Avaya™ Interaction Center Release 6.1.3 Telephony Connectors Programmer Guide

07-300105
Issue 3
June 2004

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. Customer and/or End User agree to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation to the extent made by the Customer or End User.

Link disclaimer

Avaya Inc. is not responsible for the contents or reliability of any linked Web sites and 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.

Warranty

Avaya Inc. provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product, while under warranty, is available through the following Web site:

http://www.avaya.com/support

Preventing toll fraud

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

Avaya fraud intervention

If you suspect that you are being victimized by toll fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Web site:

http://www.avaya.com/support

Providing telecommunications security

Telecommunications security (of voice, data, and video communications) is the prevention of any type of intrusion to (that is, either unauthorized or malicious access to or use of) your company's telecommunications equipment by some party.

Your company's "telecommunications equipment" includes both this Avaya product and any other voice/data/video equipment that could be accessed via this Avaya product (that is, "networked equipment").

An "outside party" is anyone who is not a corporate employee, agent, subcontractor, or person working on your company's behalf. Whereas, a "malicious party" is anyone (including someone who may be otherwise authorized) who accesses your telecommunications equipment with either malicious or mischievous intent.

Such intrusions may be either to/through synchronous (time-multiplexed and/or circuit-based) or asynchronous (character-, message-, or packet-based) equipment or interfaces for reasons of:

- Use (of capabilities special to the accessed equipment)
- Theft (such as, of intellectual property, financial assets, or toll-facility access)
- Eavesdropping (privacy invasions to humans)
- Mischief (troubling, but apparently innocuous, tampering)
- Harm (such as harmful tampering, data loss or alteration, regardless of motive or intent)

Be aware that there may be a risk of unauthorized intrusions associated with your system and/or its networked equipment. Also realize that, if such an intrusion should occur, it could result in a variety of losses to your company (including, but not limited to, human and data privacy, intellectual property, material assets, financial resources, labor costs, and legal costs).

Your responsibility for your company's telecommunications security

The final responsibility for securing both this system and its networked equipment rests with you, an Avaya customer's system administrator, your telecommunications peers, and your managers. Base the fulfillment of your responsibility on acquired knowledge and resources from a variety of sources, including, but not limited to:

- Installation documents
- System administration documents
- Security documents
- Hardware-/software-based security tools
- Shared information between you and your peers
- Telecommunications security experts

To prevent intrusions to your telecommunications equipment, you and your peers should carefully program and configure:

- Your Avaya-provided telecommunications systems and their interfaces
- Your Avaya-provided software applications, as well as their underlying hardware/software platforms and interfaces
- Any other equipment networked to your Avaya products.

Trademarks

Avaya is a trademark of Avaya Inc.

Insert all other Avaya Trademarks here, then delete this paragraph. DO NOT include other company's trademarks.

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

Document ordering information:
Avaya Publications Center

Voice: +1-207-866-6701
1-800-457-1764 (Toll-free, U.S. and Canada only)

Fax: +1-207-626-7269
1-800-457-1764 (Toll-free, U.S. and Canada only)

Write: Globalware Solutions
200 Ward Hill Avenue
Haverhill, MA 01835 USA
Attention: Avaya Account Manager

Web: http://www.avaya.com/support

E-mail: totalware@gwsmail.com

Order: Document No. 07-300105, Issue 3
June 2004

For the most current versions of documentation, go to the Avaya support Web site:

http://www.avaya.com/support

COMPAS

This document is also available from the COMPAS database. The COMPAS ID for this document is 101687.

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

Avaya™ Interaction Center
Release 6.1.3
Telephony Connectors Programmer Guide

Contents

Before You Begin

This section includes the following topics:

Typographical Conventions

This guide uses the following font conventions:

Font Type	Meaning
`command`	This font signifies commands, information that you enter into the computer, or information contained in a file on your computer.
`commandvariable`	This font indicates variables in a command string.
italics	This font is used to add emphasis to important words and for references to other chapter names and manual titles.
link	Blue underlined text in online documents indicates a hypertext jump to related information. To view the related material, click the blue underlined text.

Notes, Tips, and Cautions

Note:

A note calls attention to important information.

Important:

An important note calls attention to a situation that has the potential to cause serious inconvenience or other similar repercussions.

Tip:

A tip offers additional how-to advice.

CAUTION:

A caution points out actions that may lead to data loss or other serious problems.

Contacting Technical Support

If you are having trouble using Avaya software, you should:

1. Retry the action. Carefully follow the instructions in written or online documentation.
2. Check the documentation that came with your hardware for maintenance or hardware-related issues.
3. Note the sequence of events that led to the problem and the exact messages displayed. Have the Avaya documentation available.
4. If you continue to have a problem, contact Avaya Technical Support by:
   - Logging in to the Avaya Technical Support Web site http://www.avaya.com/support/qq
   - Calling or faxing one of the following numbers from 8:30 a.m. to 8:30 p.m. (Eastern Standard Time), Monday through Friday (excluding holidays):
     - Toll free in the U.S. and Canada: 1-888-TECH-SPT (1-888-832-4778)
     - Direct line for international and domestic calls: 1-512-425-2201
     - Direct line for faxes: 1-512-997-4330

- Sending email with your question or problem to crmsupport@avaya.com. You may be asked to email one or more files to Technical Support for analysis of your application and its environment.

Note:

If you have difficulty reaching Avaya Technical Support through the above URL or email address, please go to http://www.avaya.com for further information.

Product Documentation

Most Avaya product documentation is available in both printed and online form. However, some reference material is available only online, and certain information is available only in printed form. A PDF document with detailed information about all of the documentation for the Avaya Interaction Center is included in the `Doc` directory on the product CD-ROM. This PDF document is also included on the separate documentation CD-ROM.

Readme File

The Readme file is a PDF file included on the Avaya Interaction Center software CD-ROM. This file contains important information that was collected too late for inclusion in the printed documentation. The Readme file can include installation instructions, system requirements, information on new product features and enhancements, suggested work-arounds to known problems, and other information critical to successfully installing and using your Avaya software. Avaya may also deliver an Addendum to the Readme, which will be posted on the Avaya Technical Support Website. The Readme Addendum will contain similar information uncovered after the manufacture of the product CD-ROM. Review the Readme file and the Readme Addendum before you install your new Avaya software.

Electronic Documentation

The electronic documentation (in PDF or HTML format) for each Avaya Interaction Center product is installed automatically with the program. Electronic documentation for the entire Avaya product suite is included on the product CD-ROM and the documentation CD-ROM.

You can also view the documentation set online at http://www.avayadocs.com.

Printed Documentation

You can purchase printed copies of these manuals separately. For details, see Order: Document No. 07-300105, Issue 3 June 2004 on the back of this manual's title page.

License to Print the Electronic Documentation

Online copies of documentation are included on the CD-ROM that accompanies every software release. An Avaya customer who has licensed software (a "Licensee") is entitled to make this online documentation available on an internal network or "intranet" solely for the Licensee's use for internal business purposes. Licensees are granted the right to print the documentation corresponding to the software they have purchased solely for such purposes.

Right-To-Print License Terms

Documents must be printed "as-is" from the provided online versions. Making changes to documents is not permitted. Documents may be printed only by any employee or contractor of Licensee that has been given access to the online documentation versions solely for Licensee's internal business purposes and subject to all applicable license agreements with Avaya. Both online and printed versions of the documents may not be distributed outside of Licensee enterprise or used as part of commercial time-sharing, rental, outsourcing, or service bureau use, or to train persons other than Licensee's employees and contractors for Licensee's internal business purposes, unless previously agreed to in writing by Avaya. If Licensee reproduces copies of printed documents for Licensee's internal business purposes, then these copies should be marked "For internal use only within <Licensee> only." on the first page or cover (where <Licensee> is the name of Licensee). Licensee must fully and faithfully reproduce any proprietary notices contained in the documentation. The copyrights to all documentation provided by Avaya are owned by Avaya and its licensors. By printing any copy of online documentation Licensee indicates its acceptance of these terms and conditions. This license only governs terms and conditions of printing online documentation. Please reference the appropriate license agreement for terms and conditions applicable to any other use, reproduction, modification, distribution or display of Avaya software and documentation.

Educational Services

Avaya University provides excellent training courses on a variety of topics. For the latest course descriptions, schedules, and online registration, you can get in touch with us:

- Through the web at http://www.avaya-learning.com/logon_form.asp
- Over the telephone at 800-288-5327 (within the U.S.) +001 303-406-6089 (outside of the U.S.)
- Through email at Avaya.U.Helpdesk@accenture.com

Chapter 1: Introduction

Computer Telephony Integration (CTI) is the technology that applies computer intelligence to telecommunications devices by integrating telephone hardware (PBXs, IVRs, and ACDs) with computers and software. The Telephony Connector server is the heart of the Avaya Computer Telephony Integration solution.

This section includes the following topics:

The Telephony Connector server

The Telephony Connector server is commonly referred to as Avaya Telephony server or Avaya TS (TS). It is the Avaya server software responsible for linking Avaya Interaction Center (Avaya IC) to CTI products; such as private branch exchange (PBX) systems, automatic call distribution (ACD) systems, and interactive voice response (IVR) units.

Using the Avaya TS, other servers and clients in Avaya IC request telephony services (such as transfer or route a call, hang up a call) by invoking Avaya TS methods. These servers and clients also assign to the Avaya TS to receive telephone-related events (for example, “there is a call from customer 999-1111”).

The Avaya TS separates client applications from the telephone switch (ACD) and its interface by encapsulating switch-specific features. Client applications are then portable across multiple types of switches, CTI links, and other types of telephony interfaces, such as Integrated Services Digital Network (ISDN).

Multiple Avaya TSes can be implemented for redundancy to enable transparent failover in the events of a Telephony server failure. For more information on server failover on Avaya IC, refer to *IC Administration Volume 1: Servers & Domains*.

For more information on the TS, refer to Telephony Server on page 47

The Telephony Server Queue Statistics server

The Telephony Server Queue Statistics server (TSQS) is the Avaya server that monitors the voice channel and maintains queue statistics on the Agent Data Unit (ADU) server. When running On the Avaya™ DEFINITY and the Avaya™ Communication Manager switches, the TSQS relies on the Avaya TS to interact with the switch. On the Aspect CallCenter and the Nortel Meridian switches, the TSQS interacts directly with the switch.

Although the TSQS is an independent software server, it is documented is in this guide for convenience.

For more information on the TSQS, refer to Queue Statistics Server on page 89.

How this book is organized

This guide is intended for internal and external audiences.

For external users, this book is structured and sequenced according to the other Avaya Interaction Center (IC) guides, namely, pre-requisites, installation and administration, and provides cross-references to specific chapters of these documents.

Chapters dedicated to the operation of the server present additional details, such as the data containers, alarms, events generated by the server, as well as its method calls (API).

Terminology and Acronyms

This section lists and defines some of the terms and acronyms used in the *Telephony Connectors Programmer Guide*.

Term	Definition
ACD	Automatic Call Distributor - a device at the contact center that handles and routes calls to queues and agents. It also provides information that can be used to determine the operational efficiency of the contact center.
ACD-DN	Nortel's nomenclature for a queue.
ADU	Agent Data Unit - a record of an agent's activities during an Avaya Agent session.
ADUID	A unique identifying number assigned to the ADU by the ADU server when it creates the ADU.
ADU server	Agent Data Unit server - the custodian of ADUs on Avaya IC, handles current information on all active agents and queues.
Adjunct Routing	Definity's nomenclature for a Route Point
ANI	Automatic Number Identification - the telephone number of the telephone from which the call into the contact center originated.
API	Application Programming Interface - a group of functions or methods used by programmers to customize the server.
Avaya CM	Avaya Communication Manager - the Avaya PBX/Call Center Media Servers previously known as DEFINITY and MultiVantage software.

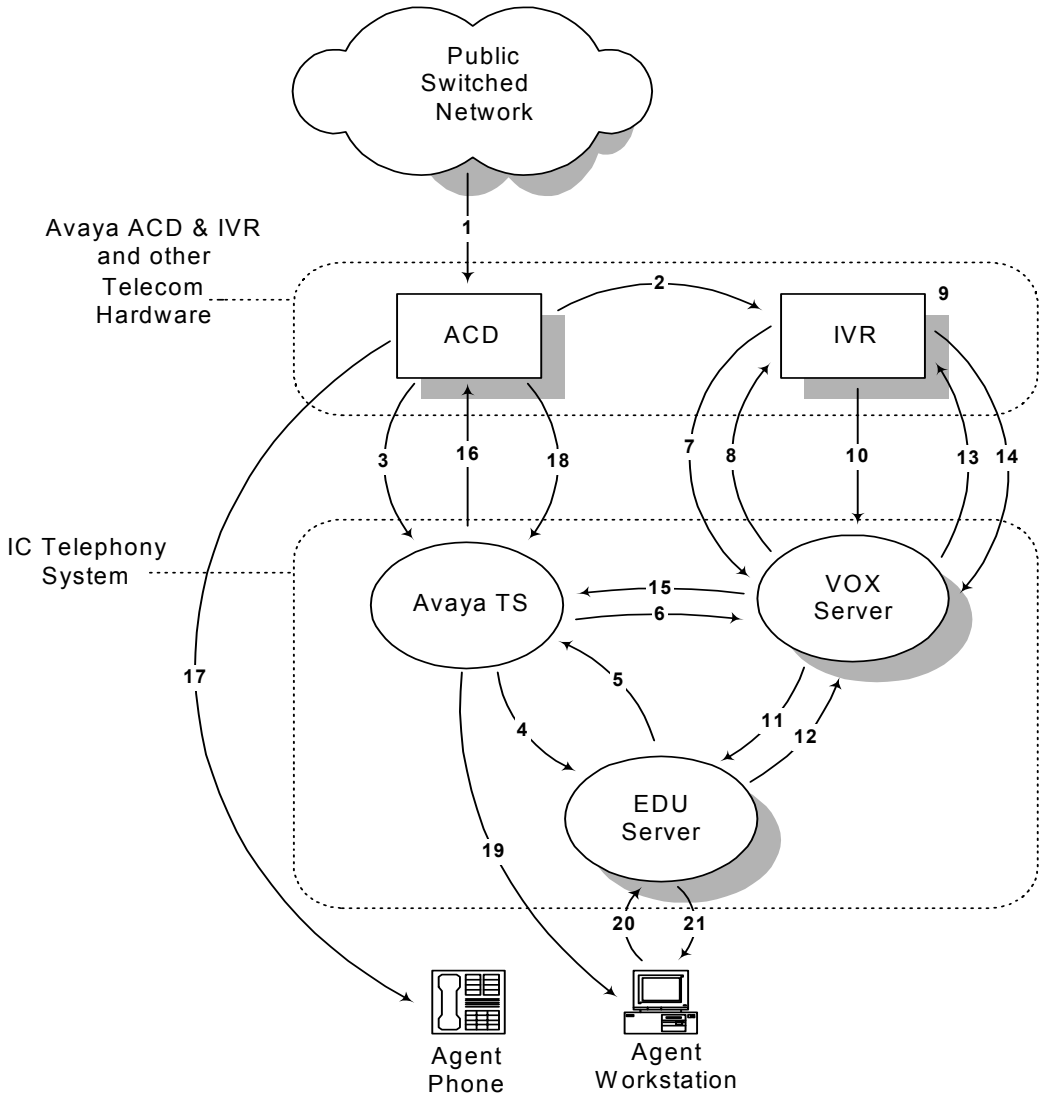
Term	Definition
Business Advocate	Avaya Business Advocate - the Avaya resource management system
Call Reference ID	Call reference ID as assigned by the switch.
CCT	Call Control Table - Aspect's nomenclature for a vector.
CDN	Control Directory Number - Nortel's nomenclature for a route point.
CVLAN	An application that extends the CallVisor® PC API over the LAN. The CVLAN application allows multiple clients with different operating systems to access the CVPC API.
DN	Directory Number - also "Dialable" Number
DNIS	Dialed Number Identification Service - the number assigned to the telephone that was dialed by the caller. For example, if the caller dials 1-800-123-4567, the DNIS is not 1-800-123-4567. When this call arrives in the PBX, it is assigned a number, called the DNIS, to identify the telephone number that was dialed. In this example, the DNIS could be 4567. DNIS is a service provided by the carriers to help customers identify the telephone numbers they dialed.
DLG	DEFINITY Lan Gateway - a software application in the MAPD that serves as an ISDN router of ASAI messages through a TCP "tunnel" via 10 Base-T Ethernet.
EDU	Electronic Data Unit - a record of a caller's activities during an Avaya Agent session.
EDUID	A unique identifying number assigned to the EDU by the EDU server when it creates the EDU.
EDU server	Electronic Data Unit server - the server that maintains all active EDUs on all of the media channels on Avaya IC.
ISDN	Integrated Services Digital Network - a system of digital phone connections which allows data to be transmitted using end-to-end digital connectivity.
IVR	Interactive Voice Response - device which interacts with callers using prerecorded messages that prompt callers to select available options.
MAPD	Multi-Application Platform for DEFINITY - an Intel based secondary processor available as an Avaya CM TN board complex and is housed in an Avaya CM carrier. MAPD serves as a platform for various applications.

Term	Definition
Parking Device	The switch resource used by Business Advocate to hold calls until it finds a contact resource to handle the call. • On the Avaya CM, it is a VDN pointing to a vector which contains an adjunct routing step. • On the Aspect CallCenter, it is a CCT. • On the Nortel Meridian, it is a CDN.
PBX	Private Branch Exchange - private telephone exchange used within an enterprise and located on the premises.
PDK	Predictive Dialing Kernel - Generic name for the Predictive Dialing server that enables automated outbound dialing functionality.
PSTN	Public Service Telephone Network - typically refers to a local long distance exchange carrier.
Resource Manager	The Avaya IC server that matches contacts to those agents who are best qualified to handle them. The Resource Manager is used to support the optional Avaya Business Advocate feature of Avaya IC.
RONA	Redirected on No Answer - a feature on the switch that allows the switch to redirect a queue call that is not answered by an agent after a certain number of rings or time.
Route Point	Directory Number (DN) on the switch, which requests call routing instructions from a host application (Avaya TS) outside the switch.
Service Class	The service class defines a type of interaction work for Business Advocate using a qualifier set. Each service class is assigned a goal that is used to measure how quickly contacts in that service class should be serviced. Each Advocate queue has a one-to-one mapping with the service class.
UUI	User to User Information - a field within an ASAI message that normally contains the EDUID of a call.
VDN	Vectored Directory Number (DN) - used on the Avaya CM.
Vector	Set of instructions or scripts that are implemented by the switch to conduct call processing.
TS	Avaya Telephony server
TSA	Avaya Telephony Server Adapter server - connection point between the Resource Manager server and the Avaya TS. The TSA is used to support the optional Avaya Business Advocate feature of Avaya IC.

Term	Definition
TSQS	Avaya Telephony Server Queue Statistics server - the server that monitors the voice channel and keeps current queue statistics in the ADUID.
Virtual Queue	A logical abstraction that represents a number of real queues or service classes. An agent can pick a virtual queue to transfer a call to a qualified agent.

Example call flow

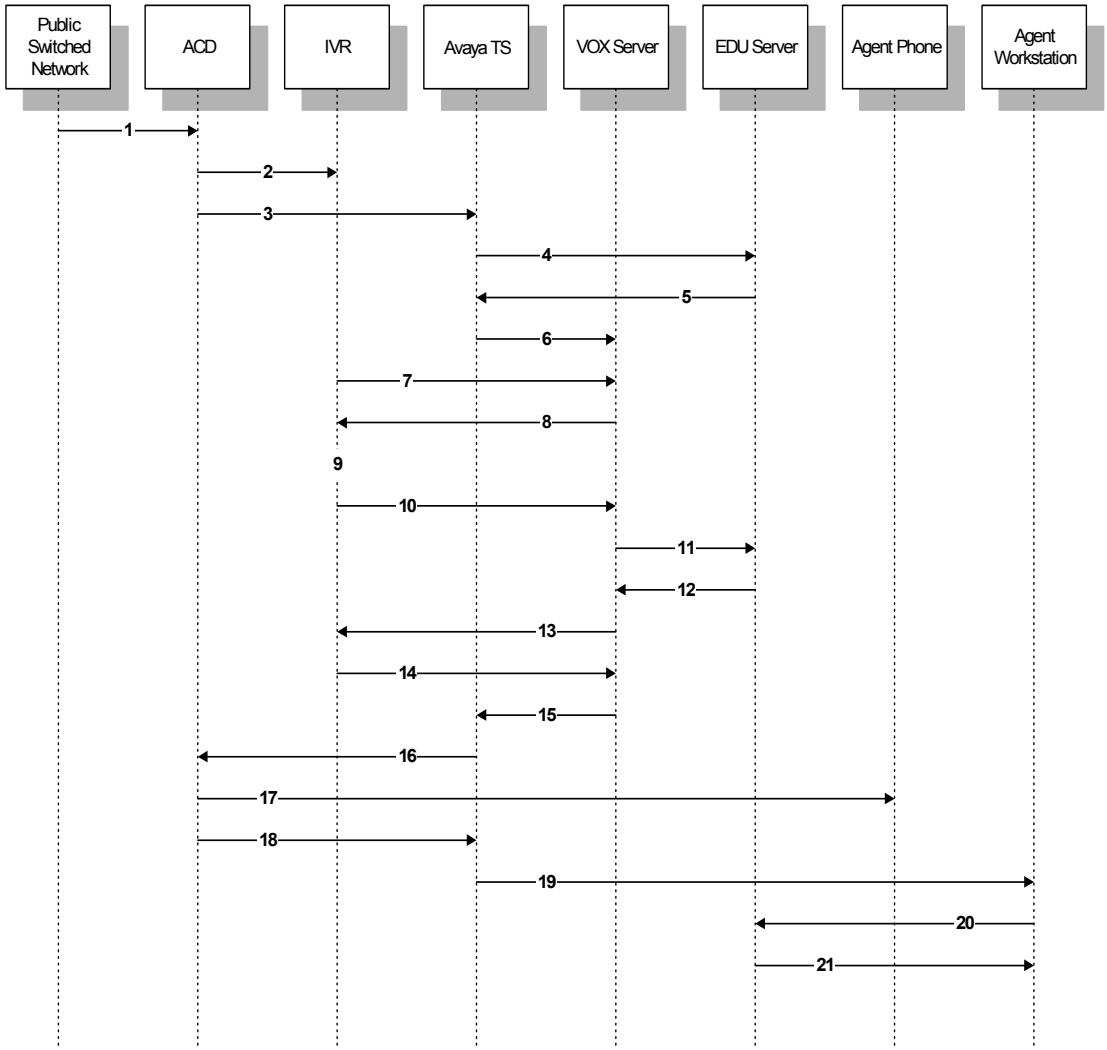
The function of the Avaya TS is illustrated in the following call flow. Please note that call flows vary, depending on business rules. Additional call flow diagrams are provided in Generic Call Flows on page 297.



Step	Description
1	A call arrives at the switch (ACD) from the Public Switched Network.
2	The ACD, in this example, is set up to route calls directly to an Interactive Voice Response (IVR) unit. The call moves from the ACD to the IVR.
3	The ACD notifies the Avaya TS that it has received a call on a line that the Avaya TS was previously monitoring.
4	The Avaya TS invokes a method on the EDU server, which instructs the EDU server to create a new EDU for the call. The method specifies the elements to place into the EDU (such as ANI and DNIS).
5	The EDU server creates the EDU with the specified elements and sends it to the Avaya TS.
6	The Avaya TS checks its tables to see which clients or servers previously indicated an interest in monitoring events on the phone line to which the call was routed. The Avaya TS discovers that the VOX server is an interested party, so the Avaya TS sends an incoming call event to the VOX server. The event includes the extension to which the call was routed and the call's EDU identifier (EDUID).
7	The IVR notifies the VOX server of the new call. The notification includes the channel number on which the call arrived.
8	The VOX server associates the IVR channel number with a telephone extension. The VOX server then matches that extension with the incoming call event received in Step 6. The EDUID from the incoming call event is then returned to the IVR.
9	The IVR greets and prompts the caller for an account number and collects the digits entered by the caller via a touch-tone keypad.
10	The IVR instructs the VOX server to place the account number in the EDU as a name/value pair (for example, “account”, “123456789”).
11	The VOX server translates the IVR request into a CORBA method call to the EDU server (VDU.SetOneValue).
12	The EDU server stores the value and returns a response to the VOX server (VDU.SetOneValue.response).
13	The VOX server informs the IVR that its request (see Step 10) has been granted.
14	The IVR invokes a method on the VOX server (VOX.Transfer) to route the call to an agent queue.
15	The VOX server invokes the Avaya TS Transfer method on behalf of the IVR.
16	The Avaya TS instructs the ACD to transfer the call.

Step	Description
17	The ACD places the call in a queue and transfers it to the next available agent, whose telephone rings.
18	The ACD notifies the Avaya TS of the transferred call. The Avaya TS keeps track of which EDUID is associated with the call.
19	The Avaya TS sends an incoming call event (TS.IncomingCall.event) to the client application assigned to the extension to which the call was transferred. The event includes the EDUID for the call.
20	The client application uses the EDUID to request an account ID from the EDU server (VDU.GetOneValue).
21	The EDU server responds to the client application (VDU.GetOneValue.response) with the requested account ID.

The following diagram illustrates the same call flow in a different format.



Chapter 2: Installation

This chapter provides general installation and configuration information for the Avaya Telephony server (TS). It also provides links to the appropriate sections of this manual or other manuals in the Avaya IC documentation set for more detailed information.

This section includes the following topics:

Configuration considerations

This section provides general voice channel configuration considerations for Avaya IC. Refer to *Installation and Configuration* when installing and configuring the voice channel and the other Avaya IC components, on your system.

- Make sure you configure an ACD name parameter for each ACD (switch) in your Avaya IC environment. You can include identifying information, such as department or site in the ACD name. This is done on IC Manager when you add a TS to your system.
- You may need to create voice queues to gather statistics information, to provide a literal translation for an ACD (switch) or to integrate with routing hints.You can also create a queue for each of the possible route (transfer) points where the Incoming call flow can route a voice contact.
- During configuration, you can specify the switch for the TS on the machine that hosts the TS. Specify the switch on the General tab of the Avaya IC Configuration Tool to ensure the executable file matches the executable name in IC Manager.
- Assigning too many agents to a TS could negatively impact system performance. It is recommended that you assign no more than 800 agents to each TS in your Avaya IC environment.
- The TS and TSQS may be slow to start in a system with an large number of queues (over 2,000). Be aware of the potential impact of large number of queues when you plan your Avaya IC system.

Prerequisites

The prerequisites for the Avaya Telephony server (Avaya TS) may vary based on system environment and implementation. Refer to the *Installation Planning and Prerequisites* for more information on prerequisites that apply to your implementation and installation.

Supported switches and protocols

The following table illustrates the switches (ACDs), and protocols used by those switches, that are supported by the Avaya TS in IC 6.1.3:

ACD Type	ACD Model	ACD Protocol
Aspect	CallCenter version 8 via Portal (not via Event Bridge)	AspectCMI
Avaya	Avaya DEFINITY G3 versions 8.3, 9.5, 10 Avaya Communication Manager 1.1, 1.2, 1.3	ASAI
Nortel	Meridian 1 using Meridian Link 5.0c	MLP
Nortel	Meridian 1 using Symposium Server 4.0	MLS

These switches are supported on various platforms. For more information on supported platforms and software requirements, refer to the *Installation Planning and Prerequisites*.

Chapter 3: Call Routing

This chapter describes the implementation of Host-based Call Routing on Avaya IC 6.1.3. Host-based Call Routing is the ability of an ACD (switch) client application to dynamically influence call handling. This chapter also discusses issues with Host-based Call Routing and offers recommendations on how to setup Avaya IC to take full advantage of Host-based Call Routing features.

This section includes the following topics:

Host-Based Call Routing

Call routing is how contact centers distinguish and prioritize voice contacts. Call routing lets the contact center differentiated how they handle treatment of customers, to create better relationships and retain, or increase, their market penetration. Typically, call routing is defined by a set of instructions, scripted or table driven, defined in the switch fabric, and executed upon call connection to specific end-points in the switch. The actual mechanism and types of end-points change from manufacturer to manufacturer, but the concept is the same.

Host-based Call Routing is an ACD feature, which allows a host application, like an ACD client for CTI operations (the Telephony server on Avaya IC), to determine where a call should be moved next, thus influencing prioritization of the various calls arriving at the Call Center dynamically. This determination is based on criteria foreign to the switch itself, for example, data stored in a database and linked to a customer via a phone number.

The following two services are available from the Public Switched Telephone Network (PSTN) to assist in identifying the source and destination of a call:

- Automatic Number Identification (ANI)
- Dialed Number Identification Service (DNIS)

The methodology on how this all happens varies from manufacturer to manufacturer. Therefore, it is important to understand what is available, and issues behind it, so that proper exploitation of the feature is possible. This chapter presents this information in sections for each of the switches supported in Avaya IC 6.1.3.

Avaya DEFINITY/Communication Manager

Scripts are stored in vectors on the Avaya DEFINITY/Communication Manager switches, and these vectors are associated with Vector Directory Numbers (VDNs). A route request is initiated when a vector encounters the command “adjunct routing”. At this point the vector processing stops, and a route request (C_RT_REQ) is posted to the host application (the TS) associated with the ASAI station specified in the adjunct route step.

Note:

The Avaya DEFINITY/Communication Manager has an adjunct route limit of 2000 calls. When this limit is exceeded, calls are not dropped, but the Adjunct Route step in the vector processing falls through to the next Route step fails. There is no way to detect the cause of the adjunct route failure inside the vector. Nothing returned to the TS because the adjunct route failed.

This architecture means the TS receives route requests per link extension, not per VDN. The VDN and vector are simply the “connecting elements” between a call, a link extension, a MAPD signal, and the client application.

The route request carries a “route id” (on Avaya CM it is an association ID, such as “saoid”), which is used by the host application (the TS), upon route response (C_RT_SEL), to indicate which call to route.

The TS keeps track internally of the "route id" for a route request, and associates it with the EDUID for a given contact. This enables the client application (typically IC Workflow Server) to specify the EDUID and the destination to which a call is to be routed. The TS performs tasks like formatting the ASAI route response.

The client application tells the TS where to route the call via TS.Route() request.

Vectors

Vector scripts for route points use the following layout:

```
1    wait-time 1 sec hearing ringback
2    adjunct routing link 140
3    wait-time 5 secs hearing ringback
4    route to number 451 with cov n if unconditionally
```

The vector in this example indicates that the customer will hear ringing for 1 second, then the switch will post a route request to the TS associated with link 1 (the ASAI station, such as a link extension). The switch then waits for 5 seconds (wait-time step) for a route response from the TS.

Route Request/Responses

In a successful route response, step 4 in the previous example, is not be executed. The call is routed to a different Directory Number (DN) in the switch, and, if appropriate, scripting continues from there.

Typically, route requests are used as a placeholders for incoming calls from customers. They allow the Avaya IC to determine how to best route a call based on available data. Calls are usually routed within a 4 to 6 second window.

With Avaya Business Advocate (Business Advocate) on Avaya IC, VDNs similar to the one in the previous example are configured with a wait-time period of 1 hour. This provides Business Advocate with the flexibility to “store” (or park) a call for longer periods of time, then route the call to a given agent.

Route End

When the switch moves to step 4, it cancels the original route request post to the TS with a C_RT_END (a route end). If a route response is not received within the specified time in the wait-time step, the route is terminated and step 4 is executed. In this case, the switch moves the call to DN 451, which could be another VDN (a queue, or a route point), an agent, or a station.

For some calls, the logic in Routing Engine may indicate to the TS that no special routing is to be conducted. For example, the vector continues as if the TS never responded to the route request. This operation is called "default routing". In the previous script, step 3 is terminated before 5 seconds and step 4 is executed.

Link Failure

If the Telephony server (host application) fails, the CTI link is severed, and the switch waits 5 seconds before going to step 4 in the previous example. The TS will not receive the route request in step 2 until the CTI link is re-established, and therefore will not respond. Calls on step 3 are processed from step 4, during the link failure.

TS Assignment

TS clients (such as IC Agent, Routing Engine, and TSQS) need to assign with servers in order to receive events, and to indicate to the TS that an association with the switch should be created for the given device the client wants to monitor or control.

The switch maintains a one-to-one relationship between the device and the switch association, one device, one switch association. However, multiple clients can be assigned to the same device, and the TS will multiplex events/requests/responses appropriately.

Some rules that apply are:

1. Assigned clients cannot have different ADUs.
   - Two agents cannot be assigned to the same device.
   - However, an agent and a server or an agent and two servers can be assigned to the same device because servers do not have ADUs.
   - The same agent can be assigned to the same device from more than one workstation.
2. Race conditions between applications are not resolved by the TS.
   - If two Routing Engines assigned to the same route run into a race condition between them, the TS will not resolve the race condition.

   Note:
   Race conditions on resolving a route request do not cause duplicate EDUs, but they may cause incorrect call handling, and/or application errors.

The only method used by a client to associate for a device is TS.Assign(). To assign for a route point, the criterion is TS.Assign("*r40010"), where "40010" is a VDN front-ending a vector, which is set to post a Route Request via Adjunct Routing step. The Avaya TS allows for a criterion of "*r*default", which has special meaning and indicates "all-non-monitored-VDNs". For more information, refer to TS.Assign("*r*default") on page 36.

IC 5.6

TS.Assign("*r")

On eContact 5.6, the TS configured for the Avaya DEFINITY/Communication Manager allowed for only one client application to be associated (assigned) for route requests. This was accomplished via the method call TS.Assing("*r").

Restrictions

The immediate impact of only one association receiving all route requests is scalability. If the client application could not handle the volume of incoming route requests, there was no way to increase system throughput.

The second issue is related to filtering route requests arriving from different VDNs. Typically, a contact center has calls routed based on incoming VDNs, because they identify a group handling a certain contact center service. For instance, a customer dialing customer service does not want to be connected to billing, and vice-versa. In eContact 5.6, this filtering and consistency mechanism had to be built inside the various flows executed by the Avaya Workflow server.

IC 6.x

On Avaya IC 6.1, the ability of the TS to further extend the TS.Assign() criteria and the filtering mechanism per VDN is built into the TS. The assign method syntax is enhanced and changed to identify the VDN in which a client application is interested. The TS informs the client application of a route request arriving on the particular VDN and filters out (and gives to other client applications if applicable) route requests from other VDNs.

On Avaya IC 6.1, the TS was modified to support multiple assignments to same device. For example, two Routing Engines could receive route-requests from same VDN. For more information, refer to Multiple assignments on page 37).

TS.Assign(“*rVDN”)

On Avaya IC 6.1, the syntax used is TS.Assign(“*r4001”), where 4001 is a VDN front ending a vector in the form presented above.

TS.Assign(“*r*default”)

The departure from TS.Assign(“*r”) had a migration impact on the customer configuration built around it. In order to allow for some backward compatibility with eContact 5.6, the Avaya IC 6.1 TS provides similar functionality via TS.Assign(“*r*default”).

This method tells the TS to send all of the route requests to all of the VDNs that are not individually monitored by the associated client application. In a configuration where there are 4 VDNs and one Routing Engine assigned to "*r*default", and another Routing Engine assigned to a specific VDN, the one assigned to "*r*default" receives route requests for 3 VDNs. All the other route requests pertain to the individually assigned VDN and, as such, go to the second Routing Engine.

If a call is routed based on an assign criteria of *r*default, the VDN number is added to the switch specific field as "route_vdn". This causes this field to be written to the EDU in the IncomingCall event.

The behavior of the “catch-all-route-requests” device on eContact 5.6, and Avaya IC 6.1, is similar, but not the same. This is indicated by the assignment criteria. On Avaya IC 6.1, it is “*r*default”, on eContact 5.6 it is “*r”. It is worth noting that migration issues must be considered when moving from eContact 5.6 to Avaya IC 6.1.

Multiple assignments

Since multiple assignments per device are allowed on Avaya IC 6.1.3, you should be aware of the implications of double assignments.

There are two possible configurations:

- One Telephony server, two Routing Engines
- Two Telephony servers, two Routing Engines

Typically, these combinations exist to distribute traffic, load, and/or promote redundancy, which improves crash-recovery and increases system reliability.

One TS – Two Routing Engines

In this scenario, the two Routing Engines are assigned to the same VDN, and receiving route requests from the same TS.

Without the appropriate coordination between Routing Engines, a race condition is established. The first Routing Engine to respond to a route request moves the call, while the second Routing Engine receives a request failure response.

This condition is difficult to diagnose and it places an unnecessary load on the system starting with the TS, passing through the CTI link, and reaching all the way inside the switch.

Two TSes – Two Routing Engines

In this scenario, the two Routing Engines are assigned to the same VDN, but through different Telephony servers, and in turn, different CTI links whether it is through same MAPD or not.

Note:

A route request is associated with a link extension as illustrated in the example in Vectors on page 33. Depending on how the vector is configured, Routing Engines might never see a request originating from the “cross-domain” VDN. This would be the case in which an adjunct route step does not exist on a vector to send a route request to the Routing Engine on the “other domain”.

Having adjunct routes cascaded appears to improve this scenario, but it also has issues. Examine the following two vectors:

```
VDN 100 - Vector1            (route point 1)
     adjunct routing link 1400
     wait-time 5 seconds
     adjunct routing link 1401
     wait-time 5 second
     route to vector 99 unconditionally

VDN 200 - Vector2            (route point 2)
     adjunct routing link 1401
     wait-time 5 seconds
     adjunct routing link 1400
     wait-time 5 seconds
     route to vector 99 unconditionally
```

In this case, with two Telephony servers and two Routing Engines, one could have better link redundancy. However, a route end due to timeout, on either vector, would cause two EDUs to be associated with same customer contact. Unfortunately, route ends due to timeout are a common occurrence, even more so when the system is overloaded and database matches are slow.

*"*r*default"*

The TS can contain the following assignment combinations:

- one Routing Engine assigned to "*r*default" – no other "*r" assignment
  An eContact 5.6 setting, only one Routing Engine and all route requests are processed by this single Routing Engine.
- one Routing Engine assigned to "*r*default" – other Routing Engine assigned to "*rVDN"
  With Routing Engine1 assigned to "*r*default" and Routing Engine2 assigned to "*r*VDN", a RoutingEnginge2 crash causes route requests to "*rVDN" to be posted to workflow1. If Routing Engine1 is not properly configured to handle calls arriving from this VDN, these calls might be routed incorrectly.
- two Routing Engines assigned to "*r*default" – no other "*r" assignments
  With two Routing Engines assigned to "*r*default", a race condition is established between them. The first Routing Engine to respond with TS.Route() moves the call to a new destination, and the other Routing Engine receives a response indicating failure from the TS.
- two Routing Engines assigned to "*r*default" – other Routing Engines assigned to "*rVDN"
  With Routing Engine1 and Routing Engine2 assigned to "*r*default", and Routing Engine3 and Routing Engine4 assigned to "*rVDN", as long as Routing Engines3 and 4 are running, any discrepancy in behavior, or race conditions is restricted to Routing Engines 1 and 2.

  Routing Engines 3 and 4 compete to route on "*rVDN", which causes the first Routing Engine to send a TS.Route() request to succeed, and the other to receive a failure.

Therefore, for the most part this behaves like the single “*r*default” and “*rVDN”, except that a race condition exists on every request, there is more traffic to/from the switch, and a failure is reported on every second attempt. This can an extra load on an already busy TS.

Events

When a call arrives at the adjunct route step in a vector, the switch sends a route request (C_RT_REQ) to the TS. The TS translates the route request to a TS.IncomingCall event and delivers it to the client application, which is typically a Routing Engine.

When the TS.IncomingCall.Event reaches the Workflow server, it triggers a flow that instructs the TS to route the call to another destination via a TS.Route() request. When the TS requests this work from the switch, and an ACK is received confirming the call was processed, the TS issues a TS.Divert.Event informing the Routing Engine that the call left the route point and was delivered to the destination indicated in the TS.Route().

Immediately after a TS.Divert.Event, the TS issues a TS.Disconnect.event, which informs the Routing Engine the call is no longer at the adjunct route step.

If the Routing Engine does not respond in a timely fashion, the route request will timeout, and the switch will send a route end (C_RT_END) to the TS. The TS converts the route end into a TS.Divert.Event. The difference between this event and the event created during the TS.Route() is that, in the case of a timeout, the TS does not know the final destination for this call, so the TS.Divert.Event informs a divert took place, but does not carry the destination of the call.

Event	Comment
TS.IncomingCall.Event	Route Request posted from PBX to TS.
TS.Divert.Event	Call got routed or timed out while in route point.
TS.Abandoned.Event	Call got terminated by far end while in route point
TS.Disconnect.Event	Indicator that call is no longer associated with route point.

Business Advocate

On Avaya Business Advocate (Business Advocate), the Telephony Server Adapter (TSA) assigns to the TS using the following two criteria:

- ‘*r” (route point)
- “*w” (wait device, a.k.a., parking device)

Both are of these criteria are implemented via Adjunct Routing and care should be taken when implementing hybrid systems with Business Advocate and non-Business Advocate environments.

Since Business Advocate parking devices are implemented via Adjunct Routing, factoring in a Routing Engine using “*r*default” might be convoluted and subject to failures due to switch configuration changes. The Routing Engine assigned to “*r*default” may need to be informed of parking devices in order to properly operate in case of a TSA failure. This may not present a problem on small systems, but will become a burden in large environments with large number of parking devices. More detailed information about Avaya Business Advocate is provided in *Business Advocate Configuration and Administration* .

Recommendations

Some recommendations for proper usage of “*r*default” are:

- Do not assign more than one Routing Engine to “*r*default”.
- Do not assign more than one Routing Engine to the same “*rVDN” number.
- Plan and document how to make “*r*default” and “*rVDN” work together.
- When using Business Advocate, consider a dedicated link extension/signal and TS to handle Business Advocate traffic. Avoid using “*r*default” on this link.
- Routing Engines assigning for “*r*default” will typically have flows (scripts) designed to handle a set of VDNs. Take dare when designing these flows, so that route requests arriving from “unexpected” VDNs (VDNs outside the interested set) are properly handled and calls are not dropped or incorrectly routed.
- When configuring a voice device for the TSQS in IC Manager, make sure you use a queue number with *vm as the entity to which to issue the TS.Assign. If you mistakenly enter a routing point, the TSQS thinks that routing point is a queue. This creates confusion when a call arrives after the Workflow server has issued a *r*default request for all route points. The TSQS thinks the call is in queue and the Workflow treats it like a routing point.

Nortel Meridian

The Nortel Meridian uses Controlled Directory Number (CDN) devices, which behave similar to an Avaya DEFINITY VDN in Avaya IC. However, the vectors on the Nortel Meridian are not used by Avaya IC. This functionality is handled in a table-like paradigm.

The following Nortel Meridian switches are supported on Avaya IC 6.1.3.

- Nortel Meridian-1 using Symposium Call Center Server 4.0
- Nortel Meridian-1 using Meridian Link 5.0

Route Request/Responses

A route response is issued via TS.Route() request. The Nortel Meridian also allows for music, ring back, or silence treatment for calls on a CDN. These treatments can be specified in the CDN or activated via software through TS.Route() request.

Route End

Route ends occur due to timeouts, abandonment, or link failure. However, the Nortel Meridian considers a CDN inactive and default routes all calls arriving to it, if more than 10 consecutive route requests go unanswered. The switch imposes a 4 second timeout, which can be lengthened by issuing music, ringback, or silence treatment as a first route response.

Link Failure

In the event of a link failure, the switch moves all calls on a CDN to an associated ACDDN. This is called default route.

TS Assignment

On Nortel, one has to indicate individual CDNs to monitor, i.e., the only syntax available is TS.Assign("*rCDN"). Issues on IC 6.x are comparable to those on Definity, with the obvious exception that the Nortel does not support the "*r*default" assignment criteria.

