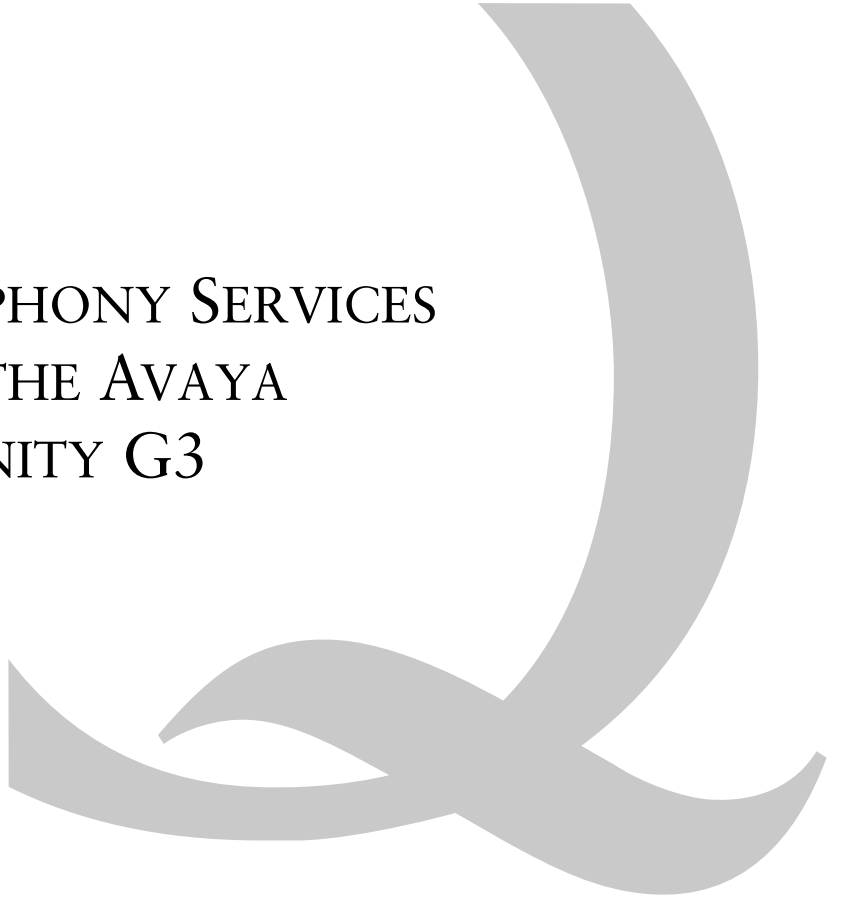
QUINTUS

TELEPHONY SERVICES FOR THE AVAYA DEFINITY G3



Part Number:DXX-1036-01

Notice

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

Avaya Web Page

The world wide web home page for Avaya is http://www.avaya.com

Preventing Toll Fraud

"Toll fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or 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 you are being victimized by toll fraud and you need technical support or assistance, call 1-800-643-2353.

Providing Telecommunications Security

Telecommunications security (of voice, data, and/or 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 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:

- Utilization (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/data privacy, intellectual property, material assets, financial resources, labor costs, and/or legal costs).

Trademarks

Quintus, WebQ, CustomerQ, VESP, CONVERSANT and DEFINITY are registered trademarks of Avaya Inc. The Quintus logo, the WebCenter logo, Quintus eContact, HelpQ, CallCenterQ, HRQ, SalesQ, Quintus CTI, NabCTI, QCTI, QManager, QFlow Designer, WebCenter, WebCenter WRU, WebACD, ComHub, WebCenter Email Response, DataWake, Defining Web-Based Customer Interaction, Real-Time Enterprise, and DEFINITY One are trademarks of Avaya Inc.

