.

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 role of the IVR Server DIP interface.

In This Chapter

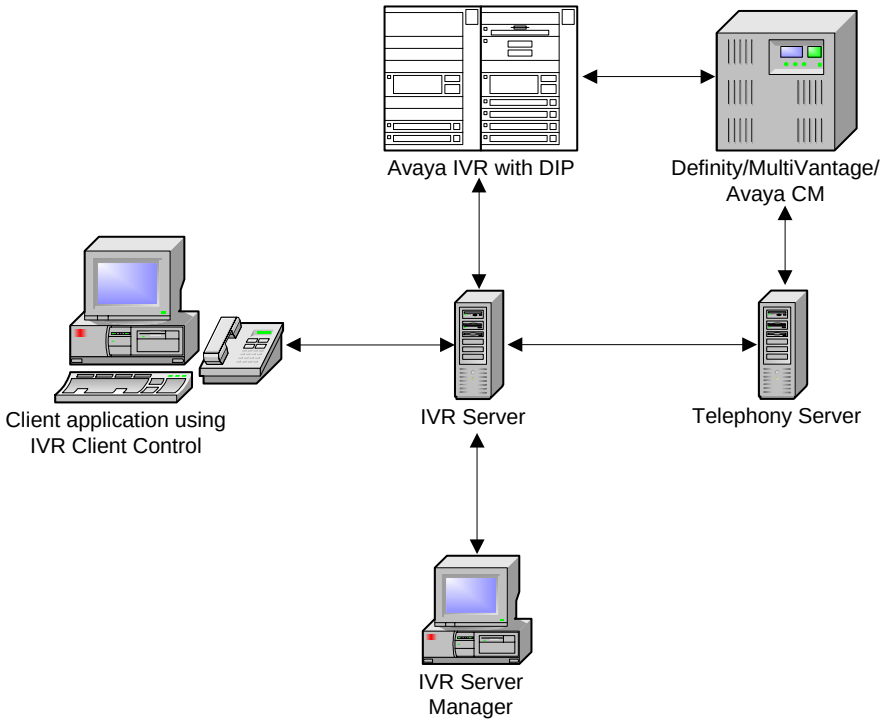
What is the IVR Server DIP?

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



CHAPTER 3

Data Interface Process (DIP) Installation

This chapter explains how to install and remove IVR Server DIP components on the Conversant Version 7, Avaya IVR Version 8, and Avaya IR 1.2.

In This Chapter

Pre-Installation Requirement

The IVR Server Data Interface Process (DIP) communicates with the IVR Server over the local network using the TCP/IP protocol. For the DIP to work correctly, the Avaya IVR must have a working network card and the TCP/IP protocol installed and working.

Install AIVRS DIP on Conversant Version 7

Note: These steps should be completed using the root (or equivalent level) access rights.

Check Previous Version

If this install is an upgrade from a previous release of the IVR Server DIP, you must remove the previously installed version first.

1 To check if the IVR Server DIP has already been installed, execute the command:

```
pkginfo -x aivrsV7
```

If there is a previous version, the following text returns:

```
aivrsV7  Active IVR Server for Conversant V7
(i386) x.x.x
```

2 If the above text displays, you need to ***remove the previous version*** (see "Remove AIVRS DIP from Conversant Version 7" on page 15).

Expand Tar File

You must manually transfer installation files from the Contact Center Express CD to the Avaya IVR using ftp (or similar) protocol. A tar file, aivrsv7.tar, is located in the Contact Center Express\Server\IVR Server\AvayaIVR directory.

To transfer and expand the tar file:

1 Telnet to the Avaya IVR, log in and create a temporary directory in your default home directory, for example, mkdir temp.

2 Open a separate ftp session and transfer the tar file into the newly created temp directory on the Avaya IVR. Note: Remember to set the session for binary transfer.

3 Close the ftp session.

4 To expand the tar file in the telnet session, execute the command:

```
tar -xvf aivrsv7.tar
```

Install

To install the DIP:

1 From the expanded tar file location, execute the following command:

```
pkgadd -d {full path to location of expanded tar file}
```

The following text appears:

```
The following packages are available:
1  aivrsV7     Active IVR Server for Conversant V7
(i386) 1.0.1
```

```
Select package(s) you wish to process (or 'all' to
process all packages). (default: all) [?,??,quit]:
```

2 Press [Enter] to accept the default and continue the installation.

3 If the Avaya IVR Voice System is running, the following text appears:

```
The voice system is currently running. The voice system
must be stopped before the install can continue. Do you
wish to stop the voice system now? [y,n]
```

Enter [y]. Note: If you do not to stop the voice system, the installation will abort.

The following text appears:

```
Once the install is complete, do you wish to restart
the voice system? [y,n]
```

Enter 'y'.