Events

The Nortel Meridian does not allow for CDN monitoring. It sends route requests to assigned applications, but does not report when calls have left a CDN. This is also true for route responses or route ends due to timeout.

Events that are reporting on a call's progress from a CDN are generated by the TS based on a timer for route ends or on return code for route selects.

A TS.Divert.Event is issued by the TS when the request to move the call is accepted by the switch, not when it actually was moved. If a call is abandoned while in a CDN or if the route request times out, the TS will not know about it and this could cause the TS to allocate more memory than necessary. To deal with this, the TS implements a mechanism to catch these conditions. The mechanism is such that upon route request arrival a structure is saved and a timer initiated. When a route select arrives, the TS clears the data structure and issues the TS.Divert.Event. If the timer expires, the TS issues a TS.Divert.Event without a destination.

Note:

This matches two conditions: abandonment or actual timeout. There is no way for the TS to distinguish between them. On the Nortel Meridian, determination of call abandonment is not possible on CDNs.

Business Advocate

On Business Advocate, the TSA assigns to the TS using the following two criteria:

- '*r" (route point)
- "*w" (wait device, a.k.a., parking device).

Both are of these criteria are implemented via Adjunct Routing and care should be taken when implementing hybrid systems with Business Advocate and non-Business Advocate environments.

More detailed information about Avaya Business Advocate is provided in *Business Advocate Configuration and Administration* .

Recommendations

There are no special recommendations for the Nortel Meridian.

Aspect CallCenter

The Aspect CallCenter 8.0 is supported on Avaya IC 6.1.3. This switch has similar behavior to the Nortel Meridian, via Portal, and the Avaya DEFINITY/Communication Manager, but because the Aspect switch has a set of 6 variables associated with the call and Call Control Tables (CCTs) that can act based on these variables, the implementation of route points is different than that used for the Avaya DEFINITY/Communication Manager or the Nortel Meridian.

The set of variables on a call are:

- subtype
- data_a
- data_b
- data_c
- data_d
- data_e

Subtype is type string and 12 bytes long. Data_e is type string and 40 bytes long. All of the variables are read/write, except subtype, which is set by the CCT, and cannot be directly modified by the TS.

The EDUID is associated with the call by storing it with data_e. In spite of the fact that CCTs can manipulate data_e, it should not be done.

Call Control Table

In order to implement Route Points, CCTs must be coded with SEND DATA and RECEIVE DATA steps. The SEND DATA step passes a CIM message to the TS. The TS examines field “subtype” and if there is an assigned client for the value in the “subtype” field, a TS.IncomingCall.Event is posted. The TS treats this as a route request.

After the SEND DATA step, the CCT steps into a RECEIVE DATA where it remains until timeout (usually 15 minutes per RECEIVE DATA), or until a 601 message is passed from the TS.

Upon receiving the Routing Engine response, the TS.Route() request is translated into a 601 message, which satisfies the RECEIVE DATA, and the call is routed in accordance to instructions in the 601 message.

Route Request/Responses

The Aspect CallCenter does not have a concept for a device associated with a route point, so the TS implements one using the “subtype” field.

The route response is issued via TS.Route() request, similar to Avaya DEFINITY and Nortel Meridian.

Note:

The Aspect CallCenter has another method, which can be used to respond to a route request: TS.ReceiveData(). This method overrides all variables associated with a call. It exists to extend to the client switch features, and for backward compatibility with eContact 5.6.

Route End

Route ends occur due to timeouts, abandonment, or link failure. The CCT should be coded so calls are default routed and treated by the contact center in some fashion (usually a direct queuing to an agent pool).

Link Failure

The CCT is not informed about a host application failure on a RECEIVE DATA step. As a consequence, calls do not continue CCT processing via “ON ERROR” code path and remain in the RECEIVE DATA step until timeout.

This would allow definition, at the CCT level, on how to behave in case of link failure. The behavior is usually to queue calls to an agent pool.

TS Assignment

The assignment for a route point on the Aspect CallCenter is somewhat different than on the Avaya DEFINITY and the Nortel Meridian because rather than a DN to assign to, there is a “subtype”, which is of type string.

The command becomes TS.Assign(“*rR001”), where “R001” is a subtype in an incoming CCT.

The issues with Avaya IC 6.1.3 assignments are similar to those on the Avaya DEFINITY, except the Aspect CallCenter does not support the “*r*default” assignment.

Events

The Aspect CallCenter does not report when a call leaves a route point. Events reporting on call progress from a route point is generated by the TS based on a timer for route ends or on return code for route selects.

A TS.Divert.Event is issued by the TS when the request to move the call is accepted by the switch, not when it actually was moved.

If a call is abandoned while in a route point or if the route request times out, the TS will not know about it, which could cause the TS to allocate more memory than necessary. The TS implements a mechanism to deal with these conditions. In this mechanism, when a route request arrives, a structure is saved and a timer initiated. When a route select arrives, the TS clears the data structure and issues the TS.Divert.Event. If the timer expires, the TS issues a TS.Divert.Event without a destination.

Note:

This matches two conditions, abandonment, or actual timeout. There is no way for the TS to distinguish between them, so determination of call abandonment on route points on the Aspect CallCenter is not possible.

Business Advocate

On Business Advocate, the TSA assigns to the TS using the following two criteria:

- ‘*r” (route point)
- “*w” (wait device, a.k.a., parking device).

Both are of these criteria are implemented via Adjunct Routing and care should be taken when implementing hybrid systems with Business Advocate and non-Business Advocate environments.

The implementation of a parking device on Business Advocate requires multiple RECEIVE DATA steps in a CCT, because each RECEIVE DATA step can wait only 15 minutes. This timeout is set at the CTI link level.

More detailed information about Avaya Business Advocate is provided in *Business Advocate Configuration and Administration* .

Recommendations

The Avaya IC implementation on the TS configured for the Aspect CallCenter switch requires a set of CCT adjustments. Among these adjustments are specific settings relative to call progress reporting via reserved subtypes. Take care to not assign to these special subtypes from Routing Engine.

For more information on the reserved subtypes used in IC 6.1.3, refer to *Installation Planning and Prerequisites*.

Chapter 4: Telephony Server

This chapter provides Telephony server configuration and administration information for each of the switches that are supported in IC 6.1.3.

This section includes the following topics:

Avaya DEFINITY/Communication Manager

This section provides information to configure and administer the Avaya TS to work with the Avaya DEFINITY 8.3, 9, 10 and Avaya Communication Manager 1.1, 1.2, 1.3.

You should configure the switch for the Avaya TS before proceeding with the steps in this section. For detailed instructions on configuring the Avaya DEFINITY/Communication Manager for the Avaya TS, refer to *Installation and Configuration*.

Important:
When configured for the Avaya DEFINITY/Communication Manager, the Avaya TS requires the latest version of the CVLAN client, which can be downloaded from the Avaya web site. The CVLAN client is not distributed with the IC 6.1.3 installation disks.

Refer to *Installation and Configuration* and *Installation Planning and Prerequisites* for specific instructions on downloading and installing the CVLAN client.

Supported platforms

The servers that support the Avaya DEFINITY/Communication Manager can be run on various operating systems for IC 6.1.3. The following table lists operating systems on which the servers can run for the Avaya DEFINITY/Communication Manager:

Operating System	Avaya TS	TSQS
Windows 2000 SP4	Yes	Yes
Windows 2003	Yes	Yes
Solaris 8	Yes	Yes
Solaris 9	Yes	Yes
AIX 5.1L ML3	Yes	Yes
AIX 5.2L	Yes	Yes

Hardphone vs. softphone capabilities

The following table summarizes the capabilities of the Avaya DEFINITY/Communication Manager by comparing its Hardphone and softphone capabilities. Events listed in the Hardphone column are supported by the TS when evoked manually through the hardphone. The events listed in the Softphone column are supported by the Avaya TS when evoked through softphone.

Note:

The Avaya DEFINITY/Communication Manager does not complete transferred calls to a busy line if the call transfer is initiated from a hardphone with the speaker phone on. To successfully complete a call transfer to a busy line on the Avaya DEFINITY/Communication Manager, you must take the hardphone off speaker mode before initiating the transfer.

Capability	Hardphone	Softphone
alternate call	yes	yes
answer call, connect	yes	yes
auto-ready (auto-in)	yes	yes
blind transfer	yes	yes
bridge two calls	yes	no
busy (aux-work)	yes	yes
conference call	yes	yes
consultative transfer	yes	yes
consultative transfer, one-step	yes	yes
forward calls, cancel forward calls	yes	no
hang up call	yes	yes
hold call	yes	yes
listen disconnect/reconnect (mute)	yes	no
login	yes[1]	yes
logout	yes[a]	yes
make call	yes	yes
make call, predictive (via Softdialer)	no	yes

Capability	Hardphone	Softphone
message waiting, cancel message waiting	yes	no
off hook	yes	no
on hook	yes	no
ready (in service/manual-in)	yes	yes
reconnect held call	yes	yes
send all calls, cancel send all calls	yes	no
send DTMF tones	yes	yes
service observing	yes	yes
single-step conference[1]	yes	no
third party drop	no	yes
walk away, return from walk-away	yes	no
wrap-up (after-call work)	yes	yes

1. Not supported if done by the hardphone.

Telephony server configuration

The Avaya TS is configured on the Server Editor of IC Manager. This section provides the the required and optional parameters that must be set to configure the Avaya TS for the Avaya DEFINITY/Communication Manager.

For information on adding and configuring servers on Avaya IC using IC Manager, refer to *IC Administration Volume 1: Servers & Domains*.

Enter or modify the following parameters on the **TS** tab of the **Server Editor**. Right mouse click on the screen and select the **Show Advanced Properties** option to display all of these parameters.

Field	Recommended entry	Notes
ACD Name	Select the name of the ACD assigned to the Avaya switch.	This is the name of the ACD that this TS is serving from a pick list of names assigned to the ACD during system configuration.
ACD Type	Select Avaya	The type of ACD with which the TS will communicate.
ACD Model	Select Definity	The model of the ACD that corresponds to the selected ACD Type.
ACD Protocol	Select asai	The protocol to be used between the TS and the ACD. If your system includes a Legacy TS with switch with Avaya DEFINITY software, see *Migration Guide*.
Site	Select the site of your TS.	Select the site that this server is associated with. The TS uses this information to retrieve the queues for internal monitoring.
ACD Link	Enter the IP address (or a name if it can be resolved into an IP) of the MAPD card set. Maximum length is 32.	The ACD Link is the device through which the Avaya TS communicates with the ACD.
Signal Number	Specify a signal number when configuring multiple Avaya users on a single machine. There is no default value in the TS, but IC Manager sets this value to 1. Maximum length is 8.	The signal extension number of the ASAI line associated with each TS. The signal number is mandatory, if it is not specified, the TS will not be able to establish the link between users.

Field	Recommended entry	Notes
Advanced Properties		
Enable Call Containers	Check to create call containers for the TS. Default is checked.	If checked, the TS creates call containers to store information about the different legs of calls.
Enable Agent Containers	Check to turn on agent containers for the TS. Default is checked.	If checked, the ADU containers for the TS are turned on.
Use 5.6 State Fields	Check to give containers for agent states entries in the 5.6 style. Default is unchecked.	Example, ts.loginid. For more information, refer to the *Telephony Connectors Programmer Guide*.
Use 6.0 State Fields	Check to give containers for agent states entries in the 6.0 style. Default is checked.	Example, voice.loginid. For more information, refer to the *Telephony Connectors Programmer Guide*.
Wrap up by Client	Check to have the TS wait for the wrap-up process to be completed before removing the call information from memory. Default is unchecked.	If unchecked, the TS may remove the call information from memory before wrap-up is complete.
Wrap up Client Time to Live (min)	Enter the period of time, in minutes, that the TS waits for the wrap-up process to be completed by the client. Default is 15.	If unchecked, the TS issues a request for wrap up on behalf of the client.
Wrap up Server Time to Live (min)	Enter the period of time, in minutes, that the TS waits for the wrap-up process to be completed by the server. Default is 2.	If unchecked, the TS issues a request for wrap up on behalf of the server
Call Plan	Enter the number of digits on the external extension numbers used by the contact center. Default is 6.	This helps identify the call as internal to the switch.
Thread Pool Size	Enter the number of threads to allocate for the server thread pool. Default is 20.	

Field	Recommended entry	Notes
Queues Owned	Enter the name of the queue or queues for which the TS is responsible for creating queue ADUs.	The TSQS should not be associated with this queue in any way.
Default ANI	Enter the default ANI value, for incoming calls, which did not carry ANI information.	
Enable Avaya Soft Dialer	Check to configures the switch to use the Soft Dialer. Default is checked.	The Soft Dialer is used for outbound dialing.
Abort on Link Down	Check to abort (stop) the TS if the link goes down. Default is checked.	
Call Record Time to Live	Enter the maximum period of time, in hours, that a CtsCallRecord can remain in memory before the TS cleans it up. Default is 24.	A TpDisconnected event is generated for this record.
Enable Reason Codes	Check to configure the ACD for reason codes. Default is unchecked.	The application can use reason codes for agent logouts and agent changeState to Busy.
Default Aux Reason Codes	Enter the code to use when the agent does not enter a code when changing their state to Busy. Default is 0.	This is only used if reason codes are enabled.
Dial by Equipment	Check to determine if the ACD requires the destination to be reached by equipment. Default is unchecked.	If you are using Avaya Business Advocate, this field must be checked.
ACD Mode	Select the ACD mode that is compatible with the settings of the switch. Default is EAS.	This field must be set to EAS for the TS to support CVLAN Agent Events. If this field is set to ACD, the TS will not receive agent events.
ACD Version	Enter the version number of the switch software.	For example: Enter 10 for Avaya DEFINITY version 10.
PDK Login	Enter the PDK login ID.	The login ID that enables the agent to integrate with the predictive outbound dialing feature.

Field	Recommended entry	Notes
RONA Divert	Check to enable RONA (Redirect on No Answer). Default is checked.	This tells the TS that the switch is capable of handling a call deflection (divert) on an alerting device or if the call needs to be answered before it can be transferred or conferenced to another destination.
Ccti Trace Level	Select the desired log level for Ccti. Default is info.	Valid values are debug, info, warning, and error layer.
Cpbx Trace Level	Select the desired log level for Cpbx. Default is info.	Valid values are debug, info, warning, and error layer.
TSV5 Trace Level	Select the desired log level for TSV5. Default is info.	Valid values are debug, info, warning, and error layer.
DTMF Tone Duration	Enter the duration period of each tone. Default is 35 seconds	This lets you specify the length of each tone in .01 second increments. This value can be anything between 1 and 35 seconds.
DTMF Pause Duration	Enter the duration of each pause. Default is 10 seconds.	This lets you specify the length of the pause between tones in .01 second increments. This value can be anything between 1 and 35 seconds.

Configuration tab

The following configuration parameter is not presented on the TS tab in IC Manager. Set this parameters on the Configuration tab:

Field	Recommended entry	Notes
allow_names_mixed_case	Enter "true" to enable processing of queue names in the case that comes in. If set to "false", queue names will be set to lower case.	Values: true or false.
blockedani	Enter the ANI number you want to display if the ANI currently displays as ***** because the caller has caller ID blocked.	If an ANI is not present, translates to the default ANI specified in the Default ANI parameter. If the ANI is ***** or #####, translates to this value.
cutthrough_waittime	Enter the number of seconds the TS will wait for an event after it receives a C_CUT_THROUGH event. Once this time expires, the TS assumes the call has been answered.	Values: 0 - 30 (seconds).

Agent configuration

The following agent configuration parameters are set on IC Manager when adding an agent to Avaya IC. These parameters relate to the switch parameters set on the Avaya DEFINITY/Communication Manager. For information on configuring agents in hunt groups, refer to Defining hunt and skill groups on page 341. The table lists the label used when configuring with IC Manager, followed in parentheses by the parameter name as it is required internally by the Avaya TS.

Label and Internal Name	Description
Phone ID (phone)	For EAS agents, this is the agent's login ID. For non-EAS, this can be any identifier for the agent. For a direct connection (one without any queue involvement) this is the physical teleset extension.
Password (phone_ passwd)	For EAS, password to log in phone. This can be supplied at TS.Login time. Not used for non-EAS agents.
Phone Type (phone_type)	Agent work mode. ACD signifies non-EAS. If empty, a direct phone (one with no queue involvement) is assumed.
Equipment (equipment)	Agent station extension. If used, the agent does not have to enter an equipment number when logging in. This number can be overridden at log in time.

Note:

IC 6.1.3 does not support pending work (same as Preset) mode changes and Agent Events if it is used with an Avaya DEFINITY G3 v6.4 or earlier. To take advantage of the pending work mode changes and Agent Events capabilities, you must run your Avaya TS with an Avaya DEFINITY G3, v7.1 or later.

Preset States

While in call, agents can preset the state to which they will transition when the call ends to Busy or Wrapup. The agent can also decide to change this preset state while still in the call. For example, when the call begins the agent presets the state to Busy. After being in the call, the agent knows there will be after call work to do related to the call. The agent can then change the preset to Wrapup to be able to perform this after call work. The agent cannot back out of a preset during a call. That is, once the agent has set preset, that agent can only go to a Busy or Wrapup state after the call. The agent can go to another state like Ready from the preset state once the call is over.

Agent events

Agent state changes on the TS are not synchronously transmitted over the CVLAN interface to CTI applications. This can result in the softphone and the CTI applications' agent states getting "out of sync" with the agent states on the hardphone. This creates confusion for the agent and can cause incorrect call routing on non-voice interactions in a multi-channel environment.

The TS has been enhanced to support CVLAN Agent Events to keep the agent state on the softphone in sync with the agent state on the hardphone. The TS only receives agent events after an agent's softphone has assigned to the TS. The TS does not get agent events, such as a login entry, if the agent has only logged into the hardphone.

Important:
The Preset phone state must be in sync on the hardphone and the softphone to support CVLAN Agent Events.
The TS must be configured in EAS mode (not ACD mode) as described in ACD Mode on page 53 to support CVLAN Agent Events.

When the Avaya DEFINITY TS is enabled for Agent Events:

1. Agent state changes (such as login, logout, busy, ready, readyauto, and after call work) done by the agent over the hardphone are reported to the ACD.
2. The ACD sends events for these agent state changes to the Avaya TS.
3. The TS generates and sends corresponding events to the softphone to keep it in sync with the events sent to the ACD by the hardphone.
4. The Avaya TS updates the EDU and the ADU with agent state changes.

Note:
The person responsible for the installation and administration of the Core Services on Avaya IC must have root permission for Unix machines and Administrator permission for Windows-based machines on the machine where the TS is installed when they use the Configuration tool to do the following.

- Add and configure the Telephony server
- Create and configure the web applications for the website, Email Template Administration, Web License Manager, and Letter Generator

Prerequisites for agent events

To enable Agent Events on an Avaya TS running with an Avaya DEFINITY switch, install the following prerequisites on IC 6.1.3.

- MAPD CVLAN server Release 8.2.1 or higher
- CVLAN Client Release 8.2.5 or higher
- The signal on the MAPD must be configured as an ADJLK (not ASAI) and associated with a port (ASAI station) in the Avaya DEFINITY switch.

For more detailed information about these requirements, consult the administrator responsible for the Avaya DEFINITY/Communication Manager at your site.

Aspect CallCenter

This section provides information necessary to configure and administer the Avaya TS to work with the Aspect CallCenter (version 8.x).

You should configure the Aspect switch for the Avaya TS before proceeding with the steps in this section. For instructions on configuring the Aspect CallCenter switch for the Avaya TS, refer to *Installation and Configuration*.

Software prerequisites

The following table lists the Aspect software that is required for the Avaya TS and TSQS to work properly with the Aspect CallCenter switch:

Software	Purpose
Aspect CMI server version 4.X	Installed and available via an Ethernet connection. Enables the Avaya TS to establish a connection to the Aspect CallCenter System. The Avaya TS uses no external libraries.
Aspect RealTime Data Server	Installed and available via an Ethernet connection. Enables the TSQS to gather statistics from the Aspect CallCenter System.
Aspect RealTime Receiver Custom Control (RealTime Runtime version)	Install on the same machine as the TSQS.

Refer to the *Installation Planning and Prerequisites* for more detailed information about IC 6.1.3 prerequisites.

Supported platforms

The servers that support the Aspect CallCenter System version 8 can be run on various operating systems for IC. 6.1.3. The following table lists operating systems on which the IC servers can run for the IC Aspect CallCenter:

Operating System	Avaya TS	TSQS
Windows 2000 SP4	Yes	Yes
Windows 2003	Yes	Yes

Hardphone vs. softphone capabilities

The following table summarizes the capabilities of the Aspect CallCenter switch by comparing its hardphone (manual) and softphone (telephony) capabilities. Events listed in the Hardphone column are supported by the Aspect version of the Avaya TS when evoked manually through the hardphone. Events listed in the Softphone column are supported by the Aspect version of the Avaya TS when they are evoked through the softphone.

Capability	Hardphone	Softphone
alternate call	yes	yes
announce call transferred thru CCT	yes	no
answer call, connect	yes	yes
blind transfer (trunk calls only)	no	yes
bridge two calls	yes	no
busy (aux-work)	no	yes
conference call	yes	yes
consultative transfer	yes	yes
dial an agent supergroup	yes	no
dial a network access code	yes	no
hang up call	yes	yes
help	yes	no
hold call	yes	yes

Capability	Hardphone	Softphone
last number dialed	yes	no
listen to customer messages	yes	no
listen to mailbox messages	yes	no
login	yes	yes
logout	yes	yes
make call	yes	yes
make call, predictive	no	yes (via Soft Dialer, Aspect 8 only)
microphone	yes	no
monitor and control DN	no	yes
network transfer using inband signaling	no	yes, via MSHS
notepad	yes	no
off hook	yes	no
on hook	yes	no
play back message while recording	yes	no
ready (in service/manual-in)	yes	yes
reconnect held call	yes	yes
record data	yes	no
record mailbox greeting	yes	no
record name	yes	no
repeat message	yes	no
report emergency call	yes	no
report line problems	yes	no
return customer call	yes	no
review mailbox name or greeting	yes	no
review message before sending	yes	no
send DTMF tones	yes	no

Capability	Hardphone	Softphone
send mailbox message to another agent	yes	no
send a message to another extension	yes	no
speaker phone	yes	no
supervisor	yes	no
third-party drop	yes	no
view statistics	yes	no
wrapup (after call work)	yes	yes

Note:

The Avaya TS configured for the Aspect CallCenter switch does not support "chained" consultation scenarios. If an agent initiates a consultative transfer with a second agent, who answers the call before the transfer is complete, then that second agent cannot transfer this call to a third agent. In effect, the second agent cannot transfer a call that was an uncompleted transfer from the first agent.

Telephony server configuration

The Avaya TS is configured on the Server Editor of IC Manager. This section provides the parameters that must be set to configure the Avaya TS for the Aspect CallCenter 8 via Portal, not Event Bridge.

For information on adding and configuring servers on Avaya IC using IC Manager, refer to *IC Administration Volume 1: Servers & Domains*.

Enter or modify the following parameters on the **TS** tab of the **Server Editor**. Right mouse click on the screen and select the **Show Advanced Properties** option to display all of these parameters.

Field	Recommended entry	Notes
ACD Name	Select the name of the ACD assigned to the Aspect switch.	This is the name of the ACD that this TS is serving from a pick list of names assigned to the ACD during system configuration.
ACD Type	Select Aspect	The type of ACD with which the TS will communicate.
ACD Model	Select Aspect8	The model of the ACD that corresponds to the selected ACD Type.
ACD Protocol	Select AspectCMI	The protocol to be used between the TS and the ACD. Avaya IC automatically selects AspectCMI when ACD Type of Aspect is selected.
Site	Select the site of your TS.	Select the site that this server is associated with. The TS uses this information to retrieve the queues for internal monitoring.
Aspect Contact Server Host	Enter the IP address of the machine that hosts the Aspect Contact server.	
Data Interlink Number	Enter the Data Interlink number.	The Data Interlink number from the Aspect ACD thE TS uses to communicate with the ACD.
Aspect Header	Enter the link definition on the Aspect switch.	If this field is empty, the server uses the machine name of the ACD.

Field	Recommended entry	Notes
Device	Enter the TCP/IP port where the TS needs to create a connection. Default is 7,046.	For a list of default port numbers for components in the Avaya IC suite, see the *Installation and Configuration*.
Monitored Agent Group	Enter the number assigned to the agent group to monitor. Default is 1.	Do not complete this field if the TS is monitoring all the agent groups on Avaya IC.
Monitored Trunk Group	Enter the number assigned to the trunk group to monitor. Default is 1.	Do not complete this field if the TS is monitoring all the trunk groups on Avaya IC.
Monitor All Agent Groups	Check this box if you want the TS to monitor all agent groups. Default is unchecked.	This overrides the entry in the Monitored Agent Group field.
Monitored Super Agent Group	Enter the number assigned to the super agent group to monitor. Default is 1.	Leave this field blank if you do not want the TS to monitor the super agent group.
Monitor All Trunk Groups	Check this field if you want the TS to monitor all trunk groups. Default is unchecked.	This overrides the entry in the Monitored Trunk Group field.
Blind Transfer CCT	Enter the Call Control Table number used to perform blind transfers.	
Make Call CCT	Enter the Call Control Table number used to make calls within the Aspect switch.	
External Calls CCT	Enter the Call Control Table number used to make calls external to the Aspect switch.	
Route CCT	Enter the Call Control Table number used to route calls external to the Aspect switch.	
Transfer CCT	Enter the Call Control Table number used during call transfers.	
Transfer Init CCT	Enter the Call Control Table number used during transfer init operations.	

Field	Recommended entry	Notes
Predictive CCT	Enter the Call Control Table number used during predictive operations.	
Advanced Properties		
Enable Call Containers	Check to create call containers for the TS. Default is checked.	If checked, the TS creates call containers to store information about the different legs of calls.
Enable Agent Containers	Check to turn on agent containers for the TS. Default is checked.	If checked, the ADU containers for the TS are turned on.
Use 5.6 State Fields	Check to give containers for agent states entries in the 5.6 style. Default is unchecked.	Example, ts.loginid. For more information, refer to the *Telephony Connectors Programmer Guide*.
Use 6.0 State Fields	Check to give containers for agent states entries in the 6.0 style. Default is checked.	Example, voice.loginid. For more information, refer to the *Telephony Connectors Programmer Guide*.
Wrap up by Client	Check to have the TS wait for the wrap-up process to be completed before removing the call information from memory. Default is unchecked.	If unchecked, the TS may remove the call information from memory before wrap-up is complete.
Wrap up Client Time to Live (min)	Enter the period of time, in minutes, that the TS waits for the wrap-up process to be completed by the client. Default is 15.	If unchecked, the TS issues a request for wrap up on behalf of the client.
Wrap up Server Time to Live (min)	Enter the period of time, in minutes, that the TS waits for the wrap-up process to be completed by the server. Default is 2.	If unchecked, the TS issues a request for wrap up on behalf of the server
Call Plan	Enter the number of digits on the external extension numbers used by the contact center. Default is 6.	This helps identify the call as internal to the switch.

Field	Recommended entry	Notes
Thread Pool Size	Enter the number of threads to allocate for the server thread pool. Default is 20.	
Queues Owned	Enter the name of the queue or queues for which the TS is responsible for creating queue ADUs.	The TSQS should not be associated with this queue in any way.
Default ANI	Enter the default ANI value, for incoming calls, which did not carry ANI information.	
Abort on Link Down	Check to abort (stop) the TS if the link goes down. Default is checked.	
Allow Unregistered Subtype	Check to provide backwards compatibility with the old data synchronization mechanism. Default is checked.	
Application Data	Check to indicate this switch is capable of persistent application data. Default is unchecked.	
Aspect Header Upper	Check to post the aspect_header to the switch in upper case format. Default is unchecked.	
Call Record Time to Live	Enter the maximum period of time, in hours, that a CtsCallRecord can remain in memory before the TS cleans it up. Default is 24.	A TpDisconnected event is generated for this record.
Monitor Route Point	Check to enable the switch to monitor route points internally. Default is unchecked.	
Pdk Login	Enter the PDK login ID.	The login ID that enables the agent to integrate with the predictive outbound dialing feature.

Field	Recommended entry	Notes
RONA Divert	Check to enable RONA (Redirect on No Answer). Default is unchecked.	This tells the TS that the switch is capable of handling a call deflection (divert) on an alerting device or if the call needs to be answered before it can be transferred or conferenced to another destination. On the Aspect TS, the Route CCT is used to handle route events in place of divert.
Request Timeout (sec)	Enter the period of time, in seconds, for the TS to wait for a response after posting a request to the switch. Default is 15.	
Route Point Time to Live (sec)	Enter the number of seconds a call can remain on an unmonitored route point before the TS cleans it up. Default is 5.	After this specified period of time, a TpDisconnect event is posted to the call. Advocate users should set this value as high as possible (999) to provide ample time to deliver the call from the queue to the agent.
TSV5 Trace Level	Select the desired log level for TSV5. Default is info.	Valid values are debug, info, warning, and error layer.
Ccti Trace Level	Select the desired log level for Ccti. Default is info.	Valid values are debug, info, warning, and error layer.
Cpbx Trace Level	Select the desired log level for Cpbx. Default is info.	Valid values are debug, info, warning, and error layer.
EDU Updates	Check to allow EDU updates. Default is checked.	EDU updates are triggered by CCT SendData/SendConnect commands.

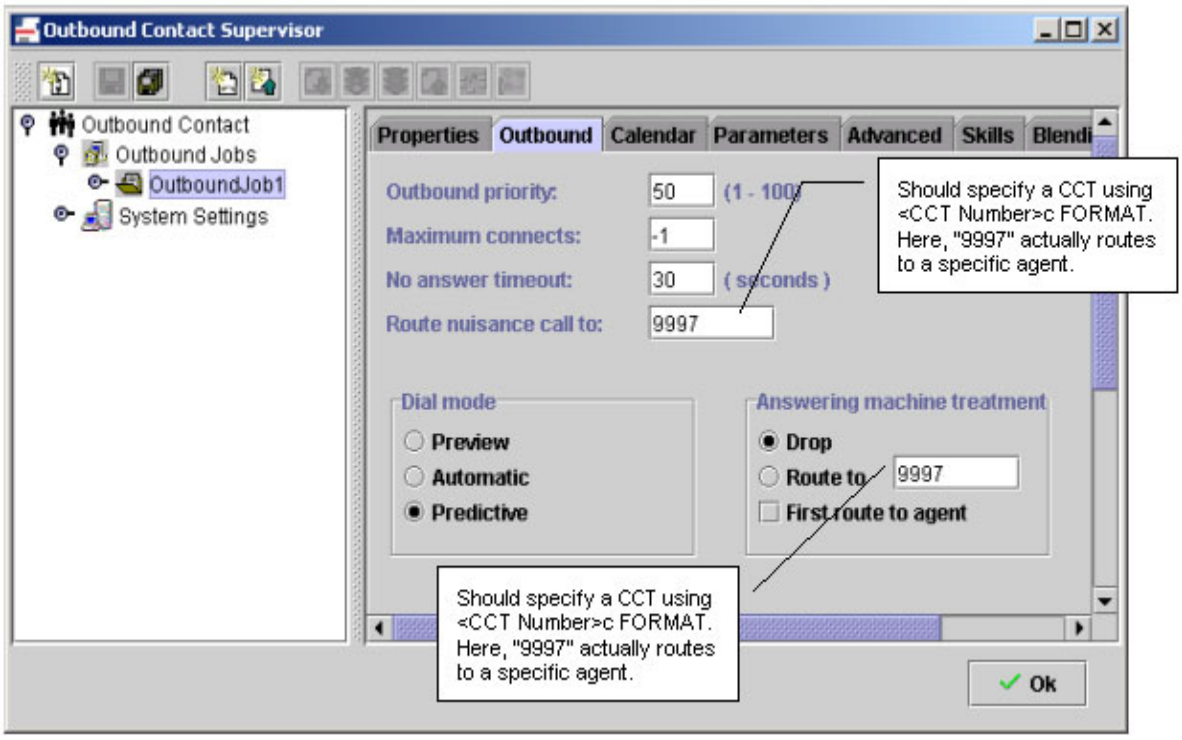
Configuration tab

The following configuration parameter is not presented on the TS tab in IC Manager. Set this parameters on the Configuration tab:

Field	Recommended entry	Notes
allow_names_mixed_case	Enter "true" to enable processing of queue names in the case that comes in. If set to "false", queue names will be set to lower case.	Values: true or false.
digits_in_data_x	Enter a single character value to indicate the Aspect data variable in which the dialed digits number string will be placed during Make Calls and Blind Transfers. Use A to allow 20 characters for the DNIS value. Default is D.	Values: A to D

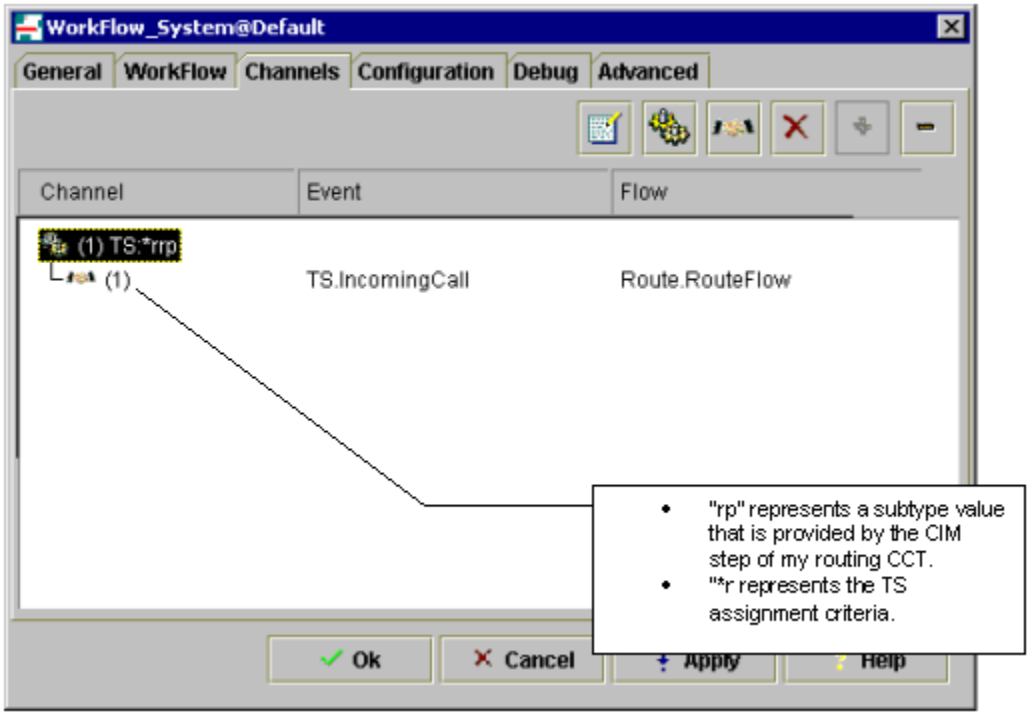
Outbound Supervisor configuration

You must set certain configuration parameters to use the Outbound Supervisor with the Aspect version of the Avaya TS. Set the parameters in IC Manager as they are displayed in the following example using the instructions provided in *IC Administration Volume 1: Servers & Domains*.

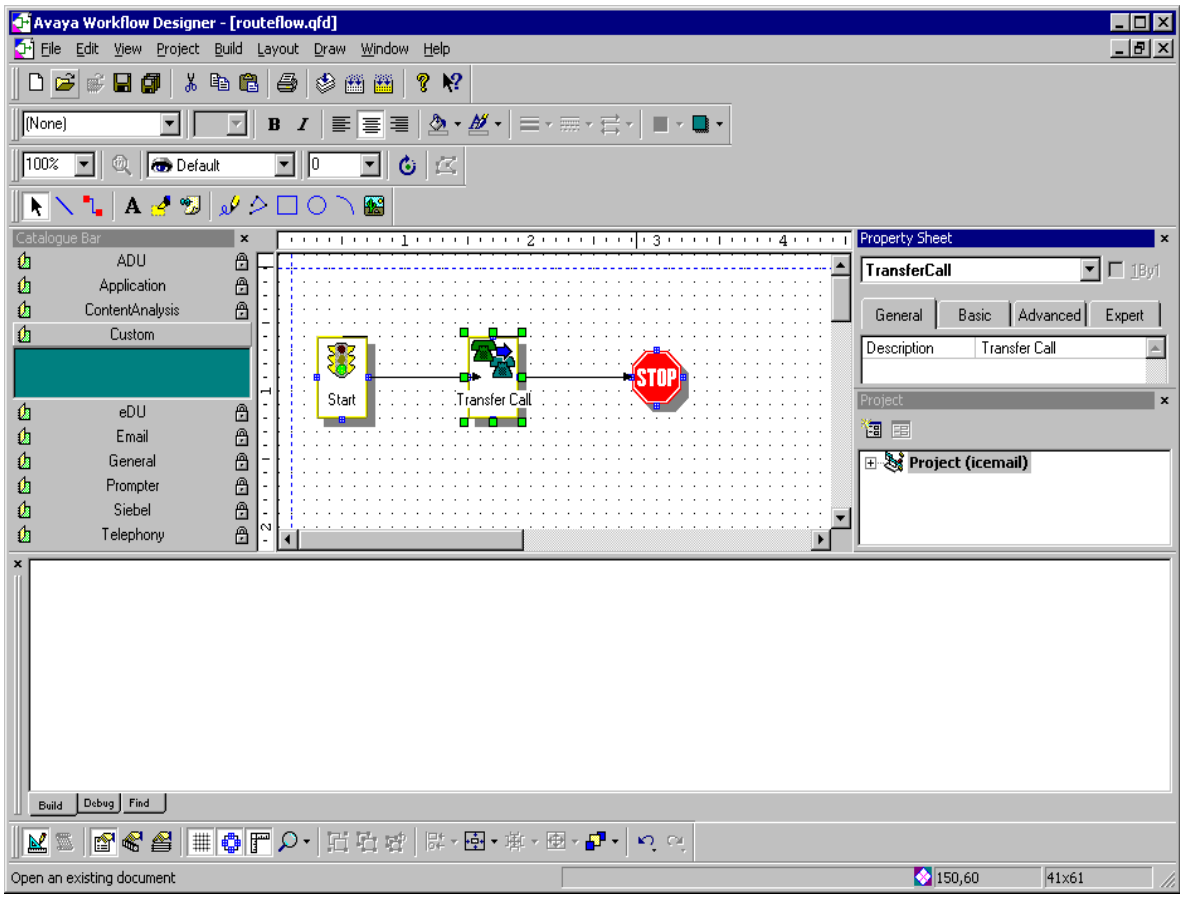


IC Workflow considerations

Configure the workflow in the Avaya Workflow Designer as illustrated in the following example using the instructions provided in the *Avaya Workflow Designer User Guide*. You must set “pbxType” workflow to type “Lucent” for the Aspect version of the Avaya TS because the Aspect version of the Avaya TS performs host-based-routing like an Avaya G3 switch



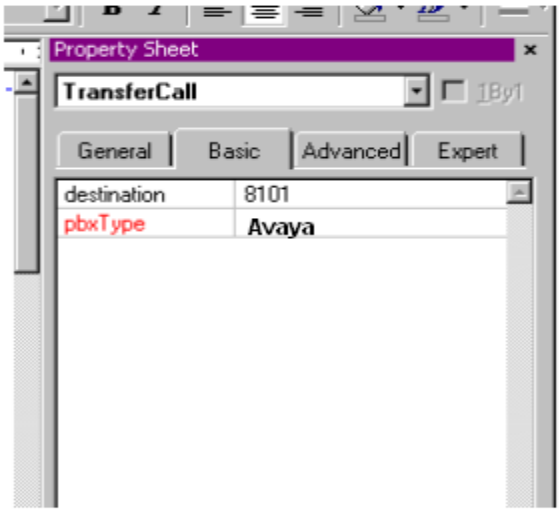
The following series of illustrations from the Workflow Designer shows a sample route flow for processing a call arriving at a routing CCT.



Basic Properties tab

Set the “pbxType” workflow to type “Avaya” at the Basic tab of the Property Sheet frame because the Aspect TS performs host-based-routing like an Avaya DEFINITY switch.

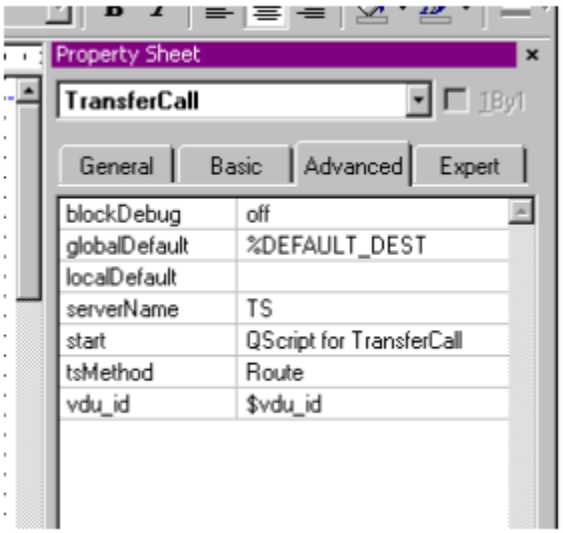
Basic Properties Tab



Advanced Properties tab

Set the properties at the Advanced tab of the Property Sheet frame to those in your system that match the properties in the following example. The tsMethod properties must be set to Route.

Advanced Properties Tab



Nortel Meridian-1: Symposium Call Center Server

This section chapter provides the information necessary to configure the Avaya TS to work with the Nortel Meridian-1 switch using Symposium Call Center Server. Symposium Call Center Server acts as bridge between the Nortel Meridian-1 switch and the Avaya TS. The Avaya TS communicates with Symposium Call Center Server via Intel NetMerge Call Processing software (formerly CT Connect). The Symposium Call Center Server sends the information that it receives from the Avaya TS to the Meridian-1 switch using MLS communication protocol.

You should configure the switch and the Intel NetMerge Call Processing software for the Avaya TS before proceeding with the steps in this section.

- For instructions on configuring the Meridian-1 switch using Symposium Call Center Server for the Avaya TS, refer to *Installation and Configuration*.
- For instructions on configuring the Intel NetMerge software, refer to the documentation provided with your Intel NetMerge Call Processing software.

Refer to *Installation Planning and Prerequisites* for more detailed information about software requirements.

This chapter also compares the capabilities of the switch during manual operation with those of the switch when telephony is enabled.

Supported platforms

You can run the IC servers that support Symposium Call Center Server on various operating systems in IC. 6.1.3. The following table lists operating systems on which the IC servers can be run for the IC Symposium Call Center Server:

Operating System	Avaya TS	TSQS
Windows 2000 SP4	Yes	Yes
Windows 2003 Server	Yes	Yes

Hardphone vs. softphone capabilities

The following table summarizes the capabilities of the Meridian-1 switch using Symposium Call Center Server and the MLS protocol by comparing its hard phone (manual) and softphone (telephony) capabilities. The events listed in the Hardphone column are supported by the TS when evoked manually through the hardphone. The events listed in the Softphone column are supported by the TS when they are evoked through the softphone.