Portions of the Quintus eContact Suite include technology used under license as listed below, and are copyright of the respective companies and/or their licensors:

ActivePerl is a trademark of ActiveState Tool Corp. This product includes software developed by the Apache Software Foundation (http://www.apache.org/). Cognos, Impromptu and Powerplay are registered trademarks of Cognos Incorporated. YACC++ is a registered trademark of Compiler Resources, Inc. APEX, VideoSoft, and True DBGrid are registered trademarks, and ComponentOne, VSVIEW, SizerOne, VS-OCX, VSFlexGrid Pro, VSVIEW, VSFORUM, VSREPORTS, VSDOCX, VSSPELL, TrueDBListPro, COMPONENTONE CHART, and ComponentOne Query are trademarks of ComponentOne LLC. CT-Connect is a registered trademark of Dialogic Corporation. Dialogic and the Dialogic logo are trademarks of Dialogic Corporation. Fulcrum is a registered trademark of Fulcrum Technologies, Inc. Searchserver is a trademark of Fulcrum Technologies, Inc. AIX and RISC System/6000 are trademarks of International Business Machines Corporation. DB2, IBM, OS/2, AS/400, and CICS are registered trademarks of International Business Machines Corporation. VisualX is a registered trademark of Intergroup Technologies, Inc. ActiveX, Visio, Internet Explorer, Windows, Windows NT, Windows 2000, Win32s, SQL Server, Visual Basic, Visual C++, Outlook, Windows, and FrontPage are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Oracle is a registered trademark of Oracle Corporation. Oracle8i and Oracle SQL/Services are trademarks of Oracle Corporation. Rogue Wave and .h++ are trademarks of Rogue Wave Software, Inc. Siebel is a trademark of Siebel Systems, Inc. Basicscript is a registered trademark of Henneberry Hill Technologies Corporation d/b/a Summit Software Company. Sun, iPlanet, Java, Solaris JRE, J2EE, JavaServer Pages, and Java are trademarks or registered trademarks of Sun Microsystems, Inc. SPARC is a registered trademark of SPARC International, Inc. Products bearing SPARC trademarks are based on an architecture developed by Sun Microsystems, Inc. Formula One is a licensed trademark and Tidestone Technologies, Inc. Visual Components, First Impression, and VisualSpeller are registered trademarks of Tidestone Technologies, Inc. JRun is a trademark of Macromedia, Inc. in the United States and/or other countries. Pentium is a registered trademark of Intel Corporation. Acrobat is a registered trademark of Adobe Systems. Other product and brand names are trademarks of their respective owners.

Obtaining Products

To learn more about Avaya products and to order products, contact Avaya Inc. or your authorized Avaya dealer.

U.S. GOVERNMENT RESTRICTED RIGHTS.

This documentation and the Quintus eContact Suite software were developed at private expense and no part of them is in the public domain. Any use of this documentation or any of the Quintus eContact Suite software which on behalf of the United States of America, its agencies and/or instrumentalities ("U.S. Government"), is provided with Restricted Rights. The Quintus eContact Suite software and this documentation is "Restricted Computer Software" as that term is defined in Clause 52.227-19 of the Federal Acquisition Regulations ("FAR") and is "Commercial Computer Software" as that term is defined in Subpart 227.471 of the Department of Defense Federal Acquisition Regulation Supplement ("DFARS"). The Quintus eContact Suite software is classified as "Commercial Computer Software" and the U.S. Government may only acquire "restricted rights" in the Quintus eContact Suite software and this documentation as that term is defined in Clause 252.227-7013(c)(1) of the DFARS, and if the Quintus eContact Suite software is supplied to any unit or agency of the U.S. Government other than Department Of Defense, the U.S. Government's rights in the Quintus eContact Suite software and this documentation will be as defined in Clause 52-227-19(c)(2) of the FAR; and use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(I)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or in subparagraphs (c)(1) and (2); Commercial Computer Software -- Restricted Rights at 48 CFR 52.227-19, as applicable. Licensor is Avaya Inc., 211 Mt Airy Road, Basking Ridge, N.J. 07920.