4 Once the install is complete, the text from Step 1 re-appears. Type quit to end the installation.

5 If the voice system is running and the Avaya IVR system is at run level 4, the IVR Server DIP should automatically load and be available for connection from the IVR Server. To check the IVR Server DIP is running, execute the command:

```
ps -ef | grep -i "aivrs"
```

If the DIP is running, output returns.

6 If the DIP is not running (no output returns), you need to determine the ***cause*** (see "Troubleshooting" on page 22).

Remove AIVRS DIP from Conversant Version 7

Note: These steps should be completed using the root (or equivalent level) access rights.

1 Execute the command:

```
pkgrm aivrsV7
```

The following text appears:

```
The following package is currently installed:
aivrsV7  Active IVR Server for Conversant V7
(i386) 1.0.7

Do you want to remove this package [yes,no,?,quit]
```

2 Enter 'y' to start the removal process.

3 If the voice system is running, the following text appears:

```
The voice system is currently running. The voice system
must be stopped before the remove can continue. Do you
wish to stop the voice system now? [y,n]
```

Enter 'y' to stop the voice system and continue the removal process.

The following text appears:

```
After the removal is completed, would you like to
restart the voice system? [y,n]
```

Enter 'y'.

Install AIVRS DIP on Avaya IVR Version 8

Note: These steps should be completed using the root (or equivalent level) access rights.

Check Previous Version

If this install is an upgrade from a previous release of the IVR Server DIP, you must remove the previously installed version first.

1 To check if the IVR Server DIP has already been installed, execute the command:

```
pkginfo -x aivrsV8
```

If there is a previous version, the following text returns:

```
aivrsV8  Active IVR Server for Conversant V8
(i386) 1.0.5
```

2 IIf the above text displays, you need to ***remove the previous version*** (see "Remove AIVRS DIP from Avaya IVR Version 8" on page 18).

Expand Tar File

You must manually transfer installation files from the Contact Center Express CD to the Avaya IVR using ftp (or similar) protocol. A tar file, aivrsv8.tar, is located in the Contact Center Express\Server\IVR Server\AvayaIVR directory.

To transfer and expand the tar file:

1 Telnet to the Avaya IVR, log in and create a temporary directory in your default home directory, for example, mkdir temp.

2 Open a separate ftp session and transfer the tar file into the newly created temp directory on the Avaya IVR. Note: Remember to set the session for binary transfer.

3 Close the ftp session.

4 To expand the tar file in the telnet session, execute the command:

```
tar -xvf aivrsv8.tar
```

Install

To install the DIP:

1 From the expanded tar file location, execute the following command:

```
pkgadd -d {full path to the location of expanded tar
file}
```

The following text appears:

```
The following packages are available:
1  aivrsV8     Active IVR Server for Conversant V8
(i386) 1.0.5
```

```
Select package(s) you wish to process (or 'all' to
process all packages). (default: all) [?,??,quit]:
```

2 Press [Enter] to accept the default and continue the installation.

3 If the Avaya IVR Voice System is running, the following text appears:

```
The voice system is currently running. The voice system
must be stopped before the install can continue. Do you
wish to stop the voice system now? [y,n]
```

Enter [y]. Note: If you do not to stop the voice system, the installation will abort.

The following text appears:

```
Once the install is complete, do you wish to restart
the voice system? [y,n]
```

Enter 'y'.

4 Once the installation is complete, the text from step 1 re-appears. Type quit to end the installation.

5 If the voice system is running and the Avaya IVR system is at run level 4, the IVR Server DIP should automatically load and be available for connection from the IVR Server. To check the IVR Server is running, execute the command:

```
ps -ef | grep -i "aivrs"
```

If the DIP is running, output returns.

6 If the DIP is not running (no output returns), you need to determine the ***cause*** (see "Troubleshooting" on page 22).

Remove AIVRS DIP from Avaya IVR Version 8

Note: These steps should be completed using the root (or equivalent level) access rights.

1. Execute the command:

   ```
   pkgrm aivrsV8
   ```

   The following text appears:

   ```
   The following package is currently installed:
   aivrsV8  Active IVR Server for Conversant V8
   (i386) 1.0.7

   Do you want to remove this package [yes,no,?,quit]
   ```

2. Enter 'y' to start the removal process.

3. If the voice system is running, the following text appears:

   ```
   The voice system is currently running. The voice system
   must be stopped before the remove can continue. Do you
   wish to stop the voice system now? [y,n]
   ```

   Enter 'y' to stop the voice system and continue the removal process.

   The following text appears:

   ```
   After the removal is completed, would you like to
   restart the voice system? [y,n]
   ```

   Enter 'y'.

Install AIVRS DIP on Avaya IR 1.2

Note: These steps should be completed using the root (or equivalent level) access rights.