The Meridian-1 switch does not support the transfer of a conference call. If you want to replace a party on a conference call with a more appropriate second party, you cannot transfer the conference call to that second party. You must conference the second party into the conference call;

Capability	Hardphone	Softphone
alternate call	no	no
answer call, connect[1]	yes	yes
auto-ready (auto-in)	yes	yes
blind transfer	yes	yes
bridge two calls	no	no
busy (aux-work)	yes	yes
conference call	yes	yes
consultative transfer	yes	yes
forward calls, cancel forward calls	yes	no
hang up call	yes	yes
hold call[2]	yes	yes
listen disconnect/reconnect (mute)	no	no
login	yes	yes
logout	yes	yes
make call	yes	yes
message waiting, cancel message waiting	yes	no
off hook	yes	no
on hook	yes	no

Capability	Hardphone	Softphone
reconnect held call	yes	yes
send all calls, cancel send all calls	yes	no
wrap-up (after-call work)	no	yes simulated by the TS

1. You can resolve a “Bad Object State” error message from the Softphone on Symposium Call Center Server by turning on Security Option.
2. You can resolve a “Bad Object State” error message from the Softphone on Symposium Call Center Server by turning on Security Option. The Meridian-1 switch does not support an agent initiating a new call while the current call is on hold.
   You cannot initialize a conference or a transfer on the hardphone, but once a conference or transfer is initiated on the softphone, you can complete it on the hardphone.

Important:

Makecall, Conference, and Transfer can be initiated from hard phone, but the Avaya TS is not informed about call progress by the switch. The softphone is not synchronized with the hardphone. Workaround: Do not use a mixture of the hardphone & softphone with Nortel switches. If you start a conference on the hardphone, you must finish it on the hardphone. For Nortel switches, the controlling party on a consultative call will have a phantom line appearance in the event that the caller hangs up. The switch attempts to re-synchronize the phone set if the agent uses the softphone. Actions through the hardphone are not posted by the switch to the TS, so the hardphone and the softphone get out of sync. In this scenario, in spite of the fact that the voice path is gone, the TS will attempt remove the line appearance and re-synchronize the hardphone and the softphone if the softphone is used to terminate the existing line appearance via TS.HangupVDU().

Note:

The Avaya TS configured for the Nortel Meridian switch does not support “chained” consultation scenarios. If an agent initiates a consultative transfer with a second agent, who answers the call before the transfer is complete, then that second agent cannot transfer this call to a third agent. In effect, the second agent cannot transfer a call that was an uncompleted transfer from the first agent.

Telephony server configuration

The Avaya TS is configured on the Server Editor of IC Manager. This section provides the the required and optional parameters that must be set to configure the Avaya TS for the Meridian-1 switch using Symposium Call Center Server.

For information on adding and configuring servers on Avaya IC using IC Manager, refer to *IC Administration Volume 1: Servers & Domains*.

Enter or modify the following parameters on the **TS** tab of the **Server Editor**. Right mouse click on the screen and select the **Show Advanced Properties** option to display all of these parameters.

Field	Recommended entry	Notes
ACD Name	Select the name of the ACD assigned to the switch.	This is the name of the ACD that this TS is serving from a pick list of names assigned to the ACD during system configuration.
ACD Type	Select Nortel	The type of ACD with which the TS will communicate.
ACD Model	Select Symposium	The model of the ACD that corresponds to the selected ACD Type.
ACD Protocol	Select MLS.	This is the protocol used by the Symposium Server.
Site	Select the site of your TS.	Select the site that this server is associated with. The TS uses this information to retrieve the queues for internal monitoring.
Link	The logical identifier in the server where NetMerge is running.	The Link establishes a path between a NetMerge client and a given PBX interface. The logical identifier is established in the NetMerge server configuration program.
Node	Enter the name of the server where NetMerge is running.	The Node needs to translate to an IP address via DNS or host table.
Advanced Properties		
Enable Call Containers	Check to create call containers for the TS. Default is checked.	If checked, the TS creates call containers to store information about the different legs of calls.

Field	Recommended entry	Notes
Enable Agent Containers	Check to turn on agent containers for the TS. Default is checked.	If checked, the ADU containers for the TS are turned on.
Use 5.6 State Fields	Check to give containers for agent states entries in the 5.6 style. Default is unchecked.	Example, ts.loginid. For more information, refer to the *Telephony Connectors Programmer Guide*.
Use 6.0 State Fields	Check to give containers for agent states entries in the 6.0 style. Default is checked.	Example, voice.loginid. For more information, refer to the *Telephony Connectors Programmer Guide*.
Wrap up by Client	Check to have the TS wait for the wrap-up process to be completed before removing the call information from memory. Default is unchecked.	If unchecked, the TS may remove the call information from memory before wrap-up is complete.
Wrap up Client Time to Live (min)	Enter the period of time, in minutes, that the TS waits for the wrap-up process to be completed by the client. Default is 15.	If unchecked, the TS issues a request for wrap up on behalf of the client.
Wrap up Server Time to Live (min)	Enter the period of time, in minutes, that the TS waits for the wrap-up process to be completed by the server. Default is 2.	If unchecked, the TS issues a request for wrap up on behalf of the server
Call Plan	Enter the number of digits on the external extension numbers used by the contact center. Default is 6.	This helps identify the call as internal to the switch.
Thread Pool Size	Enter the number of threads to allocate for the server thread pool. Default is 20.	
Queues Owned	Enter the name of the queue or queues for which the TS is responsible for creating queue ADUs.	The TSQS should not be associated with this queue in any way.

Field	Recommended entry	Notes
Default ANI	Enter the default ANI value, for incoming calls, which did not carry ANI information.	
Abort on Link Down	Check to abort (stop) the TS if the link goes down. Default is checked.	
Application Data	Check to indicate this switch is capable of persistent application data. Default is checked.	
Call Record Time to Live	Enter the maximum period of time, in hours, that a CtsCallRecord can remain in memory before the TS cleans it up. Default is 24.	A TpDisconnected event is generated for this record.
Logout Before Login	Check to log the user out before attempting to login. Default is checked	
Dial by Equipment	Check to determine if the ACD requires the destination to be reached by equipment. Default is checked.	If you are using Avaya Business Advocate, this field must be checked.
Make Busy After Logout	Check to set the phone set to the MakeBusy phone state after logout is concluded. Default is unchecked.	
Monitor Route Point	Check to indicate this switch is capable of monitoring route points internally. Default is unchecked.	
Route Point Time to Live (sec)	Enter the number of seconds a call can remain on an unmonitored route point before the TS cleans it up. Default is 5.	After this specified period of time, a TpDisconnect event is posted to the call. Advocate users should set this value as high as possible (999) to provide ample time to deliver the call from the queue to the agent.

Field	Recommended entry	Notes
Suppress Logout Event	Check to ignore the logout event generated when the agent presses “Logout” on the hard phone. Default is unchecked.	This setting provided backward compatibility with the TS Meridian, version 5.1. It also allows for Multiple Queue Assignment support because the agent can logout from the hard phone without terminating the softphone session. IMPORTANT: When this parameter is set to true, the TS also suppresses logout events that are caused by an agent logging out of the hardphone. In this case, the softphone and hardphone become out of sync.
RONA Divert	Check to enable RONA (Redirect on No Answer). Default is unchecked.	This tells the TS that the switch is capable of handling a call deflection (divert) on an alerting device or if the call needs to be answered before it can be transferred or conferenced to another destination.
TSV5 Trace Level	Select the desired log level for TSV5. Default is info.	Valid values are debug, info, warning, and error layer.

Configuration tab

The following configuration parameter is not presented on the TS tab in IC Manager. Set this parameters on the Configuration tab:

Field	Recommended entry	Notes
allow_names_mixed_case	Enter "true" to enable processing of queue names in the case that comes in. If set to "false", queue names will be set to lower case.	Values: true or false.

Meridian-1: Meridian Link

This section provides the information necessary to configure the Avaya TS to work with the Nortel Meridian-1 switch using Meridian Link. The Meridian Link acts as a bridge between the Avaya TS and the Meridian-1 switch. The Avaya TS sends data to the Meridian Link via Intel NetMerge Call Processing software (formerly CT Connect). The Meridian Link sends this information to the Meridian-1 switch using the MLP communication protocol.

You should configure the switch and the Intel NetMerge software for the Avaya TS before proceeding with the steps in this section.

- For instructions on configuring the Meridian-1 switch using Meridian Link for the Avaya TS, refer to *Installation and Configuration*.
- For instructions on configuring the Intel NetMerge software, refer to the documentation provided with your Intel NetMerge Call Processing software.

Refer to *Installation Planning and Prerequisites* for more detailed information about software requirements.

Note:

If you are using the Generic ACD to connect to the Meridian-1 switch via Meridian Link over the MLP protocol, the vtel.ini file should indicate the Avaya DEFINITY/Communication Manager. The Generic ACD is configured to operate the Meridian-1 switch for the DEFINITY switch specified in the vtel.ini file.

This chapter also compares the capabilities of the switch during manual operation with those of the switch when telephony is enabled.

Supported platforms

The IC servers that support Meridian Link can be run on various operating systems in IC. 6.1.3. The following table lists operating systems on which the IC servers can be run in a Meridian Link environment:

Operating System	Avaya TS	TSQS
Windows 2000 SP4	Yes	Yes
Windows 2003	Yes	Yes
Solaris 8	Yes	No
Solaris 9	Yes	No

Hardphone vs. softphone capabilities

The following table summarizes the capabilities of the Meridian-1 switch, using Meridian Link and the MLP communication protocol by comparing its hard phone (manual) and softphone (telephony) capabilities. Those events listed in the Hardphone column are supported by the TS when evoked manually through the hardphone. The events listed in the Softphone column are supported by the TS when they are evoked through the softphone.

Meridian Link does not support the transfer of a conference call. If you want to replace a party on a conference call with a more appropriate second party, you cannot transfer the conference call to that second party. You must conference the second party into the conference call.

Capability	Hardphone	Softphone
alternate call	yes	yes
answer call, connect	yes	yes
auto-ready (auto-in)	yes	yes
blind transfer	yes	yes
bridge two calls	no	no
busy (aux-work)	yes	yes
conference call	yes	yes
consultative transfer	yes	yes
consultative transfer, one-step	no	no
forward calls, cancel forward calls	yes	no
hang up call	yes	yes
hold	yes	yes
listen disconnect/reconnect (mute)	no	no
login	yes	yes
logout	yes	yes
make call	yes	yes
make call, predictive	no	no
message waiting, cancel message waiting	no	no

Capability	Hardphone	Softphone
monitor and control DN	no	yes
monitor queue	no	no
network transfer using inband signaling	no	no
off hook	yes	yes
on hook	yes	yes
ready (in service/manual-in)	yes	yes
reconnect held call	yes	yes
redirect alerting call	no	no
route call	no	no
send all calls, cancel send all calls	no	no
send DTMF tones	no	no
single-step conference	no	no
third party drop	yes	no
walk away, return from walk-away	no	no
wrap-up (after-call work)	no	no

Important:

Makecall, Conference, and Transfer can be initiated from hard phone, but the Avaya TS is not informed about call progress by the switch. The softphone is not synchronized with the hardphone. Workaround: Do not use a mixture of the hardphone & softphone with Nortel switches. If you start a conference on the hardphone, you must finish it on the hardphone.
For Nortel switches, the controlling party on a consultative call will have a phantom line appearance in the event that the caller hangs up. The switch attempts to re-synchronize the phone set if the agent uses the softphone. Actions through the hardphone are not posted by the switch to the TS, so the hardphone and the softphone get out of sync. In this scenario, in spite of the fact that the voice path is gone, the TS will attempt remove the line appearance and re-synchronize the hardphone and the softphone if the softphone is used to terminate the existing line appearance via TS.HangupVDU().

Note:

The Avaya TS configured for the Nortel Meridian switch does not support “chained” consultation scenarios. If an agent initiates a consultative transfer with a second agent, who answers the call before the transfer is complete, then that second agent cannot transfer this call to a third agent. In effect, the second agent cannot transfer a call that was an uncompleted transfer from the first agent.

Telephony server configuration

The Avaya TS is configured on the Server Editor of IC Manager. This section provides the the required and optional parameters that must be set to configure the Avaya TS for the Meridian-1 switch using Meridian Link.

For information on adding and configuring servers on Avaya IC using IC Manager, refer to *IC Administration Volume 1: Servers & Domains*.

Enter or modify the following parameters on the **TS** tab of the **Server Editor**. Right mouse click on the screen and select the **Show Advanced Properties** option to display all of these parameters.

Field	Recommended entry	Notes
ACD Name	Select the name of the ACD assigned to the switch.	This is the name of the ACD that this TS is serving from a pick list of names assigned to the ACD during system configuration.
ACD Type	Select Nortel	The type of ACD with which the TS will communicate.
ACD Model	Select Meridian	The model of the ACD that corresponds to the selected ACD Type.
ACD Protocol	Select MLP.	This is the protocol used by the Symposium Server.
Site	Select the site of your TS.	Select the site that this server is associated with. The TS uses this information to retrieve the queues for internal monitoring.
Link	The logical identifier in the server where NetMerge is running.	The Link establishes a path between a NetMerge client and a given PBX interface. The logical identifier is established in the NetMerge server configuration program.

Field	Recommended entry	Notes
Node	Enter the name of the server where NetMerge is running.	The Node needs to translate to an IP address via DNS or host table.
Advanced Properties		
Enable Call Containers	Check to create call containers for the TS. Default is checked.	If checked, the TS creates call containers to store information about the different legs of calls.
Enable Agent Containers	Check to turn on agent containers for the TS. Default is checked.	If checked, the ADU containers for the TS are turned on.
Use 5.6 State Fields	Check to give containers for agent states entries in the 5.6 style. Default is unchecked.	Example, ts.loginid. For more information, refer to the *Telephony Connectors Programmer Guide*.
Use 6.0 State Fields	Check to give containers for agent states entries in the 6.0 style. Default is checked.	Example, voice.loginid. For more information, refer to the *Telephony Connectors Programmer Guide*.
Wrap up by Client	Check to have the TS wait for the wrap-up process to be completed before removing the call information from memory. Default is unchecked.	If unchecked, the TS may remove the call information from memory before wrap-up is complete.
Wrap up Client Time to Live (min)	Enter the period of time, in minutes, that the TS waits for the wrap-up process to be completed by the client. Default is 15.	If unchecked, the TS issues a request for wrap up on behalf of the client.
Wrap up Server Time to Live (min)	Enter the period of time, in minutes, that the TS waits for the wrap-up process to be completed by the server. Default is 2.	If unchecked, the TS issues a request for wrap up on behalf of the server
Call Plan	Enter the number of digits on the external extension numbers used by the contact center. Default is 6.	This helps identify the call as internal to the switch.

Field	Recommended entry	Notes
Thread Pool Size	Enter the number of threads to allocate for the server thread pool. Default is 20.	
Queues Owned	Enter the name of the queue or queues for which the TS is responsible for creating queue ADUs.	The TSQS should not be associated with this queue in any way.
Default ANI	Enter the default ANI value, for incoming calls, which did not carry ANI information.	
Abort on Link Down	Check to abort (stop) the TS if the link goes down. Default is checked.	
Application Data	Check to indicate this switch is capable of persistent application data. Default is checked.	
Call Record Time to Live	Enter the maximum period of time, in hours, that a CtsCallRecord can remain in memory before the TS cleans it up. Default is 24.	A TpDisconnected event is generated for this record.
Logout Before Login	Check to log the user out before attempting to login. Default is checked.	
Dial by Equipment	Check to determine if the ACD requires the destination to be reached by equipment. Default is checked.	If you are using Avaya Business Advocate, this field must be checked.
Make Busy After Logout	Check to set the phone set to the MakeBusy phone state after logout is concluded. Default is unchecked.	
Monitor Route Point	Check to indicate this switch is capable of monitoring route points internally. Default is unchecked.	

Field	Recommended entry	Notes
Route Point Time to Live (sec)	Enter the number of seconds a call can remain on an unmonitored route point before the TS cleans it up. Default is 5.	After this specified period of time, a TpDisconnect event is posted to the call. Advocate users should set this value as high as possible (999) to provide ample time to deliver the call from the queue to the agent.
Suppress Logout Event	Check to ignore the logout event generated when the agent presses “Logout” on the hard phone. Default is unchecked.	This setting provided backward compatibility with the TS Meridian, version 5.1. It also allows for Multiple Queue Assignment support because the agent can logout from the hard phone without terminating the softphone session. IMPORTANT: When this parameter is set to true, the TS also suppresses logout events that are caused by an agent logging out of the hardphone. In this case, the softphone and hardphone become out of sync.
RONA Divert	Check to enable RONA (Redirect on No Answer). Default is unchecked.	This tells the TS that the switch is capable of handling a call deflection (divert) on an alerting device or if the call needs to be answered before it can be transferred or conferenced to another destination.
TSV5 Trace Level	Select the desired log level for TSV5. Default is info.	Valid values are debug, info, warning, and error layer.
Ccti Trace Level	Select the desired log level for Ccti. Default is info.	Valid values are debug, info, warning, and error layer.
Cpbx Trace Level	Select the desired log level for Cpbx. Default is info.	Valid values are debug, info, warning, and error layer.

Configuration tab

The following configuration parameter is not presented on the TS tab in IC Manager. Set this parameters on the Configuration tab:

Field	Recommended entry	Notes
allow_names_mixed_case	Enter "true" to enable processing of queue names in the case that comes in. If set to "false", queue names will be set to lower case.	Values: true or false.

Chapter 5: Queue Statistics Server

This chapter provides Telephony Server Queue Statistics server (TSQS) configuration and administration information for each of the switches supported in IC 6.1.3. The TSQS is the Avaya server that monitors the voice channel and maintains queue statistics on the Agent Data Unit (ADU) server.

This section includes the following topics:

Important:
If there are more than 500 queues defined on the system, the TSQS may timeout in a failover situation.

In order for the TSQS to start properly, there must be at least 1 queue defined on the Avaya IC system.

Avaya DEFINITY/Communication Manager

This section provides information to configure and administer the Avaya TSQS to work with the Avaya DEFINITY 8.3, 9, 10 and Avaya Communication Manager 1.1, 1.2, 1.3.

Supported platforms

The TSQS that supports the Avaya DEFINITY/Communication Manager can be run on various operating systems in IC 6.1.3. The following table lists the operating systems on which the TSQS servers can be run for the Avaya DEFINITY/Communication Manager:

Operating System	TSQS
Windows 2000 SP4	Yes
Windows 2003	Yes
Solaris 8	Yes
Solaris 9	Yes
AIX 5.1L ML3	Yes
AIX 5.2L	Yes

Queue Statistics server configuration

The Avaya TSQS is configured on the Server Editor of IC Manager. This section provides the required and optional parameters that must be set to configure the Avaya TSQS for the Avaya DEFINITY/Communication Manager.

For information on adding and configuring servers on Avaya IC using IC Manager, refer to *IC Administration Volume 1: Servers & Domains*.

Enter or modify the following parameters on the TSQS tab of the TSQS Server Editor. Right mouse click on the screen and select the Show Advanced Properties option to display all of these parameters

Field	Recommended entry	Notes
ACD Name	Select an ACD Name.	The name of the ACD (switch) that this TSQS is serving. Provides a pick list of name(s) assigned to the switch during system configuration. Each TSQS on the system must have a unique ACD Name.
Site	Select the site where the TSQS is located.	The site used by your TS configured for the Avaya switch.
ACD Type	Select Avaya.	The ACD Type option used by your TS configured for the Avaya switch.
ACD Model	Displays Definity after you select Avaya as ACD Type.	The ACD Model option used by your TS configured for the Avaya switch.
ACD Protocol	Select asai.	The ACD Protocol option used by your TS configured for Avaya switch.
Advanced Properties		
Update Interval (secs)	Enter the number of seconds between updates to the ADU server with queue statistics. Default is 10 seconds.	
Oldest ADU Timestamp	Check to enable backward compatibility to eContact 5.6. Default is unchecked.	Lets the TSQS keep the oldest field in the ADU as the length of time (in seconds) that the oldest call is in queue. The default uses the timestamp of when the oldest call arrived.

Field	Recommended entry	Notes
Forced ADU Update	Check to enable the server to issue an ADU.SetValues command even if none of the parameter values changed. Default is unchecked.	
Maximum Calls to Track	Enter the maximum number of calls to track in a single queue. Default is 4,096	

Aspect CallCenter

This section provides information necessary to configure and administer the Avaya TSQS to work with the Aspect CallCenter (version 8.x).

Supported platforms

The TSQS that supports the Aspect CallCenter version 8 can be run on various operating systems in IC 6.1.3. The following table lists the operating systems on which the TSQS servers can be run for the Aspect CallCenter:

Operating System	TSQS
Windows 2000 SP4	Yes
Windows 2003	Yes
Solaris 8	No
Solaris 9	No
AIX 5.1L ML3	No
AIX 5.2L	No

Software prerequisites

The following table lists the Aspect software that is required for the Avaya servers to work properly with the Aspect CallCenter switch:

Software	Purpose
Aspect CMI server version 4.X	Installed and available via an Ethernet connection. Enables the Avaya TS to establish a connection to the Aspect CallCenter System. The Avaya TS uses no external libraries.
Aspect RealTime Data Server	Installed and available via an Ethernet connection. Enables the TSQS to gather statistics from the Aspect CallCenter System.
Aspect RealTime Receiver Custom Control (RealTime Runtime version)	Install on the same machine as the TSQS.

Refer to the *Installation Planning and Prerequisites* for more detailed information about IC 6.1.3 prerequisites.

Queue Statistics server configuration

The Avaya TSQS is also configured on the Server Editor of IC Manager. This section describes the parameters that must be set to configure the Avaya TSQS for the Aspect CallCenter 8.

For information on adding and configuring servers on Avaya IC using IC Manager, refer to *IC Administration Volume 1: Servers & Domains*.

Enter or modify the following parameters on the **TSQS** tab of the **Server Editor**. Right mouse click on the screen and select the **Show Advanced Properties** option to display all of these properties.

Field	Recommended entry	Notes
ACD Name	Select an ACD Name	The name of the ACD (switch) that this TSQS is serving. Provides a pick list of name(s) assigned to the switch during system configuration. Each TSQS on the system must have a unique ACD Name.
Site	Select the site where the TSQS is located.	The Site used by your TS configured for the Aspect switch
ACD Type	Select Aspect.	The ACD Type option used by the TS configured for the Aspect switch.
ACD Model	Select Aspect8.	The ACD Model option used by your TS configured for the Aspect switch.
ACD Protocol	Displays AspectCMI after you select Aspect8 as the ACD Model.	Select the ACD Protocol option used by your Telephony server configured for the Aspect switch.
Switch Name	Enter the IP address of the link that connects the switch to the TSQS.	This is different from the IP address used by the switch.
Switch Port	Enter the number of the port used by the switch.	
Advanced Properties		
Update Interval (secs)	Enter the number of seconds between updates to the ADU server with queue statistics. Default is 10 seconds.	

Field	Recommended entry	Notes
Oldest ADU Timestamp	Check to enable backward compatibility to eContact 5.6. Default is unchecked.	Lets the TSQS keep the oldest field in the ADU as the length of time (in seconds) that the oldest call is in queue. The default uses the timestamp of when the oldest call arrived.
Forced ADU Update	Check to enable the server to issue an ADU.SetValues command even if none of the parameter values changed. Default is unchecked.	
Switch Poll Interval	Enter the number of seconds between TSQS requests for queue information from the switch. Default is 10.	

Enter the following parameters on the **Configuration** tab of the **Server Editor**.

To enter parameters on the Configuration tab:

1. Select the **Configuration** tab to display the **Server Editor**.
2. Click the **New** button to display the **CTI Type Editor** dialog box.
3. Select the **Couple** option from the **CTI Type** drop down menu.
4. Enter the name of the parameter in the **Name** field.
5. Enter the value of the parameter in the **Value** field.
6. Click **OK** to add the new parameter and close the **CTI Type Editor** dialog box.
7. The name and value of the new parameter are displayed on the **Server Editor**.

The following table provides the names and default values of the parameters you must enter using the procedures described above. Descriptions of each parameter are provided to help you determine your system values.

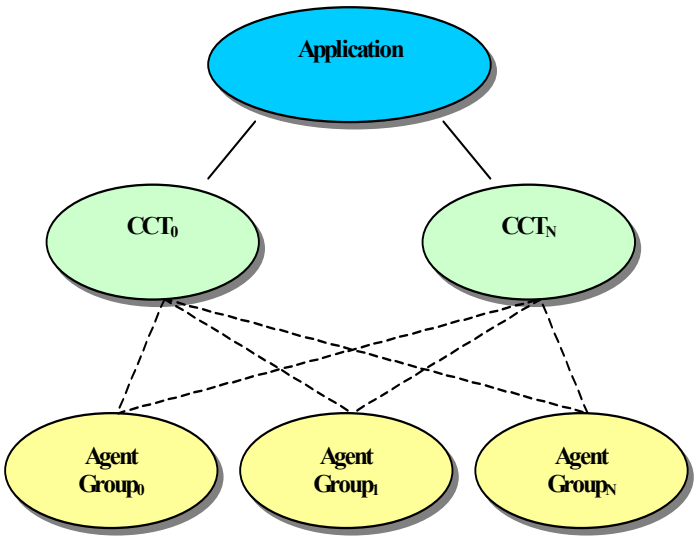
Name	Default Value	Description
log_file	NULL	Name of the TSQS log file.
log_size	10,000,000	Size of the TSQS log file.
log_level	3	Determines the level of information that should be in the log file.
auto_activate	TRUE	Sets the auto activate property of the Aspect ActiveX control.

Name	Default Value	Description
base_port	9000	Sets or returns the base TCP/IP port of the Aspect ActiveX control.
infograms	NULL	List of the infogram numbers the Aspect ActiveX control will listen for. Enter in the format: infogram#<tab>infogram#<tab>...
stats_infogram_num	110	Infogram number that contains the queue statistics.
ip_octet_1	225	First part of the IP octet composing the IP address used to monitor Aspect RealTime Data Server messages.
ip_octet_2	10	Second part of the IP octet composing the IP address used to monitor Aspect RealTime Data Server messages.
ip_octet_3	10	Third part of the IP octet composing the IP address used to monitor Aspect RealTime Data Server messages.
ip_octet_4	10	Fourth part of the IP octet composing the IP address used to monitor Aspect RealTime Data Server messages.
use_default_IP	FALSE	If set to TRUE, the Aspect ActiveX control listens for infograms on IP 224.0.0.10. If set to FALSE, the Aspect ActiveX control uses the IP address defined in the ip_octet_1, ip_octet2, ip_octet3, and ip_octet4 parameters.
use_default_port	TRUE	If set to TRUE, the Aspect ActiveX control listens for infograms on port 9000. If set to false, the Aspect ActiveX control listens for infograms on the port that is defined in the base_port parameter.

TSQS CONSIDERATIONS

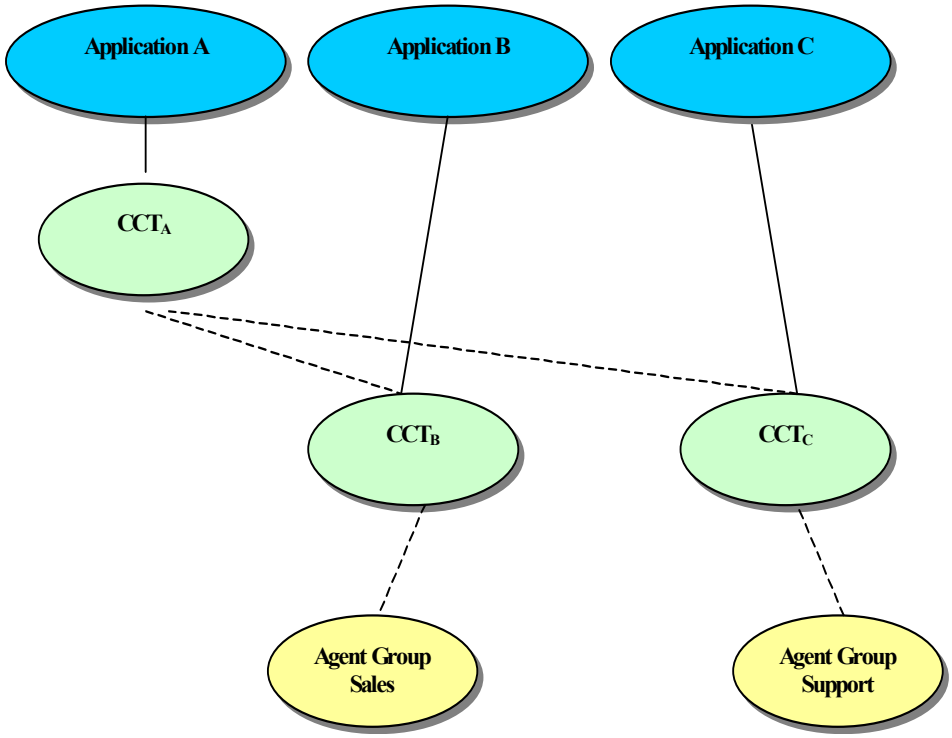
This section describes some considerations related to the collection of queue statistics from the Aspect CMI server. The Aspect CallCenter System organizes statistical information related to Agent Groups (queues) in the application for which the Agent Group received the call.

The following diagram illustrates this relationship:



In this diagram, queue type statistics are maintained for all Agent Groups that have been “triggered” by a CCT defined under a specific application. For example, if an inbound call is received by a CCT that has been created under application A, regardless of the Agent Group that ultimately receives the call, the statistics for that “interaction” are maintained by the application whose CCT handled the call (Application A). If the Agent Groups in question are Sales and Support, and they were each assigned to the same application (as above), then the TSQS would only see an aggregate of both Agent Groups.

To avoid this, you should create multiple applications for maintaining separate statistics. This makes the application structure more complex and the location of the CCT “routing” the calls also becomes a factor. The following example illustrates how the Applications, CCTs, and Agent Groups must be arranged to allow statistical information to be maintained for each Agent Group individually (not aggregated).



The dashed lines in this diagram represent a potential path that can be traversed using CCT programming steps. The solid lines represent ownership. Given that CCT_B is owned by $Application_B$, and only CCT_B presents calls to the Agent Group Sales, the statistics maintained by $Application_B$ will only be those for Agent Group Sales. This holds true for the Agent Group Support.

Transferring calls

You cannot transfer calls to TSQS devices in an Aspect CallCenter environment because they are not physical switch resources. TSQS devices are entities used by the switch for statistical purposes.

For an agent to transfer calls to a device queue in an Aspect CallCenter environment, the TSQS requires one queue device to be configured in Avaya IC for statistics monitoring and the TS requires another device to be configured and marked "addressable" to enable agents to transfer calls to them. The TSQS uses an application number (as described above) as a device. For the TS, the device must be a CCT, which is marked as "addressable".

Nortel Meridian-1: Symposium Call Center Server

This section chapter provides the information necessary to configure the Avaya TSQS to work with the Nortel Meridian-1 switch using Symposium Call Center Server. Symposium Call Center Server acts as bridge between the Nortel Meridian-1 switch and the Avaya TS. The Avaya TS communicates with Symposium Call Center Server via Intel NetMerge Call Processing software (formerly CT Connect). The Symposium Call Center Server sends the information that it receives from the Avaya TS to the Meridian-1 switch using MLS communication protocol.

Supported platforms

The TSQS that supports the Symposium Call Center Server can be run on various operating systems in IC 6.1.3. The following table lists the operating systems on which the TSQS servers can be run for the Symposium Call Center Server:

Operating System	TSQS
Windows 2000 SP4	Yes
Windows 2003	Yes
Solaris 8	No
Solaris 9	No
AIX 5.1L ML3	No
AIX 5.2L	No

Queue Statistics server configuration

The Avaya TSQS is configured on the Server Editor of IC Manager. This section provides the required and optional parameters that must be set to configure the Avaya TSQS for the Meridian-1 switch using Symposium Call Center Server.

For information on adding and configuring servers on Avaya IC using IC Manager, refer to *IC Administration Volume 1: Servers & Domains*.

Enter or modify the following parameters on the **TSQS** tab of the **Server Editor**. Right mouse click on the screen and select the **Show Advanced Properties** option to display all of these parameters.

Field	Recommended entry	Notes
ACD Name	Select an ACD Name.	The name of the ACD (switch) that this TSQS is serving. Provides a pick list of name(s) assigned to the switch during system configuration. Each TSQS on the system must have a unique ACD Name.
Site	Select the site where the TSQS is located.	The site used by the TS configured for your Nortel switch using Symposium Server.
ACD Type	Select Nortel.	The ACD Type option used by your TS configured for the Nortel switch using Symposium Server.
ACD Model	Select Symposium.	The ACD Model option used by your TS configured for the Nortel switch using Symposium Server.
ACD Protocol	Select MLS.	The ACD Protocol option used by your TS configured for the Nortel switch using Symposium Server.
Switch Name	Enter the IP address of the link that connects the switch to the TSQS.	This is different from the IP address used by the switch.
Switch Port	Enter the number of the port used by the switch.	
Advanced Properties		
Update Interval (secs)	Enter the number of seconds between updates to the ADU server with queue statistics. Default is 10 seconds.	

Field	Recommended entry	Notes
Oldest ADU Timestamp	Check to enable backward compatibility to eContact 5.6. Default is unchecked.	Lets the TSQS keep the oldest field in the ADU as the length of time (in seconds) that the oldest call is in queue. The default uses the timestamp of when the oldest call arrived.
Forced ADU Update	Check to enable the server to issue an ADU.SetValues command even if none of the parameter values changed. Default is unchecked.	
Switch Poll Interval	Enter the number of seconds between TSQS requests for queue information from the switch. Default is 10.	
Symposium User Id	Enter the Symposium User ID. Default is sysadmin.	
Symposium Password	Enter the Symposium Password. Default is nortel.	

Transferring calls

You cannot transfer calls to TSQS devices in a Symposium Call Center environment because they are not physical switch resources. TSQS devices are entities used by the switch for statistical purposes.

For an agent to transfer calls to a device queue using a Nortel Symposium Call Center Server, the TSQS requires a skill set name and ID, which are defined in the Symposium Call Center Server. The TS can transfer calls to a queue, so the device must be a queue, which is marked as "addressable" for the TS.

Meridian-1: Meridian Link

This section provides the information necessary to configure the Avaya TSQS to work with the Nortel Meridian-1 switch using Meridian Link. Meridian Link acts as a bridge between the Avaya TS and the Meridian-1 switch. The Avaya TS sends data to the Meridian Link via Intel NetMerge Call Processing software (formerly CT Connect). The Meridian Link sends this information to the Meridian-1 switch using the MLP communication protocol.

Supported platforms

The TSQS that supports Meridian Link can be run on various operating systems in IC 6.1.3. The following table lists the operating systems on which the TSQS servers can be run for Meridian Link:

Operating System	TSQS
Windows 2000 SP4	Yes
Windows 2003	Yes
Solaris 8	No
Solaris 9	No
AIX 5.1L ML3	No
AIX 5.2L	No

Queue Statistics server configuration

The Avaya TSQS is configured on the Server Editor of IC Manager. This section provides the required and optional parameters that must be set to configure the Avaya TSQS for the Nortel Meridian-1 switch using Meridian Link.

For information on adding and configuring servers on Avaya IC using IC Manager, refer to *IC Administration Volume 1: Servers & Domains*.

Enter or modify the following parameters on the **TSQS** tab of the **Server Editor**. Right mouse click on the screen and select the **Show Advanced Properties** option to display all of these parameters.

Field	Recommended entry	Notes
ACD Name	Select an ACD Name.	The name of the ACD (switch) that this TSQS is serving. Provides a pick list of name(s) assigned to the switch during system configuration. Each TSQS on the system must have a unique ACD Name.
Site	Select the site where the TSQS is located.	The site used by your TS configured for the Nortel switch using Meridian Link.
ACD Type	Select Nortel.	The ACD Type option used by your TS configured for the Nortel switch using Meridian Link.
ACD Model	Select Meridian.	The ACD Model option used by your TS configured for the Nortel switch using Meridian Link.
ACD Protocol	Select MLP.	The ACD Protocol option used by your TS configured for the Nortel switch using Meridian Link.
Switch Name	Enter the IP address of the link that connects the switch to the TSQS.	This is different from the IP address used by the switch.
Switch Port	Enter the number of the port used by the switch.	
Advanced Properties		
Update Interval (secs)	Enter the number of seconds between updates to the ADU server with queue statistics. Default is 10 seconds.	

Field	Recommended entry	Notes
Oldest ADU Timestamp	Check to enable backward compatibility to eContact 5.6. Default is unchecked.	Lets the TSQS keep the oldest field in the ADU as the length of time (in seconds) that the oldest call is in queue. The default uses the timestamp of when the oldest call arrived.
Forced ADU Update	Check to enable the server to issue an ADU.SetValues command even if none of the parameter values changed. Default is unchecked.	
Switch Poll Interval	Enter the number of seconds between TSQS requests for queue information from the switch. Default is 10.	
Vendor Id	Enter the Nortel Vendor Id number corresponding to your switch.	

Chapter 6: Multi Site Heterogeneous Switch

This chapter describes the functionality and operation of the Multi Site Heterogeneous Switch (MSHS). MSHS support for the TS enables transfers and conferences across ACDs to take place transparently (multiple ACDs behave as a single element).

This section includes the following topics:

Overview

Multi Site Heterogeneous Switch (MSHS) is a TS feature that allows supported switches (Avaya DEFINITY, Aspect CallCenter, and Nortel Meridian) co-located or not (in different physical locations) to operate, with a few exceptions, as one large contact center.

For the methods related to make call, conference, transfer, and route, the TS determines, via a MSHS protocol, how and where to route a call. This is done through a combination of elements, where the ADU plays a major role because all of the users and queues must have an ADU on the system and the physical location of the of each device (site, TS, and switch) is written to the ADU.

Note:

Users and queues are referenced by name, IC login ID or queue name. The TS is responsible for name resolution, which means it must map the name to a physical location. Name resolution is performed through an ADU query because the ADU is the central repository for this information.

Once the source TS determines the location of a given user, via name resolution if the given user is not assigned to the source TS, the source TS needs to determine if MSHS protocol can communicate with the destination. TSes participating in MSHS and assigned to a TS Group are required to cross assign to each other. TSes can only cross assign to each other during startup and only if they belong to the same TS Group. Therefore, a MSHS request will not cause a remote TS to be started by its ORB server. Once a server is configured and started, it is automatically added to the pool of TSes servicing MSHS.

MSHS is invoked via the method TS.HeteroSwitchHandoff(), which passes the EDUID and the final destination for the call to the destination TS. The source TS expects a response with a phone number to dial, which is used to establish the voice path between the two switches. The destination TS gets this phone number from a list of Reserved DNs. This list of Reserved DNs establishes a relationship between a DID (direct inward dial) and a route point (VDN). The destination TS is monitoring all VDNs on the Reserved DN list.

The destination TS scans the list and finds an entry that is not in use. It then associates the EDUID and the final call destination with that entry, and reserves the entry for a period of time.

Once informed about which number to dial, the source TS determines how it should dial the number. This is accomplished via the Dial Translation table, which defines dialing plans for different sites. For example, when dialing between two different sites, one site might have tie trunks so dialing the PSTN is not necessary. For more information, refer to the Dial Translation table description.

The source TS establishes a voice path (places a call) with the destination TS. When the call reaches the reserved VDN, the destination TS is notified, and it associates the EDUID with the call and routes (via route select) the call to its final destination, thus releasing the reserved DN, and making it available for re-use by another MSHS transaction.

Important:

In order to get an accurate "number in call" count, you need to understand when the TS adds parties to conference lists. The TS does not add a party to a conference list until the party is connected to the call. For example, when an agent conferences a call to a second agent and the call is placed in queue waiting for the second agent to answer, the call is not included in the "number in call" count while it is in the queue. It only gets included in the count when the call is answered by the second agent. This prevents the TS indicating that a conference took place with a queue, which would change the conference end points.

Network transfer

Network transfer using inband signaling is a service offered by network carriers that transfers a call from one location to another via the carrier network, avoiding trunk-to-trunk connections. Depending on the network transfer offering, the transfer operation can be requested using either inband (DTMF) signaling or out-of-band (ISDN D-channel) signaling. Avaya TS provides support for inband network transfer.

Note:

Network transfer using out-of-band signaling is not directly supported by Avaya IC, however an ACD may invoke out-of-band network transfer under certain call scenarios and/or sequences.

With inband network transfer, the carrier is notified that a call connected to one site needs to be disconnected from that site and reconnected to a second site via a DTMF dialing sequence. Because the carrier performs the actual transfer operation, no devices are tied up to the local switch to route the call from site to site, thereby minimizing trunk usage. Avaya IC supports this service for blind transfers and trunk-to-trunk connections for consultative calls.

Several carriers offer network transfer via inband signaling. Currently MCI offers an inband network transfer service called Take Back and Transfer. Sprint also offers a similar version of inband network transfer. AT&T provides an equivalent service called Inband Transfer and Connect (covered in AT&T TR50075). The AT&T offer supports sending UUI with the transferred call if Avaya IC includes the UUI in the ISDN DISConnect message for the call. Neither MCI nor Sprint support UUI transfer. The inband network transfer capability is generically referred to as “Take Back and Transfer” in the Avaya IC screens and the sending UUI support is referred to as “ISDN UUI”.

Functional unit descriptions

Destination resolution

Destination resolution, or name resolution, is performed by the TS via ADU.Find(). Every user and queue has a name and an ADU in Avaya IC. The ADU contains the necessary elements to map the name to its physical location.

Agent ADU elements

When an agent logs into Avaya IC, an ADU is created. When the agent assigns to the TS, the following elements are written to the ADU:

voice.connector	UUID of the TS where the agent is assigned/logged in.
voice.connectorname	Alias of the TS where the agent is assigned/logged in.
voice.acdname	ACD name of the switch where the agent is assigned/ logged in.
voice.device	Physical device where the agent is currently logged in.
voice.device_address	Concatenation of voice.device and voice.connector.

The TS can determine the location of any given user with this data.

Queue ADU elements

Queue ADUs are created when the TSQS assigns to the queue. The following data elements are found in queue ADUs:

connector	UUID of the TS servicing this queue.
connectorname	Alias of the TS servicing this queue.
acdname	ACD name of the switch servicing this queue.
queueid	Physical device for the queue
queuename	Name of the queue in Avaya IC.
id	Same as queueid above.
device_address	Concatenation of queueid and connector.

The TSQS can determine the location of any given queue with this data.

Note:

Unlike agent information, queue data is cached for 24 hours after a successful ADU.Find(). Therefore, the search for a given queue is performed only once every 24 hours (a configurable parameter). This is because queues are static elements and there is no reason to overload the ADU with unnecessary finds.

Multiple ways to interconnect sites

The first issue with multi-site and a pool of reserved DNs is the need for a site to know how to reach all of the other sites. Site-A might use a private network to reach site-B, but that might not be true from site-B to site-A.

Also, the destination TS knows which reserved DN it associated an EDUID with, therefore it knows the DID behind it. Passing this phone number to the requesting TS means a translation to cause the call to be routed via a private network. For instance, site-A dials site-B via speed-dial “801xxxx”, instead of the usual “91508787xxxx” for the PSTN.

The TS uses the “human” approach to implement a solution. Note that a person has a single standard universe of numbers to work from, the PSTN (usually 10 digits for calls within the U.S.). When a person is on site-A, this person knows to dial “801xxxx” to reach someone on site-B, and also knows to dial “802xxxx” from site-C. By adopting this same strategy, the TS does not need to know it is dialing site-B, from A, or C — it rather needs to know how to translate the 10-digit PSTN number into an outbound dial based on where it is located. For instance, when the TS receives (508)787-2834, it could simply substitute (508)787-2834, with “80128”, in order to generate an outbound dial via PSTN or private network. The actual path is in accordance to ACD internal rules. This approach is flexible enough because it is rule based and can be modified as the network is adjusted to satisfy the business needs of the contact center.