This documentation may only be stored, transmitted or reproduced, in whole or in part, under the terms of the Right to Print License contained in the *Quintus eContact Suite 5.6 Product Documentation Guide*, located on the documentation CD, and in the Preface to each individual manual.

CONTENTS

Before You Begin

Typographical Conventions

This guide uses the following font conventions:

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

Notes, Tips, and Cautions

Note: A note calls attention to important information.

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 Quintus software, you should:

1. Retry the action, carefully following the instructions given for that task in the 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 you see. Have the Quintus documentation available.
4. If you continue to have a problem, contact Quintus Technical Support by:
   - logging in to the Quintus Technical Support web site (*www.quintus.com/qq*).
   - Calling (888) TECHSPT or (888) 832-4778 in the U.S. and Canada from 8:30 a.m. to 8:30 p.m. (EST), Monday through Friday (excluding holidays). International users should call 512-425-2201, or calling 1-800-242-2121 in the U.S. only.
   - Email your question or problem to *support@quintus.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 Quintus Technical Support through this URL or this email address, please go to *www.avaya.com* for further information.

Product Documentation

Most Quintus 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 Quintus eContact Suite 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 an HTML file included on the Quintus eContact Suite 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 workarounds to known problems, and other information critical to successfully installing and using your Quintus software. You may also receive a printed Readme Addendum, containing similar information uncovered after the manufacture of the product CD-ROM. You should review the Readme file and the Readme Addendum before you install your new Avaya software.

Electronic Documentation

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

Printed Documentation

You can purchase printed copies of these manuals separately. Consult with your sales representative or with Customer Support for assistance.

License to Print the Electronic Documentation

Online copies of documentation are included on the CD for every software release. Quintus customers who have licensed software (each, a “Licensee”) are entitled to make this online documentation available on an internal network or “intranet” solely for Licensee's use for internal business purposes, and 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 only be printed 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 Quintus. 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 Quintus. 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 Quintus are owned by Quintus 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 Quintus 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 from *http://learning2.avaya.com*.

- Over the phone at 800-337-8941 or 800-288-LEARN (800-288-5327).
- Through email at *trainingadmin@quintus.com*.

Chapter 1
Overview of the Telephony Server

Computer Telephony Integration (CTI) involves seamlessly integrating telephone hardware (e.g., switches), computers, and software. The Telephony Server is at the core of the eContact Telephony solution.

What is the Telephony Server?

The Telephony Server (TS) is the eContact Telephony component responsible for linking Quintus eContact to CTI products such as private branch exchange (PBX) systems, automatic call distribution (ACD) systems, and integrated voice response (IVR or VRU) systems.

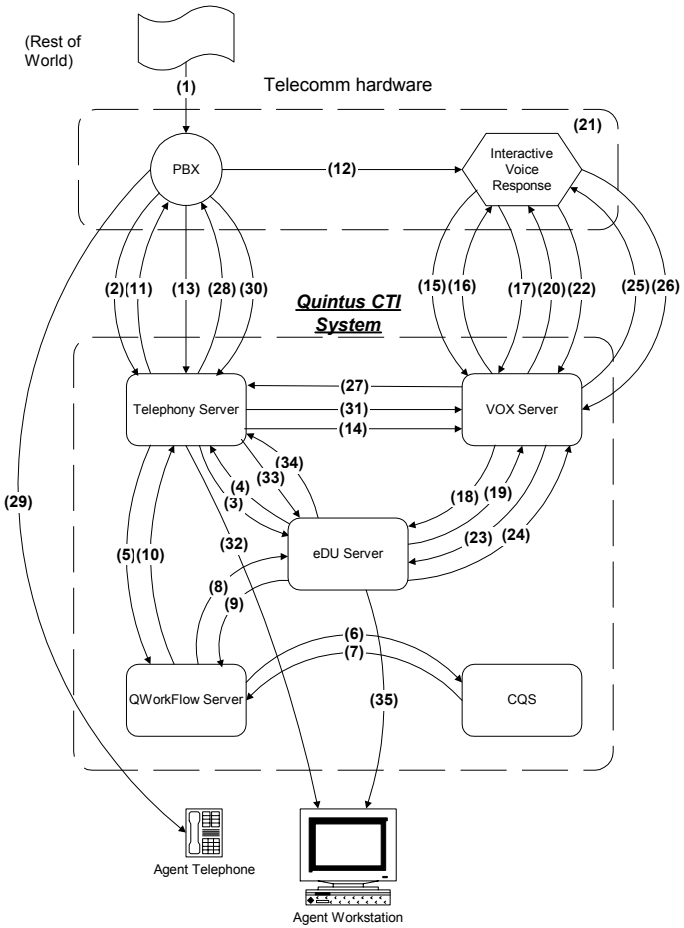
Using the TS, other servers and clients in the eContact Telephony environment request telephony services (transfer or route a call, hang up a call, and so on) by invoking TS methods. Clients and servers also "assign" to the TS to receive telephone-related events (e.g., "there is a call from customer 999-1111").