Check Previous Version

If this install is an upgrade from a previous release of the IVR Server DIP, you must remove the previously installed version first.

1 To check if the IVR Server DIP has already been installed, execute the command:

```
pkginfo -x aivrsair
```

If there is a previous version, the following text returns:

```
aivrsair Avaya CCE DIP for Avaya IR
(i386) 1.0.5
```

2 If the above text displays, you need to ***remove the previous version*** (see "Remove AIVRS DIP from Avaya IR 1.2" on page 21).

Expand Tar File

You must manually transfer installation files from the Contact Center Express CD to the Avaya IVR using ftp (or similar) protocol. A tar file, aivrsair.tar, is located in the Contact Center Express\Server\IVR Server\AvayaIVR directory.

To transfer and expand the tar file:

1 Telnet to the Avaya IVR, log in and create a temporary directory in your default home directory, for example, mkdir temp.

2 Open a separate ftp session and transfer the tar file into the newly created temp directory on the Avaya IVR. Note: Remember to set the session for binary transfer.

3 Close the ftp session.

4 To expand the tar file in the telnet session, execute the command:

```
tar -xvf aivrsair.tar
```

Install

To install the DIP:

1 From the expanded tar file location, execute the following command:

```
{file path to pkgadd}/pkgadd -d {full path of the
directory which contains the expanded tar file}
```

The following text appears:

```
The following packages are available:
1  aivrsair     Avaya CCE DIP for Avaya IR
                (sparc.sun4u) 1.4.2

Select package(s) you wish to process (or 'all' to
process all packages). (default: all) [?,??,q]:
```

2. Press [Enter] to accept the default and continue the installation.
3. If the Avaya IVR Voice System is running, the following text appears:

   ```
   The voice system is currently running and must be
   stopped in order to install this package. Is it OK to
   stop the voice system? [y,n]
   ```

   Enter [y]. Note: If you do not to stop the voice system, the installation will abort.

   The following text appears:

   ```
   Once the install change is completed, would you like
   to restart the voice system? [y,n]
   ```

   Enter 'y'.

   ```
   Do you want to continue with the installation of
   <aivrsair>? [y,n]
   ```

   Enter 'y'.
4. Once the installation is complete, the text from step 1 re-appears. Type quit to end the installation.
5. If the voice system is running and the Avaya IVR system is at run level 4, the IVR Server DIP should automatically load and be available for connection from the IVR Server. To check the IVR Server is running, execute the command:

   ```
   ps -ef | grep -i "aivrs"
   ```

   If the DIP is running, output returns.
6. If the DIP is not running (no output returns), you need to determine the ***cause*** (see "Troubleshooting" on page 22).

Remove AIVRS DIP from Avaya IR 1.2

Note: These steps should be completed using the root (or equivalent level) access rights.

1. Execute the command:

   ```
   /vs/bin/stop_vs
   ```

2. Execute the command:

   ```
   pkgrm aivrsair
   ```

   The following text appears:

   ```
   The following package is currently installed:
   aivrsair Avaya CCE DIP for Avaya IR
   (sparc.sun4u) 1.4.2

   Do you want to remove this package?
   ```

3. Enter 'y' to start the removal process.

4. The following text appears:

   ```
   This package contains scripts which will be executed
   with super-user permission during the process of
   removing this package.

   Do you want to continue with the removal of this
   package? [y,n,?,q]
   ```

   Enter 'y' to continue the removal process.

   The following text appears:

   ```
   Removal of (aivrsair) was successful.
   ```

Troubleshooting

The IVR Server DIP is installed but the IVR Server does not connect

- Check whether the IVR Server DIP is running by executing the command:

  ```
  ps -ef | grep -i "aivrs"
  ```

  This command returns no output if the IVR Server DIP is not running.
- Check the error log for errors on binding to the specified port. Ensure the TCP ports used for the server and DIP match.

The IVR Server DIP is not running

- Check the Avaya IVR system is at run level 4. The DIP is only designed to start with the voice system. This can be achieved by executing the command:

  ```
  who -r
  ```

  This command outputs the following if the Avaya IVR system is at run level 4:

  ```
  run-level 4  Apr 19 14:51    4    5    3
  ```

  If the system is not at run level 4, execute the command:

  ```
  start_vs
  ```
- Check the IVR Server DIP executable file is available and has mode settings that allow it to execute. Executable files are:
  - V7 /vs/data/aivrs/aivrs_dipv7
  - V8 /vs/data/aivrs/aivrs_dipv8

  Execute the command:

  ```
  ls -al /vs/data/aivrs*
  ```

  This command should output the following mode settings:

  ```
  -r-xr-xr-x    1 root  sys     110164 Apr 19 05:42
  aivrs_dipv7
  ```