The solution calls for a centralized table listing each site, and how to translate a PSTN number into an outbound dial. For instance,

Source TS alias	Dest. TS alias	Match	Remove	Prepend	Append
ActonDefinity	AustinNortel	512427	TRUE	804	
ActonDefinity	DublinDefinity	925479	TRUE	801	
ActonDefinity	Local	508	TRUE	9	
ActonDefinity	Boston	*	FALSE	91	
ActonDefinity	ActonNortel	77	FALSE	899	
ActonNortel	ActonDefinity	88	FALSE	899	
HolmdelDefinity	DublinDefinity	925479	FALSE	1	

The Dest TS Alias is listed here for completeness, but the TS does not use this data. The MATCH field is sufficient to determine how to translate a dial string.

Note:

A TS located in Acton knows how to translate a PSTN entry from Austin, as does a TS in Dublin. Also note that a TS located in Holmdel would dial the full 10-digits to reach Dublin using the data in the Match and the Prepend fields.

The table does not need to list all possible sites and combinations. If an entry is not in the table, the rule is to use PSTN, thus the entry for Holmdel is unnecessary, but harmless.

The destination TS knows the PSTN entry for each reserved DN, and does not need to know anything about how to be reached by other Telephony servers. This task is the responsibility of the requesting TS.

The TS raises an Alarm if the table is empty, but it will not block functionality, nor cause the TS to stop functioning, because it might be possible to still interconnect sites via ordinary PSTN outdial.

These elements are configured in IC Manager. For more information, refer to Configuring multi site heterogeneous switches on page 119.

Network transfer

In order to minimize trunk usage, Avaya IC takes advantage of the carrier's network transfer, if it is available.

The determination whether to use network transfer is defined on the Transfer Resolution Table screen as shown in the following table:

Source TS alias	Destination TS alias	Network Transfer	ISDN UUI Support
ActonDefinity	AustinNortel	TRUE	FALSE
AustinNortel	ActonDefinity	TRUE	FALSE
ActonDefinity	DublinDefinity	TRUE	FALSE

The table contains entries showing a unidirectional resolution, which provides more flexibility in the configuration.

The network transfer functionality is only available via blind transfer.

In the Network Transfer column, TRUE indicates the carrier for the incoming trunk group supports network transfer and FALSE indicates a trunk-to-trunk connection is required.

In the ISDN UUI Support column, TRUE indicates the carrier for the incoming trunk group supports transferring the UUI with the network transfer or the trunk-to-trunk connection.

Sites not listed are considered not to have network transfer capability.

Network transfer and ISDN UUI support are administered in IC Manager. Refer to Setting advanced properties on page 126.

Bypassing MSHS

After retrieving data that provides the location of a specific agent or queue, the TS decides whether or not to enable MSHS support.

The TS can bypass MSHS when the source and destination Avaya TSes are in the same switch, or when both Avaya TSes can receive UUI from interconnecting private ISDN trunks. The determination is via the Transfer Resolution Table:

Source TS alias	Destination TS alias	Network Transfer	ISDN UUI Support
ActonDefinity	AustinNortel	TRUE	FALSE
AustinNortel	ActonDefinity	TRUE	FALSE
ActonDefinity	DublinDefinity	FALSE	TRUE

This table indicates an ISDN trunk is available between ActonDefinity and DublinDefinity and that UUI transfer is possible on a trunk-to-trunk connection basis. Therefore, the source TS will not use MSHS, it will dial directly to the destination DN.

Which method to use is resolved as follows:

- Retrieve the ADU relative to the destination.
- Evaluate if direct transfer is possible (same switch):
  - use voice.acdname.
  - if source and destination are in the same switch, and that switch can keep persistent Application Data, then perform direct transfer.
- Evaluate if UUI data transfer is possible on this connection.
  - if so, the destination is resolved by the data in the ADU via voice.device.
- Evaluate if network transfer is possible.
  - if BLIND TRANSFER, use network transfer.

For information on accessing and modifying the Transfer Resolution Table on IC Manager, refer to Setting advanced properties on page 126.

Multiple ANI per site

During the MSHS handshake, the destination TS provided the source TS with a DID to which the voice is to be established. In order to enhance security, the destination TS matches the ANI of the incoming call with values found in the Multiple ANI Table. If the ANI does not match a table entry, the TS routes the call in accordance with the following rules:

Condition	Action
no ANI in incoming call	Transfer call to a default DN. TS keeps waiting for call on reserved DN, until timeout. Update statistics with invalid ANI arrival.
ANI present, but not recognized	Transfer call to a default DN. TS keeps waiting for call on reserved DN, until timeout. Update statistics with invalid ANI arrival.
ANI present, recognized, but reserved DN was expecting a different one	Transfer call to a default DN. TS keeps waiting for call on reserved DN, until timeout. Update statistics with invalid ANI arrival.
reserved DN was not set to expect a call	Transfer call to a default DN. Update statistics with invalid call arrival.

The ANI validation is performed through a central table, which lists, per TS the range of ANIs to be used. The ANI table is simply:

TS Alias	ANI range
ActonDefinity	(508)787-28xx
DublinDefinity	(925)417-28xx

- An alarm is raised if the table is not found or empty.
- These elements are configured and administered in IC Manager. For more information, refer to Configuring multi site heterogeneous switches on page 119.

TS grouping

A large company might organize Avaya TSes in groups to optimize communication, or to restrict calls from being transferred across switches even though all sites are under a single IC 6.1.3 control. This extra level of flexibility reduces WAN bandwidth requirements, and helps to logically organize what otherwise would be a complex structure.

Having a TS belonging to one or more groups and allowing the TS to communicate exclusively within a set requires a table to define the set. At startup, the TS queries the Directory server for the set list. The TS determines which groups it is a member of and proceeds to cross assign to the other TSes that belong to the same groups.

The following table defines the set of groups in which the TS can communicate. This table provides an example on how to set up the groups:

TS Group	TS Alias
SalesGrp1	AustinDefinity, ChicagoNortel, DublinDefinity
SalesGrp2	ActonDefinity, DublinDefinity
CTI	ActonDefinity, AustinNortel, DublinNortel, HolmdelDefinity
CrossCountry	ChicagoNortel, DenverAspect

During startup, the TS retrieves the set and assigns in accordance to the table definition. In the scenario above, a TS in Chicago could not send a call to Holmdel because it does not assign to Holmdel during startup.

An alarm is raised if the table is not located or empty, for in this case, the TS could not route calls to other sites.

These elements are configured and administered in IC Manager. For more information, refer to Configuring multi site heterogeneous switches on page 119.

Call interflow

Call Interflow distributes the call load between sites with minimal cost to the customer. By measuring the workload and other system wide parameters, a workflow determines where best to direct a call.

MSHS routes calls after delivery. This solution works even in networks that do not support SS7 protocol. However, routing only takes place after the call is delivered to a site. Also, site-to-site transfers are slow because there are two steps with a possible PSTN involvement. This can also cause extra resource allocation on switches due to call pass-through as well as incoming and outgoing trunk allocation.

MSHS can receive a call at a site, and route it while it is at the incoming adjunct route step, if the switch allows and the switches can be configured for trunk-to-trunk transfer. Trunk-to-trunk transfers allow an incoming call to be routed through an outgoing trunk to an external destination. The call remains active until either end point drops off. Usually this feature is turned off because it leaves the environment open for toll fraud and it takes two trunks for each call passing through the switch.

To help administration of the environment with respect to the pool of Reserved DNs and the overall workload, the following statistics are gathered by the TS and presented via TS.GetStatus.Request:

- % Reserved DN utilization
- Peak Reserved DN utilization
- Number of failed attempts due to unavailable DNs
- Number of reserved DNs that timed out
- Number of calls routed to default destination due to invalid ANI
- Number of requests for Reserved DN initiated by clients other than a TS

User interface or external API

The exposed API is constructed on top of one CORBA method that should have the following signature:

```
ORBStatus HeteroSwitchHandoff(in VDU_ID vduid,
          in string target,
          in string type_of_request,
          in string request_handle
          out string dest);
```

The HeteroSwitchHandoff method is used by the source to initiate the transfer/conference. The input parameters are:

- The EDUID related to the call.
- The alphanumeric target destination.
- An indication of the type of call (transfer, conference, etc).
- A Request Handle, which is used by the source to match the request with the response.

The destination issues a HeteroSwitchRouteStatus event to inform the source that the connection is established. The HeteroSwitchRouteStatus event indicates the correspondent RequestHandle and if the connection was completed (VESP_SUCCESS), or if one end is missing, which indicates a security fault (VESP_FAILURE). The TS uses this event to finalize its client request.

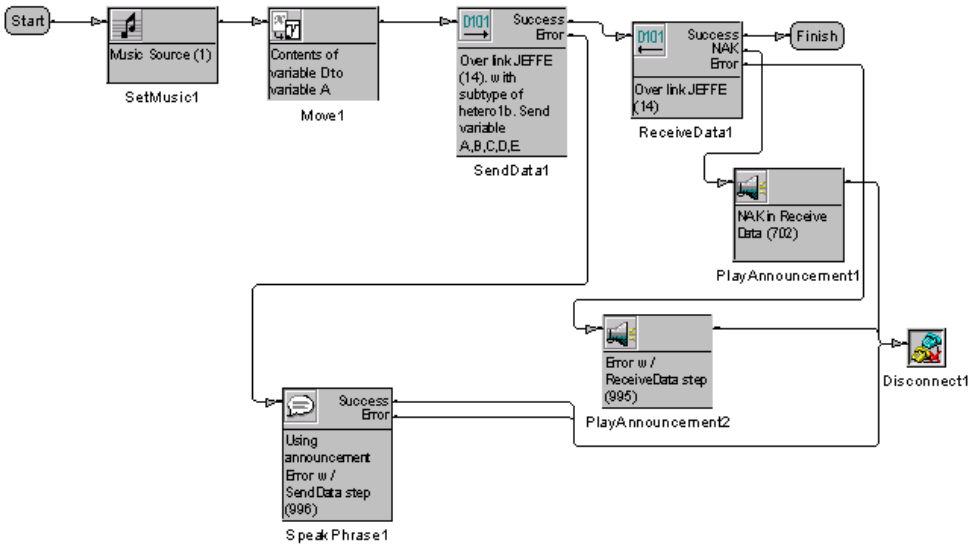
Configuring multi site heterogeneous switches

Multi Site Heterogeneous Switch (MSHS) functionality enables you to transfer and conference telephone calls regardless of switch type or location. In Avaya IC 6.1.3, MSHS is supported on selected versions of Avaya DEFINITY/Communication Manager, Nortel Meridian, and Aspect CallCenter switches. For details on the supported versions of these switches, see *Installation Planning and Prerequisites*.

Prerequisites for MSHS

Before configuring MSHS:

- Install and configure the Avaya TSes in stand-alone mode. Refer to the *Installation and Configuration* for instructions.
- Test these servers to ensure they can run in stand-alone mode. Refer to the *IC Administration Volume 1: Servers & Domains* for instructions.
- For systems using an Aspect CallCenter switch, a separate Hetero CCT is required for each Hetero call you need to transfer at the same time. You must specify a unique subtype in the SendData step for each Hetero CCT. This subtype is used in the configuration of the Reserved DN table. Refer to Aspect CallCenter on page 43 for more information about CCTs, SendData steps, and subtypes.

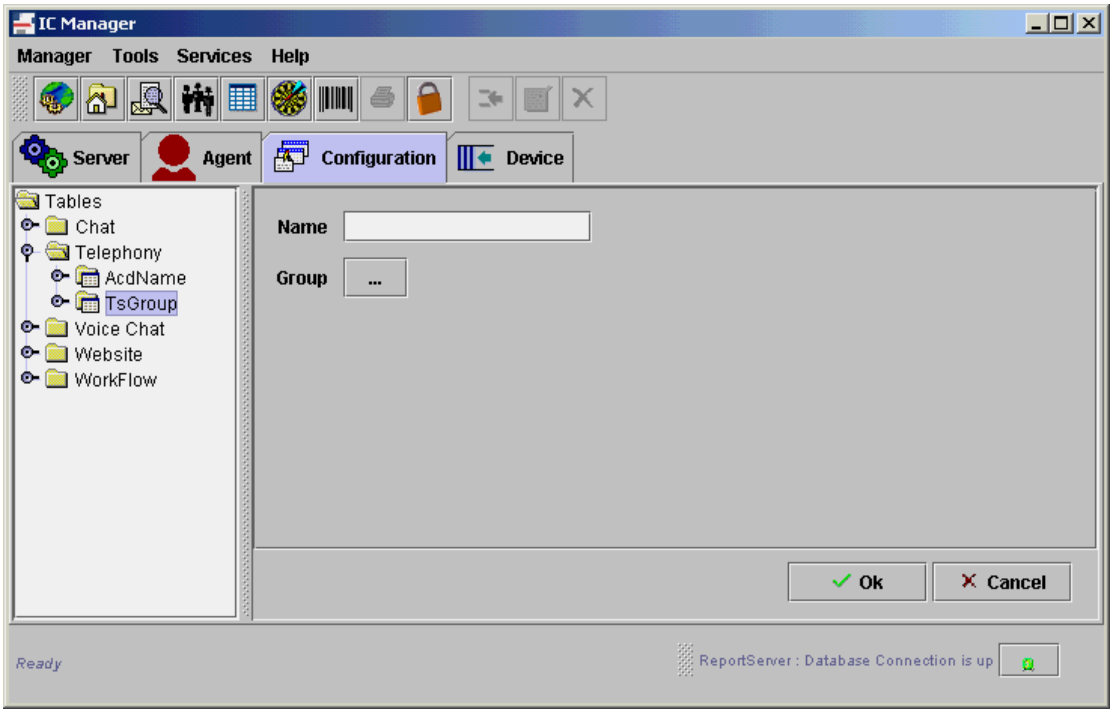


Creating TS groups

After installing, configuring, and testing each TS in stand-alone mode, assign the Avaya TSes to TS Groups based on how you want to connect them. For example, you may want to connect an Avaya DEFINITY TS at your location to a Nortel Meridian TS at another site. You would put these two servers in the same TS Group.

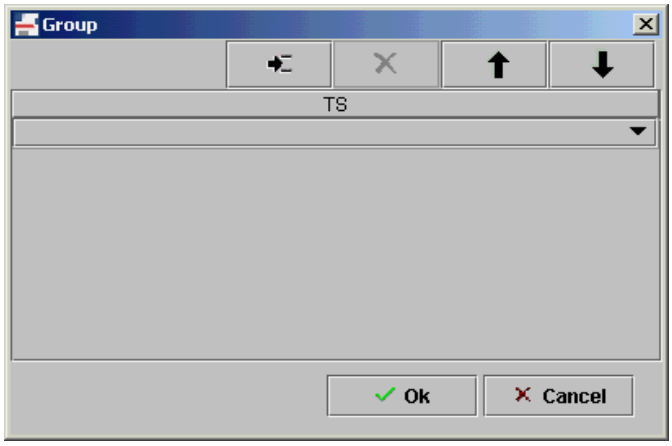
To create TS groups:

1. In IC Manager, click the **Configuration** tab.
2. Select **Tables >Telephony >TsGroup** in the left frame.
3. Click **New** to display the following dialog box:

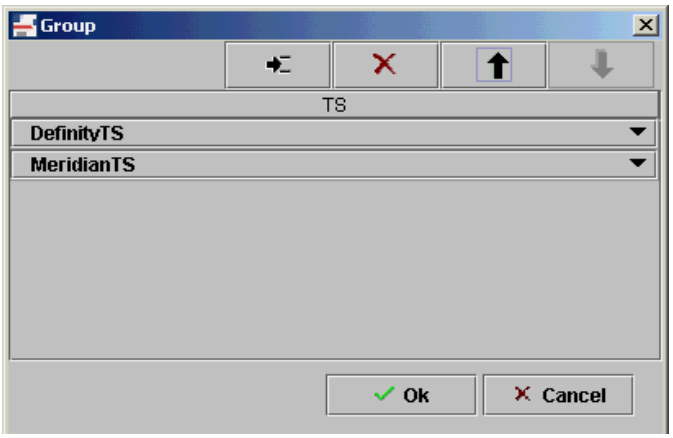


4. Enter the name of the TS Group in the **Name** field. We recommend that you use a logical name, such as Definity, for the TS at your site.

5. Click the Ellipsis (...) in the **Group** field to display the Group dialog box, shown in the following figure, where you can select servers to assign to this TS Group.



6. Click **New** and IC Manager adds a field under the TS heading.
7. Click the down arrow on the right side of the new field to display a list of the available Avaya TSes.
8. Click on the server name from the list to add it to the TS Group. Click **OK** to add the Avaya TSes to the TS Group.
9. Click **New** and repeat these steps to assign additional servers to the TS Group. The following figure illustrates a TS Group that contains the Avaya DEFINITY TS and the Meridian TS.



10. After assigning the last TS to the TS Group, click **OK**. IC Manager displays the list of servers (by name and group) that you assigned.
11. Click **Apply** to complete the server assignment process. The names of the server groups are displayed in the left frame under TS Group.

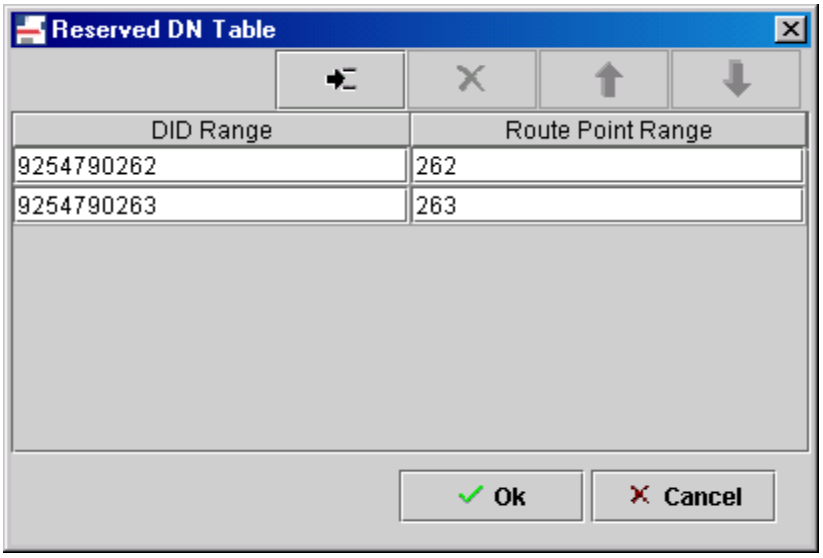
Configuring MSHS

After you have installed, configured, tested, and assigned the servers to a TS Group, configure the TS to enable the multi site heterogeneous switch functionality.

To configure Avaya IC 6.1.3 to support MSHS:

1. In IC Manager, click the **Server** tab to display a list of Avaya servers in the **Server Manager**.
2. Double-click on the name of the TS in your location to display the **Edit Server** dialog box.
3. Click the **Hetero-Switch** tab.
4. Check the **Enable Hetero Switch Transfer** check box to display the remaining configuration fields.
5. In the **Default Route Point** field, enter the VDN number that identifies the default destination for route point calls received for which the ANI does not match.
6. If you want the TS to validate the ANI when it receives a call in a reserved DN, check the check box in the **ANI Validation** field and enter the ANI number you want the TS to validate in the **Multiple ANI Table** field below. For more details, refer to Creating TS groups on page 120
7. In the **Queue Time to Live** field, enter the number of hours that a TS will consider a queue on another TS exists before rechecking to see if its ADU exists on the system.
8. In the **Hetero Lifetime** field, enter the number of seconds that reserved DNs should remain in the locked state waiting for its call to arrive.
9. Check the check box in the **Use DNIS** field to enable the TS to use DNIS resolution when a call arrives at a reserved route point, which allows a range of DID numbers to be associated with a single reserved VDN.
   - The TS monitors the VDN, as usual.
   - The one-to-one relationship between the DID and VDN is extended to DID to DNIS.
   - A range of DID entries, such as 5085551201 to 5085551299, by default expand over a DNIS range of 1201 to 1299 allowing the TS to maintain a DID-DNIS relationship via a single VDN.
10. Check the check box in the **Use Local ANI** field prior to routing the call to have the TS use the Local ANI parameter if the call does not contain an ANI. Default is checked.
11. Select the TS Group to which this TS is assigned from the drop down list in the **TS Group** field.

12. Click the Ellipsis (...) in the **Reserved DN Table** field to display the Reserved DN Table dialog box, shown in the following figure, where you reserve numbers for communicating between the Avaya TSes. You must complete both fields in this table:



a. To insert a new row under the **DID Range** and **Route Point Range** fields, click on the **New** button.

b. In the **DID Range** field, enter the range of phone numbers to be reserved for routing between the Avaya TSes:

For the numbers:	Enter	or
925479001 925479002 925479003 . . . 9254790099	92547900xx	9254790001-9254700099 **or** 9254790001-98

c. To skip some numbers within the range of numbers, enter a comma between the numbers. For example, 925479001-9254790050, 925479060-9254790099 enters all of the numbers in the original range except 925479051-9254790059.

d. In the **Route Point Range** field, enter the one of the following used by your switch to interpret the phone numbers to the second switch:

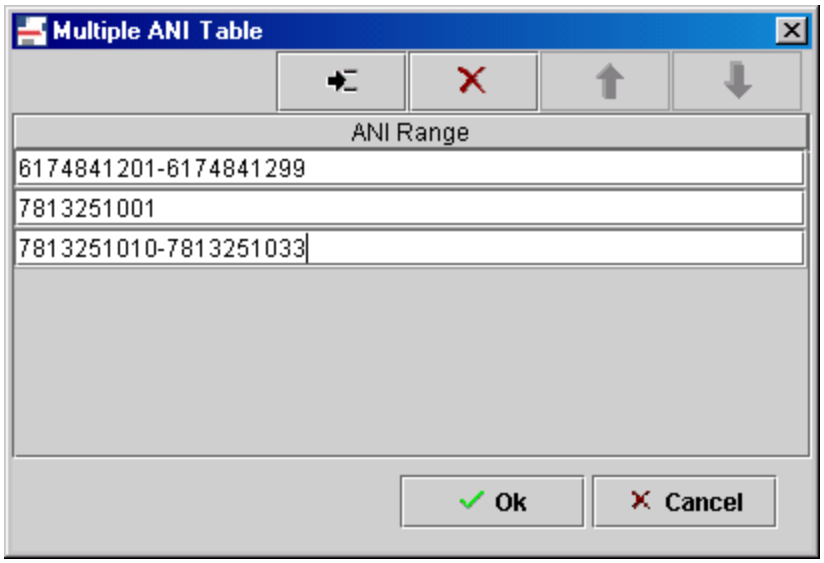
- Avaya DEFINITY - the range of internal VDN numbers.
- Nortel Meridian - the range of CDN numbers.
- Aspect CallCenter - the subtype of the SendData step of the Hetero CCT. List the incoming DID and the CCTs connected to it on a 1:1 basis.

e. Click **OK** to save your entries and return to the **Hetero-Switch** configuration window.

Note:

For same switch hetero transfers on an Aspect CallCenter switch, the DID entry must correspond to a target CCT number. For example, 76c where the "c" denotes a CCT on the Aspect CallCenter switch.
For hetero switching from a non-Aspect CallCenter switch to an Aspect CallCenter switch, the DID entry must correspond to the dialed digits needed to reach the Hetero CCT on the Aspect CallCenter switch. For example, dialing the digits 9,19995551111 will reach CCT 76c.

13. Click the Ellipsis (...) in the **Multiple ANI Table** field to display the **Multiple ANI Table** dialog box, shown in the following figure. Enter the ANI number used by this TS to communicate with the other TS. Using the ANI gives the other TS another way to confirm which TS is the one making the call.



Note:

check the check box in the **ANI Validation** field to enable the Multiple ANI Table parameter. The first TS may not be validating ANIs, but it still needs to reveal its ANIs because the second TS may be doing ANI validation.

14. Click on the Ellipsis (...) in the **Dial Translation Table** field to display the **Dial Translation Table** dialog box, shown in the following example.

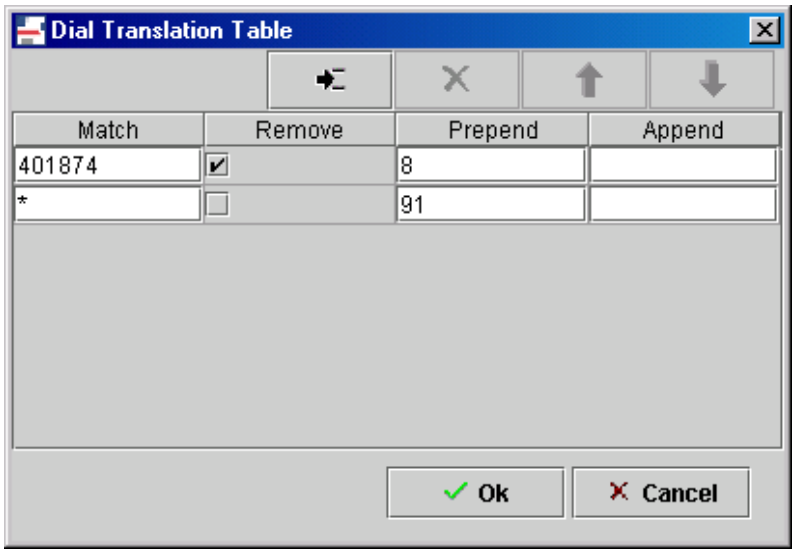
Example

The **Dial Translation Table** parameter enables you to automate the dialing of the phone number that connects to the other TS in a speed dial fashion.

The following example describes a typical scenario:

1. You have an Avaya DEFINITY TS is at your site and you want to connect to a Meridian TS at another site.
2. The two switches (Avaya DEFINITY and Meridian) have each other's route information.
3. In the **Match** field, enter the sequence of elements the local TS will use as the key for the rule.
   - In the following example, 401874 is entered in the **Match** field to create a rule that handles phone number type 401874xxxx.
   - The rule is read across: The TS removes 401874 from the phone number and prepends the number 8 (which is entered in the Prepend field).
   - Phone number 4018746754 becomes 86754.
   - The 8 might correspond to a speed dial setup in the switch.
4. The Avaya DEFINITY TS at your site parses the DID of the Meridian (destination) TS by replacing the numbers in the **Match** field with the number you entered in the **Prepend** field.

Another destination causes the second rule to execute and the number in the second **Prepend** field (91 in this example) is prepended to the original string, a destination TS indicating 6178883421 would cause the TS to dial 96178883421



5. Click the **New** button to insert a row of fields on the dialog.
6. Enter the number that corresponds to the DID (depending on the switch) extension in the **Match** field.

   Note:

   The match can be any number of digits you want to match in order to determine what to dial in order to reach that number. An asterisk (*) can be used to indicate the whole number.
7. Enter the digits of the phone number to which you want to prepend the match value in the **Prepend** field.
8. Click **OK** to save these entries.
9. Click **OK** at the **Hetero Switch** configuration screen to save all the configuration parameters.
10. Restart the TS to have the MSHS configuration take effect.

Setting advanced properties

Advanced Properties are provided for using the carrier's network transfer functionality using inband signaling. Inband network transfer is generically referred to as "Take Back and Transfer" on the Avaya IC screens.

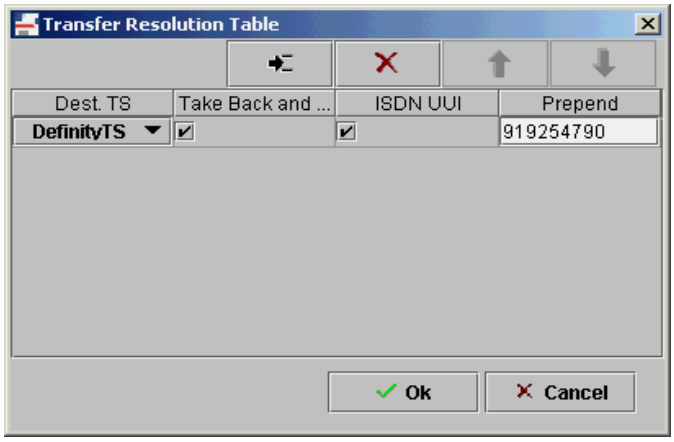
Note:

Configuring these parameters requires an advanced knowledge in telephony and switch functionality. We recommend that you consult with trained Avaya Professional Services or your Avaya BusinessPartners personnel before proceeding.

To set Advanced Properties for MSHS functionality:

1. To display the **Show Advanced Properties** prompt, right click on the **Hetero Switch** configuration screen.
2. To display the advanced parameters, check the check box at the **Show Advanced Properties** prompt.

3. To display the **Transfer Resolution Table** dialog box, shown in the following figure, click on the Ellipsis (...) in the **Transfer Resolution Table** field.



   a. To insert a row of fields on the dialog, click the **New** button.

   b. Select the TS to which you want to connect in the **Dest TS** drop down field.

   c. To enable the network transfer service, check the check box in the **Take Back and Transfer** field.

   d. If both the Avaya TSes can receive UUI from interconnecting private ISDN trunks, check the check box in the **ISDN UUI** field.

   e. Enter the digits of the phone number to which you want to prepend the match value in the **Prepend** field.

   Note:

   This is similar to the entries you made in the **Data Translation Table** field. The first TS needs to know what to dial to reach the second TS even if they are connected directly using UUI.

   f. Click **OK** to save these entries.

4. In the **Take Back and Transfer Sequence** field, enter the sequence of characters that request the connection to send to the TS to activate the network transfer service.

5. In the **Take Back Transfer Type** field, enter the type of network transfer service that you want to implement, inbound or outbound.

6. To enable trunk-to-trunk connections for consultative calls in the ACD and the TS, check the check box in the **Trunk to Trunk Transfer Enabled** field.

7. Click **OK** to save the **Advanced Properties** configuration parameters.

8. Restart the TS to have the Take Back and Transfer configuration take effect.

Chapter 7: Containers

This chapter describes the objects the Avaya TS uses the store information about the contacts and agents who use Avaya IC. These objects are called containers.

This section includes the following topics:

Overview

The Avaya TS automatically collects information about the contacts who call Avaya IC and the agents who handle them. This information is stored in objects known as containers**.** Containers are structures that hold repeated instances of call data, such as the names of interested clients and call point information or agent activity.

The Avaya TS creates two kinds of containers to store this information.

- Voice contact containers store information about the contacts using Avaya IC. Voice contact containers are discussed in detail in Voice contact containers on page 131.
- Agent containers store information about the agents who handle contacts on Avaya IC. Detailed information on agent containers is provided in Agent containers on page 146.

Containers are enabled or disabled through the "Enable Call Containers" configuration parameter in IC Manager. The "Enable Call Containers" and the "Use 6.0 State Fields" parameters are enabled by default on the Server Manager in IC Manager to enable both types of containers and call state changes. These parameters are required for the creation of containers on Avaya IC.

The following table describes the configuration parameters that pertain to containers. Refer to *IC Administration Volume 1: Servers & Domains* for more details.

Parameter Name	Default	Description
Enable Agent Containers	enabled	Turns ADU Containers on/off for agent containers.
Enable Call Containers	enabled	Turns EDU Containers on/off for voice contact containers.
Use 6.0 State Fields	enabled	Determines if “call state” statistics are written to the container in the new style (6.0 and later).

Voice contact containers

Voice contact containers (EDU containers) store information about specific contacts as they move through Avaya IC. This information includes the ANI and DNIS associated with the contact, the duration of the contact (in real-time), and the reason for terminating the contact.

Voice contact containers are created when the contact is received by the Avaya TS from the switch. The contact is assigned an Electronic Data Unit (EDU) identifier, known as an EDUID. When the call enters a device that is monitored by Avaya IC, the Avaya TS verifies if the data in the UUI is a valid EDUID. If it is not valid, the Avaya TS creates a UUI field in the EDU and populates this field with the information retrieved from the UUI. From that point, the UUI data is populated by the EDUID.

The information that is collected in the voice contact container as the contact moves from one leg of the call to another is placed into the contact's Electronic Data Unit (EDU) based on the EDUID.

The various parties involved in a voice contact (such as agents, callers, and IVRs) are known as end points. Each time a voice contact is connected to a new end point, the Avaya TS creates a new container in which to store information about the new end point.

A voice contact's first end point is named voice.1, the second is voice.2, the third is voice.3 and so on. Information pertaining to each end point is stored in a subordinate structure: voice.2.talktime, voice.2.ringtime, voice.2.holdtime, and so on.

A voice contact container accumulates information about each successive end point until the voice contact connection is terminated. When the connection is terminated, the accumulated information for all the end points is sent to the EDU server.

This section describes the EDU containers that are created in IC 6.1.3. The tables list the contents of an example container. X represents the unique identification number for each end point. Y and Z represent sequence numbers within each end point's activities.

EDU elements

The EDU server updates the following EDU elements in the voice contact container when an EDU is created. These EDU elements are also updated when the contact reaches an end point in the call. The EDU server updates the EDU the elements that apply to the specific end point.

Container Name	Value	Description
agent	container number	Number of the agent container to be incremented with the information from this leg of the call.
agent_key	unique database record key	The key that retrieves the agent record from the database.
ani	phone number	Automatic Number Identification. The caller's 10-digit telephone number.
contactduration	number of seconds	Difference between contactcreatetime and contactendtime. This field is not meaningful if the EDU was stored and restored, or if the EDU was removed due to idleness. Note this value is the length of time the EDU was in memory, not the length of the call.
contactendtime	time in seconds	Time of the last Terminate method to the EDU or equivalent, such as removal of the EDU due to idleness.
dnis	phone number	Dialed Number Identification Service. Telephone number dialed by the contact.
dest	phone number	Telephone number of the destination you are trying to reach
ext	phone number	Telephone extension of the agent.
loginid	loginid	Loginid of the agent or end point.
orig	phone number	Telephone of the originator of this leg of the call.
phone	phone id	Agent identifier (ID) on the switch that corresponds with the loginid field.
queue	queue id	Identifier of the queue where the call is placed while waiting for an agent.

Container Name	Value	Description
ucid (Avaya DEFINITY only)	8-byte binary tag	Universal Call ID (UCID) - A unique tag the Avaya DEFINITY/Communication Manager assigns at the origination of each call that is not already associated with an existing UCID.
voice	container number	Number of the voice contact container to be incremented with the information from this leg of the call.

EDU containers

EDU containers are created after the call reaches an end point and a call state is entered. For example, an agent answers the call, the call is abandoned, the call is transferred and answered by a second agent, etc.

EDU containers are created for a queue only if the call is abandoned while in the queue. These queue containers are populated only for voice contact containers running on a system with an Avaya DEFINITY/Communication Manager. The following information is added to the voice contact container as the contact moves from one leg of the call to another.

Container Name	Value	Description
voice.X	delta time	For the first voice contact container (voice.1) in the call, this is base time (should be 0). For subsequent containers in the call, it is elapsed time in seconds since the creation of the EDU.
voice.X.abandon	reason for abandoning the call: "in queue", "while ringing", "while on hold"	If the exit reason of a call end point is "abandon", this container provides additional information.
voice.X.abandontime	number of seconds	The period of time, in seconds, that the contact was on the call before it was abandoned.
voice.X.agent_key	unique database record key	The key that retrieves the agent record from the database.
voice.X.conference.Y	time in seconds	The number of seconds between the creation time of the EDU and the beginning of the conference call.
voice.X.conference.Y.parties	number of parties	The number of parties participating in the conference call.
voice.X.conference.Y.dest.Z	phone number	The destination phone number (Z) of a conference call, where Z = 1, 2, 3, ...
voice.X.conference.Y.key	database key	The database key of the agent to whom the call was conferenced. This matches the value in voice.X.conference.Y.dest.
voice.X.connect	delta time	The elapsed time in seconds between the creation of the EDU and the time the call was connected.

Container Name	Value	Description
voice.X.destination	phone number	The phone number of the client at end point X.
voice.X.direction	inbound/outbound	The direction of the call to or from the agent.
voice.X.exit_reason	reason for exit: "normal", "transfer", "abandon", "other"	The switch supplies the reason for the termination of a call end point.
voice.X.holdtime.Y	time in seconds	The period of time spent on hold during instance Y.
voice.X.leg_id	unique id (UUID)	The unique leg_id of the current leg of the call.
voice.X.loginid	loginid	The loginid of the client at end point X.
voice.X.origin	phone number	The phone number (ANI) at the other end of end point X.
voice.X.queue	delta time	The elapsed time in seconds between the creation of the EDU and the time at which the switch reports the call as queued.
voice.X.queue_number	queue number	The queue number configured with the contact center switch as the VDN that front ends the queue.
voice.X.queuetime	time in seconds	The time reported by the switch between a call arriving on an inbound trunk of the switch and the call being sent to an agent queue.
voice.X.ringtime	time in seconds	The time reported by the switch between the agent phone starting to ring and the agent answering the phone.
voice.X.talktime	time in seconds	The time reported by the switch between the agent answering the phone and the agent hanging up the phone.

Container Name	Value	Description
voice.X.transfer	phone number	The phone number to which the call was transferred.
voice.X.ucid (Avaya DEFINITY TS only)	8-byte binary tag	Universal Call ID (UCID) - A unique tag the Avaya DEFINITY/Communication Manager switch assigns at the origination of each call that is not already associated with an existing UCID.

Advocate Qualifiers in the EDU container

This section describes how Business Advocate qualifiers are recorded in an EDU container and how they are associated with the agents and end points of the call where they are used. Avaya Business Advocate is documented in *Business Advocate Configuration and Administration*.

This section provides descriptions and examples for the following call scenarios:

- initial contact route
- call transfer to a service class
- abandoned call

This section also describes the data that is written by the Workflow server and the Telephony Server Adapter (TSA).

Initial Contact Route

To record Business Advocate qualifiers in the EDU and associate them with the leg of the contact, where the contact was processed, the following scenario is performed. This example refers to a call received over a voice channel on Avaya IC.

1. The contact is presented to the TS. The TS sends a request, to match the contact to an agent, to the TSA for the TS. The TSA sends the contact to the Workflow server to be qualified.
2. The Workflow server records the information used to route the contact in a new container named advocate.n.routinginfo.m.
   a. If the container number is zero, the Workflow server creates a new advocate container by writing the qualifier information to "advocate.+.routinginfo.+”.
   b. If the container number is non-zero, the Workflow server writes the qualifier information to "advocate.n.routinginfo.+”.

   Important:
   The flow must return the number of the advocate container ("n") to the TSA in the postqualification call. On the first request to qualify the contact, the TSA passes "0" to the flow. On subsequent requests, the TSA passes the “n” that was used in the first request.

   The TSA must keep a record of the number of times (“m”) it requested the qualification for a single contact and the number of the advocate container (“n”) that this request is using (the number returned in postqualification.)

3. When an agent is matched to the work, the TSA records the following information in the EDU, where agentid is the login id of the agent.

```
advocate.<agentid> = advocate.n
advocate.<agentid>.qualifiers = <qualifier string>
advocate.n.routinginfo.m.agentmatch=true

For example:
advocate.joe = advocate.1
advocate.joe.qualifiers= 2.2,4.6,5.3
advocate.1.routinginfo.1.agentmatch = true
```

4. The TSA asks the TS to route the call to “agentid”. When the TS creates a new container for the contact for the destination, the TS reads the advocate.<agentid>and advocate.<agentid>.qualifiers fields in the EDU and writes the following information into the voice.x container.

   Agentid is the login id of the agent.
   Leg_id# is a UUI value (32 bytes unique id: 3d35dcf9000000008723591d232f0002) that is generated by the VESP library upon requesting the number of the leg for this connection.
   <qualifier string> is the string of qualifiers written by the TSA into advocate.<agentid>.qualifiers field.

```
voice.x.qualifiers= <qualifier string>
voice.x.leg_id = leg_id#
advocate.n.leg_id = leg_id#

For example:
voice.2.qualifiers = 2.2,4.6,5.3
voice.2.leg_id = 3d35dcf9000000008723591d232f0002
advocate.1.leg_id = 3d35dcf9000000008723591d232f0002
```

5. The TS also writes voice.<x>.<qualifiers = <qualifier string> into the ADU container for the leg of the contact.
6. The TS then nulls out the <qualifier string> value in advocate.<agentid>.qualifiers.

Initial Contact Route Example

The following example illustrates a new call being routed to an agent.

1. The TSA requests call qualification from the Workflow server for a qualification and passes "0" to indicate that this is a first request.
2. The Workflow server qualifies the call. For this example, assume it uses the qualifiers set "2.2, 5.2, 6.3".
3. The Workflow server writes the routinginfo information to advocate.+.routinginfo.+ and returns "1" to the TSA in the postqualification call, as the advocate container for the contact.
4. The TSA sends a request to the Resource Manager server for an agent match. For this example, assume that the Resource Manager server returns “joe”, as the agent login.
5. The TSA writes the following fields to the EDU:

```
advocate.joe.qualifiers = 2.2,5.2,6.3
advocate.joe = advocate.1
advocate.1.routinginfo.1.agentmatch = true
```

6. Then the TSA sends a request to the TS to route the call to “joe”.
7. When the call arrives at “joe's" phone, the TS creates a voice.x container for the call and reads the value “advocate.1” from advocate.joe from the EDU and the 2.2,5.2,6.3 qualifiers string from advocate.joe.qualifiers, then writes it to the EDU:

```
voice.1.qualifiers = 2.2,5.2,6.3
voice.1.leg_id = 3d35dcf9000000008723591d232f0002
advocate.1.leg_id = 3d35dcf9000000008723591d232f0002
```

For this example, the EDU contains the following fields:

```
advocate.1.routinginfo.1.qualifiers = 2.2,5.2,6.3
advocate.1.routinginfo.1.waittreatmentstyle =  2
advocate.1.routinginfo.1.specificagentid=”
advocate.1.routinginfo.1.excludedagentid=”
advocate.1.routinginfo.1.action=route
advocate.1.routinginfo.lrmid_key = "

advocate.1.routinginfo.1.agentmatch = true
advocate.1.leg_id = 3d35dcf9000000008723591d232f0002
advocate.joe = advocate.1
advocate.joe.qualifiers = 2.2,5.2,6.3
voice.1.qualifiers = 2.2,5.2,6.3
voice.1.leg_id = 3d35dcf9000000008723591d232f0002
```

After these fields are written to the EDU, the media connector changes to null.

```
advocate.joe.qualifiers = “
```

Transfer Calls to Service Class

Service Classes are used to define a type of interaction work for Business Advocate.

When transferring a call or requesting a conference for a call:

1. The agent client calls the Unified Agent Directory (UAD) to transfer the call to a service class.
2. The UAD calls a transfer workflow.
3. The transfer workflow creates a new advocate container by writing the qualifier information to "advocate.+.routinginfo.+”.
4. The agent client sends a requests the TS to transfer the call to the wait treatment queue returned by the transfer workflow.
5. When the receiving TS sees the call queued to the wait treatment queue, this TS sends an incoming event to the TSA. (This may not be the same TS that transferred the call.)
6. The TSA accesses the highest numbered advocate container to get the qualifiers.
7. The TS routes calls to the agent using the steps starting from step 3.

Abandoned Calls

When a voice call is abandoned the following occurs:

1. The switch sends a CTI event to the TS indicating an abandoned call.
2. The TS informs the TSA of the abandoned call.
3. When the TS creates the voice container for the leg of the call representing the abandoned call, the TS writes:

   ```
   voice.x.abandoned = true
   ```

4. The TSA writes:

   ```
   advocate.n.routinginfo.m.abandoned = true
   ```

If a call is abandoned before an agent match is found, there are no “agentmatch” fields set in the advocate container. If the contact is abandoned after an agent match is found, the abandoned field exists in addition to the “agentmatch” field.

The TSA may send a route request to the TS, and the TS simultaneously sends an abandoned event to the TSA. Since the TSA receives a route fail if the call is abandoned, the TSA can write the abandon information in the routinginfo container.

However, if the call is abandoned after the TSA receives the route confirmation and before the agent receives the call, there is no abandon indication in the routinginfo container.