The Telephony Server "insulates" client applications from the telephone switch and its interface by encapsulating switch-specific features. Client applications are thus portable across multiple types of switches, CTI links, and other types of telephony interfaces, such as ISDN.

Multiple Telephony Servers can be implemented for redundancy to enable automatic restart and recovery, without user intervention, in case the ACD link, server network, or server software fails.

Example Call Flow

The function of the TS can best be understood by examining a typical call flow. Note that call flows vary for each installation. Additional call flow diagrams are presented in Appendix A.



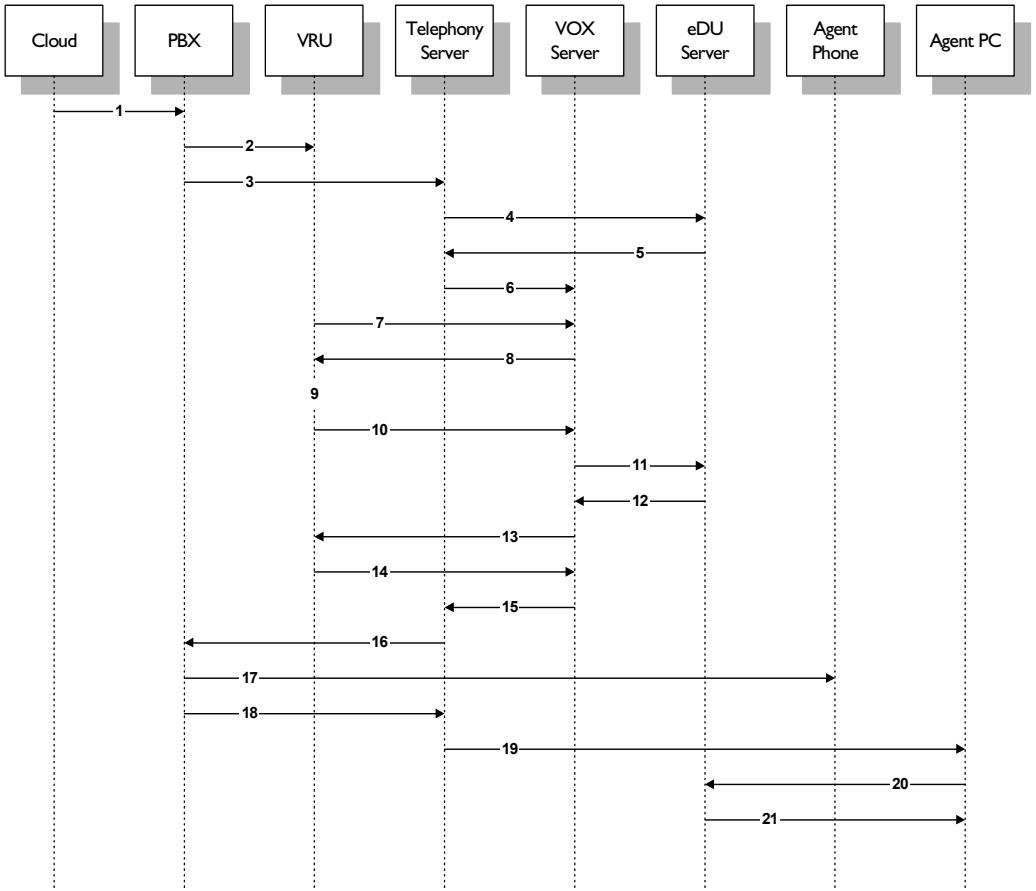
Call Flow Model: Explanation of Server Interactions