  If these mode settings are not present, execute the command:

  ```
  chmod 755 /vs/data/aivrs/aivrs*
  ```
- Check the inittab file has been correctly updated in the /etc directory. You can check this by executing the command:

  ```
  cat /etc/inittab | grep -i "aivrs"
  ```

  If the file has the correct entry, this command outputs the following:

  ```
  AIVR:4:respawn:/vs/data/aivrs/aivrs_dipv7
  >/dev/null 2>&1
  ```

  If this output is not present, check the directory /etc/conf/init.d contains the file AIVRSdip. This file contains the entry (Version 7 example):

  ```
  #
  # Startup Active IVR server DIP For Version 7
  Conversant
  ```

#

```
AIVR:4:respawn:/vs/data/aivrs/aivrs_dipv7
>/dev/null 2>&1
```

If the file is present, reboot the Avaya IVR. If the file is not present, it should be added.

Change Default AIVRS DIP Configuration

Note: Items discussed in this section assume a sound knowledge of both the Avaya IVR system and Unix operating system on which it runs. Changing system parameters and system files can have severe consequences to the operation and reliability of the Avaya IVR platform.

Startup

The IVR Server DIP is started automatically by the Unix system when the Avaya IVR reaches run level 4. This indicates the successful startup of the voice system. Startup is achieved by an entry in the inittab file in the /etc directory. This file is rebuilt when the system is restarted from entries found in the /etc/conf/init.d directory. The IVR Server DIP installation adds a file to this directory named AIVRSdip. This file contains the entry (Version 7 example):

```
#

# Startup Active IVR server DIP For Version 7 Conversant

#

AIVR:4:respawn:/vs/data/aivrs/aivrs_dipv7 >/dev/null
2>&1
```

This file allows the DIP to start once the system comes to run level 4 and will restart should it fail. The default configuration has no additional command line parameters.

TCP Port

The IVR Server DIP communicates with the IVR Server via a TCP/IP connection on the local network. The DIP "listens" for connections from the server on a specified port. By default this port number is assigned to 29095.

You can change this value by adding a command line switch. By executing the command aivrs_dipv7 -port 1024, you instruct the IVR Server DIP to use port 1024 to listen for incoming connections from the IVR Server. If this change is required permanently, you should change the AIVRSdip file in the /etc/conf/init.d directory. Once the change is made, you need to rebuild the inittab file and re-initialize using init -q or restart the system.

Note: The TCP/IP port the IVR Server DIP listens on for incoming connections must match the value configured in the IVR Server.

Voice System DIP name

When the IVR Server DIP process starts, it registers itself with the Avaya IVR voice system. It does this by specifying a DIP name. By default, the name is "aivrsdip".

You can change this value by adding a command line switch. By executing the command aivrs_dipv7 -dipname newdipname, you can instruct the IVR Server DIP to register the DIP name "newdipname" with the voice system.

Note: The script functions used in Avaya IVR Designer and ScriptBuilder interact with the IVR Server DIP via the Transaction State Machine (TSM) using the DIP name. If you change the DIP name used in the DIP start up, you must also change all script functions.

Error Logging

IVR Server DIPs write error information relating to their operation to a series of log files.

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

Log files are stored in the same directory folder as the DIP component: /vs/data/aivrs.

CHAPTER 4

IVR to IVR Server

The IVR Server communicates with the IVR using TCP/IP.

This chapter covers the messages sent from the IVR script to the IVR Server.

In This Chapter

Initialise

Syntax: *Initialise(ScriptName)*

Description: A message sent from the script to the IVR Server to indicate that the script has started. This is the first request which should be sent by the script. This function has to be executed once at the start of the script before any other Active Telephony functions.

Note: The “aivrsBuffer” variable must also be declared in the beginning of the script.

Parameters

ScriptName	The name of the script running on the specified port. This script name will be passed on client applications with events.

Return Code

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
     #Initialise variables

1.   Set Field Value

           Field: aivrsBuffer = ""

           Field: ScriptName = "aivrsTest"


     #Initialise the dip and its connectivity to AIVRS

     #FUNCTION: aiInit - initialise DIP connectivity

     #INPUT Parameter:

     #Parameter 1 - Script Name

     #RETURN CODE:

     #Return code zero (0) means successful

     #Minus return code means error occurred

5.   External Function

        Function Name: aiInit

        Use Arguments: ScriptName
```

```
        Return Field:  ExtFuncReturn


6.   Evaluate
     If ExtFuncReturn   != 0
        #Log aiInit Error
     End Evaluate
```

MakeCall

Syntax: *MakeCall(Destination, UUI)*

Description: A request from the IVR to IVR Server to make a call to the specified destination. The specified UUI will be passed with the call.

Parameters

Destination	The destination of the call.
UUI	Any user-to-user information sent with the call.

Return Code

If the call fails (ie. the IVR port is not off hook), the failure code will be placed into the *ReturnCode* field.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
   Set Field Value
           Field: DestinationNo = "8888"
           Field: UUIData = "UUI Test"


     #FUNCTION: aiInit - initialise DIP connectivity
     #INPUT Parameter:
     #Parameter 1 - Script Name
     #RETURN CODE:
     #Return code zero (0) means successful
     #Minus return code means error occurred
```

```
External Function
     Function Name: aiMakeCall
     Use Arguments: DestinationNo  UUIData
     Return Field:  ExtFuncReturn

Evaluate
  If ExtFuncReturn   != 0
    #Log aiMakeCall Error
  End Evaluate
```