Also, if a voice contact is abandoned between the time the call is transferred from the agent to the wait treatment queue in a voice transfer, and the time the TS passes the event to the TSA, there may be no abandon in the routinginfo container.

Flow Data

The following is the information recorded by the workflow when qualifying a contact for Business Advocate. This information must be recorded by:

- the Business Advocate qualification flows for voice, email and chat
- the Business Advocate exception flows for voice, email and chat.
- the transfer flows processing virtual queues containing service classes for voice and email (there currently is no chat transfer to a service class).

This information is placed in a new container: advocate.n.routinginfo.m, where “n” is the number of the request to find a Business Advocate agent for this contact and “m” is the number of tries that this request has made to match the contact to an agent.

The fields that are written are:

Name	Value
advocate.n.routinginfo.m.qualifiers	The qualifiers for the contact, formatted as a comma separated list of category.qualifier e.g, 2.2,5.2,6.3.
advocate.n.routinginfo.m.waittreatmentstyle	Integer value representing the wait treatment style for this contact.
advocate.n.routinginfo.m.specificagentid	Agent identifier for specific agent. (an empty string indicates no specific agent specified.)
advocate.n.routinginfo.m.excludedagentid	Agent identifier for excluded agent (an empty string indicates no excluded agent specified.)
advocate.n.routinginfo.m.action	This is a string set to “route” indicating a route qualification, or transfer indicating a transfer qualification.
advocate.n.routinginfo.m.lrmid_key	pkey of the LRMid for a specific agent, (empty string if agentid empty)

Adapter Server Data

The information in this section is written by the Adapter server (TSA) when the server receives an agent match for a contact. This information is placed in a new container:

```
advocate.n.routinginfo.m
```

where “n” is the number of the request to find a Business Advocate agent for this contact and “m” is the number of tries that this request has made to match the contact to an agent.

Name	Value
advocate.n.routinginfo.m.agentmatch	true

The TSA also writes the couples:

Name	Value
advocate.<agentid>	advocate.n
advocate.<agentid>.qualifiers	The qualifiers for the call, formatted as a comma separated list of category.qualifier e.g, 2.2, 5.2, 6.3

where agentid = the login id of the agent to which the contact should be routed

Name	Value
<media>.<x>.qualifiers	The qualifiers for the contact, formatted as a comma separated list of category.qualifier e.g, 2.2,5.2,6.3
<media>.<x>.leg_id	The string representation of the WACD taskID.
advocate.<n>.leg_id	The string representation of the WACD taskID.

If the Adapter server (TSA) receives an abandon for the contact before the contact is delivered, the server writes the following to the EDU.

Name	Value
advocate.n.routinginfo.m.abandoned	true

Reporting from EDU containers

The Avaya TS does not calculate state durations (for example, connect duration, talk time, hold time) nor does it calculate event counts (for example, the number of times a voice contact was placed on hold). However, a reporting application can derive these values from the time stamps and other information in a voice contact container.

The following table explains how useful process measurement data can be derived from the information stored in voice contact containers.

Measurement	Definition	Derivation
Hold Count	The total number of times a voice contact was placed on hold.	The total number of voice.X.holdtime.Y events for a given instance of X.
Hold Time	The time spent on hold during hold instance Y.	Derived from voice.X.holdtime.Y.
Queue Time	The time reported by the switch between a voice contact arriving on an inbound trunk of the switch and the voice contact being sent to an agent queue. Queue time is available only on the switches that provide direct queue monitoring, which is the Avaya DEFINITY. The Nortel switches do not support direct queue monitoring.	Derived from value stored in voice.X.queuetime.
Ring Time	The time reported by the switch between agent phone starting to ring and agent answering the phone.	Derived from value stored in voice.X.ringtime.
Talk Time	The total time that the agents were connected to the telephone call, but not on hold. (Aspect CallCenter-specific.)	Derived from value stored in voice.X.talktime.

Switch support for EDU containers

The following table lists the EDU containers that are created on each of the switches supported in IC 6.1.3. Some are supported for agent and queue activity, some are supported only for agent activity, and some are not supported for any activity on the switch.

Container Name	Avaya DEFINITY	Aspect CallCenter	Nortel Meridian
voice.X	agent & queue	agent	agent
voice.X.abandon	agent & queue	agent	agent
voice.X.abandontime	agent & queue	agent	agent
voice.X.agent_key	agent	agent	agent
voice.X.conference.Y	agent	agent	agent
voice.X.conference.Y.parties	agent	agent	agent
voice.X.conference.Y.dest.Z	agent	agent	agent
voice.X.conference.Y.key	agent	agent	agent
voice.X.connect	agent	agent	agent
voice.X.destination	agent & queue	agent	agent
voice.X.direction	agent & queue	agent	agent
voice.X.exit_reason	agent & queue	agent	agent
voice.X.holdtime.Y	agent	agent	agent
voice.X.leg_id	agent & queue	agent	agent
voice.X.loginid	agent & queue	agent	agent
voice.X.origin	agent & queue	agent	agent
voice.X.queue	queue	no	no
voice.X.queue_key	agent & queue	no	agent (only if the call is external, coming through a trunk)
voice.X.queue_number	queue	no	no
voice.X.queuetime	queue	no	no
voice.X.ringtime	agent	agent	agent

Container Name	Avaya DEFINITY	Aspect CallCenter	Nortel Meridian
voice.X.talktime	agent	agent	agent
voice.X.transfer	agent	agent	agent
voice.X.ucid	agent & queue	no	no

Agent containers

As a voice contact moves between agents in the contact center, the Avaya TS stores information about the voice contact in a way that does not interfere with the statistics from the next or last agent to handle the voice contact. To do this, the Avaya TS automatically creates an agent container. An agent container is created for an agent at login. The agent is assigned an Agent Data Unit (ADU) identifier, known as an ADUID.

There are two types of agent containers**.**

- The first type of agent container stores information about a specific agent, such as the agent's login ID, the time the agent logged in, and the current number of active contacts for that agent.
- The second type of agent container stores aggregate information about a group of agents assigned to a specific system queue. The second type of agent container receives information from a server that supports the Avaya TS. This supporting server is known as the Telephony Queue Statistics server (TSQS).

The information that is collected in an agent container about the agent and the contacts the agent handles goes into that agent's ADU based on the ADUID. Information from an agent queue container goes into that agent's queue ADU.

Similarly, when a queue is defined in IC Manager, a separate ADU is created for the queue, and the queue is assigned an ADUID. This ADU stores information about contact traffic for the queue. When a service class is created, two ADUs are created for the service class.

Each agent involved in the voice contact gets a separate portion of the agent container. The first agent to handle the voice contact is assigned the name agent.1. Activities involving agent 1 are represented by values such as agent.1.callduration and agent.1.reasoncode. The next agent to handle the voice contact is assigned the name agent.2 with activity values agent.2.callduration, agent.2.reasoncode, and so on.

This section describes the ADU containers that are created in IC 6.1.3. The tables list the contents of an example container. X represents the unique identification number for each end point. Y and Z represent sequence numbers within each end point's activities.

Agent container example

This example follows an agent container from creation to termination.

1. The Avaya TS requests an EDU for a new voice contact, determines the login ID (or UUID) of the agent client that first handles the voice contact. It sets the name agent.+*loginid* to a value of *loginid*.
2. An agent client who already assigned to the Avaya TS using agent.*.*loginid* gets a watch event, because agent.1 was created with a matching value. From this watch event the agent client gets the EDUID and the current values in the EDU. Using the * token in assigns significantly reduces the overhead of multiple assigns. Refer to the *Electronic Data Unit Server Programmer's Guide* for a complete description of the * token and other container naming conventions.
3. The agent client handles the EDU, typically storing the data in the agent.! container, which expands to agent.1 because this is the first agent client to handle the call. The voice contact is routed to a second agent client.
4. The Avaya TS recognizes the transfer and marks the EDU with an agent.+*newloginid* and a value of *newloginid*. (This becomes agent.2.)
5. A different agent client, also assigned to the Avaya TS with agent.*.*newloginid,* sees this new EDU. The first agent client still sees the EDU, because the first agent's criteria still match (the agent.1 container still contains the first agent's login ID).
6. The first agent client continues to add data to the EDU during voice contact wrapup. The second agent client sees these changes in real time.
7. The first agent client decides to receive no more events about the EDU changes and sets agent.! to an empty string. This sets agent.1 and the first agent client gets a drop event, because the first agent client's assign criteria no longer matches. Agent.1 is empty and agent.2 is not the first agent's login ID.
8. At some point, the first agent client invokes a VDU.Terminate method to the EDU server. The EDU is no longer kept active for the first agent.

ADU containers

ADU containers are created for a queue only if the call is abandoned while in the queue. These queue containers are only populated for voice contact containers running on a system with an Avaya DEFINITY/Communication Manager.

The following table lists the ADU contents of an example voice contact container. X represents the contact number for this agent on the media channel.

Container Name	Contents
voice.X.eduid	The EDU associated with this contact.
voice.X.queue_info	The queue from which the call was received. This could be the ACD split or a VDN.
voice.X.queue_key	The database key for the queue from which the contact was received. This value is returned only if queue monitoring is enabled on the switch and is supported only on the Avaya DEFINITY TS because the other IC 6.1.3 switches do not support queue monitoring.
voice.X.sourcequeue	The queue from which the contact was received. This value is returned only if queue monitoring is enabled on the switch and is supported only on the Avaya DEFINITY TS because the other IC 6.1.3 switches do not support queue monitoring.
voice.X.state	The state of the agent: "loggedin", "loggedout", "active", "available", "busy", "wrapup".
voice.X.stdstate	The media channel state of the agent.
voice.X.stdstate.Y.active	The current media channel state of the agent as "active".
voice.X.stdstate.Y.active.reason	The reason the agent exited the "active" media channel state. This container is empty.
voice.X.stdstate.Y.active.starttime	The time that the current agent state (active) started.
voice.X.stdstate.Y.alerting	The current media channel state of the agent as "alerting".
voice.X.stdstate.Y.alerting.reason	The reason the agent exited the "alerting" media channel state. This container is empty.
voice.X.stdstate.Y.alerting.starttime	The time that the current agent state (alerting) started.

Container Name	Contents
voice.X.stdstate.Y.inactive	The current media channel state of the agent as "inactive".
voice.X.stdstate.Y.inactive.reason	The reason the agent exited the "inactive" media channel state. This container is empty.
voice.X.stdstate.Y.inactive.starttime	The time that the current agent state (inactive) started.
voice.X.stdstate.Y.initiating	The current media channel state of the agent as "initiating".
voice.X.stdstate.Y.initiating.reason	The reason the agent exited the "initiating" media channel state. This container is empty.
voice.X.stdstate.Y.initiating.starttime	The time that the current agent state (initiating) started.
voice.X.stdstate.Y.terminated	The current media channel state of the agent as "terminated".
voice.X.stdstate.Y.terminated.reason	The reason the agent exited the "terminated" media channel state. Contents are "terminated".
voice.X.stdstate.Y.terminated.starttime	The time that the current agent state (terminated) started.
voice.X.stdstate.Y.wrapup	The current media channel state of the agent as "wrapup".
voice.X.stdstate.Y.wrapup.reason	The reason the agent exited the "wrapup" media channel state. This container is empty.
voice.X.stdstate.Y.wrapup.starttime	The time that the current agent state (wrapup) started.

Switch support for ADU containers

The following table lists the ADU containers that are created on each of the switches supported in IC 6.1.3. Some are supported for agent and queue activity, some are supported only for agent activity, and some are not supported for any activity on the switch.

Container Name	Avaya DEFINITY	Aspect CallCenter	Nortel Meridian
voice.X.eduid	agent	agent	agent
voice.X.queue_info	agent	no	agent (external calls)
voice.X.queue_key	agent	no	agent (external calls)
voice.X.sourcequeue	agent	agent	agent (external calls)
voice.X.state	agent	agent	agent
voice.X.stdstate	agent	agent	agent
voice.X.stdstate.Y.active	agent	agent	agent
voice.X.stdstate.Y.active.reason	agent	agent	agent
voice.X.stdstate.Y.active.starttime	agent	agent	agent
voice.X.stdstate.Y.alerting	agent	agent	agent
voice.X.stdstate.Y.alerting.reason	agent	agent	agent
voice.X.stdstate.Y.alerting.starttime	agent	agent	agent
voice.X.stdstate.Y.inactive	agent	agent	agent
voice.X.stdstate.Y.inactive.reason	agent	agent	agent
voice.X.stdstate.Y.inactive.starttime	agent	agent	agent
voice.X.stdstate.Y.initiating	agent	agent	agent
voice.X.stdstate.Y.initiating.reason	agent	agent	agent
voice.X.stdstate.Y.initiating.starttime	agent	agent	agent
voice.X.stdstate.Y.terminated	agent	agent	agent
voice.X.stdstate.Y.terminated.reason	agent	agent	agent
voice.X.stdstate.Y.terminated.starttime	agent	agent	agent

Container Name	Avaya DEFINITY	Aspect CallCenter	Nortel Meridian
voice.X.stdstate.Y.wrapup	agent	agent	agent
voice.X.stdstate.Y.wrapup.reason	agent	agent	agent
voice.X.stdstate.Y.wrapup.starttime	agent	agent	agent

Reporting from an agent container

The Avaya TS updates the information in agent containers when a contact is terminated. This information is used for reports on agent activities. The following table lists the data the Avaya TS records and the reporting information provided by that data:.

Data Recorded	Calculation
Number of contacts offered	A tally of items posted to a specific ADU.
Number of contacts handled	Derived from container end point 1. Number of contacts handled is a combination of voice.1 and voice.1.talktime where talk time is greater than a specific period of time (for example, 2 seconds.)
Number of contacts abandoned while in queue, ringing, or on hold	Derived from container end points that include voice.abandon items. Data is recorded for each agent.
Average voice contact handling time	The amount of time from the start of the contact to the end of the contact. For example, from the time when the EDU is created to the time the contact is disconnected.
Average talk time per contact	Total talk time divided by the number of contacts handled. Because the switch reports talk time from its own perspective, talk time does not include the period in which a contact is on hold. Therefore, talk time should first be reduced by the following: voice.X.holdtime.Y (for all Ys in the container) divided by the number of contacts handled.
Average talk time per agent	Similar to calculation for average talk time per contact, except this one is specific to each container end point item voice.X.loginid, grouped by login ID. Data can be derived from the ADU by dividing total agent talk time by the number of agents.

Data Recorded	Calculation
Average wrap up time	Raw data is in the ADU itself. Every agent has an ADU after a successful voice.assign. After hanging up a voice contact, the agent transitions to a wrap up state, and the time of this transition is recorded in the agent's ADU. The time of the agent's next state transition is also recorded in the ADU. Therefore, wrap up time is the period between these two transition times. The average, therefore, is derived by dividing the total wrap up time for all voice contacts by the total number of voice contacts.
Average answer time	Total ring time for the number of contacts offered, divided by the number of contacts.
Average time before abandoning a voice contact	Raw data is derived from container end points that include voice.abandon items. Data is recorded for each agent. Average time before abandoning a voice contact is the sum of ring time, queue time, and talk time for all abandoned voice contacts, divided by the number of abandoned voice contacts.
Average number of transfers	Derived by dividing the number of contacts handled by the number of voice.X.transfer items in the container.
Number of times a contact is placed on hold	Raw data comes from the number of contacts handled. Calculation is as follows: number of voice.X.holdtime.Y items for all Ys within each X.
Duration of the current agent state	Raw data is in the ADU. For a given ADU, subtract the current time from the time the current agent state was recorded in the ADU.

Queue ADU fields

When a queue is defined in IC Manager, a separate ADU is created for the queue, and the queue is assigned an ADUID. This queue ADU stores information about contact traffic for the queue.

Note:

When section refers to queue ADUs, that includes Service Class ADUs. For more information, refer to the *Agent Data Unit Server Programmer's Guide*.

The following table contains the ADU values that are set by the Avaya TS and TSQS when the queue ADU is created. These values are static, they do not change throughout the life of the ADU.

Field Name	Description	Set By
acdname	The database key for the switch associated with this queue.	TS only (not TSQS)
connector	The UUID of the media connector server responsible for the queue.	TS only (not TSQS)
connectorname	The name of the media connector responsible for this queue.	TS only (not TSQS)
id	The media channel specific id of the queue. The id must be unique for a media channel in a specific site.	TS and TSQS
key	The database key assigned to this queue.	TS and TSQS
media	The type of media channel over which the queue is receiving contacts: voice, email, or web.	TS and TSQS
minimumagents	The fewest number of agents that should be assigned to one queue.	TS and TSQS
priority	The priority level assigned to the queue. It can be 1 for the highest priority down to 10 for the lowest.	TS and TSQS
queue_key	The database key assigned to this queue.	TS and TSQS
queueid	The media channel specific id of the queue. The id must be unique for a media channel in a specific site.	TS and TSQS
queuename	The name assigned to this queue when it was created in IC Manager.	TS and TSQS

Field Name	Description	Set By
servicelevel	The interval within which a queued contact should be assigned to an agent on the system. This time period value varies across media channels.	TS and TSQS
site	The name of the site to which the queue was assigned when it was created in IC Manager.	TS and TSQS
site_key	The unique database key assigned to the queue's site.	TS and TSQS
type	The type of ADU that is created. This value is "queue".	TS and TSQS

Service Class Record

Service Class records apply to Avaya Business Advocate. Avaya Business Advocate is documented in *Business Advocate Configuration and Administration*.

Service Class records define types of interaction work for Business Advocate using qualifier sets. Each service class is assigned a goal that is used to measure how quickly contacts in that service class should be serviced. Each Advocate queue has a one-to-one mapping with the Service Class.

This section describes the Service Class ADUs that Avaya Business Advocate (Advocate) sends to the ADU server. Service Class information is stored in 3 ADU data records:

- Service Class Summary record
- Service Class State record
- Service Class Detail record

Service Class Summary record

The Service Class Summary record contains data about a single service class on Avaya IC. This data is static (not dynamic) and is only updated when service class administration changes.

Advocate sends the following data to the Service Class Summary record on all of the ADU servers to which Advocate is assigned.

Data Item	Description	Updated when...
scid	Identifier of the service class as a string.	the service class is created or administered.
name	Name of the service class as a string.	the service class is created or administered.
description	Description of the service class as a string.	the service class is created or administered.
type	Type of resource, set to "serviceclasssummary" for this record as a string.	the service class is created.
lrmid_key	Identifier of the logical Resource Manager server associated with this ADU as an integer.	the service class queue is administered.
createtime	Time the record was initially created in the format: YYYY:MM:DD HH:MM:SS	the record is initially created.

Data Item	Description	Updated when...
createtimet	Time the record was created, in seconds, from 1/1/70 as an integer.	the record is initially created.
lastupdate	Time the record was last updated in the format: YYYY:MM:DD HH:MM:SS	the record is updated.
lastupdatetimet	Time the record was last updated, in seconds, since /1/70 as an integer.	the record is updated.
qualifiers	Service class qualifier as a string in a comma separated list arranged in ascending order of the category ids.	the service class is administered.
channel	Media channel represented by the service class. Allows media connectors to listen for notifications from only those service class that correspond to their media channel.	the service class is administered.
state	State of the service class as an integer from 1 to 8.	the email service class state changes. This field is only set for email service classes on distributed Advocate.
definedagents	Indicates that agents are administered for the service class as T (true) or F (false).	the first agent is defined for the service class or the last agent is removed from the service class. This field is only set for email service classes on distributed Advocate.
loggedinagents	Indicates the agents logged in to this service class are capable of servicing this service class as T (true) or F (false). Some agents may be logged in to the service class, but unavailable to Advocate due to a CTI link failure. They are not counted.	the first agent logs in to the service class or the last agent logs out of the service class. --OR-- the first agent is made available after covering a failed CTI link or the last agent is made available because that agent covers a failed CTI link. This field is only set for email service classes on distributed Advocate.

Data Item	Description	Updated when...
scgoalid	Service goal used for this service class as a string. (32 byte service goaln id)	the goal assigned to the service class changes.
lowerthreshold	Lower threshold value, in seconds, for the service goal associated with this service class as an integer.	the threshold changes or the goal associated with the service class changes.
upperthreshold	Upper threshold value, in seconds, for the service goal associated with this service class as an integer.	the threshold changes or the goal associated with the service class changes.
criticalthreshold	Critical threshold value, in seconds, for the service goal associated with this service class as an integer.	the threshold changes or the goal associated with the service class changes.
criticalduration	Specifies whether or not the critical threshold is used for the service class.	the criticalduration changes or the goal associated with the service class changes.

Service Class State record

The Service Class State record contains real time for all of the service classes on a single instance of Advocate. In a system with multiple instances of Advocate, there is a Service Class State record on each instance of Advocate. The load balancing flow uses this data to load balance between the Advocate instances. The Service Class State record is updated at a pre-configured time interval, default is every 5 seconds.

Advocate sends the following data to the Service Class State record on all of the ADU servers to which Advocate is assigned.

Data Item	Description	Updated when...
type	Type of resource, set to "allserviceclassstate" as a string.	the Resource Manager initializes.
lrmid_key	Identifier of the logical Resource Manager server associated with this ADU as an integer.	the Resource Manager initializes.
createtime	Time the record was initially created in the format: YYYY:MM:DD HH:MM:SS	the record is initially created.
createtimet	Time the record was created, in seconds, since 1/1/70 as an integer.	the record is initially created.
lastupdatetime	Time, in seconds, since the record was last updated in the format: YYYY:MM:DD HH:MM:SS.	the record is updated.
lastupdatetimet	Time the record was last updated, in seconds, since 1/1/70 as an integer.	the record is updated.
sc<scid>	Service class state for the service class with the id "scid". Refer to sc<scid> format on page 160 for more details. There is a sc<scid> field in each service class in the Resource Manager server.	one of the attributes on the list is changed.

sc<scid> format

The sc<scid> field in the Service Class State record uses the following format:

```
sc<scid>={0,6{"qualifiers", ">qualifiers>"},{"state","<n>"},
{"wat","<n>"},{"ewt","<n>"},{"ewtused","<1/0>"},
{"loggedinagentcount","<n>"}}
```

where <scid> is the 32 byte scid for this service class.

These fields may appear in any order in the array. When a single field in the array is written, the remaining fields must also be written. None of these fields should ever have a value of NULL.

The following table described the sc<scid> fields from the above example:

Data Item	Description	Updated when...
qualifiers	Service class qualifier as a string.	the service class is created or administered.
state	State of the service class as an integer from 1 to 8.	the state has changed, updated every 5 seconds.
wat	Waited advanced time of the service class, in milliseconds.	the wait advanced time has changed, updated every 5 seconds.
ewt	Estimated wait time of the service class.	the estimated wait time has changed, updated every 5 seconds.
ewtused	Indicates if the estimated wait time value is reliable and can be used as T (true) or F (false).	the value of the ewtused field has changed, updated every 5 seconds.
loggedinagentcount	Indicates the agents logged in to this service class are capable of servicing this service class as T (true) or F (false). Some agents may be logged in to the service class, but unavailable to Advocate due to a CTI link failure. They are not counted.	the number of logged in agents has changed, updated every 5 seconds.
recordstatus	Indicates that this record is no longer valid as "deleted".	the service class is deleted.

Service Class Detail record

The Service Class Detail record contains real time data for a single service class for a single instance of Advocate for a single media connector. The Service Class Detail record is updated at a pre-configured time interval, default is every 5 seconds.

Advocate sends the following data to the Service Class Detail record on all of the ADU servers to which Advocate is assigned.

Data Item	Description	Updated when...
qualifiers	Service class qualifier string.	The service class queue is created or administered.
type	Type of resource, set to "serviceclassdetail" as a string.	The service class queue is administered.
lrmid	Identifier of the logical Resource Manager server associated with this ADU as a string.	The service class queue is administered.
connector	UID of the media connector that created the record.	the record is initially created.
createtime	Time the record was initially created in the format: YYYY:MM:DD HH:MM:SS	the record is initially created.
createtimet	Time the record was created, in seconds, since 1/1/70 as an integer.	the record is initially created.
lastupdate	Time the record was last updated in the format: YYYY:MM:DD HH:MM:SS	the record is updated.
lastupdatetimet	Time the record was last updated, in seconds, since /1/70 as an integer.	the record is updated.
contactcount	Number of contacts currently in queue. Not including calls routed to a specific agent (see directcount). Represented as an integer that is >= 0.	a contact is sent to a queue, abandoned, or delivered to an agent (ringing). Contacts are removed from contactcount when they are alerting at the agent.

Data Item	Description	Updated when...
contactsoffered	Number of contacts that were sent to this queue since midnight. Not including contacts routed to a specific agent (see directcount). Represented as an integer that is >= 0.	a contact is sent to a queue. This count is still incremented if an agent is available and the contact is not sent to a queue.
deliveredqueuetime	Total queue time of all contacts delivered to agents since midnight as an integer. Not including contacts routed to a specific agent (see directdeliveredqueuetime). Represented as an integer.	a contact is delivered to an agent (ringing). Any contacts delivered to an agent without first being sent to a queue do not contribute queue time in this field.
deliveredahead	Number of contacts that resided in this queue for less time than the value set in the LowerThreshold field before they were delivered to agents, since midnight. Represented as an integer that is >= 0.	a contact is delivered to an agent. If the LowerThreshold field value is 0, no contacts are counted.
deliveredtarget	Number of contacts that resided in this queue for at least the value set in the LowerThreshold field, but less time than the value set in the UpperThreshold field before they were delivered to agents, since midnight. Not including contacts routed to a specific agent. Represented as an integer that is >= 0.	a contact is delivered to an agent. If the LowerThreshold field value is 0, contacts delivered to an agent without residing in the queue are counted.
deliveredbehind	Number of contacts that resided in this queue for more time than the value set in the UpperThreshold field, but less time than the value set in the CriticalThreshold field before they were delivered to agents, since midnight. Not including contacts routed to a specific agent. Represented as an integer that is >= 0.	a contact is delivered to an agent.

Data Item	Description	Updated when...
deliveredcritical	Number of contacts that resided in this queue for more time than the value set in the CriticalThreshold field, before they were delivered to agents, since midnight. Not including contacts routed to a specific agent. Represented as an integer that is >= 0.	a contact is delivered to an agent.
abandonedqueuetime	Total queue time, in seconds, of all abandoned contacts, since midnight. Not including contacts routed to a specific agent (see directdeliveredqueuetime). Represented as an integer.	a contact is abandoned. Queue time may start before midnight, but is recorded in the period in which the contact abandons.
abandonedahead	Number of contacts that resided in this queue for less time than the value set in the LowerThreshold field before they were abandoned, since midnight. Represented as an integer that is >= 0.	a contact abandoned.
adandonedontarget	Number of contacts that resided in this queue for at least the value set in the LowerThreshold field, but less time than the value set in the UpperThreshold field before they were abandoned, since midnight. Not including contacts routed to a specific agent. Represented as an integer that is >= 0.	a contact is abandoned.

Data Item	Description	Updated when...
abandonedbehind	Number of contacts that resided in this queue for more time than the value set in the UpperThreshold field, but less time than the value set in the CriticalThreshold field before they were abandoned, since midnight. Not including contacts routed to a specific agent. Represented as an integer that is >= 0.	a contact is abandoned.
abandonedcritical	Number of contacts that resided in this queue for more time than the value set in the CriticalThreshold field, before they were abandoned, since midnight. Not including contacts routed to a specific agent. Represented as an integer that is >= 0.	a contact is abandoned.
outflowed	Number of contacts left in the queue since midnight. These are contacts that were not abandoned or delivered to an agent. This may occur under certain error conditions or when the contact is transferred from one media connector to another.	a contact leaves the queue.
outflowqueuetime	Total queue time, in seconds, of all the contacts that left the queue without abandoning or being delivered to an agent, since midnight. Represented as an integer.	a contact leaves the queue.
specificcount	Current number of contacts in the service class queue for a specific agent. Represented as an integer that is >=0.	a contact is abandoned or delivered to an agent. Contacts are removed from specifccount when they are alerting at the agent.

Data Item	Description	Updated when...
specificoffered	Number of contacts sent to this queue for a specific agent, since midnight. Represented as an integer that is >=0.	a contact is sent to a queue. This count is still incremented if an agent is available and the contact is not sent to a queue.
specificdelivered	Number of contacts sent to this queue for a specific agent and delivered to an agent, since midnight. Represented as an integer that is >=0.	a contact is delivered to an agent
specificdeliveredqtime	Total queue time, in seconds, of all the contacts delivered to specific agents, since midnight. Represented as an integer.	a contact is delivered to an agent. Any contacts delivered to an agent without first being sent to a queue do not contribute queue time in this field.
specificabandoned	Number of contacts sent to this queue for a specific agent that abandoned, since midnight. Represented as an integer that is >=0.	a contact is abandoned.
specificabandonedqtime	Total queue time, in seconds, for all contacts for specific agents that abandoned from queue, since midnight. Represented as an integer.	a contact is abandoned. Queue time may start before midnight, but is recorded in the period in which the contact abandons.

ADU fields for statistical reporting

The following table contains the ADU values that represent the statistics of a given queue. These values are maintained for monitored queues except where indicated in the Notes column. They are changed by the Telephony Queue Statistics server (TSQS) with each update of the ADU.

Field Name	Definition	Notes
abandoned	The number of contacts that abandoned from this queue or from alerting since midnight.	
abandonedlasthour	The number of contacts that abandoned from this queue in the last hour.	
abandonedthishour	The number of contacts that abandoned from this queue during the current hour.	
abandonmiss	The number of contacts that abandoned from this queue or from alerting since midnight after waiting longer than the service level interval.	Not available on Aspect CallCenter
abandonqueuetime	The total queue time of all contacts that were abandoned from queue since midnight.	Not available on Nortel Meridian
acceptqueuetime	The total queue time of all contacts that were answered/accepted by agents since midnight.	Not available on Nortel Meridian
averagedelay	The average expected wait time for the queue.	
avgqueuetime	The average expected wait time for the queue. The same value as averagedelay.	
contactcount	The current number of contacts in the queue.	
contactsaccepted	The number of contacts routed to this queue that were answered/accepted by agents since midnight.	

Field Name	Definition	Notes
contactsoffered	The number of contacts routed to this queue since midnight.	
handledlasthour	The number of contacts answered/accepted in the last hour.	
handledthishour	The number of contacts answered/accepted during the current hour.	
minutes_in_first_hour	The number of minutes that make up the first hourly statistic.	
offeredlasthour	The number of contacts routed to this queue in the last hour.	
offeredthishour	The number of contacts routed to this queue during the current hour.	
oldest	The number of seconds the oldest contact has been in the queue. If the “oldest_as_timestamp” configuration parameter is set to true in IC Manager, then this value is the arrival time of the oldest contact in the queue.	Not available on Nortel Meridian
servicelevelmiss	The number of contacts that were not handled during the prescribed service level time. This includes all contacts that abandoned.	
servicelevelper	The percentage of calls that were not handled within the prescribed service level time.	
servicelevel	Number of seconds within which a contact is to be answered or accepted.	
ttlqueuetime	The total waiting time of all the contacts in the queue.	
updatetime	The time of the last update to the queue.	

ADU fields for MSHS support

The following table describes the ADU field names that were added to Avaya IC 6.1.3 to support Multi Site Heterogeneous Switch (MSHS) functionality. These fields are used by all of the switch versions of the Avaya TS:.

Field Name	Description	Set By
voice.acdname	The name of the switch that the Avaya TS is serving.	Avaya TS
voice.agent_key	The database key for the agent.	Avaya TS
voice.connector	The UUID of the Avaya TS that is responsible for agent events.	Avaya TS
voice.connectorname	The name assigned Avaya TS that is responsible for agent events.	Avaya TS
voice.device	The physical phone set where the agent is located.	Avaya TS
voice.device_address	Concatenation of the following fields: ● voice.device ● voice.connector Layout is: 423@400f013b0001000087235 91d580002	Avaya TS

Container implementation

Container implementation was modified in the Avaya IC 6.0.1 release to provide better call data for reporting. This section describes when containers are created in the course of the call in an eContact 5.6 environment versus an Avaya IC 6.x environment.

Refer to Call scenarios on page 170 for more information about container contents.

Interaction Center 6.0.1 and subsequent releases

In Avaya IC 6.0.1, the first container is not created until the call reaches an agent. Additional containers are created at each leg of the call.

If a call is abandoned in the queue while waiting for an agent, a container is created. Containers are not created for route points in Avaya IC 6.1.3.

The following example illustrates what occurs when a call is received by Avaya IC 6.0.1. For information on the contents of the containers in different call scenarios, refer to Call scenarios on page 170.

1. An Electronic Data Unit (EDU) is created when the call is received and ready for routing.
2. A container is not created when the call is routed to a queue to wait for an available agent.
3. The first container (voice.1) is created when the call is routed to an agent. Any data pertinent to queue times prior to the delivery of the call to the agent is also reported in this container. If the call is abandoned before it reaches the agent, this container is still created and holds data about this call in queue.
4. A second container (voice.2) is created when a call that is transferred to or conferenced with a second agent reaches the second agent. If a conference is cancelled or the first agent abandons the call, the second container is still created and holds information pertaining to the abandonment of the second leg of the call.
5. If the call is terminated at any point in this scenario, the statistics in the previous container are used for the reporting of the call.

Call scenarios

This section describes the EDU elements and values that are created in the following call scenarios in the Avaya IC 6.1.3 implementation of voice contact containers. This section also describes the voice contact containers that are created and the call state changes that occur during each call scenario.

- Route point
- Simple call scenario
- Abandoned call scenario (in queue, while ringing, on hold)
- Transfer call scenario (at a queue, at an agent, transfer to second agent)
- Conference call scenario (at a queue, at an agent, conference with a second agent)
- Consultative cancel scenario (during ring, after answered by the second agent)

Restrictions

Voice.X.queue_key containers are not provided if voice.X.queue_number containers are not supported by the switch. For example, not supported by Aspect CallCenter and supported on Nortel Meridian only for external calls.

Voice.X.queuetime containers are not supported on Aspect CallCenter and Nortel Meridian switches.

Route point

In all of the call scenarios in this section, the EDU is created when the call is received by the Avaya TS from the switch. This is the route point of Avaya IC. The following example lists the EDU elements and their corresponding values as they are found in the system log file:

```
"agent.+3d35dcf90000000008723591d237f0002" = ""
"agent_key" = ""
"ani" = "20002"
"calltype" = "direct"
"contactduration" = "0"
"contactendtime" = "1031331966"
"ctype" = "direct"
"dest" = "40010"
"dnis" = "40010"
"ext" = "40010"
"loginid" = "3d34943600040000 8723591d14250002"
"orig" = "20002"
"phone" = ""
"primary_ani" = "20002"
"primary_dnis" = "40010"
"type" = "voice"
"ucid" = "190020002010313319 66" (Avaya DEFINITY only)
```

Note:

The call is routed to a queue to wait for an available agent. The new elements that are written to the EDU at the queue overwrite these elements with the exception of the “primary_ani” element and the “primary_dnis” element. The “primary_ani” element and the “primary_dnis” element do not change throughout the call.

Simple call scenario

The call is sent to a queue to wait for an available agent. The call is routed to an agent when one becomes available.

This section contains the EDU elements and values that are created when the call is routed to the queue and then to the agent in this scenario. It also provides an example of the voice contact container that is created when the call reaches the agent.

Call at a queue

The Avaya TS notifies the EDU server the call is at a queue. The EDU server then updates the EDU with the following elements and values.

Note:

These elements are not populated in the voice contact container when a call is in queue for these switches.

```
"agent.+3d35dcf90000000008723591d232f0002" = ""
"agent_key" = ""
"ani" = "20002"
"contactduration" = "0"
"contactendtime" = "1031331966"
"dnis" = "40010"
"dest" = "40050"
"ext" = "40050"
"loginid" = "dd50117"
"orig" = "40010"
"phone" = ""
"ucid" = "19002000201031331966" (Avaya DEFINITY only)
```

Call at an agent

The Avaya TS notifies the EDU server the call has reached an agent. The EDU server updates the EDU with the following elements and values.

```
"agent.+dd50117" = "dd50117"
"agent_key" = "100006"
"ani" = "20002"
"contactduration" = "2"
"contactendtime" = "1031331968"
"dnis" = "40010"
"dest" = "20127"
"ext" = "20127"
"loginid" = "dd50117"
"orig" = "40010"
"phone" = "50117"
"queue" = "40050" (not present for Business Advocate)
"ucid" = "19002000201031331966" (Avaya DEFINITY only)
"voice.+dd50117" = "0"
```

Voice Contact Container

The following voice contact container is created when the agent answers the call.

```
"voice.1" = "0"
"voice.1.acdname" = "1fe"
"voice.1.agent_key" = "100006"
"voice.1.connect" = "1"
"voice.1.destination" = "20127"
"voice.1.direction" = "inbound"
"voice.1.exit_reason" = "normal"
"voice.1.loginid" = "dd50117"
"voice.1.leg_id" = "3d78e07e000a00008723591d23330002"
"voice.1.origin" = "40010"
"voice.1.queue" = "0" (Avaya DEFINITY only)
"voice.1.queue_number" = "40050" (Avaya DEFINITY only)
"voice.1.queuetime" = "0" (Avaya DEFINITY only)
"voice.1.ringtime" = "1"
"voice.1.talktime" = "1"
"voice.1.ucid" = "19002000201031331966" (Avaya DEFINITY only)
```

The following elements are written by the TSA on systems that are running Avaya Business Advocate:

```
"voice.1.qualifiers" = "a,b,c"
"advocate.<n>.leg_id" = "3d78e07e000a00008723591d23330002"
```

Call State Changes

The voice contact container records the following call state changes for a call that was answered by an agent in the appropriate sections of the voice contact container.

```
state.1031331966" = "created"
"voice.1.stdstate.1031331966.created.reason" = ""
"voice.1.stdstate.1031331966.created.starttime" = "1031331966"

"voice.1.stdstate.1031331966" = "alerting"
"voice.1.stdstate.1031331966.alerting.reason" = ""
"voice.1.stdstate.1031331966.alerting.starttime" = "1031331966"

"voice.1.stdstate.1031331967" = "active"
"voice.1.stdstate.1031331967.active.reason" = ""
"voice.1.stdstate.1031331967.active.starttime" = "1031331967"

"voice.1.stdstate.1031331968" = "wrapup"
"voice.1.stdstate.1031331968.wrapup.reason" = "101"
"voice.1.stdstate.1031331968.wrapup.starttime" = "1031331968"

"voice.1.stdstate.1031331969" = "terminated"
"voice.1.stdstate.1031331969.terminated.reason" = "101"
"voice.1.stdstate.1031331969.terminated.starttime" = "1031331969"
```

Abandoned call scenario

The call is sent to a queue to wait for an available agent. The call is routed to an agent when one becomes available.

The caller may abandon the call at various points in the call.

- At the queue if the caller who does not want to wait for an agent.
- At the agent if the call is put on hold.
- At the agent while the agent's phone is ringing.

This section provides the EDU elements and values that are created when the call is abandoned at each of these points in the call. It also provides an example of the voice contact container that is created in each of these instances.

Abandoned in queue scenario

The Avaya TS notifies the EDU server the call was abandoned at a queue. The EDU server updates the EDU with the following elements and values.

Note:

These elements are not populated in the voice contact container when a call is in queue for these switches.

```
"agent.+3d35dcf90000000008723591d232f0002" = ""
"agent_key" = ""
"ani" = "20002"
"contactduration" = "5"
"contactendtime" = "1031321506"
"dest" = "40050"
"dnis" = ""
"ext" = "40050"
"loginid" = "3d3494360004000008723591d14250002"
"orig" = "20002"
"phone" = ""
"ucid" = "190020000041026920227" (Avaya DEFINITY only)
```

Voice Contact Container

The following voice contact container is created when the call is abandoned in the queue.

```
"voice.1" = "0"
"voice.1.abandon" = "while in queue"
"voice.1.abandontime" = "5"
"voice.1.acdname" = "1fe"
"voice.1.destination" = "40050"
"voice.1.direction" = "inbound"
"voice.1.exit_reason" = "abandon"
"voice.1.leg_id" = "3d358f24000e00008723591d1f450002"
"voice.1.loginid" = "3d3494360004000087235 91d14250002"
"voice.1.origin" = "20002"
"voice.1.queue_key" = "6"
"voice.1.ucid" = "190020000 41026920227" (Avaya DEFINITY only)
```

Call State Changes

The voice contact container records the following call state changes for a call that was abandoned in queue in the appropriate sections of voice contact container, where "X" =1 and is incremented every time the stdstate changes.

```
"voice.1.stdstate.X" = "created"
"voice.1.stdstate.X" = ""
"voice.1.stdstate.X.starttime" = "1031321506"

"voice.1.stdstate.X" = "alerting"
"voice.1.stdstate.X.alerting.reason" = ""
"voice.1.stdstate.X.alerting.starttime" = "1031321506"

"voice.1.stdstate.X" = "terminated"
"voice.1.stdstate.X.terminated.reason" = "100"
"voice.1.stdstate.X.terminated.starttime" = "1031321506"
```

Abandoned on hold scenario

The Avaya TS notifies the EDU server the call was abandoned while on hold at the agent's phone. The EDU server updates the EDU with the following elements and values.

```
"agent.+dd50117" = "dd50117"
"agent_key" = "10006"
"ani" = "20002"
"contactduration" = "3"
"contactendtime" = "1031351528"
"dest" = "20127"
"dnis" = "40010"
"ext" = "20127"
"loginid" = "dd50117"
"orig" = "20002"
"phone" = "50117"
"queue" = "40050"
"ucid" = "190020002810313515 24" (Avaya DEFINITY only)
"voice.+dd50117" = "0"
```

Voice Contact Container

The following voice contact container is created when the call is abandoned while on hold.

```
"voice.1" = "0"
"voice.1.abandon" = "while on hold"
"voice.1.abandontime" = "1"
"voice.1.acdname" = "1fe"
"voice.1.agent_key" = "100006"
"voice.1.connect" = "1"
"voice.1.destination" = "20127"
"voice.1.direction" = "inbound"
"voice.1.exit_reason" = "abandon"
"voice.1.leg_id" = "3d792ce5000500008723591d23330002"
"voice.1.loginid" = "dd50117"
"voice.1.origin" = "20002"
"voice.1.queue" = "0" (Not present for Aspect and Nortel)
"voice.1.queue_number" = "40050" (Not present for Aspect and Nortel)
"voice.1.queue_key" = "6" (Not present for Aspect and Nortel)
"voice.1.queuetime" = "0" (Not present for Aspect and Nortel)
"voice.1.ringtime" = "1"
"voice.1.talktime" = "1"
"voice.1.ucid" = "190020002810313515 24" (Avaya DEFINITY only)
```

The following elements are created systems that are running Avaya Business Advocate:

```
"voice.1.qualifiers" = "a,b,c"
"advocate.<n>.leg_id" = "3d78e07e000a00008723591d23330002"
```

Call State Changes

The voice contact container records the following call state changes for a call that was abandoned while on hold at the agent's phone in the appropriate sections of voice contact container.

```
"voice.1.stdstate.1031351525" = "created"
"voice.1.stdstate.1031351525.created.reason" = ""
"voice.1.stdstate.1031351525.created.starttime" = "1031351525"

"voice.1.stdstate.1031351525" = "alerting"
"voice.1.stdstate.1031351525.alerting.reason" = ""
"voice.1.stdstate.1031351525.alerting.starttime" = "1031351525"

"voice.1.stdstate.1031351526" = "active"
"voice.1.stdstate.1031351526.active.reason" = ""
"voice.1.stdstate.1031351526.active.starttime" = "1031351526"

"voice.1.stdstate.1031351527" = "inactive"
"voice.1.stdstate.1031351527.inactive.reason" = ""
"voice.1.stdstate.1031351527.inactive.starttime" = "1031351527"

"voice.1.stdstate.1031351528" = "terminated"
"voice.1.stdstate.1031351528.terminated.reason" = "100"
"voice.1.stdstate.1031351528.terminated.starttime" = "1031351528"
```

Abandoned while ringing scenario

The Avaya TS notifies the EDU server the call was abandoned while the agent's phone was ringing. The EDU server updates the EDU with the following elements and values.

```
"agent.+dd50117" = "dd50117"
"agent_key" = "100006"
"ani" = "20002"
"contactduration" = "7"
"contactendtime" = "1031321570"
"dest" = "20127"
"dnis" = "40010"
"ext" = "20127"
"loginid" = "dd50117"
"orig" = "20002"
"phone" = "50117"
"queue" = "40050"
"ucid" = "19002000091031321563" (Avaya DEFINITY only)
"voice.+dd50117" = "0"
```

Voice Contact Container

The following voice contact container is created when the call is abandoned while the agent's phone is ringing.

```
"voice.1" = "0"
"voice.1.abandon" = "while ringing"
"voice.1.abandontime" = "7"
"voice.1.acdname" = "1fe"
"voice.1.agent_key" = "100006"
"voice.1.destination" = ""
"voice.1.direction" = "inbound"
"voice.1.exit_reason" = "abandon"
"voice.1.leg_id" = "3d78b7db000f00008723591d23330002"
"voice.1.loginid" = "dd50117"
"voice.1.origin" = "40050"
"voice.1.queue" = "0" (Not present for Aspect and Nortel)
"voice.1.queue_number" = "40050" (Not present for Aspect and Nortel)
"voice.1.queue_key" = "6"
"voice.1.queuetime" = "0" (Not present for Aspect and Nortel)
"voice.1.ucid" = "19002000091031321563" (Avaya DEFINITY only)
```

The following elements are created systems that are running Avaya Business Advocate:

```
"voice.1.qualifiers" = "a,b,c"
"advocate.<n>.leg_id" = "3d78e07e000a00008723591d23330002"
```

Call State Changes

The voice contact container records the following call state changes for a call that was abandoned while the agent's phone was ringing in the appropriate sections of voice contact container.

```
"voice.1.stdstate.1031321563" = "created"
"voice.1.stdstate.1031321563.created.reason" = ""
"voice.1.stdstate.1031321563.created.starttime" = "1031321563"

"voice.1.stdstate.1031321563" = "alerting"
"voice.1.stdstate.1031321563.alerting.reason" = ""
"voice.1.stdstate.1031321563.alerting.starttime" = "1031321563"

"voice.1.stdstate.1031321570" = "wrapup"
"voice.1.stdstate.1031321570.wrapup.reason" = "100"
"voice.1.stdstate.1031321570.wrapup.starttime" = "1031321570"

"voice.1.stdstate.1031321570" = "terminated"
"voice.1.stdstate.1031321570.terminated.reason" ="100"
"voice.1.stdstate.1031321570.terminated.starttime" ="1031321570"
```

Transfer call scenario

The call is sent to a queue to wait for an available agent. When an agent becomes available, the call is sent to that agent. The agent then transfers the call to another agent on the system.