Step	Explanation
1	Call arrives at the switch (PBX) from the public network.
2	In this example, the switch is set up to request routing information from the QCTI system across the link to the TS. This is referred to as an adjunct routing request.
3	The TS invokes a method on the eDU Server (VDU.Create) to create a eDU for this call. The method call includes a list of the initial elements to place in the eDU (ani, dnis, etc.).
4	The eDU Server creates the eDU and returns a response (VDU.Create.response) to the TS. The response contains a unique eDUID for this call.
5	The TS sees in its tables that the QWorkFlow Server has previously assigned to monitor all calls associated with a request for routing information. The TS sends an event (TS.IncomingCall.event) to the QWorkFlow Server, including the eDUID, ani and dnis for the call. The call is still waiting unanswered at the switch, the caller hears a ring.
6	The QWorkFlow Server invokes a flow to respond to incoming call events. In this example, the flow queries the CQS Server to find an account number for the call based on the ani (ANI.SelGetRecords).
7	The CQS Server returns a response containing data from the matching row or rows in the database (ANI.SelGetRecords.response).
8	If multiple accounts existed for the same ani so that unique identification was not possible based on ani alone, the only value set might be: {“ani_flag”, “AMBIGUOUS”}.
9	The eDU Server stores the values and returns a response to the QWorkFlow Server (VDU.SetValues.response).
10	At the end of the incoming call routine, the QWorkFlow Server invokes a method on the TS to route the call (TS.Route). The request includes the eDUID and the routing extension. The appropriate extension may be deduced by the QWorkFlow Server from information in the CQS database.
11	The TS instructs the switch to route the call (The TS afterwards responds to the QWorkFlow Server with TS.Route.response, not shown).
12	In this example, the switch routes the call to an extension associated with an Interactive Voice Response unit.
13	The switch notifies the TS that the call has arrived at the IVR.

Step	Explanation
14	The TS sends an event (TS.IncomingCall.event) to the VOX Server, including the extension to which the call was routed and the eDUID of the call.
15	At the start of the IVR script, the IVR notifies the VOX of the new call (VOX.newcall), including the IVR channel number on which the call was received.
16	The VOX Server checks its tables to associate the IVR channel number with a telephone extension. It matches the extension with an incoming call event from the TS. The eDUID from the corresponding incoming call event is returned to the IVR script.
17	The IVR tells the VOX to retrieve the account number and related information (i.e. ani_flag, name) from the eDU. This is accomplished with multiple calls to VOX.getvdu.
18	The VOX Server translates each IVR request into a CORBA method call to the eDU Server (VDU.GetOneValue).
19	The eDU Server returns a response to the VOX Server (VDU.GetOneValue.response) containing the requested data if available.
20	The VOX returns a response to the IVR script.
21	If the ani_flag from the eDU is set to "UNIQUE", the IVR prompts the caller for the last 4 digits of their account number to verify identity. If not, it prompts the caller for their entire account number.
22	The eDU is updated to reflect the new account number if it was collected. An additional data element, id_security, could be set to "TRUE" in the eDU if the ani_flag was "UNIQUE" and the caller entered the correct digits to verify their identity, and set to "FALSE" otherwise. This flag could be viewed by the CSR who receives the call to determine if additional steps are necessary to identify the caller. These values are set in the eDU with multiple calls to VOX.setvdu (only one is shown).
23	The VOX Server translates each IVR request into a CORBA method call to the eDU Server (VDU.SetOneValue).
24	The eDU Server returns a response to the VOX Server (VDU.SetOneValue.response).
25	The VOX returns a response to the IVR script.
26	The IVR invokes VOX.transfer to send the call to a CSR queue.
27	The VOX Server calls TS.Transfer on behalf of the IVR.
28	The TS instructs the switch to transfer the call.