CallDelivered

Syntax:	*CallDelivered(AParty, BParty, VDN, CollectedDigits, UUI)*
Description:	A message from the IVR to the IVR Server to indicate that a new script has been launched on a particular port. This allows the IVR Server to be used without a connection to the Avaya CT Server. The message allows up to five pieces of script data to be sent with the message.

Parameters

AParty	The party that instigated the call.
BParty	The party to receive the call.
VDN	The VDN delivery the call.
CollectedDigits	Any information collected via the Definity/MultiVantage/Avaya CM server vector prompting feature.
UUI	Any user-to-user information sent with the call.

Return Code

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
#Use of external function - aiDelivered
#It returns the following parameters -
#OUTPUT Parametes:
#Parameter 1 - AParty number
#Parameter 2 - BParty number
```

```
      #Parameter 3 - Incoming call VDN number
      #Parameter 4 - Collected Digits
      #Parameter 5 - UUI Data
      #RETURN CODE:
      #Return code zero (0) means successful
      #Minus return code means error occurred
      External Function
         Function Name: aiDelivered
         Use Arguments: MyAParty  MyBParty  MYVDN
MyCollectedDigits  MyUUI
         Return Field:  ExtFuncReturn


   Evaluate
      If ExtFuncReturn   != 0
        #Log aiDelivered Error
      End Evaluate
```

ClearCall

Syntax: *ClearCall(UUI)*

Description: A request from the IVR script to the IVR Server to use the Avaya CT Server connection for releasing the call. User-to-user information can, optionally, be sent with the call release. If the call is a conference call with three or more parties, only the IVR port will be dropped from the call.

Parameters

UUI	Any user-to-user information sent with the call.

Return Code

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
      #Use of external function – aiClear
      #INPUT Parameter:
      #Parameter 1 – UUI Data
      #RETURN CODE:
      #Return code zero (0) means successful
      #Minus return code means error occurred
```

```
External Function
      Function Name: aiClear
      Use Arguments: UUIData
      Return Field:  ExtFuncReturn

Evaluate
   If ExtFuncReturn   != 0
     #Log aiClear Error
   End Evaluate
```

RetrvCall

Syntax: *RetrvCall*

Description: A request from the IVR to the IVR Server to retrieve a call that has been placed on hold at the current IVR port.

Note: The message is not passed to any client applications.

Return Code

A message from the IVR Server to the IVR to indicate that the request to retrieve then call at the current port has been processed.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
     #Use of external function – aiRetrvCall
     #RETURN CODE:
     #Return code zero (0) means successful
     #Minus return code means error occurred
  External Function
      Function Name: aiRetrvCall
      Return Field:  FuncRtnCode

Evaluate
   If ExtFuncReturn   != 0
     #Log aiRetrvCall Error
   End Evaluate
```

HoldCall

Syntax: *HoldCall*

Description: A request from the IVR to the IVR Server to place the current call on hold.

Return Code

A message sent from the IVR Server to the IVR to indicate that the request to hold the call at the current port has been processed.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
     #Use of external function – aiHoldCall
     #RETURN CODE:
     #Return code zero (0) means successful
     #Minus return code means error occurred
   External Function
        Function Name: aiHoldCall
        Return Field:  FuncRtnCode

  Evaluate
     If ExtFuncReturn   != 0
       #Log aiHoldCall Error
     End Evaluate
```

TransferCall

Syntax: *TransferCall(Destination, UUI)*

Description: A request from the IVR script to the IVR Server to use the Avaya CT Server connection for blind transfer of the call to a specified destination.

A request from the IVR script to the IVR Server to use the Avaya CT Server connection for transferring the call to a specified destination. The transfer is performed in a blind manner.

Parameters

Destination	The destination to receive the current call.
UUI	Any user-to-user information sent with the call.