This section provides the EDU elements and values that are created when the call is received by the first agent and when the call is transferred to and received by the second agent. It contains an example of the voice contact container that is created in each of these instances.

Call at agent 1

The Avaya TS notifies the EDU server the first agent answered the call. The EDU server updates the EDU with the following elements and values.

```
"agent.+dd50117" = "dd50117"
"agent_key" = "100006"
"ani" = "20002"
"contactduration" = "0"
"contactendtime" = "1031322397"
"dest" = "20127"
"dnis" = "40010"
"ext" = "20127"
"loginid" = "dd50117"
"orig" = "40010"
"phone" = "50117"
"queue" = "40050"
"ucid" = "190020001310313223 97" (Avaya DEFINITY only)
```

Voice Contact Container

The following voice contact container is created when the first agent answers the call:

```
"voice.1" = "0"
"voice.1.acdname" = "1fe"
"voice.1.agent_key" = "100006"
"voice.1.connect" = "1"
"voice.1.destination" = "20127"
"voice.1.direction" = "inbound"
"voice.1.exit_reason" = "transfer"
"voice.1.holdtime.+" = "3"
"voice.1.leg_id" = "3d78bb1d000b00008723591d23330002"
"voice.1.loginid" = "dd50117"
"voice.1.origin" = "40010"
"voice.1.queue" = "0"
"voice.1.queue_number" = "40050"
"voice.1.queue_key" = "6"
"voice.1.queuetime" = "0"
"voice.1.ringtime" = "1"
"voice.1.talktime" = "5"
"voice.1.transfer" = "50001"
"voice.1.ucid" = "19002000131031322397" (Avaya DEFINITY only)
```

The following elements are created systems that are running Avaya Business Advocate:

```
"voice.1.qualifiers" = "a,b,c"
"advocate.<n>.leg_id" = "3d78e07e000a00008723591d23330002"
```

Call State Changes

The voice contact container records the following call state changes for a call answered by the first agent in the appropriate sections of voice contact container.

```
"voice.1.stdstate.1031322397" = "created"
"voice.1.stdstate.1031322397.created.reason" = ""
"voice.1.stdstate.1031322397.created.starttime" = "1031322397"

"voice.1.stdstate.1031322397" = "alerting"
"voice.1.stdstate.1031322397.alerting.reason" = ""
"voice.1.stdstate.1031322397.alerting.starttime" = "1031322397"

"voice.1.stdstate.1031322398" = "active"
"voice.1.stdstate.1031322398.active.reason" = ""
"voice.1.stdstate.1031322398.active.starttime" = "1031322398"

"voice.1.stdstate.1031322403" = "inactive"
"voice.1.stdstate.1031322403.inactive.reason" = ""
"voice.1.stdstate.1031322403.inactive.starttime" = "1031322403"

"voice.1.stdstate.1031697454" = "terminated"
"voice.1.stdstate.1031697454.terminated.reason" = "102"
"voice.1.stdstate.1031697454.terminated.starttime" = "1031322406"
```

Transfer to agent 2

The Avaya TS notifies the EDU server the call was transferred to a second agent and the second agent answered the call. The EDU server updates the EDU with the following elements and values.

```
"agent.+dd50001" = "dd50001"
"agent_key" = "100007"
"ani" = "20127"
"contactduration" = "11"
"contactendtime" = "1031322408"
"dest" = "20001"
"dnis" = "20001"
"ext" = "20127"
"loginid" = "dd50001"
"orig" = "20127"
"phone" = "50001"
"ucid" = "19002000141031322403" (Avaya DEFINITY only)
```

Voice Contact Container

The following voice contact container is created when the second agent answers the transferred call.

```
"voice.2" = "6"
"voice.2.acdname" = "1fe"
"voice.2.agent_key" = "100007"
"voice.2.connect" = "8"
"voice.2.destination" = "20001"
"voice.2.direction" = "inbound"
"voice.2.exit_reason" = "normal"
"voice.2.leg_id" = "3d78bb23000500008723591d23330002"
"voice.2.loginid" = "dd50001"
"voice.2.origin" = "20127"
"voice.2.ringtime" = "2"
"voice.2.talktime" = "3"
"voice.2.ucid" = "190020001410313224O3" (Avaya DEFINITY only)
```

The following elements are created systems that are running Avaya Business Advocate:

```
"voice.1.qualifiers" = "a,b,c"
"advocate.<n>.leg_id" = "3d78e07e000a00008723591d23330002"
```

Call State Changes

The voice contact container records the following call state changes for a transferred call that was answered by a second agent in the appropriate sections of voice contact container.

```
"voice.2.stdstate.1031322403" = "created"
"voice.2.stdstate.1031322403.created.reason" = ""
"voice.2.stdstate.1031322403.created.starttime" = "1031322403"

"voice.2.stdstate.1031322403" = "alerting"
"voice.2.stdstate.1031322403.alerting.reason" = ""
"voice.2.stdstate.1031322403.alerting.starttime" = "1031322403"

"voice.2.stdstate.1031322405" = "active"
"voice.2.stdstate.1031322405.active.reason" = ""
"voice.2.stdstate.1031322405.active.starttime" = "1031322405"

"voice.2.stdstate.1031322408" = "terminated"
"voice.2.stdstate.1031322408.terminated.reason" = "101"
"voice.2.stdstate.1031322408.terminated.starttime" = "1031322408"
```

Note:

During a blind transfer, the switch issues the same set of events as a two step transfer, therefore, a blind transfer is reflected as a consultative transfer with the difference that “voice.1.holdtime.+ “ = “0”.

Conference call scenario

The call is sent to a queue to wait for an available agent. When an agent becomes available, the call is sent to the first agent. The first agent sets up a conference call between the first agent and another agent on Avaya IC.

This section provides the EDU elements and values that are created when a conference call is set up between agents and when the call is shared by both agents. It contains an example of the voice contact container that is created in each of these instances.

The EDU elements for the call at the queue are the same as the example in Simple call scenario on page 172, but the values associated with these elements vary based on the details of the call.

Note:

These elements are not populated in the voice contact container when a call is in queue for these switches.

Call at agent 1

The Avaya TS notifies the EDU server the first agent answered the call. The EDU server updates the EDU with the following elements and values.

```
"agent.+dd50117" = "dd50117"
"agent_key" = "100006"
"ani" = "20002"
"contactduration" = "18"
"contactendtime" = "1031322568"
"dest" = "20127"
"dnis" = "40010"
"ext" = "20127"
"loginid" = "dd50117"
"orig" = "20002"
"phone" = "50117"
"queue" = "40050"
"ucid" = "19002000161031322550" (Avaya DEFINITY only)
"voice.+dd50117" = "0"
```

Voice Contact Container

The following voice contact container is created when the first agent answers the call:

```
"voice.1" = "0"
"voice.1.acdname" = "1fe"
"voice.1.agent_key" = "100006"
"voice.1.conference.+" = "15"
"voice.1.conference.!.parties" = "3"
"voice.1.conference.!.dest.+" = "20002"
"voice.1.conference.!.dest.+" = "20001"
"voice.1.conference.!.dest.+" = "20127"
"voice.1.conference.!.key.+" = ""
"voice.1.conference.!.key.+" = "100007"
"voice.1.conference.!.key.+" = "100006"
"voice.1.connect" = "2"
"voice.1.destination" = "20127"
"voice.1.direction" = "inbound"
"voice.1.exit_reason" = "transfer"
"voice.1.holdtime" = "1"
"voice.1.loginid" = "dd50117"
"voice.1.leg_id" = "3d78bbb6000f00008723591d23330002"
"voice.1.origin" = "40050"
"voice.1.queue" = "0" (Not present on Aspect and Nortel)
"voice.1.queue_number" = "40050" (Not present on Aspect and Nortel)
"voice.1.queue_key" = "6"
"voice.1.queuetime" = "0" (Not present on Aspect and Nortel)
"voice.1.ringtime" = "2"
"voice.1.talktime" = "16"
"voice.1.ucid" = "19002000161031322550" (Avaya DEFINITY only)
```

The following elements are created systems that are running Avaya Business Advocate:

```
"voice.1.qualifiers" = "a,b,c"
"advocate.<n>.leg_id" = "3d78e07e000a00008723591d23330002"
```

Note:

The sub-container "voice.1.exit_reason" = "transfer" indicates that this party is the controller party and that it dropped from the call after the conference was established.

Call State Changes

The voice contact container records the following call state changes for a call that was answered by the first agent in the appropriate sections of voice contact container.

```
"voice.1.stdstate.1031322550" = "created"
"voice.1.stdstate.1031322550.created.reason" = ""
"voice.1.stdstate.1031322550.created.starttime" = "1031322550"

"voice.1.stdstate.1031322550" = "alerting"
"voice.1.stdstate.1031322550.alerting.reason" = ""
"voice.1.stdstate.1031322550.alerting.starttime" = "1031322550"

"voice.1.stdstate.1031322552" = "active"
"voice.1.stdstate.1031322552.active.reason" = ""
"voice.1.stdstate.1031322552.active.starttime" = "1031322552"

"voice.1.stdstate.1031322561" = "inactive"
"voice.1.stdstate.1031322561.inactive.reason" = ""
"voice.1.stdstate.1031322561.inactive.starttime" = "1031322561"

"voice.1.stdstate.1031322568" = "terminated"
"voice.1.stdstate.1031322568.terminated.reason" = "102"
"voice.1.stdstate.1031322568.terminated.starttime" = "1031322568"
```

Conference with agent 2

The Avaya TS notifies the EDU server the second agent answered the call after the first agent conferenced the second agent. The EDU server updates the EDU with the following elements and values.

```
"agent.+dd50001" = "dd50001"
"agent_key" = "100007"
"ani" = "20127"
"contactduration" = "19"
"contactendtime" = "1031322569"
"dnis" = "20001"
"dest" = "20001"
"ext" = "20001"
"loginid" = "dd50001"
"orig" = "20127"
"phone" = "50001"
"ucid" = "19002000171031322561" (Avaya DEFINITY only)
"voice.+dd50001" = "11"
```

Voice Contact Container

The following voice contact container is created when the first agent conferences the call to the second agent and the second agent answers the conferenced call:

```
"voice.2" = "11"
"voice.2.acdname" = "1fe"
"voice.2.agent_key" = "100007"
"voice.2.connect" = "13"
"voice.2.conference.+" = "15"
"voice.2.conference.!.parties" = "3"
"voice.2.conference.!.dest.+" = "20002"
"voice.2.conference.!.dest.+" = "20001"
"voice.2.conference.!.dest.+" = "20127"
"voice.2.conference.!.key.+" = ""
"voice.2.conference.!.key.+" = "100007"
"voice.2.conference.!.key.+" = "100006"
"voice.2.destination" = "20001"
"voice.2.direction" = "inbound"
"voice.2.exit_reason" = "normal"
"voice.2.leg_id" = "3d78bbc1000300008723591d23330002"
"voice.2.loginid" = "dd50001"
"voice.2.origin" = "20127"
"voice.2.ringtime" = "2"
"voice.2.talktime" = "6"
"voice.2.ucid" = "19002000171031322561" (Avaya DEFINITY only)
```

The following elements are created systems that are running Avaya Business Advocate:

```
"voice.1.qualifiers" = "a,b,c"
"advocate.<n>.leg_id" = "3d78e07e000a00008723591d23330002"
```

Call State Changes

The voice contact container records the following call state changes for a call in which when the first agent conferences the call to the second agent and the second agent answers the conferenced call in the appropriate sections of voice contact container.

```
"voice.2.stdstate.1031322561" = "created"
"voice.2.stdstate.1031322561.created.reason" = ""
"voice.2.stdstate.1031322561.created.starttime" = "1031322561"

"voice.2.stdstate.1031322561" = "alerting"
"voice.2.stdstate.1031322561.alerting.reason" = ""
"voice.2.stdstate.1031322561.alerting.starttime" = "1031322561"

"voice.2.stdstate.1031322563" = "active"
"voice.2.stdstate.1031322563.active.reason" = ""
"voice.2.stdstate.1031322563.active.starttime" = "1031322563"

"voice.2.stdstate.1031322569" = "terminated"
"voice.2.stdstate.1031322569.terminated.reason" = "101"
"voice.2.stdstate.1031322569.terminated.starttime" = "1031322569"
```

Note:

The TS can only provide the voice..conference elements if the switch provides enough data for the TS to determine a conference is in progress. During Multi-Site Heterogeneous Switching, the destination party on another switch does not receive the information that conference is in progress, and therefore, the TS cannot report on it.

Consultative cancel scenario

The call is sent to a queue to wait for an available agent. When an agent becomes available, the call is sent to the agent. This agent sets up a consultative call with another agent on the system. The agent then cancels the consultative call.

This section provides the EDU elements and values that are created when the call reaches the first agent. They are updated when a consultative call is cancelled at one of the following points in the call:

- While phone of the second agent is ringing
- After the second agent answered the phone

This section provides a voice contact container example for each of these instances.

Call at agent 1

The Avaya TS notifies the EDU server the first agent answered the call. The EDU server updates the EDU with the following elements and values.

```
"agent.+dd50117" = "dd50117"
"agent_key" = "100006"
"ani" = "20002"
"calltype" = "direct"
"contactduration" = "21"
"contactendtime" = "1031695356"
"ctype" = "direct"
"dnis" = "20127"
"dest" = "20127"
"ext" = "20127"
"loginid" = "dd50117"
"phone" = "50117"
"primary_ani" = "20002"
"primary_dnis" = "20127"
"orig" = "20002"
"type" = "voice"
"ucid" = "19002000051031695335" (Avaya DEFINITY only)
"voice.+dd50117" = "0"
```

Voice Contact Container

The following voice contact container is created when the first agent answers the call.

```
"voice.1" = "0"
"voice.1.acdname" = "1fe"
"voice.1.agent_key" = "100006"
"voice.1.connect" = "1"
"voice.1.destination" = "20127"
"voice.1.direction" = "inbound"
"voice.1.exit_reason" = "normal"
"voice.1.holdtime.+" = "3"
"voice.1.holdtime.+" = "7"
"voice.1.leg_id" = "3d7e6be7000500008723591d23330002"
"voice.1.loginid" = "dd50117"
"voice.1.origin" = "20002"
"voice.1.ringtime" = "1"
"voice.1.talktime" = "11"
"voice.1.ucid" = "19002000051031695335" (Avaya DEFINITY only)
```

The following elements are created systems that are running Avaya Business Advocate:

```
"voice.1.qualifiers" = "a,b,c"
"advocate.<n>.leg_id" = "3d78e07e000a00008723591d23330002"
```

Call State Changes

The voice contact container records the following call state changes for the call when the first agent answers it in the appropriate sections of voice contact container.

```
"voice.1.stdstate.1031695335" = "created"
"voice.1.stdstate.1031695335.created.reason" = ""
"voice.1.stdstate.1031695335.created.starttime" = "1031695335"

"voice.1.stdstate.1031695335" = "alerting"
"voice.1.stdstate.1031695335.alerting.reason" = ""
"voice.1.stdstate.1031695335.alerting.starttime" = "1031695335"

"voice.1.stdstate.1031695336" = "active"
"voice.1.stdstate.1031695336.active.reason" = ""
"voice.1.stdstate.1031695336.active.starttime" = "1031695336"

"voice.1.stdstate.1031695338" = "inactive"
"voice.1.stdstate.1031695338.inactive.reason" = ""
"voice.1.stdstate.1031695338.inactive.starttime" = "1031695338"

"voice.1.stdstate.1031695341" = "active"
"voice.1.stdstate.1031695341.active.reason" = ""
"voice.1.stdstate.1031695341.active.starttime" = "1031695341"
```

The following section of the voice contact container indicates a second connection to the second agent, where the consultation call is cancelled after the second agent answers the call.

```
"voice.1.stdstate.1031695346" = "inactive"
"voice.1.stdstate.1031695346.inactive.reason" = ""
"voice.1.stdstate.1031695346.inactive.starttime" = "1031695346"

"voice.1.stdstate.1031695353" = "active"
"voice.1.stdstate.1031695353.active.reason" = ""
"voice.1.stdstate.1031695353.active.starttime" = "1031695353"

"voice.1.stdstate.1031695356" = "terminated"
"voice.1.stdstate.1031695356.terminated.reason" = "101"
"voice.1.stdstate.1031695356.terminated.starttime" = "1031695356"
```

Consultation cancelled during ring with agent 2

The Avaya TS notifies the EDU server the first agent cancelled the consultative call while the second agent's phone was ringing. The Avaya TS considers this an abandoned call. The EDU server updates the EDU with the following elements and values.

```
"agent.+dd50001" = "dd50001"
"agent_key" = "100007"
"ani" = "20127"
"calltype" = "direct"
"dnis" = "20001"
"dest" = "20001"
"ctype" = "direct"
"contactduration" = "3"
"contactendtime" = "1031695341"
"ext" = "20001"
"loginid" = "dd50001"
"phone" = "50001"
"orig" = "20127"
"ucid" = "19002000061031695338" (Avaya DEFINITY only)
"voice.+dd50001" = "3"
```

Voice Contact Container

The following voice contact container is created when the consultative call is cancelled while the second agent's phone is ringing.

```
"voice.2" = "3"
"voice.2.acdname" = "1fe"
"voice.2.abandon" = "while ringing"
"voice.2.abandontime" = "3"
"voice.2.agent_key" = "100007"
"voice.2.exit_reason" = "abandon"
"voice.2.destination" = "20001"
"voice.2.direction" = "inbound"
"voice.2.leg_id" = "3d7e6bea000300008723591d23330002"
"voice.2.loginid" = "dd50001"
"voice.2.origin" = "20127"
"voice.2.ucid" = "19002000061031695338" (Avaya DEFINITY only)
```

The following elements are created systems that are running Avaya Business Advocate:

```
"voice.1.qualifiers" = "a,b,c"
"advocate.<n>.leg_id" = "3d78e07e000a00008723591d23330002"
```

Call State Changes

The voice contact container records the following call state changes for a consultative call that was cancelled while the second agent's phone was ringing in the appropriate sections of voice contact container.

```
"voice.2.stdstate.1031695338" = "created"
"voice.2.stdstate.1031695338.created.reason" = ""
"voice.2.stdstate.1031695338.created.starttime" = "1031695338"

"voice.2.stdstate.1031695338" = "alerting"
"voice.2.stdstate.1031695338.alerting.reason" = ""
"voice.2.stdstate.1031695338.alerting.starttime" = "1031695338"

"voice.2.stdstate.1031695341" = "terminated"
"voice.2.stdstate.1031695341.terminated.reason" = "100"
"voice.2.stdstate.1031695341.terminated.starttime" = "1031695341"
```

Consultation cancelled after agent 2 answered

The Avaya TS notifies the EDU server the first agent cancelled the consultative call after the second agent answered the call. The Avaya TS considers this a normal termination of the call. The EDU server updates the EDU with the following elements and values.

```
"agent.+dd50001" = "dd50001"
"agent_key" = "100007"
"ani" = "20127"
"dnis" = "20001"
"dest" = "20001"
"contactduration" = "7"
"contactendtime" = "1031695353"
"orig" = "20127"
"ext" = "20001"
"loginid" = "dd50001"
"phone" = "50001"
"ucid" = "19002000071031695346" (Avaya DEFINITY only)
"voice.+dd50001" = "11"
```

Voice Contact Container

The following voice contact container is created when the consultative call is cancelled after the second agent answered the call.

This is the third voice contact container in this consultative cancel call scenario. It assumes the first agent previously cancelled the consultation while the second agent's phone was ringing, then re-initiated the consultative call only to have it cancelled after the second agent answered the call.

```
"voice.3" = "11"
"voice.3.acdname" = "1fe"
"voice.3.agent_key" = "100007"
"voice.3.connect" = "15"
"voice.3.leg_id" = "3d7e6bf2000300008723591d23330002"
"voice.3.loginid" = "dd50001"
"voice.3.destination" = "20001"
"voice.3.direction" = "inbound"
"voice.3.exit_reason" = "normal"
"voice.3.origin" = "20127"
"voice.3.ringtime" = "4"
"voice.3.talktime" = "3"
"voice.3.ucid" = "19002000071031695346" (Avaya DEFINITY only)
```

The following elements are created systems that are running Avaya Business Advocate:

```
"voice.1.qualifiers" = "a,b,c"
"advocate.<n>.leg_id" = "3d78e07e000a00008723591d23330002"
```

Call State Changes

The voice contact container records the following call state changes for a consultative call that was cancelled after the second agent answered it in the appropriate sections of voice contact container.

```
"voice.3.stdstate.1031695346" = "created"
"voice.3.stdstate.1031695346.created.reason" = ""
"voice.3.stdstate.1031695346.created.starttime" = "1031695346"

"voice.3.stdstate.1031695346" = "alerting"
"voice.3.stdstate.1031695346.alerting.reason" = ""
"voice.3.stdstate.1031695346.alerting.starttime" = "1031695346"

"voice.3.stdstate.1031695350" = "active"
"voice.3.stdstate.1031695350.active.reason" = ""
"voice.3.stdstate.1031695350.active.starttime" = "1031695350"

"voice.3.stdstate.1031695353" = "terminated"
"voice.3.stdstate.1031695353.terminated.reason" = "101"
"voice.3.stdstate.1031695353.terminated.starttime" = "1031695353"
```

Chapter 8: Telephony Server API

This chapter describes the Avaya TS Application Programming Interface (API). It contains descriptions of each method in the Interface Definition Language (IDL) Specification, including input parameters, returns, exceptions, and examples.

This section includes the following topics:

Method Descriptions

TS.AnswerVDU

Syntax

```
ORBStatus AnswerVDU( in VDU_ID vduid );
```

Description

This method answers a telephone call, changing its state from Alerting to Answered. This method is invoked in response to an incoming call event.

Parameters

Value	Description
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_parameter	EDUID is invalid.
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problems.
vesp_resource_not_available	There is no call to answer.
vesp_service_not_available	Switch does not have this function.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";

status = Vesp_Request( "TS.AnswerVDU", callback, user_data, session,
                       my_vduid );
```

TS.Assign

Syntax

```
ORBStatus Assign( in string criteria );
```

Description

This method creates a session with the Avaya TS. When a session is created, events corresponding to the monitoring criteria are sent to the client.

Parameters

Input parameters correspond to criteria for monitoring and controlling devices during the current session.

Criteria	Description
number	Identifier of the physical phone. The type of device is qualified by the DS entry for the agent requesting the assignment. The TS recognizes ACD (EAS or MRD) devices and non-ACD devices.
*p	Indicates the value following this criteria is a physical device that allows for phone calls. The device can be EAS, ACD, or MRD.
*q	Queue (to monitor the activity on the queue). IC 6.1.3 also supports *vm is for backward compatibility.
*rVDN	Route point (receives a route request and and makes it available for a routing decision). This returns the Adjunct Route Requests, via association of TS signal on the MAPD, and filters the requests to return only those that originated from such a VDN. **Note**: Special case: *r*default, which indicates all route requests issued over the TS's signal are routed to the assigned client.
*s	SendData step posted from a CCT to a subtype that does not start with "gv" or "sv" that triggers a TS.SendData.Event to the client application. This parameter is only available on systems running with an Aspect CallCenter switch.
*v	VOX server port (to determine if the server sending the events is a VOX server and, if so, the port number used by the server).

Criteria	Description
*vm	VDN to monitor. A TS.CallOffered.Event is issued on the incoming call, but routing is not available and no EDUID is created.
*w	Route point that is a waiting device where a call is parked waiting to be routed by Business Advocate.
*wt	Defines a parking device like any other parking device, but as the default device to which the TS moves redirected calls Advocate calls triggered by the client application via a TS.RONA() request.
*a	The assign criteria for an ACD split (Avaya DEFINITY/Communication Manager only). Agent group whose skill set satisfies the requirements of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_assign_failure	Assign has failed.
vesp_bad_session	Client ID is invalid.

Examples

```
status = Vesp_Assign_Request( "TS.Assign", &ev, callback, user_data,
event_callback, session, "*p5112" );

status = Vesp_Assign_Request( "TS.Assign", &ev, callback, user_data,
event_callback, session, "*q701" );

status = Vesp_Assign_Request( "TS.Assign", &ev, callback, user_data,
event_callback, session, "*r1221" );

status = Vesp_Assign_Request( "TS.Assign", &ev, callback, user_data,
event_callback, session, "*v8000" );
```

TS.Busy

Syntax

```
ORBStatus Busy( void );
```

Description

This method makes an ACD teleset unavailable for ACD calls. No calls are received until a BusyTerminate() or Ready() is received. ACD calls are blocked. Direct calls are not blocked. If this method is called when the phone is in call, the phone state of Busy is pending instead of immediate because the phone is in call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problems.
vesp_service_not_available	Service not available.

Example

```
status = Vesp_Request( "TS.Busy", callback, user_data, session );
```

TS.BusyWithReason

Syntax

```
ORBStatus BusyWithReason( in string reasoncode );
```

Description

This method changes an agent's work mode to "AuxWork" and stores a reason code for this state change. This method is only supported on Avaya MultiVantage switches.

Parameters

Value	Description
reasoncode	Code that represents the reason for work mode change.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_failure	Request has failed; possible internal protocol problems.

Example

```
status = Vesp_Request( "TS.BusyWithReason", callback, user_data,
                       session, reasoncode );
```

TS.ConferenceCancelVDU

Syntax

```
ORBStatus ConferenceCancelVDU( in VDU_ID vduid );
```

Description

This method cancels a conference that began with the TS.ConferenceInit() method.

Parameters

Value	Description
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_bad_vduid	EDUID is invalid.
vesp_failure	Request has failed. Possible internal protocol problems.
vesp_illegal_state	State of the phone is not compatible with operation.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";
status = Vesp_Request( "TS.ConferenceCancelVDU", callback, user_data,
                       session, my_vduid );
```

TS.ConferenceCompleteVDU

Syntax

```
ORBStatus ConferenceCompleteVDU( in VDU_ID vduid );
```

Description

This method completes a conference initiated with the TS.ConferenceInitVDU() method. The party on Hold is joined to the other parties on the call.

Parameters

Value	Description
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_parameter	EDUID is invalid.
vesp_bad_session	Session is invalid.
vesp_bad_vduid	EDUID is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.
vesp_resource_not_available	No call to conference.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";
status = Vesp_Request( "TS.ConferenceCompleteVDU", callback, user_data,
                       session, my_vduid );
```

TS.ConferenceInitVDU

Syntax

```
ORBStatus ConferenceInitVDU( in VDU_ID vduid, in string dest );
```

Description

This method places a caller on Hold and dials a third party. If this method fails, every effort is made to automatically retrieve the caller on Hold. The EDUID is passed in the incoming call event the end point receives.

If successful, this method places the third party on the list of interested parties belonging to the EDU.

With MSHS activated, if the destination of the call is referenced by name and that name is not logically resolved, the TS uses MSHS to find out where to post the call.

Parameters

Value	Description
dest	Destination of the conference. The destination can be any of the following: Agent ID Number–The agent's teleset number is included in the conference. Name–The name is looked up in the Avaya IC Directory. If the telephone number field associated with the name contains an extension number, that extension is included in the conference. A sequence of digits to dial.
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_bad_vduid	EDUID is invalid.
vesp_busy	Destination was busy.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";
status = Vesp_Request( "TS.ConferenceInitVDU", callback, user_data,
                       session, my_vduid, "12345" );
```

TS.CreateQueueADU

Syntax

```
ORBStatus CreateQueueADU( in string queuename, out string aduid );
```

Description

This method enables the TS to create queue unique ADUs for the TSQS. When the TSQS requests a queue ADU from the TS, the TS checks the list of existing ADUs for the specified queuename. The TS provides the TSQS their queue ADUs from this list. If more ADUs are need, this method enables the TS to creates them.

Parameters

Value	Description
queuename	Name of the queue the TS checks for the list of existing ADUs.
aduid	Identifier of the ADUs on the list.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.

TS.Deassign

Syntax

```
oneway void Deassign( void );
```

Description

This method disconnects (deassigns) a session with the Avaya TS. When a session is deassigned, the flow of events from the Avaya TS to the client stops.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.

Example

```
status = Vesp_Deassign_Request( "TS.Deassign", &ev, NULL, OUL,
                                session );
```

TS.Divert

Syntax

```
TS.Divert ( in VDU_ID vduid, in string dest )
```

Description

This method moves a call from a queue, agent, or phone set to a new destination. To move a call from a phone set, the call must be in the Alerting state.

- For CSTA, calls can be moved from queues, agents, or phone sets.
- For Avaya DEFINITY/Communication Manager, calls can only be moved from agents and phone sets, not queues.

Parameters

Value	Description
vduid	EDUID of the call to be routed.
dest	Destination of the call. The destination can be any of the following: • Agent ID Number – The call is placed to the agent's teleset. • Name – The name is looked up in the Avaya IC Directory. If the telephone number field associated with the name contains an extension number, the call is routed to that extension. • Sequence of digits to dial.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_parameter	Invalid parameter.
vesp_bad_session	Session is invalid.
vesp_bad_vduid	Invalid EDUID.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
char         dest ="4001";

ORBStatus    status;

VDU_ID       my_vduid = "3016ace000700007800002c1b580002";

status = Vesp_Request("TS.Divert", callback, user_data, session, my_vduid,
                      destination);
```

TS.DropVDU

Syntax

```
ORBStatus DropVDU( in VDU_ID vduid, in string dest );
```

Description

This method is only supported on the Avaya DEFINITY/Communication Manager. It is used to relinquish a party from a conference.

For this method to operate properly, you need to know the party ID associated with the connection. The TS keeps track of the party ID internally by associating a connected number to the party ID.

For example, if you call 16175551212 and the call is redirected, the connected party could become 17815551313. The TS will inform IC of this on the "dest" field of a TS.Connect event. As a result, the TS will expect a TS.DropVDU() for the number 17815551313.

This method also allows for direct indication of the party ID through "*#n" syntax, where n is the party number to be dropped. TS.DropVDU(*#n) drops the second party in the call from a conference.

Note:

The TS requires CALL CONTROL in order to drop a party. If your environment contains any third party application that issues Third Party Call Control on the call against which you want to operate, the TS.DropVDU() fails.

Parameters

Value	Description
vduid	The EDUID of the call.
dest	The destination to be dropped, found in the conference event. A party cannot drop itself.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_failure	Request has failed; possible internal protocol problems.

Example

```
status = Vesp_Request( "TS.DropVDU", callback, user_data, session,
                       my_vduid, destination );
```

TS.GenericUpdate

Syntax

```
ORBStatus GenericUpdate( in string info );
```

Description

This method forces the Telephony server to update some of its internal settings.

Parameters

Value	Description
info	Telephony server internal settings.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
status = Vesp_Request( "TS.GenericUpdate", in string info );
```

TS.GetPBXTime

Syntax

```
ORBStatus GetPBXTime( out string time );
```

Description

This method gets the current time from the switch. This method is only supported on the Avaya DEFINITY/Communication Manager.

Parameters

Value	Description
time	The current time

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
status = Vesp_Request( "TS.GetPBXTime", out string info );
```

TS.GetPhoneInfo

Syntax

```
ORBStatus GetPhoneInfo( in string selection, out SeqCouple info );
```

Description

This method returns switch-dependent information about the state of a phone. If the client is an ACD or EAS agent, this method returns the ACD mode.

This method is not restricted to the Avaya DEFINITY/Communication Manager. However, for the Aspect CallCenter and the Nortel Meridian, the information that is returned is based on the TS's internal elements, not in the values returned from the switch. As such, the status passed by the TS may not match the status of the switch. Also, the data furnished by the TS to the Aspect and Nortel switches is restricted to "mode", which can contain the null (agent not logged in), ready, busy, and wrapup values.

Parameters

Value	Description
selection	Kind of information to be retrieved. You must be logged into the switch to retrieve all of the following values except login: ● "type" — Domain type and extension type of this agent's teleset. ● "login" — List of agents logged in to the split. login=[split]{,print}, where ",print" is optional and, if present, indicates the information should also be written to the log for compatibility with 5.6. ● "eas" or "acd" — Agent state information about both talk state (idle or in-call) and work mode state (busy, wrap-up, auto-ready, ready). ● empty string (" ") — If the agent's type is "eas" or "acd", requests the same information as "eas" or "acd", above. If the agent's type is neither "eas" or "acd", requests the same information as "station".
info	A sequence of couples containing call state, agent mode, and reason code. State can be "active", "idle", or "null". Agent mode can be "busy", "wrapup", "autoready", "ready", "null", or "loggedout".

Returns

Value	Description
vesp_success	Request was successful

Exceptions

Value	Description
vesp_failure	Request has failed; possible internal protocol problems.

Example

```
status = Vesp_Request( "TS.GetPhoneInfo", callback, user_data,
                       session, "", values );
```

TS.GetQueueInfo

Syntax

```
ORBStatus GetQueueInfo( in string queue, in string selection,
                        out SeqCouple info );
```

Description

This method returns switch-specific information about the queue. This method is only supported on the Avaya DEFINITY/Communication Manager.

Parameters

Value	Description
queue	Name of the queue to retrieve information about.
selection	Not used, but must be supplied as an empty string.
info	A sequence of couples containing the specified information.

Returns

Value	Description
vesp_success	Request was successful

Exceptions

Value	Description
vesp_failure	Request has failed; possible internal protocol problems.

Example

```
status = Vesp_Request( "TS.GetQueueInfo", callback, user_data,
                       queue, "", values );
```

TS.GetStatus

Syntax

```
ORBStatus GetStatus( out SeqCouples status );
```

Description

This method issues a request to the Telephony server to report its current status.

Parameters

Value	Description
status	A sequence of couples containing the current status of the Telephony server.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
status = Vesp_Request( "TS.GetStatus", out SeqCouples status );
```

TS.HangupVDU

Syntax

```
ORBStatus HangupVDU( in VDU_ID vduid );
```

Description

This method hangs up the voice portion of a call. The EDU remains active for any call wrap-up activities that are wanted by the business application.

Parameters

Value	Description
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_bad_vduid	EDUID is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.
vesp_service_not_available	Service is not available.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";

status = Vesp_Request( "TS.HangupVDU", callback, user_data, session,
                       my_vdu_id );
```

TS.HeteroSwitchHandoff

Syntax

```
ORBStatus HeteroSwitchHandoff( in VDU_ID vduid, in string target, in string
                               requesttype, in string requesthandle,
                               out string dest );
```

Description

This method is provided for a Multi Site Heterogeneous Switch environment. This method requests a phone number from the remote Telephony server to dial to connect to the target Telephony server for a given call.

Refer to Multi Site Heterogeneous Switch on page 107 for more information.

Parameters

Value	Description
vduid	EDUID of the call.
target	The alphanumeric target destination (usually an Agent ID or Queue name).
request type	Type of connection as makecall, transfer, or conference.
request handle	Internal pointer to the request map entry.
dest	Telephone number to dial from the remote TS to make the connection.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
status = Vesp_Request( "TS.HeteroSwitchHandoff", in VDU_ID vduid, in string
                       target, in string requesttype, in string requesthandle,
                       out string dest );
```

TS.HoldReconnectVDU

Syntax

```
ORBStatus HoldReconnectVDU( in VDU_ID vduid );
```

Description This method takes a call off Hold, making the call active.

Parameters

Value	Description
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_bad_vduid	EDUID is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.
vesp_resource_not_available	There is no call to take off Hold.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";
status = Vesp_Request( "TS.HoldReconnectVDU", callback, user_data, session,
                       my_vdu_id );
```

TS.HoldVDU

Syntax

```
ORBStatus HoldVDU( in VDU_ID vduid );
```

Description

This method places the voice portion of the call on Hold. The EDU may still be acted on by the application.

Parameters

Value	Description
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_bad_vduid	EDUID is invalid.
vesp_service_not_available	Service is not available.
vesp_resource_not_available	There is no call to put on Hold.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";
status = Vesp_Request ( "TS.HoldHoldVDU", callback, user_data, session,
                        my_vdu_id );
```

TS.Login

Syntax

```
ORBStatus Login( in string login, in string password, in string queue,
                 in string ext );
```

Description

This method logs the agent into the ACD to be able to receive, and handle, “queued” calls for the contact center. The agent does not have to login into the switch to receive direct calls.

Parameters

Value	Description
login	Logical identification of the agent (in other words, the Agent ID). For the Meridian Link, this is the position id “+” the agent id (406+0406), if, and only if, agent software is enabled in the ACD.
password	Agent’s password for login.
queue	Queue to which agent is logging in. For an ACD agent, a queue can be specified here. If a queue is not specified here, the Avaya TS retrieves the queue value that was specified in the agent configuration utility in IC Manager. For an EAS agent, this parameter is not applicable. The switch provides the queue information. Any value specified here is ignored.
ext	Equipment number.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request failed.

Example

```
status = Vesp_Request( "TS.Login", callback, user_data, session, "5009",
                       "", "", "4009" );
```

TS.Logout

Syntax

```
ORBStatus Logout( in string queue, in string ext );
```

Description

This method logs the agent out of the ACD. No more calls that are queued for the contact center are delivered to the teleset.

Parameters

Value	Description
queue	The queue from which the agent is logging off.
ext	Ignored. Included for compatibility with other versions of the Avaya TS.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request failed.

Example

```
status = Vesp_Request( "TS.Logout", callback, user_data, session,
                       "", "" );
```

TS.LogoutWithReason

Syntax

```
ORBStatus LogoutWithReason( in string queue, in string ext,
                            in string reasoncode );
```

Description

This method logs out an agent. If the session's phone is of type EAS, the TS requests the switch to log out this agent with the supplied reason code.

This method is only supported on the Avaya DEFINITY/Communication Manager.

Parameters

Value	Description
queue	Queue that the agent logged in to.
ext	Extension number of agent phone.
reasoncode	Code that represents the reason for the agent logout.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad-parameter	Queue or ext parameter null or length too long.
vesp_failure	An assign cannot be found for the TS request, the TS request is not from an ACD or EAS agent, or the TS request is from an agent with an empty equipment field.

Example

```
status = Vesp_Request( "TS.LogoutWithReason", callback, user_data,
                       session, queue, ext, reasoncode );
```

TS.MakeBusy

Syntax

```
ORBStatus MakeBusy( void );
```

Description

This method activates "do not disturb" which sets the phone set to a completely Busy state. The agent cannot perform any other functions while in this state.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
status = Vesp_Request( "TS.MakeBusy", callback, user_data_session );
```

TS.MakeBusyTerminate

Syntax

```
ORBStatus MakeBusyTerminate( void );
```

Description

This method deactivates "do not disturb" which takes the phone set out of a completely Busy state. The agent can now perform other functions on Avaya IC.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
status = Vesp_Request( "TS.MakeBusyTerminate", callback,
                        user_data_session);
```

TS.MakeCallSetVDU

Syntax

```
ORBStatus MakeCallSetVDU( in VDU_ID vduid, in string dest );
```

Description

This method initiates a call attempt using a specified EDU.

This method succeeds even if the destination is busy and an event reporting that the destination is busy is created.

With MSHS activated, if the destination of the call is referenced by name and that name is not logically resolved, the TS uses MSHS to find out where to post the call.

Parameters

Value	Description
dest	Destination of the make call, which can be any of the following: • Agent ID Number–The call is placed to the agent's teleset. • Name–The name is looked up in the Avaya IC Directory. If the telephone number field associated with the name contains an extension number, the call is placed to that extension. • Sequence of digits to dial.
vduid	EDUID of the call.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_bad_vduid	EDUID is invalid.
vesp_busy	Destination is busy.
vesp_failure	Request failed. Possible internal protocol problem.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";
status = Vesp_Request( "TS.MakeCallSetVDU", callback, user_data, session,
m_vduid, "400" );
```

TS.MakeCallVDU

Syntax

```
ORBStatus MakeCallVDU( in string dest, out VDU_ID vduid );
```

Description

This method initiates a call attempt. It generates a new EDU and returns it. The status of the call attempt is then reported back to the client.

This method succeeds even if the destination is busy and an event reporting that the destination is busy is created.

With MSHS activated, if the destination of the call is referenced by name and that name is not logically resolved, the TS uses MSHS to find out where to post the call.

Parameters

Value	Description
dest	Destination of the make call. The destination can be any of the following: Agent ID Number–The call is placed to the agent's teleset. • Name–The name is looked up in the Avaya IC Directory. If the telephone number field associated with the name contains an extension number, the call is placed to that extension. • Sequence of digits to dial.
vduid	EDUID of the call.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_bad_vduid	EDUID is invalid.
vesp_busy	Destination is busy.
vesp_failure	Request failed. Possible internal protocol problem.

Example

```
VDU_ID my_vduid;

status = Vesp_Request( "TS.MakeCallVDU", &ev, callback, user_data,
                       session, "400"' &my_vduid );
```

TS.PropertyUpdate

Syntax

```
ORBStatus PropertyUpdate( in string loginid, in SeqCouples SeqData );
```

Description

This method is used by the client application to inform the Telephony server of an ADUID change that resulted from a client crash.

Parameters

Value	Description
loginid	Login id of the client application.
SeqData	A sequence of couples containing the specified data with which to update the Telephony server. The name value of the couple must be a valid ADUID.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
status = Vesp_Request( "TS.PropertyUpdate", in loginid,
                                            in SeqCouples SeqData);
```

TS.Ready

Syntax

```
ORBStatus Ready( void );
```

Description

This method places a phone in the Ready state in preparation for receiving a telephone call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.
vesp_service_not_available	Service is not available.

Example

```
status = Vesp_Request( "TS.Ready", callback, user_data, session );
```

TS.ReadyAuto

Syntax

```
ORBStatus ReadyAuto( void );
```

Description

This method automatically places a phone set in the Ready state in preparation for receiving a call every time the phone is hung up.

This method is primarily supported on the Avaya DEFINITY/Communication Manager. The Nortel Meridian is configured for auto-in functionality by default, so it can call this method. The Aspect CallCenter does not use this method at all.

Returns

Value	Description
vesp_success	Request was successful

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.
vesp_service_not_available	Service is not available.

Example

```
status = Vesp_Request( "TS.ReadyAuto", callback, user_data, session );
```

TS.ReceiveData

Syntax

```
ORBStatus ReceiveData( in SeqCouples values );
```

Description

This method receives information from the Workflow server after a TS.SendData event is sent to the Telephony server from the Workflow server. This method should only be used for a host query operation.

This method is only supported on the Aspect CallCenter switch.

Parameters

Value	Description
values	A sequence of couples containing the values.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
status = Vesp_Request( "TS.ReceiveData", in SeqCouples values );
```

TS.ResetPhone

Syntax

```
ORBStatus ResetPhone( in string );
```

Description

This method causes the Avaya TS to terminate all the EDUIDs associated with a specific softphone. It also clears the ADUID sub-tree and contact count for the agent and sets the phone state to Busy or Ready depending on the softphone configuration.

Parameters

Value	Description
busy	Sets the phone to not Ready after the Reset is completed. The agent is unavailable for calls.
ready	Sets the phone to Ready after the Reset is completed. The agent is available for calls.

Returns

Value	Description
vesp_success	Request was successful.

Example

```
status = Vesp_Request( "TS.ResetPhone", callback, user_data, session,
                       "ready");
```

TS.Rona

Syntax

```
ORBStatus Rona( in VDU_ID vduid );
```

Description

This method re-routes a call to a second end point when there is no answer at the first route point.

In Business Advocate, this method moves the call back into a service queue.

Parameters

Value	Description
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
status = Vesp_Request( "TS.Rona", VDU-ID vduid );
```

TS.Route

Syntax

```
ORBStatus TS.Route( in VDU_ID vduid, in string dest )
```

Description

This method moves a call from a routing point to a destination such as a queue, agent, another routing point, or a phone set.

With MSHS activated, if the destination of the call is referenced by name and that name is not logically resolved, the TS uses MSHS to find out where to post the call.

Parameters

Value	Description
vduid	EDUID of the call to be routed.
dest	Destination of the call. The destination can be any of the following: Agent ID Number–The call is placed to the agent's teleset. Name–The name is looked up in the Avaya IC Directory. If the telephone number field associated with the name contains an extension number, the call is routed to that extension.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_parameter	Invalid parameter.
vesp_bad_session	Session is invalid.
vesp_bad_vduid	Invalid EDUID.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
char          dest[25]="4001";

ORBStatus     status;

VDU_ID        my_vduid = "3016ace000700007800002c1b580002";

status = Vesp_Request("TS.Route", callback, user_data, session, my_vduid,
                      destination);
```

TS.RouteWithInfo

Syntax

```
ORBStatus TS.RouteWithInfo( in VDU_ID vduid, in string dest,
                            in string info)
```

Description

This method moves a call from a routing point to a destination such as a queue, agent, another routing point, or a phone set.

With MSHS activated, if the destination of the call is referenced by name and that name is not logically resolved, the TS uses MSHS to find out where to post the call.

Parameters

Value	Description
vduid	EDUID of the call to be routed.
dest	Destination of the call. The destination can be any of the following: Agent ID Number–The call is placed to the agent's teleset. Name–The name is looked up in the Avaya IC Directory. If the telephone number field associated with the name contains an extension number, the call is routed to that extension.
info	Information to be carried across to the destination from the routing point. This field is passed to the vector and received as dialed digits. This allows the passing of EWT to Advocate calls.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_parameter	Invalid parameter.
vesp_bad_session	Session is invalid.
vesp_bad_vduid	Invalid EDUID.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
char          dest[25]="4001";

ORBStatus     status;

VDU_ID        my_vduid = "3016ace000700007800002c1b580002";

status = Vesp_Request("TS.RouteWithInfo", callback, user_data, session,
                      my_vduid, destination);
```

TS.SendDTMFtonesVDU

Syntax

```
ORBStatus SendDTMFtonesVDU( in VDU_ID vduid, in string tones );
```

Description

This method sends DTMF tones from a softphone to the switch, as though they were generated on the phone set keypad.

This method is only supported on the Avaya DEFINITY/Communication Manager.

Parameters

Value	Description
vduid	EDUID of the call.
tone	Maximum of 32 characters from the set 0-9, *, #.

Returns

Value	Description
vesp_success	Request was successful

Exceptions

Value	Description
vesp_failure	The input parameter exceeds 32 characters, or the write to the ASAI socket failed.

Example

```
status - Vesp_Request( "TS.SendDTMFtonesVDU", callback, user_data,
                        session, my_vduid, tones, destination, "12345" );
```

TS.SetContactState

Syntax

```
ORBStatus SetContactState( in VDU_ID vduid, in string state,
                           in string reason);
```

Description

This method enables the Telephony server Adapter to explicitly set the state of a contact.

Parameters

Value	Description
state	The standard state of the contact.
reason	The optional reason for the current state.
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
status = Vesp_Request( "TS.SetContactState", callback, user_data, session,
                       my_vduid, "terminated", "102" );
```

TS.StartTimer

Syntax

```
ORBStatus StartTimer( in SeqString strAgentIds);
```

Description

This method supports failover and recovery of the soft ACD to the hard ACD for IC environments running in Avaya Business Advocate mode. This method is used by the Resource Manager to inform the Avaya TS to start the agent's recovery timer for a passed list of agents.

Parameters

Value	Description
agent_ids	Sequence of strings that contain the agents login ids.

Returns

Value	Description
vesp_success	Request was successful

Exceptions

Value	Description
vesp_bad_session	Session is invalid
vesp_failure	Request has failed; possible internal protocol problems.
vesp_service_not_available	Service not available.

Example

```
status = Vesp_Request( "TS.StartTimer", callback, user_data, session,
                       pSeqString );
```

TS.SwapHeld

Syntax

```
ORBStatus SwapHeld( void );
```

Description

This method is used in a consultative transfer or conference call. It puts the active call on Hold and make the primary call, which is on hold, active.

This method is supported for the Avaya DEFINITY/Communication Manager and the Aspect CallCenter switches.

Parameters

None

Returns

Value	Description
vesp_success	Request was successful

Exceptions

Value	Description
vesp_bad_session	Session is invalid
vesp_failure	Request has failed; possible internal protocol problems.
vesp_service_not_available	Service not available.

Example

```
status = Vesp_Request( "TS.SwapHeld", callback, user_data, session );
```

TS.TransferCancelVDU

Syntax

```
ORBStatus TransferCancelVDU( in VDU_ID vduid );
```

Description This method cancels a transfer that was started with the TS.TransferInit() method.

Parameters

Value	Description
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed.
vesp_illegal_state	State of phone is not compatible with operation.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";

status = Vesp_Request( "TS.TransferCancelVDU", callback, user_data,
                       session, my_vduid );
```

TS.TransferCompleteVDU

Syntax

```
ORBStatus TransferCompleteVDU( in VDU_ID vduid );
```

Description

This method completes the transfer that was started with the TS.TransferInit() method. The party on Hold is connected with the called third party and the first party (the originator of the transfer) is hung up.

Parameters

Value	Description
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed.
vesp_illegal_state	State of phone is not compatible with operation.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";

status = Vesp_Request( "TS.TransferCompleteVDU", callback, user_data,
                       session, my_vduid );
```

TS.TransferInitVDU

Syntax

```
ORBStatus TransferInitVDU( in VDU_ID vduid, in string dest );
```

Description

This method places the call initiator on Hold and calls a third party. If this method fails, the call initiator is retrieved from Hold. The EDUID is passed in the incoming call event that the end point receives.

If successful, this method places the third party on the list of interested parties for the EDU.

With MSHS activated, if the destination of the call is referenced by name and that name is not logically resolved, the TS uses MSHS to find out where to post the call.

Parameters

Value	Description
dest	The destination of the transfer, which can be any of the following: Agent ID Number–The call is transferred to the agent's teleset. Name–The name is looked up in the Avaya IC Directory. If the telephone number field associated with the name contains an extension number, the call is transferred to that extension. Sequence of digits to dial.
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_illegal_state	State of phone is not compatible with operation.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";

status = Vesp_Request( "TS.TransferInitVDU", callback, user_data, session,
                       my_vduid, "5000" );
```

TS.TransferVDU

Syntax

```
ORBStatus TransferVDU( in VDU_ID vduid, in string dest );
```

Description

This method transfers a call and its EDU to a destination. The second party (being transferred) is momentarily placed on Hold, and the third party (receiving the transfer) receives an incoming call event. If successful, this method places the third party on the list of interested parties for the EDU.

With MSHS activated, if the destination of the call is referenced by name and that name is not logically resolved, the TS uses MSHS to find out where to post the call.

Parameters

Value	Description
dest	The destination of the transfer, which can either of the following: • Agent ID Number–The call is transferred to the agent's teleset. • Name–The name is looked up in the Avaya IC Directory. If the telephone number field associated with the name contains an extension number, the call is transferred to that extension.
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_illegal_state	State of phone is not compatible with operation.
vesp_failure	Request has failed. Possible internal protocol problem.
resource_not_available	The resource is currently unavailable.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";

status = Vesp_Request( "TS.TransferVDU", callback, user_data, session,
                       my_vduid, "4500" );
```

TS.Unpark

Syntax

```
ORBStatus Unpark( in VDU_ID vduid, in dest );
```

Description

This method is used in Business Advocate environments to divert a call that is waiting in a parking device.

Parameters

Value	Description
dest	The destination of the transfer, which can either of the following: • Agent ID Number–The call is transferred to the agent's teleset. • Name–The name is looked up in the Avaya IC Directory. If the telephone number field associated with the name contains an extension number, the call is transferred to that extension.
vduid	EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
VDU_ID my_vduid = "3016ace000700007800002c1b580002";

status = Vesp_Request( "TS.TransferUnpark", callback, user_data, session,
                       my_vduid, "4500" );
```

TS.WrapUp

Syntax

```
ORBStatus WrapUp( void );
```

Description

This method places the phone in a wrap-up state. This method is only useful for ACD-type sets. If this method is called when the phone is in call, the phone state of WrapUp is pending instead of immediate because the phone is in call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.
vesp_service_not_available	Service is not available.

Example

```
status = Vesp_Request ( "TS.WrapUp", callback, user_data, session );
```

TS.WrapUpComplete

Syntax

```
ORBStatus WrapUpComplete( in VDU_ID vduid );
```

Description

This method completes the wrap-up state on the phone.

Parameters

Value	Description
vduid	The EDUID of the call.

Returns

Value	Description
vesp_success	Request was successful.

Exceptions

Value	Description
vesp_bad_session	Session is invalid.
vesp_failure	Request has failed. Possible internal protocol problem.

Example

```
status = Vesp_Request ( "TS.WrapUpComplete", callback, user_data, session,
                        my_vduid, "4500" );
```

Obsolete Methods

The following TS methods are no longer supported by Avaya IC. The system returns an "Unsupported Procedure" error when you try to invoke them.

- TS.Answer
- TS.CallForward
- TS.CallForwardCancel
- TS.ConferenceCancel
- TS.ConferenceInit
- TS.ConferenceComplete
- TS.GetPBXConfig
- TS.GetPhoneConfig
- TS.GetQueueConfig
- TS.HangUp
- TS.Hold
- TS.HoldReconnect
- TS.MakeCall
- TS. MessageWaiting
- TS.MessageWaitingCancel
- TS.Transfer
- TS.TransferCancel
- TS.TransferComplete
- TS.TransferInit

Chapter 9: Telephony Server Events

This chapter provides descriptions of the Avaya TS Events, including their returns.

These events may contain additional information, which is passed from the switch, that is not provided in this chapter. For example, the Nortel switches may pass an "oparty" element and the Avaya DEFINITY/Communication Manager could pass a "partyid" element. These elements are not included in this chapter because that are not generated consistently and they are subject to change.

This section includes the following topic:

Event descriptions

TS.Abandoned

A call was abandoned before it was connected to an agent (for example, while waiting in a queue). TS.Abandoned is also issued if the call is terminated while it is ringing (alerting) at an agent desktop. It can also be issued if the call is on hold and is terminated by the phone that made the call into Avaya IC.

This event is sent to the Telephony Queue Statistics server (TSQS) for reporting purposes and to all assigned clients to the Avaya TS.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
vdu_id	EDUID of the abandoned call.
monitor	The ID of the device to which this event pertains.
number_in_queue	The calls currently waiting in the queue if the monitored device is a queue and there are calls in queue.

TS.AgentOtherWork

An agent is involved in work, not necessarily related to a prior call (for example, a meeting), and is not yet ready to receive a new incoming call.

Returns

Value	Description
agent_id	Agent ID on the switch of the agent who is now engaged in other work.
monitor	Device ID to which this event pertains.

TS.AuxWork

An agent is unavailable, but not performing wrap-up, and is not ready to receive a new incoming call. TS.AuxWork is very similar to TS.AgentOtherWork.

Returns

Value	Description
agent_id	Agent ID on the switch of the agent whose phone is involved in auxiliary work.
monitor	The ID of the device to which this event pertains.

TS.Busy

A call attempt received a busy signal, and the call was not connected.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
called	Number called, which was busy.
vdu_id	EDUID of the call.
monitor	The ID of the device to which this event pertains.

TS.Conference

A call was successfully conferenced.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
dest	The phone number or extension numbers of each of the parties participating in the conference call. For example, {"dest", 5087870428"}, {"dest", "234"}, {"dest", "236"}, {"dest", "238"}.
new_call_ref_id	Call reference ID as assigned by the switch.
number_in_call	The number of parties participating in the conference call.
orig	The phone number of the first party (the call's originator).
vdu_id	EDUID of original call.
monitor	The ID of the device to which this event pertains.

TS.Connect

A call was successfully connected.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
dest	Phone number to which the call was routed by the switch.
orig	The phone number of the party who placed the call.
vdu_id	EDUID of the call.
monitor	The ID of the device to which this event pertains.

TS.Disconnect

A call was disconnected.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
vdu_id	EDUID of the disconnected call.
monitor	The ID of the device to which this event pertains.
number_in_queue	The calls currently waiting in the queue if the monitored device is a queue and there are calls in queue.

TS.Diverted

A call has been taken from a queue and routed to an agent, or a call has been moved by the ACD from an alerting device to another destination.

This event is sent to the TSQS (for reporting purposes), to other monitoring parties associated with a given call.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
vdu_id	EDUID of the call.
dest	The called party of the event. If there is no called party, there is no dest value in this event.
monitor	The ID of the device to which this event pertains.

TS.Drop

A party has been dropped from a call involving two or more parties (for example, a conference or consultative transfer).

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
dest	The phone number to which the dropped party had been connected.
vdu_id	EDUID of the call.
monitor	The ID of the device to which this event pertains.

TS.Hold

A call has been put on Hold.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
dest	The number to which the call is connected (and now on Hold).
vdu_id	EDUID of the call.
monitor	The ID of the device to which this event pertains.

TS.HoldReconnect

A call has been retrieved from Hold.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
dest	The number to which the call is connected.
vdu_id	EDUID of the call.
monitor	The ID of the device to which this event pertains.

TS.IncomingCall

A phone is in a ringing state or has received a call set-up request. This event passes the EDUID of the incoming call to the appropriate client. When a contact is routed to different agents and queues, the ANI, DNIS, and queue values are changed accordingly.

Returns

Value	Description
ani	Automatic Number Identification, the caller's 10-digit telephone number.
called	The number used by the Avaya TS to place the call. This could be an equipment number, a logical ID, or a queue number.
call_ref_id	Call reference ID as assigned by the switch.
call_route_id	Call route ID, assigned by the switch, used to reference a route request.
dest	The phone number to which the call is being directed by the switch.

Value	Description
dnis	Dialed Number Identification Service, the phone number dialed by the contact.
orig	The phone number of the party who placed the call (or initiated the transfer or conference).
vdu_id	EDUID of the call.
calltype	Indication that the call was assigned from the queue or it was assigned directly to the agent. The same information is also included as "ctype" for backward compatibility.
monitor	The ID of the device to which this event pertains.
digits	DTMF digits entered along with the call.
queuetime	Period of time the call was in queue before it was delivered to an agent. This value is only available if the queue is monitored by the TSQS or the TS itself by having the TS in the queue's ts_set. Queue monitoring is only supported on the Avaya switches.

TS.Login

An agent has logged on to a phone set.

Returns

Value	Description
agent_id	Agent ID on the switch of the agent who logged on.
monitor	The ID of the device to which this event pertains.

TS.Logout

An agent has logged out of a phone set.

Returns

Value	Description
agent_id	Agent ID on the switch of the agent who logged off.
monitor	The ID of the device to which this event pertains.

TS.ObserverConnected

A service observer connected to listen on a call in progress at an agent/station.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
monitor	The ID of the device to which this event pertains.
vdu_id	EDUID of the call.
dest	Device "observing" this agent.

TS.ObserverDropped

A service observer disconnected from an agent/station.

This method is only supported on the Avaya DEFINITY/Communication Manager.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
monitor	The ID of the device to which this event pertains.
vdu_id	EDUID of the call.
dest	Device "observing" this agent.

TS.Queued

A call has been placed in queue.

Returns

Value	Description
acd_split	Supported on Avaya DEFINITY switches only, the ACD split to which the call was queued. This value may or may not be present in this event.
call_ref_id	Call reference ID as assigned by the switch.
vdu_id	The EDUID of the call.
monitor	The ID of the device to which this event pertains.

TS.Ready

An agent has become available to take incoming calls.

Returns

Value	Description
agent_id	Agent ID on the switch of the agent who is ready to receive incoming calls.
monitor	The ID of the device to which this event pertains.

TS.Ring

TS.Ring is issued when the call is delivered to a destination, but it is not answered yet.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
dest	The phone number to which the call was directed by the switch.
orig	The phone number of the party who placed the call (or initiated the transfer or conference).
vdu_id	EDUID of the call.
monitor	The ID of the device to which this event pertains.

TS.Rona

RONA (Redirected On No Answer) is identified by the TS when a sequence of events comprised of: Incoming, Divert, AfterCallWork, Disconnect are detected, with no agent activity between these events. The TS then generates the TS.Rona event. Note that the PBX redirected the call, and the agent was made unavailable. The switch must be configured for RONA.

Returns

Value	Description
agent_id	Agent ID on the switch of the agent who is ready to receive incoming calls.
monitor	The ID of the device to which this event pertains.

TS.SendData

TS.SendData is only supported on the Aspect CallCenter. The Aspect CallCenter supports the ability to send and receive data from a Call Control Table (CCT). TS.SendData posts information received from a CCT to the client application. The client application can respond to TS.SebdData in various ways. Typically it replies with either a TS.ReceiveData() request or a TS.Route() request.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
dest	Phone number to which the call was routed by the switch.
orig	Phone number of the party who placed the original call (first party).
route_id	Call route ID, assigned by the switch, used to reference a route request in debugging.
varA	Legacy Aspect TS specific call variable.
varB	Legacy Aspect TS specific call variable.
varC	Legacy Aspect TS specific call variable.
varD	Legacy Aspect TS specific call variable.
varE	Legacy Aspect TS specific call variable. The switch carries the vduid on this variable.
vdu_id	EDUID of the call.
monitor	The ID of the device to which this event pertains.

TS.SessionFailed

This event indicates that the session with the TS is no longer valid and should be treated like a TS.ServerFailed.Event.

TS.ServerFailed

The following possible scenarios generate a TS.ServerFailed.Event: an actual server crash, and a "simulated" device disconnection by the TS, or if the link between the PBX and the TS goes down, but the TS is configured with AbortOnLinkDown = false.

In either case the client has to recover via TS.Deassign() and TS.Assign() operations, which might cause reassignment to another TS if fail over is configured.

This event is generated if the link between the Avaya TS and the client goes down.

Returns

Value	Description
dest	The device where this event is posted.
reason	The reason this event is posted as either: • LinkDown • OutOfService

TS.Transfer

A call has been transferred.

Returns

Value	Description
call_ref_id	Call reference ID as assigned by the switch.
dest	The phone number or extension numbers of each of the parties to whom the call was transferred. For example, {"dest", 5087870428"}, {"dest", "234"}, {"dest", "236"}, {"dest", "238"}.
new_call_ref_id	Call reference ID as assigned by the switch.
number_in_call	The number of parties participating in the call transfer.
vdu_id	EDUID of the call.
monitor	The ID of the device to which this event pertains.

TS.Wrapup

An agent is involved in a wrap-up activity related to a previous call and is not ready to receive a new incoming call. The TS.Wrapup event, which is also called AfterCallWork on some switches, is issued in response to a TS.Wrapup() request on the softphone. It is also issued when the agent state is changed by the switch through the hardphone, RONA, or some other circumstance.

Returns

Value	Description
agent_id	ID of agent who is involved in after call work.
monitor	The ID of the device to which this event pertains.

Chapter 10: Telephony Server Alarms

This chapter describes the alarms that the Avaya TS generates. For each alarm, a description, alarm name, cause, and remedial course of action is provided.

If an alarm contains information specific to a call, that information is represented with a [*] followed by a description of the information below the alarm description. Second pieces of call specific information are indicated with a (**) followed by their description and third pieces of call specific information are indicated with a {***} followed by their description.

This section includes the following topic:

Alarms

The following table provides the alarm description that is displayed, the name of the alarm from the Name field in the Alarm Monitor, the priority level of the alarm, the cause of the alarm, and any remedial action that can be taken as a result of the alarm.

Note:

The name assigned to an alarm may vary based on where in the code the alarm is found. As such, an alarm can have more than one name.

Alarm	Name	Level	Cause
ADU.Find failed - cannot recover.	ADUContainers	high	A problem with the interaction of the TS and the ADU. **Remedial Action** Restart the ADU server. If the problem persists, report the problem to Avaya Technical Support.
ADUID processing turned OFF–ADU server not found.	Assign	info	ADU server not present in the Avaya IC environment. Not an error. **Remedial Action** None
Agent events [*] enabled. [*] = are/are not/aren't ● are not - configuration issues with the PBX or CVLAN server ● aren't - configuration issues with the TS installation or the CVLAN client	CpbxASAI	info	Indicates whether agent events such as AfterCallWork are (or are not) provided by the switch. **Remedial Action** None
Aspect CallCenter DataInterlink not specified	CpbxAspect	emergency	A required field for the Aspect CMI server was not provided. **Remedial Action** Specify a value for the Aspect CallCenter DataInterlink.

Alarm	Name	Level	Cause
Aspect CMI Server hostname not specified.	CpbxAspectCMI	emergency	A required field for the Aspect CMI server was not provided. **Remedial Action:** Specify a value for the Aspect CallCenter DataInterlink.
Assign to [*] (*) was not successful. Retry in [*] seconds. [*] = Other TS UUID (*) = Error description	AssignFail	low	The Telephony server was unable to connect to another Telephony server. **Remedial Action** Restart the Telephony server that could not make the connection. If the problem persists, report the problem to Avaya Technical Support.
Call [*] failed ANI validation, routed to default RP. [*] = call reference id	CtsServer	emergency	The TS could not validate the ANI for this call. The call was routed to the default routing point. **Remedial Action** If this pertains to all Multi Site Hetero Switch (MSHS) calls, update the ANI tables for all of the MSHS Telephony Severs to include the ANI for this call.
Call record deletion (in pbxDeassign) failed!	CpbxASAI	high	There is a call record that could not be found in the memory database. **Remedial Action** If this alarm occurs repeatedly, report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Call Record timed cleanup failed to initialize.	onInit	high	The program that cleans up existing call records could not be initialized. **Remedial Action** Restart the TS. If the problem persists, restart the ORB server and report the problem to Avaya Technical Support.
Could not create server default session.	onInit	high	The TS could not create a server default session. **Remedial Action** Restart the TS. If the problem persists, restart the ORB server and report the problem to Avaya Technical Support.
Could not initialize Request Collector.	onInit	high	The program that cleans up existing requests could not be initialized. **Remedial Action** Restart the TS. If the problem persists, restart the ORB server and report the problem to Avaya Technical Support.
Could not initialize Request Expire cleanup.	onInit	high	The program that cleans up existing requests could not be initialized. **Remedial Action** Restart the TS. If the problem persists, restart the ORB server and report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Could not load error table.	onInit	high	Error table could not be loaded into memory. Error messages at this layer do not contain any information. **Remedial Action** The most likely reason is that the process is out of memory. Attempt to adjust the process size and restart the TS.
Could not load reserved DN list.	RPDNLoadList	emergency	The reserved DN list could not be loaded. **Remedial Action** Verify the configuration of the reserved DN's table.
Could not load TS list.	TSLoadList	emergency	The Directory server (DS) did not provide a TS List to the TS. The DS might be down or it might not be functioning properly. **Remedial Action** Restart both the Directory server and the TS. If the problem persists, report the problem to Avaya Technical Support.
Could not load UserList from DS.	GenericUpdate onInit	emergency high	The Directory server (DS) did not provide a User List to the TS. The DS might be down or it might not be functioning properly or there may not be any agents configured for the site that the TS is using. **Remedial Action** Make sure agents are assigned to the site where that the TS is using. Restart both the Directory server and the TS. If the problem persists, report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Could not register timer for cross-assign to TS	OtherTS	high	There is an internal vesp error. **Remedial Action** Restart the TS for MSHS capabilities and report the problem to Avaya Technical Support.
CpbxCSTA:: DCE/RPC Exception Caught: [*] [*] = Error description	CpbxMeridian	high	There is a communication failure related to CTConnect. **Remedial Action** Restart the TS and the CTConnect server. If the problem persists, report the problem to Avaya Technical Support.
CSTA link partially initialized [*] threads up [*] = Number of monitor threads that were started.	CpbxCSTA	high	A partial failure of the TS integration occurred. **Remedial Action** Restart the TS and the CTConnect server. If the problem persists, restart the ORB server and report the problem to Avaya Technical Support.
Data received is greater than buffer size.	UUI Data	info	The data received in the UUI field from the switch does not conform to the expected size. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
DEFAULT_ROUTE_POINT not specified, will use specified DN.	onInit	low	Avaya IC is using the system default for the Default Route Point configuration parameter because a value was not specified for this parameter in IC Manager. **Remedial Action** Specify a value for the Default Route Point parameter in IC Manager.
Deferred Request NULL during ADU.FindByKey callback.	Assign	emergency	During a TS.Assign request, the TS attempts to obtain an ADUID for the client. This operation failed, which indicates memory corruption. **Remedial Action** Restart the TS and report the problem to Avaya Technical Support.
detected repeated key in UserList - load cancelled.	DSUserListLoad	emergency	There is a problem with the interaction of the TS and the Directory server while loading a list of agents. **Remedial Action** Restart the Directory server. If the problem persists, report the problem to Avaya Technical Support.
DS.Assign.Request failed, perform a TS.UPDATE to retrieve agent records.	Assign	high	The TS could not assign to the Directory server. **Remedial Action** Perform a TS.Update to retrieve agent records and report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
DS request failed for: [*]. [*] = request criteria	CtsMultiSite	low	The Directory server (DS) did not respond to a TS request. **Remedial Action** Restart both the Directory server and the TS and report the problem to Avaya Technical Support.
EDU/TS clocks are out of sync [*] - container info might be incorrect. [*] = current time (seconds)	CallContainer[1] CallContainer[2]	high high	The time on the EDU server is out of sync with the time on the TS. As a result, the information in the voice contact container could be incorrect. **Remedial Action** Use third party clock synchronization software to get the time on the servers back in sync.
Environment Variable\ AIXThread_Scope\ Is not set to [*]. TS needs that to optimize. [*] =	onInit	high	AIXTHREAD_SCOPE variable is not set to "S" within the AIX operating system. **Remedial Action** Set this variable (AIXTHREAD_SCOPE) to "S" and restart the TS.
Error loading acd queues	CtsQueueList::loadlist onInit	high high	There is a problem with the interaction of the TS and the Directory server while loading the list of ACD queues. **Remedial Action** Restart the Directory server. If the problem persists, report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Error requesting ADU for queue [*] [*] = queue id	CtsQueueList	low	There is a problem with the interaction of the TS and the ADU server. **Remedial Action** Restart the ADU server. If the problem persists, report the problem to Avaya Technical Support.
Error [*] requesting queue ADU [*] = queue id	CtsQueueList	low	There is a problem with the interaction of the TS and the ADU server. **Remedial Action** Restart the ADU server. If the problem persists, report the problem to Avaya Technical Support.
Error requesting TS.Route.	routeCallbackEvents	high	A Multi Site Hetero Switch communication could not be completed. **Remedial Action** Restart both of the TSes.
Error/Message Table load failed	CpbxASAI CpbxCSTA CctiDevice onInit	high high high high	An internal error occurred in the TS process which made the table load fail. **Remedial Action** Restart the TS. If the problem persists, report the problem to Avaya Technical Support.
Error updating acd queues list.	CtsQueue	high	There is a problem with the interaction of the TS and the Directory server while loading the list of ACD queues. **Remedial Action** Restart the Directory server. If the problem persists, report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Event Lock EXPIRED for [*] [*] = device monitor	CctiDevice	information	A monitored device waited the maximum allowed time for a VduSatisfied event. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
Event Monitor Request (510) failed: [*] (*) [*] = error code (*) = error description	CpbxAspectCMI	high	There is a communication error between the TS and the Aspect CMI server. **Remedial Action** Restart the TS. If the problem persists, report the problem to Avaya Technical Support.
Excessive call volume for [*]: (1) (2) [*] = device number (1) = number of events on previous pass. (2) = number of events on current pass.	scanForEvents	high	The call volume that is being processed by this TS is approaching the maximum supported call volume. **Remedial Action** First, review your deployment strategy, then report the problem to Avaya Technical Support.
Failed asai_open dest=[*] asai_errno (*) {*} [*] = switch address (name or ip address) (*) = error code {*} = error description	CpbxASAI	emergency	The TS was unable to open communication with the switch. **Remedial Action** Verify the network connections and the availability of switch signals. Restart the TS. If the problem persists, report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Failed asai_set_env asai_errno [*] (*) [*] = error code (*) = error description	CpbxASAI	emergency	The TS was unable to establish environment settings with the switch. **Remedial Action** Verify the network connections and the availability of switch signals. Restart the TS. If the problem persists, report the problem to Avaya Technical Support.
Failed heartbeat request, eRetCode [*] (*) [*] = error code (*) = error description	CpbxASAI	high	The TS was unable to get a heartbeat response from the switch. **Remedial Action** Restart the TS. If the problem persists, report the problem to Avaya Technical Support.
Failed set_env, asai_errno= [*] (*), TS cannot do routing errno={*} [*] = error code (*) = error description {*} = routing feature error number	CpbxASAI	high	The TS was unable to establish itself as a routing device to the switch. **Remedial Action** Restart the TS. If the problem persists, report the problem to Avaya Technical Support.
Failed to create entry on TS list for [*] [*] = other TS UUID	TSLoadFailure	high	An internal error occurred on the TS. **Remedial Action** Restart the TS. If the problem persists, report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Failed to get created agent info	DS.GetViewRecords	emergency	There is a problem with the interaction between the TS and the DS. **Remedial Action** Update the agent record within Avaya IC. If the problem persists, report the problem to Avaya Technical Support.
Failed to register for ECS heartbeats, TS will generate heartbeats	CpbxASAI	low	The TS sends periodic heartbeat requests to the switch. This error could indicate the MAPD has already registered for heartbeats or that this is a CVCT CVLAN server which does not allow heartbeats. **Remedial Action** None
Failed to retrieve switch version ABORT IN PROGRESS!!	CpbxASAI	emergency	The TS had a problem finding the version of the switch and is stopping. **Remedial Action** Restart the TS server. If the problem persists, report the problem to Avaya Technical Support.
Failed to retrieve switch version. Using Config Parameter	CpbxASAI	high	The TS failed to retrieve the switch version from the PBX. **Remedial Action** None. The TS will use the configuration parameter instead. If the problem persists, report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Failure on VDU.GetOneValue for [*] (*) [*] = vduid (*) = name	gvVDUGetValue	low	There is a problem with the interaction between the TS and the EDU server. **Remedial Action** Restart the EDU server. If the problem persists, report the problem to Avaya Technical Support.
Failure: internal queue assignment.	gvAssign queueAssign	high high	Monitoring of queues by the TS is enabled, but the assignment to a queue device failed. **Remedial Action** Restart the TS. The queue is not be monitored until the server is restarted.
Failure: internal route assignment [*]. [*] = assign criteria	routeAssign	high	A route point specified for the Multi Site Hetero Switch (MSHS) process could not be found. **Remedial Action** Verify the MSHS settings and restart the TS.
Failure: unable to create expiration timer.	RPDN: Loadlist	high	The program that controls the expiration of reserved DN's for Multi Site Hetero Switch communications could not be started. **Remedial Action** Restart the TS. If the problem persists, restart the ORB server and report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Heartbeat to test if switch connection is open failed.	CpbxASAI	high	The TS is unable to communicate with the switch. **Remedial Action** Verify the network connections and the availability of switch signals. Restart the TS. If the problem persists, report the problem to Avaya Technical Support.
Hetero-switch: ADU.FindByKey failed for destination [*]. [*] = destination number	cbFindAdvRouteAgentADU cbFindQueueADU findDestinationTS	low low low	A search for a destination resolution for a Multi Site Hetero Site communication failed. **Remedial Action** Restart the ADU server. If the problem persists, report the problem to Avaya Technical Support.
Hetero-switch: ADU request returned failure.	cbFindAdvRouteQueueADU cbFindAgentADU cbFindDeviceAddress cbFindTSFromAgentADU	low low info info	A search for a destination resolution for a Multi Site Hetero Site communication failed. **Remedial Action** Restart the ADU server. If the problem persists, report the problem to Avaya Technical Support.
Hetero-switch: Call [*] was routed with exception (**). [*] = vduid number (**) = error code	CtsServer	high	The Multi Site Hetero Site call was routed but there was an exception and the call may not reach the desired destination. **Remedial Action** Report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Hetero-switch: Failed to request info for ADU [*]. [*] = aduid number	cbFindAdvRouteAgentADU cbFindAdvRouteQueueADU cbFindAgentADU cbFindQueueADU	low low low low	There is a problem with the interaction of a TS and the ADU server in the processing of a Multi Site Hetero Switch call. **Remedial Action** Restart the ADU server. If the problem persists, report the problem to Avaya Technical Support.
Hetero-switch: Received invalid request.	cbFindAdvRouteAgentADU cbFindAdvRouteQueueADU cbFindAgentADU cbFindDeviceAddress cbFindQueueADU cbFindTSFromAgentADU cbFindTSFromQueueADU	low low low low low low low	There is a problem with the interaction of a TS and the ADU server in the processing of a Multi Site Hetero Switch call. **Remedial Action** Restart the TS and the ADU server. If the problem persists, report the problem to Avaya Technical Support.
Hetero-switch support disabled: invalid TS_Group.	onInit	high	The servers using the hetero-switch functionality are not properly setup in TS Groups. **Remedial Action** Setup the servers in their respective TS Groups in IC Manager.
Hetero-switch support disabled: no ANI in validation table.	onInit	emergency	The TS is set for Multi Site Hetero Switch support but no ANIs were defined. **Remedial Action** Fill the Multiple ANI table in IC Manager or disable validation.

Alarm	Name	Level	Cause
Hetero-switch support disabled: no reserved DNs.	onInit	emergency	The TS is set for Multi Site Hetero Switch support but no reserved DNs were defined. **Remedial Action** Fill the Reserved DNs table in IC Manager.
Hetero-switch support disabled: no TS list.	onInit	emergency	The TS is set for Multi Site Hetero Switch (MSHS) support but no other TS could be loaded. **Remedial Action** Verify the MSHS settings including the definition of the TS Group.
Hetero-switch support disabled: RESERVED_DN_TABLE not specified.	RPDN: Loadlist	low	The Reserved DN Table configuration parameter was not specified in IC Manager. **Remedial Action** Specify a value for the Reserved DN Table parameter in IC Manager.
IncomingAction with null Record|Event	CSM	high	There is an internal error on the TS. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
Internal queue [*] assignment–session mismatch. [*] = queue number	gvAssign queueAssign No Client found No Client (device) found	high high high high	A queue for monitoring was defined, but assignment failed due to a default session mismatch. **Remedial Action** Restart the TS. The queue will not be monitored until the server is restarted.

Alarm	Name	Level	Cause
Internal self assignment method failed to initialize.	onInit	high	The program that controls the internal assignments of the TS could not be started. **Remedial Action** Restart the TS. If the problem persists, restart the ORB server and report the problem to Avaya Technical Support.
Invalid argument for [*] on asai_vduid_compression [*] = argument	CpbxASAI	high	There is an internal error on the TS. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
Invalid incoming eDUID [*] - creating new one [*] = vduid	Container	high	The EDUID received in an event was not validated by the EDU server. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
Invalid session VDU.GetOneValue for [*] (*) [*] = vduid (*) = name	gvVDUGetValue	high	There is a problem with the interaction between the TS and the EDU server. **Remedial Action** Restart the TS. If the problem persists, report the problem to Avaya Technical Support.
Invalid values for CTCServer Node and/or Link Name	pbxLinkStart	emergency	The required field pbx type Csta was not provided for a Meridian TS. **Remedial Action** Specify values for all required fields.

Alarm	Name	Level	Cause
Link Down - TS restart in progress	pbxGetEvent Cpbx CpbxASAI	emergency emergency emergency	The connection between the TS and NetMerge (formerly CT Connect) or ASAI was lost. **Remedial Action** If the TS fails to restart by itself, restart the TS. If the problem persists, report the problem to Avaya Technical Support.
Link Initialization Failure	pbxLinkStart	high	The TS could not initialize the link with the CTConnect server. **Remedial Action** Restart the TS and the Intel NetMerge Call Processing (formerly CT Connect) server. If the problem persists, report the problem to Avaya Technical Support.
Link is Down	CpbxASAI Cpbx	emergency high/low	The connection was lost between the TS and the CTConnect server. **Remedial Action** Restart the TS and the Intel NetMerge Call Processing (formerly CT Connect) server. If the problem persists, report the problem to Avaya Technical Support.
Link is Up	CpbxASAI Cpbx	info info	The connection was established between the TS and the CTConnect server. **Remedial Action** None

Alarm	Name	Level	Cause
Link lost to ASAI	CpbxASAI	emergency	The connection was lost between the TS and the Avaya DEFINITY. **Remedial Action** Verify the network connections and the availability of switch signals. Restart the TS. If the problem persists, report the problem to Avaya Technical Support.
Link NOT initialized after [*] seconds. Still trying... [*] = number of seconds	pbxLinkStart	high	The TS cannot initialize a link with the NetMerge Call Processing (formerly CT Connect) server. The TS is still trying. **Remedial Action** If you get a Link Failure alarm, restart both the TS and the Intel NetMerge Call Processing server. If the problem persists, report the problem to Avaya Technical Support.
mismatch between cleanup and active lists [*] [*] = EDUID	CtsCallNode	high	A discrepancy was found in the end point of the call. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
missing acdname entry.	onInit	emergency	The ACD Name configuration parameter was not specified in IC Manager. **Remedial Action** Specify a value for the ACD Name parameter in IC Manager.

Alarm	Name	Level	Cause
missing pbx_model entry.	onInit	low	The ACD Model configuration parameter was not specified in IC Manager. **Remedial Action** Specify a value for the ACD Model parameter in IC Manager.
missing pbx_protocol entry.	onInit	low	The ACD Protocol configuration parameter was not specified in IC Manager. **Remedial Action** Specify a value for the ACD Protocol parameter in IC Manager.
missing pbx_type entry.	onInit	low	The ACD Type configuration parameter was not specified in IC Manager. **Remedial Action** Specify a value for the ACD Type parameter in IC Manager.
missing site entry.	onInit	low	The Site configuration parameter was not specified in IC Manager. **Remedial Action** Specify a value for the Site parameter in IC Manager.
missing site key entry.	onInit	low	The servers that use the same site key are out of sync. **Remedial Action** Restart the TS and report the problem to Avaya Technical Support. Have the pertinent log files available.

Alarm	Name	Level	Cause
NIVR: Could not find VDUID associated with ANI [*] [*] = ANI number	NIVR_UnknownVDUID	high	The EDUID associated with the ANI for this call could not be found. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
No ANI in validation table.	onInit	info	This TS has no ANI defined for its Multi Site Hetero Switch calls. If a destination TS is configured to perform ANI validation, calls originating from this TS will be routed to the default routing point. **Remedial Action** Fill in the Multiple ANI table in IC Manager or ensure that the other TSes participating in this group do not perform ANI validation.
No criteria for ADU.Find	ADUContainers	high	There is an internal error on the Avaya TS. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
Not enough memory	DSQueueList DSUserLoadList	emergency emergency	The TS was unable to create new structures due to a lack of available memory. **Remedial Action** Verify the processes running on the server machine and delete any unnecessary processes. Or upgrade the server with additional memory.

Alarm	Name	Level	Cause
PBX Layer: instantiation failure - check pbx_protocol entry.	onInit	high	The TS could not start communication with the switch. **Remedial Action** Check the ACD protocol parameter in IC Manager. If it is correct, report the problem to Avaya Technical Support.
PING to [*] failed [*] UUID of the other TS	Ping.Request	emergency	The TS was unable to communicate with another TS in a Multi-Site Hetero Switch environment. **Remedial Action** Verify the network connections and restart the TSes. If the problem persists, report the problem to Avaya Technical Support.
Possible ADU crash - ADU.Find initiated	ADUContainers	info	There is a problem with the interaction between the TS and the ADU server. **Remedial Action** If additional alarms are received related to the interaction of the TS and the ADU server, restart the ADU server. If the problem persists, report the problem to Avaya Technical Support.
Possible memory corruption! Garbage Collector Thread abnormally terminated. ABORT() IN PROGRESS!!	pbxGetEvent	emergency	Memory is corrupt. The internals of the TS are not working correctly and the TS is shutting down. **Remedial Action** If the TS fails to restart itself, restart the TS. If the problem persists, report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Possible memory leak on CtsRequest clearance: [*] [*] = number of pending requests	CtsClient	high	There were still some pending requests when the client disconnected from the TS. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
Reserved DN timed out: [*] from (**). [*] = reserved DN number (**) = originator TS UUID	RPDN Timeout	high	A routing point reserved for a Multi Site Hetero Switch call timed out waiting for the call. **Remedial Action** Verify the number of these occurrences in the server status at the Advanced tab in IC Manager. If this is not an isolated case, increase the time-out period at the MSHS tab in IC Manager.
Route Points MonitorChannel failed!	CpbxCSTA	emergency	The TS was unable to create monitors for routing. **Remedial Action** Restart the TS and the Intel NetMerge Call Processing (formerly CT Connect) server. If the problem persists, report the problem to Avaya Technical Support.
Route received on unreserved DN: Dest: [*] Dnis: (**) From:{***}. [*] = destination number (**) = dialed number {***} = ANI (origin) number	routeCallbackEvents	high	A call arrived at a reserved route point that was not expecting a Multi Site Hetero Switch. The call was routed to the default routing point. **Remedial Action** If the list of reserved DNs was modified after an Avaya TS in the TS Group was started, restart the TS.