Return Code

The server has indicated that the transfer has been attempted.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
#Use of external function – aiTransfer
#This external function can be used to get AIVRS to
#transfer call and pass UUI data.
#INPUT:
#Parameter 1 - Transfer Destination
#Parameter 2 - UUI Data
#5 input parameters, length between 0 and 64 chars
#RETURN CODE:
#Return code zero (0) means successful
#Minus return code means error occurred
External Function
   Function Name: aiTransfer
   Use Arguments: XferDestination  UUIData
   Return Field:  ExtFuncReturn

Evaluate
If ExtFuncReturn   != 0
  #Log aiTransfer Error
End Evaluate
```

GetCallState

Syntax: *GetCallState ()*

Description: A request from the IVR to the IVR Server to return the call state of the current call.

Parameters

CallState Sent by the IVR Server to return to the script the current call state.

Please refer to ***Call States*** (on page 52).

Return Code

Return code from the IVR Server in response to the *GetCallState* function.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
    #Use of external function – aiGetCallSt
     #This external function can be used to find the
status of current call
     #OUTPUT:
     #Parameter 1 – Current call state
     #RETURN CODE:
     #Return code zero (0) means successful
     #Minus return code means error occurred
     External Function
        Function Name: aiGetCallSt
        Use Arguments: CallState

     Evaluate
     If ExtFuncReturn   != 0
       #Log aiGetCallSt Error
     End Evaluate
```

WaitCallState

Syntax: *WaitCallState (CallState, WaitTime)*

Description: A request from the IVR to the IVR Server to not return until the call reaches the prescribed state or the timeout occurs. The message is not passed to any clients.

Parameters

CallState Please refer to ***Call States*** (on page 52).

WaitTime	The maximum length of time, in seconds, the IVR Server waits before returning the real call state.

Return Code

Return code from the IVR Server in response to the *WaitCallState* function.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
   Set Field Value

            Field: IVR_PORT_HOLD_PENDING = “2004”

            Field: IVR_WAIT_TIME= “5”


     #Use of external function - aiWaitCallSt

     #This external function can be used to make the IVR
wait for a particular call state.

     #INPUT:

     #Parameter 1 – Desired call state

     #Parameter 2 – Desired Wait Time

     #RETURN CODE:

     #Return code zero (0) means successful

     #Minus return code means error occurred

  External Function

        Function Name: aiWaitCallS

        Use Arguments: IVR_PORT_HOLD_PENDING
IVR_WAIT_TIME


     Evaluate

     If ExtFuncReturn   != 0

       #Log aiWaitCallSt Error

     End Evaluate
```

QueryACDState

Syntax:	*QueryACDState(SplitSkillExtensionNumber AvailableAgents, StaffedAgents, CallsInQuery)*
Description:	A request from the IVR to the IVR Server to retrieve information about a split/skill. The script must pass the extension number of the split/skill. The IVR Server will return split/skill extension numbers, the number of available agents, the number of staffed agents, and the number of calls in queue.

Parameters

SplitSkillNumber	The skill or split extension number to be queried.
AvailableAgents	The number of available agents in the queried split/skill returned from IVR Server.
StaffedAgents	The number of staffed agents in the queried split/skill returned from IVR Server.
CallsInQuery	The number of calls in queue in the queried split/skill returned from IVR Server.

Return Code

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
     #Use of external function – aiQueryACD
     #This external function can be used to make the IVR
wait for a particular call state.
     #INPUT:
     #Parameters 1  – SkillSplit extension to be queried
     #OUTPUT:
     #Parameters 1 – SkillSplit extension number queried
     #Parameters 2  – Number of Agents Available
     #Parameters 3  – Number of Staffed Agents
     #Parameters 4  – Number of Calls in Queue
     #RETURN CODE:
     #Return code zero (0) means successful
     #Minus return code means error occurred


  External Function
        Function Name: aiQueryACD
        Use Arguments: SplitSkillNumber AvailableAgents
StaffedAgents  CallsInQuery
        Return Field:  ExtFuncReturn


     Evaluate
     If ExtFuncReturn   != 0
```

```
        #Log aiQueryACD Error
      End Evaluate
```

ConferenceStart

Syntax:	*ConferenceStart(PartyToBeConferenced, UUI)*
Description:	A request from the IVR to IVR Server to start a conference call in a blind manner.

Parameters

PartyToBeConferenced	The party to be added to the conference call.
UUI	Any user-to-user information sent with the call.

Return Code

Return code sent from the IVR Server to the IVR to indicate that the request to start a conference call for the specified port has been processed.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
     #Use of external function – aiConfStart

     #This external function can be used to start
Conference call from IVR.