Alarm	Name	Level	Cause
Session creation failed	LoadTSList hetero queueInternalAssign CtsQueue RPDN: Loadlist gvInternalAssign routeInternalAssign DSUserChangeEvent DSUserListLoad CtsMultiSite	emergency emergency emergency emergency emergency emergency emergency emergency emergency high	The TS was unable to create a session to communicate with other IC servers. **Remedial Action** Restart the TS. If the problem persists, restart the ORB server and report the problem to Avaya Technical Support.
TS ADU containers turned [*] during runtime. [*] = On or Off	AduContainers	info	The setting that controls if agent and queue containers are created was modified. **Remedial Action** None
TS.Assign.Request failed.	hetero	emergency	Cross assignment between TSes failed. Multi Site Hetero Switch communication between these servers is not available. **Remedial Action** Restart the Directory server and the TSes defined for this TS Group. If the problem persists, report the problem to Avaya Technical Support.
TS container create failed.	TS.Problem	low	The system did not create the requested container because of invalid data or insufficient memory. Containers for this call will not be available. **Remedial Action** Check the amount of RAM available to the process and restart the TS. If the problem persists, contact Avaya Technical Support.

Alarm	Name	Level	Cause
TS EDU containers turned [*] during runtime. [*] = On or Off	EduContainers	info	The setting that controls if voice contact containers are created was modified. **Remedial Action** None
TS not in its own TS Group.	CtsMultiSite	high	The TS is not properly setup in TS Groups. **Remedial Action** Setup the TS in its proper TS Group in IC Manager.
TS set for hetero-switch support.	onInit	info	The TS is configured to support hetero-switch functionality. **Remedial Action** None
Unable to assign to ADU server	CtsQueueList	high	The TS cannot assign to the ADU server.
Unable to assign to ADU after 3 tries	CtsQueueList	high	The TS cannot assign to the ADU server after 3 attempts to do so.
Unable to request additional info for TS [*] [*] = other TS UUID	CtsOtherTSList	high	The is a problem with the interaction between the Directory server and the TS. **Remedial Action** Restart the Directory server. If the problem persists, report the problem to Avaya Technical Support.
Unable to request updated configuration for Hetero-switch support.	CtsMultiSite:: loadDSTableAsync CtsMultiSite::updateTables	high high	An attempt to update the TS with the Multi Site Hetero Switch configuration failed. **Remedial Action** Restart both the Directory server and the Avaya TS. If the problem persists, report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
Unable to set voice.contactcount to zero.	Assign	high	The interaction between the TS and the ADU server has experienced a problem in processing a Multi Site Hetero Switch call. **Remedial Action** Restart the TS and the ADU server and report the problem to Avaya Technical Support.
UUI field does not seem to contain VDUID (size < 16).	CpbxASAI	high	There was an internal error on the TS. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
UUI field does not seem to contain VDUID (size < 32).	CpbxASAI	high	There was an internal error on the TS. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
VDU.Create.Request failed - CtsCallRecord removed.	IncomingCall MakeCall	high high	The is a problem with the interaction between the TS and the EDU server. **Remedial Action** Restart the EDU server. If the problem persists, report the problem to Avaya Technical Support.
VDU.Create.Request failed.	SendData	high	The is a problem with the interaction between the TS and the EDU server. **Remedial Action** Restart the EDU server. If the problem persists, report the problem to Avaya Technical Support.

Alarm	Name	Level	Cause
VDUID compression failed.	CpbxASAI	high	There was an internal error on the TS. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
VDUID decompression failed.	CpbxASAI	high	There was an internal error on the TS. **Remedial Action** If this is not an isolated occurrence, report the problem to Avaya Technical Support.
Vesp_Request failed.	MakeCallSetVDU MakeCall ADU.GetSubTree	emergency emergency emergency	The TS is unable to communicate with the other IC servers. **Remedial Action** Restart the TS. If the problem persists, restart the ORB server and report the problem to Avaya Technical Support.
Vesp_Request VDU.GetOneValue createtime failed:[*], ignoring tscontainer. [*] = error code	TS.Problem	high	EDU touch was not successful. It is possible to retire this EDUID without the TS knowing. **Remedial Action** Verify that the EDU server is running.
Will not request DS.Assign based on Load User Error. Use Update on TS instead.	onInit	high	The TS will not receive changes to agent records in IC Manager. A TS.Update is required for agent record changes to take affect. **Remedial Action** If this is not a desired behavior and if the problem persists, report the problem to Avaya Technical Support.

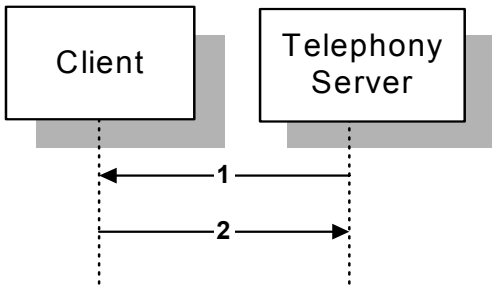
Appendix A: Generic Call Flows

This appendix illustrates and describes some generic call flows.

This section includes the following topics:

Route call

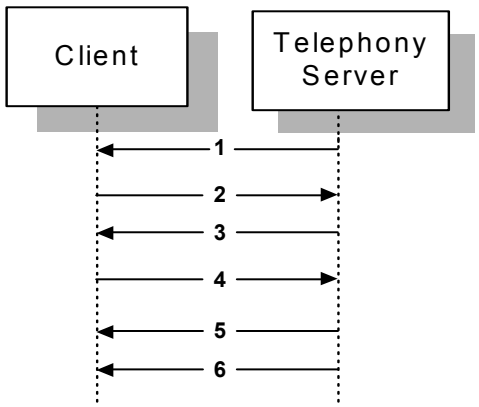
The following diagram illustrates the events and methods that are exchanged when the Telephony server routes a call to a client, for example a Workflow server.



Step	Description
1	Telephony server sends TS.IncomingCall.event(*vdu_id, call_id, ani, dnis, ctype*) to client.
2	Client invokes TS.Route(*vdu_id, dest*) method on the Telephony server.

Inbound call

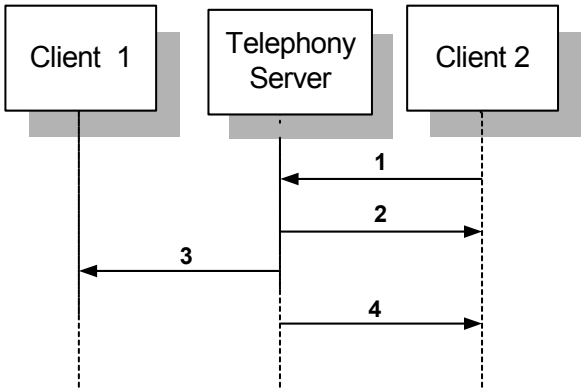
The following diagram illustrates the methods and events that are exchanged after a call arrives at the Telephony server from the switch. The sequence ends with the call being terminated by the client.



Step	Description
1	Telephony server sends a TS.Incoming.event(*vdu_id, ani, dnis, call_ref_id, ctype*) to the client.
2	Client invokes the TSAnswerVDU (*vdu_id*) method on the Telephony server.
3	Telephony server sends a TS.Connect.event(*vdu_id, call_ref_id*) to the client.
4	Client invokes the TS.HangupVDU(*vdu_id*) method on the Telephony server.
5	Telephony server returns a TS.HangupVDU.response(*vdu_id*) to the client.
6	Telephony server sends a TS.Disconnect.event(*vdu_id, call_ref_id* to the client.

Outbound call

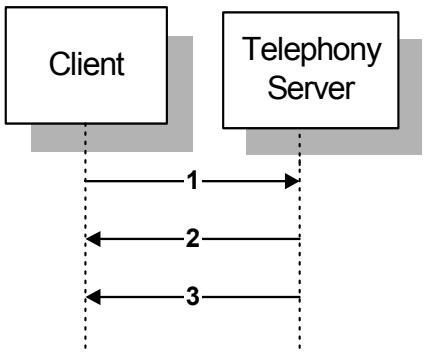
The following diagram illustrates the methods and events that are exchanged during an outbound call. In this example, Client 1 is either an agent or a queue and Client 2 is an automatic dialer.



Step	Description
1	Client 2 invokes TS.MakeCallVDU(*dest*) method on the Telephony server.
2	Telephony server returns TS.MakeCallVDU.response(*dest, vdu_id* to Client 2.
3	Telephony server sends TS.IncomingCall.event(*vdu_id, call_ref_id*) to Client 1.
4	Telephony server sends TS.Ring.event(*orig, dest, vdu_id, call_ref_id, monitor,*) to Client 2 indicating the call has reached Client 1.

Busy destination

The following diagram illustrates the methods and events that are exchanged when the call destination is busy.



Step	Description
1	Client invokes TS.MakeCallVDU(*dest*) on the Telephony server.
2	Telephony server returns TS.MakeCallVDU.response(*dest, vdu_id)* to the client.
3	Telephony server also sends TS.Busy.event (*vdu_id, call_ref_id*) to the client.

Note:

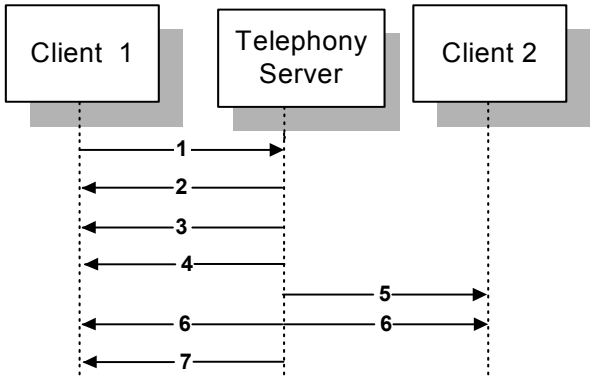
A Busy event is generated when the destination is busy. A Busy exception is raised when the originator is busy.

Blind transfer

The following diagram illustrates the methods and events that are exchanged during a blind transfer, a transfer in which the first agent hangs up before the next agent picks up the call.

The switch sets must be in the ready state. Calls are answered automatically if the switch is configured to AutoAnswer.

If the call originates within the switch system, additional events are generated that are not indicated in this model.

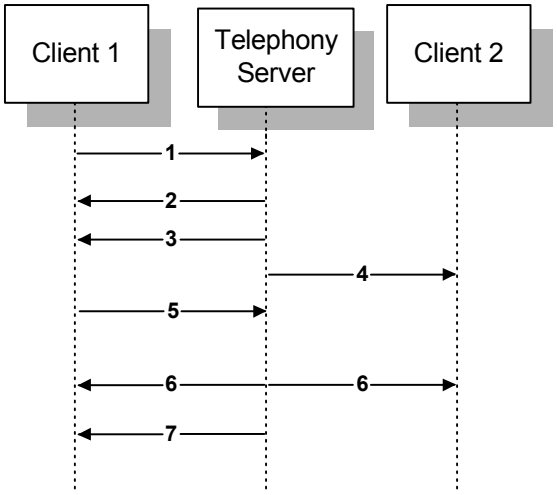


Step	Description
1	Client 1 invokes TS.TransferVDU(*vdu_id, dest*) method on the Telephony server.
2	Telephony server generates a TS.Hold.event (*vdu_id, call_ref_id*) to Client 1 that indicates the primary call is on hold.
3	Telephony server generates a TS.Ring.event (*vdu_id, call_ref_id*) to Client 1 that indicates the secondary call is in progress.
4	Telephony server returns TS.TransferVDU.response (*dest, vdu_id)* to Client 1. This response can arrive at anytime. The events and responses run in separate threads.
5	Telephony server sends TS.IncomingCall.event(*vdu_id, ani, dnis, call_ref_id, ctype*) to Client 2.

Step	Description
6	Telephony server sends a TS.Transfer.event (*vdu_id, call_ref_id*) to Client 1 and Client 2 to indicate the transfer took place.
7	Telephony server sends TS.Disconnect.event(*vdu_id, call_ref_id*) to Client 1 to indicate the EDUID is no longer associated with Client 1.

Consultative transfer

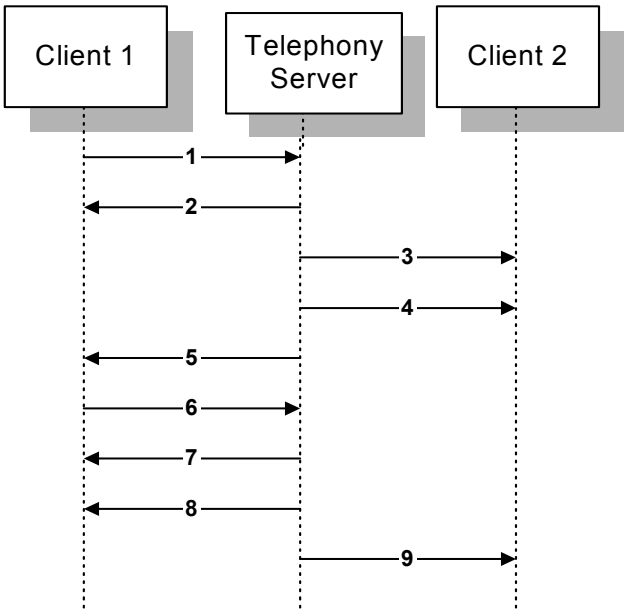
The following diagram illustrates the methods and events that are exchanged during a consultative transfer, a transfer in which the first agent remains on the line until the next agent picks up the call.



Step	Description
1	Client 1 invokes TS.TransferInitVDU(*vdu_id, dest*) method on the Telephony server.
2	Telephony server generates a TS.Hold.event (*vdu_id, call_ref_id*) to Client 1 that indicates the primary call is on hold.
3	Telephony server returns TS.TransferInitVDU.response(*dest, vdu_id*) to Client 1. This response can arrive at anytime. The events and responses run in separate threads.
4	Telephony server sends TS.IncomingCall.event(*vdu_id, call_ref_id, ani, dnis, ctype*) to Client 2.
5	Client 1 invokes TS.TransferCompleteVDU (*vdu_id*) method on the Telephony server.
6	Telephony server sends TS.Transfer.event (*vdu_id*) to Client 1 and Client 2 to indicate the transfer took place.
7	Telephony server sends TS.Disconnect.event(*vdu_id, call_ref_id*) to Client 1 to indicate the EDUID is no longer associated with Client 1.

Internal call

The following diagram illustrates the methods and events that are exchanged when a client places an internal call to another client. Note that an external call would have a similar flow except that steps 3 and 4 would not apply.



Step	Description
1	Client 1 invokes TS.MakeCallVDU(*dest*) method on the Telephony server.
2	Telephony server returns TS.MakeCallVDU.response(*dest, vdu_id)* to Client 1.
3	Telephony server sends TS.IncomingCall.event(*vdu_id, ani, dnis, call_ref_id, ctype*) to Client 2.
4	Telephony server sends TS.Connect.event(*vdu_id, call_ref_id*) to Client 2.
5	Telephony server sends TS.Connect.event(*vdu_id, call_ref_id*) to Client 1.
6	Client 1 invokes TS.HangupVDU(*vdu_id*) method on the Telephony server.
7	Telephony server sends TS.HangupVDU.response(*vdu_id*) to Client 1.

Step	Description
8	Telephony server sends TS.Disconnect.event(*vdu_id, call_ref_id*) to Client 1.
9	Telephony server sends TS.Disconnect.event (*vdu_id, call_ref_id*) or TS.Drop.event (*vdu_id, call_ref_id*) to Client 2.

Appendix B: Supported Switch Functions

This appendix lists the software functions provided by each of the supported switches and indicates which of these software functions are supported by the Avaya TS in IC 6.1.3. It also describes the limitations of the TS to normalize functionality across all of the supported switches.

This section includes the following topics:

Avaya DEFINITY/Communication Manager

The Avaya TS does not support all of the software functions available on the Avaya DEFINITY/Communication Manager. The following table lists the software functions provided by the Avaya DEFINITY/Communication and indicates which of these functions are supported by the Avaya TS in IC 6.1.3. If a function is supported with exceptions, refer to the Notes and Exceptions column for additional information.

For detailed descriptions of these Avaya DEFINITY software functions, refer to the *Avaya DEFINITY Software Features Guide*.

Important:
The Avaya DEFINITY/Communication Manager provides Converse On functionality that keeps a call's place in queue while the IVR plays predefined scripts to contacts as they wait to be connected to IC. This functionality is not intended to replace normal IVR behavior, automated service through prerecorded options. You must transfer the call directly to the IVR to use normal IVR functionality.

Avaya DEFINITY Software Function	Supported by TS	Notes and Exceptions
General:		
User-to-user data	Yes, with exceptions	Used by IC but not available to the application. This is a planned area of enhancement for future releases.
Specific party ID on multi-party calls	Yes, with exceptions	IC represents party information as "this party" and "other party" rather than indicating the individual parties on the call. IC waits for the switch to send a confirmation before it returns to the caller. IC does not deliver the information returned by the switch to the application.
Traditional ACD related control and monitoring	No	
Call control associations	No	
Wide characters in name fields	No	
Trunk ID information in messages	No	
Messages - Third Party Call Control:		
3rd Party Call Ended	Yes	

Avaya DEFINITY Software Function	Supported by TS	Notes and Exceptions
3rd Party Clear Call	Yes	
3rd Party Makecall	Yes, with exceptions	Used by IC for user classified calls and predictive calls. There is no interface for priority calls, supervisor assist calls or direct agent calls other than routing to agent ID. Non-EAS ACD calls and separate TACs are not supported.
3rd Party Merge	Yes, with exceptions	Used by IC to merge and implement call transfers and conferences. Call transfers and conferences are only done on domain control association. Individual parties and merged parties are not tracked and the party ID information is not saved. IC does not support call control associations. IC monitors VDNs to provide caller information that helps IC decide what to do with the call.
3rd Party Reconnect	Yes, with exceptions	Used by IC to reconnect a call held on a domain control association. Call control associations are not supported.
Redirect (Alerting) Call	Yes, with exceptions	Used by IC to move a call from a domain control association. IC does not use this function with call control associations.
3rd Party Relinquish Control	Yes, with exceptions	Used by IC to terminate a domain control association. IC does not use this function with call control associations.
3rd Party Send DTMF	Yes, with exceptions	Used by IC to send digits via the SendDTMFtoneVDU method. The durations of the tone and pause are configurable. Call control associations are not supported.
3rd Party Selective Drop	Yes, with exceptions	Used by IC to drop a party from a domain controlled station. IC does not use this function with call control associations.

Avaya DEFINITY Software Function	Supported by TS	Notes and Exceptions
3rd Party Selective Hold	Yes, with exceptions	Used by IC to put a call on hold from a domain controlled station. IC does not use this function with call control associations.
3rd Party Selective Listen	No	
3rd Party Single Step Conference	No	
3rd Party Take Control	Yes, with exceptions	Used by IC to cancel a secondary call in a conference or transfer. IC does not expose this feature to other applications. This function is used sporadically. Control is relinquished as soon as possible. For example, drop a party on a conference call.
Messages - Third Party Domain Control:		
3rd Party Answer	Yes	
3rd Party Auto Dial	Yes, with exceptions	Used by IC with agents who have invoked the TS.Assign method on their station to implement transfers and conferences. IC does not support priority calling or a separate TAC argument in these cases. IC exposes auto dial for use the transfers and conferences through the telephony method.
3rd Party Domain Control Request - Stations	Yes, with exceptions	Used by IC to implement TS.Assign methods on stations. The list of active calls is not returned by IC.
3rd Party Domain Control Request - Split/skills (ACD hunt group extension)	No	
3rd Party Domain Control Ended	No	
Messages - Event Notification:		
Event Notification Cancel	Yes	
Event Notification Ended (Terminate Event)	No	

Avaya DEFINITY Software Function	Supported by TS	Notes and Exceptions
Event Notification Request	Yes, with exceptions	Handled by IC for VDNs. Not supported for traditional ACD splits and AOC Trunks.
Stop Call Notification	No	
U-Abort Event	Yes, with exceptions	IC processes aborts from the switch, but it does not provide an interface for applications to issue aborts.
Messages - Event Reports:		
Agent Work Mode Change (Agent States) (using proprietary adjunct link only)	Yes	
Alerting	Yes, with exceptions	IC handles the alerting event report. Numbering plan information, trunk information, redirecting number, and party ID are not delivered to the application.
Answered	Yes, with exceptions	IC handles the answered event report. Numbering plan information and party information are not delivered to the application.
Busy/Unavailable	Yes, with exceptions	IC handles the busy event report. Numbering plan information and party information, are not delivered to the application.
Conference Call/Transfer	Yes, with exceptions	IC handles the conference and transfer event reports. Numbering plan information and party information are not delivered to the application.
Call Ended	Yes	
Call Initiated	Yes, with exceptions	IC handles the initiate event report. Party information is not delivered to the application.
Call Offered to Domain	Yes, with exceptions	IC handles the offered event report. Domain extension, II-digits, LAI information, Flexible Billing feature active, numbering plan information, and trunk information are not supported.

Avaya DEFINITY Software Function	Supported by TS	Notes and Exceptions
Call Originated	No	
Call Redirected	Yes	
Charging	No	
Connected	Yes, with exceptions	IC handles the connected event report. Numbering plan information, party information, and cause value are not delivered to the application.
Cut-Through	Yes	For more details, refer to Telephony server configuration on page 51.
Disconnect/Drop	Yes, with exceptions	IC handles the drop event report. Numbering plan information, party information, and cause value are not delivered to the application.
Entered (Collected) Digits	No	
Hold and Reconnected	Yes, with exceptions	IC handles the hold and reconnect reports. Numbering plan information and party information are not delivered to the application.
Login Logout	Yes, with exceptions	IC handles the login and logout event reports provided through station domain control of agents when the Agent Events proprietary link feature is active. The reason code provided in the logout event is not reported to IC.
Queued	Yes, with exceptions	IC handles the queued event report only to detect calls redirected by RONA. The number in queue, the ACD split extension number, and numbering plan information are not delivered to the application. IC is limited to one type of monitor per device. For example, if IC is monitoring a device as a queue (calls entering), it cannot also monitor that device for route requests.

Avaya DEFINITY Software Function	Supported by TS	Notes and Exceptions
Reorder/Denial	Yes, with exceptions	IC handles the denial event report. Numbering plan information is not delivered to the application.
Trunk Seized	Yes, with exceptions	IC handles the trunk seized event report. Numbering plan information and trunk identifier information is not delivered to the application.
Messages - Maintenance:		
Heartbeat	Yes, with exceptions	IC sends heartbeats when it cannot register as a heartbeat server. Otherwise, it responds to heartbeat requests from the switch. Heartbeat requests are not delivered to the application.
Suspend Alarms	NA	Handled from MAPD.
Resume Alarms	NA	Handled from MAPD.
Restart	NA	Handled from MAPD.
Messages - Request Feature:		
Agent Login	Yes	
Agent Logout	Yes	
Call Forward All	Yes	Only stations can forward calls. If an agent is logged into a station, that agent cannot set call forwarding.
Call Forward Busy/DA	Yes	Only stations can set this function, agents cannot. The station must be authorized to allow this function in its Class of Service.
Change Agent Work Modes	Yes	
Send All Calls	Yes	
Enable Agent Events (using proprietary adjunct link only)	Yes	
Enable EAS agent status query (using proprietary adjunct link only)	No	

Avaya DEFINITY Software Function	Supported by TS	Notes and Exceptions
Messages - Route Messages:		
Route End	Yes, with exceptions	IC handles route ends from the switch and provides an interface for the application to sends route ends to the switch. IC does not indicate specific reasons to the switch. The only cause values sent by IC are C_RESUNAVL or C_INVLDNUM.
Route Request	Yes, with exceptions	IC handles route requests from the switch. II-digits, LAI information, Flexible Billing feature active, numbering plan information and trunk information are not delivered to the application. Collected digits are received and stored in the EDU.
Route Select	Yes, with exceptions	IC provides an interface to send route selects. However, the application cannot specify priority calling, external access code or direct agent calling (flag and DAC split/skill). IC supports the passing of collected digits, but does not implement the request digit collection (collect flag, number of digits, timeout, and specific event).
Messages - Select Value:		
Message Waiting Lamp	No	
Billing Rate (Flexible Billing)	No	
Messages - Value Queries:		

Avaya DEFINITY Software Function	Supported by TS	Notes and Exceptions
Value Query - general	Yes, with exceptions	IC supports some value queries. It does not support: • classifier query • trunk group query • calls query • parties query • message waiting query • SAC query • call forwarding query • status station query The switch version query is logged, but no interface is exposed.
ACD Agent Login Query	Yes	
ACD Agent Status Query	Yes	
ACD Hunt Group Status (Split Service) Query	No	
Call Classifier Status Query	No	
Call Information Query	No	
Date and Time (Time of Day Service) Query	Yes	
EAS Agent Query (using proprietary adjunct link only)	No	
Extension (Device) Query	No	
Integrated Directory Database (Device Name) Query	No	
Party ID Query	No	
Station Feature Query: • Call Forwarding • Message Waiting Lamp • Send All Calls	 No No No	
Station Status Query	No	
Trunk Group Status Query	No	
UCID Query	No	
Version Query (undocumented capability)	No	
General Feature & Functions		

Avaya DEFINITY Software Function	Supported by TS	Notes and Exceptions
Adjunct Controlled Splits/Skills (phone control)	No	
Agent States (Agent Events)	Yes	
Adjunct Routing (Call Vectoring commands)	Yes	
Advice of Charge	No	
ASAI Customer Option Form Changes for RFA (R10)	NA	
ASAI Enhanced Adjunct Routing for Avaya IC	Yes	The call vectors for the applications and application programming need to reflect the use of this capability. This function enhances the parked call operation by providing failure handling with multiple treatments including loopback through treatment steps.
ASAI Phantom Call	No	
ASAI Proprietary Links	Yes	
ASAI Provided Dial-Ahead Digits	Yes	
ASAI Requested Digit Collection	No	
ASAI Selective Listening	No	
ASAI Send DTMF	Yes, with exceptions	Used by IC to send digits via the SendDTMFtoneVDU method. The durations of the tone and pause are configurable. Call control associations are not supported.
ASAI Single Step Conference	No	
Call Classification: Answering Machine Detection for Outbound Calls (Call Classification After Answer)	Yes	
Call Classification: Global Classification Board	Yes	
Call Classification: ISDN Call Progress Events for Call Classification	Yes	
Call Classification: Optionable Switch Classified Calls	Yes	

Avaya DEFINITY Software Function	Supported by TS	Notes and Exceptions
Computer Telephony	NA	IC interfaces directly to CVLAN through the Telephony server. No functionality is lost through this approach.
Connected Event Report for Non-ISDN Trunks	Yes	
CVLAN	Yes	
DEFINITY/MV Software Version Query (undocumented capability)	No	
Direct Agent Calling - Via Adjunct	No	
Flexible Billing (interface with AT&T MultiQuest Vari Bill Service)	No	
II-Digits Over ASAI	Yes	IC allows applications to vary the treatment given to the caller and agent selection criteria based on the originating line identity (for example, hotel phone or pay phone).
Interface Options: • BRI Link • Co-resident DEFINITY LAN Gateway (DLG) for S8300 • LAN Gateway (Ethernet Interface) using MAPD	 NA Yes Yes	
ISDN Redirecting Number for ASAI Events (MV 1.3)	No	
Multiple Monitors	No	
Multiple Outstanding Route Requests	NA	
New Reason for Redirection	No	
Pending Work Mode Changes	Yes	
Reason Codes Support - for AUX Work and Logout	Yes	IC supports sending these codes to the switch, but not reading them from the switch and delivering them to an agent.
Support Conference and Transfer Events	Yes	
Support converse vector command data return function	Yes	Converse on is not supported on IC systems running Business Advocate.

Avaya DEFINITY Software Function	Supported by TS	Notes and Exceptions
Support general PBX Switch Features (Coverage, SAC, Call Forwarding, Message Waiting)	No	
Support/Initiate Service Observing	Yes	Supported on the TS, but not by the IC applications. (IC Manager, Avaya Agent, etc.)
Support of Expanded Dial Plan (7-Digit Dialing)	Yes	
Support Vectoring (Prompting) - collecting digits in vectors	Yes	
Trunk Group Information in Event Reports	No	
UCID Support	Yes, with exceptions	IC includes UCID in the call record when received in an event report. However, IC does not use UCID nor does it query for it.
User to User Information (UUI IE) Transport - adjunct link provided to switch and received from reports	Yes, with exceptions	Used by IC but not available to the application. This is a planned area of enhancement for future releases.
UUI on Transferred Calls	Yes	
VDN Override for ISDN Trunk ASAI Events (MV 1.2)	Yes	
Version Control	No	

Aspect CallCenter

The Avaya TS does not support all the software functions available on the Aspect CallCenter switch. The following table lists the Aspect CallCenter software functions and indicates which of these functions are supported by the Avaya TS in IC 6.1.3.

For detailed descriptions of these Aspect CallCenter software functions, refer to the *Aspect CallCenter Software Features Guide*.

Aspect CallCenter Software Function	Supported by TS
AgentAfterCallWork	Yes
AgentLogin	Yes
Agent Logout	Yes
AgentNot Ready	Yes
AgentOtherWork	Yes
AgentReady	Yes
AnswerCall	Yes
Assign	Yes
Auto-ready (aux-work)	No
BlindTransfer (non-trunk calls)	No
CancelCall	Yes
ConferenceCall	Yes
ConferenceInit	Yes
ConsultativeTransfer, one step	No
DeflectCall	No
Forward Calls, Cancel Forward Calls	No
GetCallForward	No
GetDoNotDisturb	No
GetMessageWaiting	No
HangupCall	Yes
HangupUnroutedCall	Yes

Aspect CallCenter Software Function	Supported by TS
HoldCall	Yes
ListenDisconnect/Reconnect (mute)	No
MakeCall	Yes
MakePredictiveCall	Yes
MessageWaiting, CancelMessageWaiting	No
ReconnectHeld	Yes
RedirectAlertingCall	No
RespondToQuery	Yes
RetrieveHeld	Yes
RouteCall	Yes
SendAllCalls, CancelSendAllCalls	No
SendDTMF	No
SetApplicationData	Yes
SetCallForward	No
SetDoNotDisturb	No
SetMessageWaiting	No
SingleStepConference	No
SingleStepTransfer	No
SwapWithHeld	Yes
TransferCall	Yes
TransferInit	Yes
Unpark	Yes - Advocate
WalkAway, ReturnFromWalkAway	No

Nortel Meridian 1

The Avaya TS does not support all the software functions provided by Nortel Meridian 1 switch. The following table lists these Meridian 1 software functions and indicates which of these functions are supported by the Avaya TS in IC 6.1.3. This list applies to systems using either Meridian Link or Symposium Server to communicate with the Meridian 1 switch.

For detailed descriptions of these Nortel Meridian software functions, refer to the *Nortel Meridian Software Features Guide*.

Nortel Meridian Software Function	Supported by TS
Agent ID option	No
Agent Observe (advanced)	No
Analog (500/2500 type) Telephones	Yes
Automatic Call Distribution (ACD)	Yes
Automatic Number Identification • Automatic Number Identification/Centralized Automatic Message Accounting Enhancement • In-Band ANI	No
Bridging	No
Call Forward All Calls	No
Call Forward and Busy Status	No
Call Forward Busy	No
Call Forward, Internal Calls	No
Call Hold, Deluxe Individual Hold Exclusive Hold	Yes
Call Hold, Individual Hold Enhancement	No
Call Park	Yes Business Advocate only
Call Park on Unsupervised Trunks	Yes Business Advocate only
Call Transfer	Yes
Call Waiting Redirection	No

Nortel Meridian Software Function	Supported by TS
Calling and Called Number Identification	Yes
Camp-On	No
Camp-On, Forced	No
Camp-On Recalls	No
Camp-On, Station	No
Conference – Six-party conference for PBX (500/2500) Telephone	Yes (TS containers will not be correct if there are more than 3 parties involved in a conference)
Dialed Number Identification Service (DNIS) • DNIS on CDR • DNIS across Call Modifications • Name Display for DNIS • Routing by DNIS	Yes
Do Not Disturb • Group Do Not Disturb • Individual Do Not Disturb	IC only sees calls that are delivered by the switch. Setting Do Not Disturb is supported but not tested.
Hunting • Circular Hunting • Linear Hunting • Secretarial Hunting • Short Hunting • Data Port Hunting • Trunk Hunting	No
Individual Hold	No
Make Set Busy – Make Set Busy Flexible Feature Codes	No
Make Set Busy Key (basic)	Yes
Meridian MAX/ACD-MAX	Integrated with TSQS.
Multi-Tenant Service	Yes
Music • Music on Hold • Music on Delay	Yes Business Advocate only
Network ACD (NACD)	No

Nortel Meridian Software Function	Supported by TS
Not Ready Key (advanced)	Yes
Not Ready Key (basic)	Yes
Recovery of Misoperation during Call Transfer	Yes, through the hardphone.
Release Key (basic)	Yes
Routing Control	No
Six-party Conference Enhancement for PBX sets	No
Six-party Conference for PBX (500/2500) Telephone	No

Limitations on normalizing switch features

The TS makes every effort to normalize behavior across the switches that are supported on Avaya IC 6.1.3. But, each of these switches handles things differently. As such, there are features that the TS cannot normalize due to switch limitations or performance limitations. In these cases, switch behavior is apparent to the client application.

The following table provides some examples of behavior that varies across switches. This is not a complete list of examples. Developers should test their assumptions against the actual switch in their environment.

No.	Description
1.	The TS can not guarantee the sequence of events between devices. For example, a TS.IncomingCall.Event might be posted to stationA, before a TS.Ring.Event is posted to stationB, and vice-versa depending on the switch and the switch load.
2.	The TS can not guarantee that all stations involved in a transfer/conference will be informed of TS.Transfer.Event and TS.Conference.Event. The Controlling station is notified, but other stations might not be notified depending on the switch.
3.	The TS can not guarantee that data placed in the ADU and in the EDU will be the same across all switches because: ● Some switches may not provide specific data elements depending on the call flow. ● Or because device monitoring is not available. For instance, the Nortel switch does not provide queue monitoring, thus the TS can not provide data related to abandonment in queue.
4.	Some TS methods are only supported on certain switches. For example, the TS.SendData() method is only supported on the Aspect CallCenter while the Avaya DEFINITY/Communication Manager is the only switch that supports the following methods: ● TS.LogoutWithReason() ● TS.BusyWithReason() ● TS.GetPBXTime() ● TS.SwapHeld() In some cases the TS “simulates” functionality, but the switch might not have that specific feature, so access to that feature via hard phone is not available. For example, TS.WrapUp() on the Nortel Meridian is not possible via the hard phone because the switch does not support the concept of call wrap up.

No.	Description
5.	The TS may attempt to normalize some switch behavior via the Softphone. As such, attempts to operate the switch using the hard phone could yield incorrect results on some switches. This is the case with disconnections during consultative calls on a Nortel switch. The Nortel switch does not inform the TS about the disconnected party on the consultative if the primary call dropped while it was on hold. The TS can correct this problem only if the agent uses the Softphone.
6.	The TS is event driven to keep the hard phone and Softphone in sync.
7.	Not every feature in the switch is mapped over to the TS.
8.	The TS can not execute a request that is unsupported by the switch.
9.	The TS can not execute a request that is not supported by the CTI link. For example, on the Nortel Meridian switch there is no way to log an agent into more than one queue via CTI.
10.	CSTA is a recommended standard, every implementation has its nuances, and it maps to the way a given PBX operates, so the TS has to follow the PBX nuances and, in some cases, specific functionality can not be hidden from the client application.

Supported methods

The following table lists the supported TS methods per switch:

Method Name	Avaya DEFINITY	Aspect CallCenter	Nortel Meridian
TS.AnswerVDU	Yes	Yes	Yes
TS.Busy	Yes	Yes	Yes
TS.BusyTerminate	Yes	Yes	Yes
TS.BusyWithReason	Yes	No	No
TS.ConferenceCancelVDU	Yes	Yes	Yes
TS.ConferenceCompleteVDU	Yes	Yes	Yes
TS.ConferenceInitVDU	Yes	Yes	Yes
TS.DivertVDU	Yes (only for calls on stations).	No	No

Method Name	Avaya DEFINITY	Aspect CallCenter	Nortel Meridian
TS.DropVDU	Yes	No	No
TS.HangupVDU	Yes	Yes	Yes
TS.GetPhoneInfo	Yes	No	No
TS.GetQueueInfo	Yes	No	No
TS.GetPBXTime	Yes	No	No
TS.HoldVDU	Yes	Yes	Yes
TS.HoldReconnectVDU	Yes	Yes	Yes
TS.Login	Yes	Yes	Yes
TS.Logout	Yes	Yes	Yes
TS. LogoutWithReason	Yes	No	No
TS.MakeBusy	Yes	No	Yes
TS.MakeBusyTerminate	Yes	No	Yes
TS.MakeCallVDU	Yes	Yes	Yes
TS.MakeCallSetVDU	Yes	Yes	Yes
TS.Ready	Yes	Yes	Yes
TS.ReadyAuto	Yes	No	Yes
TS.ReceiveData	No	Yes	No
TS.ResetPhone	Yes	Yes	Yes
TS.Route	Yes	Yes	Yes
TS.RouteWithInfo	Yes	Partial (no data sent to the switch)	Partial (no data sent to the switch)
TS.SendData	No	Yes	No
TS.SendDTMFtonesVDU	Yes	No	No
TS.SwapHeld	Yes	Yes	No
TS.TransferCancelVDU	Yes	Yes	Yes
TS.TransferInitVDU	Yes	Yes	Yes
TS.TransferCompleteVDU	Yes	Yes	Yes

Method Name	Avaya DEFINITY	Aspect CallCenter	Nortel Meridian
TS.Wrapup	Yes	Yes	Simulated (sets agent to Not Ready)
TS.WrapupComplete	Yes	Yes	Yes

Appendix C: Troubleshooting

This appendix provides troubleshooting tips for some issues you may encounter while using the Avaya TS. We recommend that you refer to this section prior to calling Avaya Technical Support with an issue.

This section includes the following topics:

Machine address to IP address

NetMerge (formerly CT-Connect) uses name resolution to translate a machine name into its IP address. If DNS is not working, you must make an entry to identify the NetMerge server machine in the Hosts file in the C:\WINNT\System32\Drivers\etc\ directory.

Add the IP address and the server machine name to the list at the bottom of the Hosts file. The text at the top of this file provides additional information.

Operations

Operations that cannot be performed via the hard phone cannot be executed by the Avaya TS. Therefore, attempt the operation manually, if it fails, the problem is on the switch configuration. Refer to the administration guide that accompanied your switch.

Bad object state when using VTel to answer a call

The Turning Security Option in Symposium Server will resolve the “Bad Object State” error message coming from the Softphone when answering a call.

TS entering times for containers that are before EDUID create times

This is normally caused when the TS is running on a different machine than the EDU server. As a workaround, ensure that the time on the machine running the TS is after the time on the machine running the EDU server.

If the server times are out of sync, then the delta time is meaningless. Specifically, if the EDU time is ahead of the TS time, then delta time will be less then 0. The alarm is raised once only.

Thread fatal error in script (flow), routing flow fails

The TS abandons the call or diverts the call before the Workflow server invokes the TS.Route method. Two scenarios can cause this error:

1. A device/queue has been created for the same VDN to which the flow diverts (using TS.Route) the call. This causes an assignment to the same VDN twice, once from the queue and then from the WFS (by specifying the assignment criteria in the Channel tab). This is an illegal configuration, and should not be used.
2. The adjunct route wait time is set to a very low level, for instance one second or less. Set the adjunct route wait time to five seconds or more.

Consultation calls across switches

On consultation calls (TS.TransferInitVDU) across switches, where the initiating party is on a Nortel switch, the disconnection of the consulted party causes the switch to issue a TpDisconnect with both call ids. The TS can not distinguish between a transfer and a consultation, and informs the client application that both calls are gone, for example that a transfer to the consulted party occurred. The primary call, however, remains on hold at the hard phone. This problem is imposed by the Nortel link.

Workaround: Manually retrieve and terminate the call.

Conferencing agent not put in wrapup

When an agent conferences a call with a second agent, the conferencing agent does not go into the Wrapup state at the conclusion of the call. Only the agent to whom the call was conferenced goes into Wrapup at the end of the call.

This condition only occurs when agent events are disabled on a TS configured for an Avaya DEFINITY/Communication Manager. To ensure the conferencing agent goes into the Wrapup state at the conclusion of a call, enable agent events on your TS.

Making calls while in a customer call

When an agent in a customer call, puts the customer call on hold and initiates an unrelated call to a second device, the agent must make this unrelated call on the softphone. If the call is not made on the softphone, reporting on the customer call may be inaccurate and the agent may not be able to transfer the second, unrelated call.

Make all secondary calls, or calls not connected to the original customer call, on the softphone.

CTI Failure

In the event of a CTI failure, the agent should end their current phone conversation and hang up the call using the physical telephone before trying to reconnect to the Avaya TS.

Request for DS.GetViewRecords Fails

If the DS.GetViewRecords method fails, the Telephony server will not have data for agents, queues, etc. This means unqualified assignments cannot succeed because the Telephony server does not know what kind of device to associate with a given assignment request.

If this method times out, it means the Directory server is down, the database is down, or a network segment failure is in progress.

Use the manual generic update procedure on IC Manager to recover the Telephony server.

Event Lock Expired Alarms

When an agent initiates a transfer from the hard phone and the call passes through a non-monitored device, the TS locks events and delays the TS.IncomingCall to the receiving agent when receiving a C_ALERT event.

This is actually the correct TS behavior. It is a result of a timing issue between the side initiating the call and the side receiving it. During the call initialization via hard phone, the TS has to create an EDUID. By the time the EDUID is received, the call can be alerting at an agent. The TS is designed to lock events to the receiving side until an EDU is satisfied or a timeout. The receiving side waits for a timeout because the element between initialization device and destination device is not monitored.

The workaround is to change the timeout period via the device_lock_period parameter on the Configuration tab in IC Manager. The default is 5 seconds, the default is 1 second. You can also monitor the VDN/queue between the two devices.

Two EDUIDs created for the same call

In an environment with 2 Telephony servers (TS1 and TS2), both Telephony servers create an EDUIDs for the same call to an agent queue. The agent on TS2 receives the call. The Telephony Queue Statistics server (TSQS) is on the same machine as the TS1 and they both reference the same EDUID creating a queue ADU.

If you are not using the Workflow server for call routing decisions, it is important that calls routed from VDNs that are monitored by the TSQS are routed only to agents in the same domain.

Solutions:

There are three possible solutions to this limitation:

1. Put a route point before the queue and have external calls arrive at this route point. Move the call from the route point to via the TS.Route() or TS.RouteWithInfo() method. This causes the EDUID to be associated with the switch and TS2 will be properly notified what should be the EDUID during C_Alert.

   Note:

   If this is not already set this up on your system, be advised that it will require increased call handling time, increased messaging across the link, and increased system complexity for simple routing. Consider solutions 2 and 3.

2. Configure 2 queues and 2 TSQSes such that each queue services agents that are connected to a single domain. Make sure the TSQSes do not monitor queues across domains.
3. Adjust the domain and agent so the TSQS monitors the queues that service the agents who are all assigned to the same TS.

Index

Symbols

Numerical

A

B

C

D

E

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W