     #INPUT:

     #Parameters 1  – Destination

     #Parameters 2  – UUI Data

     #RETURN CODE:

     #Return code zero (0) means successful

     #Minus return code means error occurred


   External Function

        Function Name: aiConfStart

        Use Arguments: Destination  UUI

        Return Field:  ExtFuncReturn
```

```
Evaluate
If ExtFuncReturn   != 0
  #Log aiConfStart Error
End Evaluate
```

ConferenceComplete

Syntax: *ConferenceComplete(PartyToBeConferenced)*

Description: A request from the IVR to IVR Server to complete a conference call in a blind manner.

Parameters

PartyToBeConferenced The party to be added to the conference call.

Return Code

Return code sent from the IVR Server to the IVR to indicate that the request to complete the conference call for the specified port has been processed.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
     #Use of external function – aiConfComp

     #This external function can be used to start
Conference call from IVR.

     #RETURN CODE:

     #Return code zero (0) means successful

     #Minus return code means error occurred


18.  External Function

        Function Name: aiConfComp

        Return Field:  ExtFuncReturn


     Evaluate
     If ExtFuncReturn   != 0
       #Log aiConfComp Error
```

```
End Evaluate
```

ConferenceAbort

Syntax: *ConferenceAbort(PartyToBeAborted)*

Description: A request from the IVR to IVR Server to abort a conference call.

Parameters

PartyToBeAborted The party to be aborted from the conference call.

Return Code

Return code sent from the IVR Server to the IVR to indicate that the request to abort the conference call for the specified port has been processed.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
    #Use of external function – aiConfAbort

    #This external function can be used to start
Conference call from IVR.

    #RETURN CODE:

    #Return code zero (0) means successful

    #Minus return code means error occurred


  External Function

      Function Name: aiConfAbort

      Return Field:  ExtFuncReturn


    Evaluate

    If ExtFuncReturn   != 0

      #Log aiConfAbort Error

    End Evaluate
```

SetCustomerData

Syntax:	*SetCustomerData(Value1, Value2, Value3, Value4, Value5)*
Description:	A request from the IVR script that data be stored for this particular script session. The IVR Server will pass this request to the appropriate clients who will respond with a return code.

Parameters

Value1, Value2, Value3, Value4, Value5	Five key values passed by the IVR script to the server.

Return Code

Return code sent from theIVR Server to the IVR indicating the success or failure of the request.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
    #Use of external function – aiSetCustDat

     #This external function can be used to make the IVR
wait for a particular call state.

     #INPUT:

     #Parameters 1 to 5 – Desired data to be sent AIVRS

     #RETURN CODE:

     #Return code zero (0) means successful

     #Minus return code means error occurred

  External Function

        Function Name: aiSetCustDat

        Use Arguments: Data1  Data2  Data3  Data4  Data5

        Return Field:  ExtFuncReturn


     Evaluate

     If ExtFuncReturn   != 0

       #Log aiSetCustDat Error
```

```
End Evaluate
```

RequestCustomerData

Syntax: *RequestCustomerData(Key1, Key2, Key3, Key4, Key5)*

Description: A request from the IVR script to IVR Server for customer data. The appropriate client will receive this message and return data via the *RequestCustomerData* parameters. The script can pass and receive up to five key values to the server (and hence to the client).

Parameters

Key1, Key2, Key3, Key4, Key5	Five key values passed by the script to the server.

Return Code

Return code sent from the IVR Server to the IVR Indicating the success or failure of the request

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
    #Use of external function – aiGetCustDat

    #This external function can be used to make the IVR
wait for a particular call state.

    #INPUT:

    #Parameters 1 to 5 – Desired data to be sent AIVRS

    #OUTPUT:

    #Parameters 1 to 5 – Data returned from AIVRS

    #RETURN CODE:

    #Return code zero (0) means successful

    #Minus return code means error occurred

  External Function

       Function Name: aiGetCustDat

       Use Arguments: Data1  Data2  Data3  Data4  Data5

       Return Field:  ExtFuncReturn
```

```
Evaluate

If ExtFuncReturn   != 0

  #Log aiGetCustDat Error

End Evaluate
```

ExecuteCustomerFunction

Syntax:	*ExecuteCustomerFunction(FunctionID, KeyName1,Key Value1, KeyName2, Key2Value)*
Description:	A request from the IVR script to the IVR Server to have a client execute a function. The script can send and receive two sets of KeyName and KeyValue with this function.

Parameters

FunctionID	The ID or name of the function passed from the script to the client to indicate which particular function it should execute.
KeyName1, KeyValue1, KeyName2, KeyValue2	Two sets of KeyName and KeyValue that can be sent from the script to the client or received from the client through the IVR Server.

Return Code

Return code sent from the IVR Server to the IVR Indicating the success or failure of the request.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
#Use of external function – aiExeCustFnc

     #This external function can be used to make the IVR
wait for a particular call state.

     #INPUT:

     #Parameters 1  – Client function ID or function name

     #Parameters 2  – KeyName1

     #Parameters 3  – KeyValue1

     #Parameters 4  – KeyName2
```

```
        #Parameters 5  – KeyValue2
        #OUTPUT:
        #Parameters 1  – Client function ID or name executed
        #Parameters 2  – KeyName1 Returned
        #Parameters 3  – KeyValue1 Returned
        #Parameters 4  – KeyName2 Returned
        #Parameters 5  – KeyValue2 Returned
        #RETURN CODE:
        #Return code zero (0) means successful
        #Minus return code means error occurred
    Set Field Value
              Field: ExeCust1 = "TestExecCustFunction"
              Field: ExeCust2 = "ExeCustFunc Par1"
              Field: ExeCust3 = "ExeCustFunc Par2"
              Field: ExeCust4 = "ExeCustFunc Par3"
              Field: ExeCust5 = "ExeCustFunc Par4"
    External Function
           Function Name: aiExeCustFnc
           Use Arguments: ExeCust1  ExeCust2  ExeCust3
ExeCust4  ExeCust5
           Return Field:  ExtFuncReturn

        Evaluate
        If ExtFuncReturn   != 0
          #Log aiExeCustFnc Error
        End Evaluate
```

SetLoggingData

Syntax:	*SetLoggingData(LoggingKeyName, LoggingKeyValue)*
Description:	A request from the IVR script to the IVR Server to store logging data to the Interaction Data Server. If the IVR Server has a connection to the Interaction Data Server, it will store the appropriate key pairs into the value pair container for the call located at the specified port.

Parameters

LoggingKeyName, LoggingKeyValue	Logging key name and value.

Return Code

Return code sent from the IVR Server to the IVR indicating the success or failure of the request.

Please refer to ***Return Codes*** (on page 49).

Sample Code

```
     #Use of external function – aiSetLogDat

     #This external function can be used to pass data to
IVR Server for client data logging with other call-related
information.

     #INPUT:

     #Parameters 1  – Client function ID or function name

     #Parameters 1  – KeyName1

     #Parameters 2  – KeyValue1

     #RETURN CODE:

     #Return code zero (0) means successful

     #Minus return code means error occurred

  External Function

        Function Name: aiSetLogDat

        Use Arguments: "aiKeyName1"  "aiKeyValue1"

        Return Field:  ExtFuncReturn


     Evaluate

     If ExtFuncReturn   != 0

       #Log aiSetLogDat Error

     End Evaluate
```

CHAPTER 5

Appendix

In This Chapter

ai Prefix External Function Return Codes

The following table outlines the error codes that can return when IVR scripts use external functions through the IVR Server DIP.

Value	Meaning
Less than zero	Error number describes a condition detected by the DIP.
0	Function processed successfully.
1 to 99	Error number describes a condition detected by the IVR Server.
100 upwards	Error number describes a condition detected by the IVR Client.

Return Codes

DIP Return Codes

The following table documents the return codes that are passed from the IVR Server to the DIP and onto the script.

AIVR_SUCCESS = 0	Completed successfully.
DESTINATION_INVALID = 1	The destination is not valid.
DESTINATION_BUSY = 2	The destination is busy.
AVAYACT_NOT_AVAILABLE = 3	The Avaya CT link is not available.
CUSTOMER_NOT_AVAILABLE = 4	There are no clients connecting to IVR Server.
UNKNOWN_MESSAGE_FROM_IVR = 5	The IVR Server received an unknown message ID from the IVR.
CALL_CLEARED_ALREADY = 6	The call on the specified port is cleared already.
PORT_OUT_OF_SERVICE = 7	The specified port is not yet monitored.
PORT_INVALID_STATUS = 8	The specified port is not in an expected status.
UNKNOWN_PORT_NUMBER = 9	Unknown IVR port number.
AIDSSERVER_ERROR = 96	Get an error from Interaction Data Server.
AIDSERVER_NOT_AVAILABLE = 97	There is no connection to the Interaction Data Server.
UNKNOWN_ERROR_FROM_SERVER = 98	Unknown error from the IVR Server.
PORT_UNLICENSED = 99	The specified port is unlicensed.
100 and over	Return codes defined by clients.

TSM return codes

The following table documents the return codes that are passed from the DIP to the script when there is a communication problem encountered with the IVR Server.

TSM_SUCCESS = 0	
TSM_UNKNOWN_MESSAGE_TYPE = -9	The DIP received an unknown message type from the script.
TSM_SOCKET_DOWN = -10	There is no IVR Server connected to the IVR.
TSM_OUTSTANDING_LIMIT_REACHED = -11	The number of requests sent from the script is beyond the DIP's outstanding limit. This request will be ignored.
TSM_INVALID_PORT_NUM = -12	The DIP received an invalid port number from the script.
TSM_CALL_DELIVERED_DELAY = -13	This code is sent to the script if the *CallDelivered* event is received by the DIP after time-out (6 seconds).

Error Logging

IVR Server DIPs write error information relating to their operation to a series of log files.

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

Log files are stored in the same directory folder as the DIP component: /vs/data/aivrs.

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.

Index