Step	Explanation
29	The switch places the call on queue and transfers it to the next available agent. The agent telephone rings.
30	The switch notifies the TS of a transfer to this new phone line. The TS keeps track of which calls are associated with which eDUID as the switch transfers calls.
31	The TS sends a response back to the VOX server (TS.Transfer.response).
32	The TS sends an incoming call event (TS.IncomingCall.event) to the client application that has previously assigned to the new extension. The event contains the eDUID for the call.
33	Telephony server notifies eDU server of the call transfer via a VDU.SetAndTransfer.
34	eDU sends a response to telephony server, VDU.SetAndTransfer.response
35	In this case, we assume that the client application has previously assigned to the eDU Server to place a "watch" on any eDUs that contain data elements that meet a certain criteria (such as agent name). When the TS transfers the call, it also modifies the agent parameter in the eDU (step 33). This causes the eDU Server to send a VDU.Watch.event to the client application signifying that a eDU has met the watch criteria. This event contains the eDUID and all data elements stored so far in the eDU for this call. The client application can then populate the screen without a separate query to the eDU Server.

The following diagram uses a different method to illustrate the same call flow.



Supported Operating Systems

The telephony server as implemented for the Avaya Definity G3 runs on the following operating systems:

- Microsoft Windows NT 4.0, service pack 6a *or* Windows 2000
- SCO Unixware 7.1
- Solaris/SPARC 2.6 and 7.0

Supported ACD Environments

The Avaya Definity G3 supports two types of Automatic Call Distribution (ACD) environments: Expert Agent Selection (EAS), and non-EAS.

EAS Environment

In an EAS environment, each agent's ID is a dialable, virtual teleset number. An agent can be reached by dialing his or her agent ID, regardless of the phone set to which the agent has logged in.

Each EAS agent is assigned to a specific skill group. After logging in, an agent is automatically accessible through the queues assigned to his or her skill set.

Non-EAS Environment

Non-EAS is often referred to simply as ACD. In a non-EAS environment, each agent must enter his or her agent ID, queue number (group extension as defined in a hunt group) and equipment number when logging in. The agent is then accessible only through the hunt group identified at login, and through the physical equipment number entered at login.

Non-EAS agents have no virtual teleset number (the agent ID is not a dialable number). Such agents must use their physical equipment numbers for all transactions.

Note: Client applications, such as VTel, might map agent IDs to physical teleset numbers.

WAN Considerations

In a WAN environment, calls are passed between call centers using the User to User Information (UUI) field. An ISDN connection is required between each Avaya Definity G3 switch.

In a non-EAS environment, a physical extension number is required to invoke a MakeCall or Transfer method on multiple Telephony Servers. Neither MakeCall nor Transfer by login ID or logical agent number is supported in a non-EAS WAN environment.

For information on configuring servers for failover, refer to "Configuring Redundant CTI Links," on page 42. For information on configuring the switch for non-EAS agents, refer to "Configuring the Switch in an ACD Environment," on page 44.

DTMF Requirements

The Avaya Definity G3 switch and CallVisor ASAI protocol provide the ability for a call center agent to send DTMF tones to the switch, eliminating the need for the agent to switch back and forth between a hard phone and a soft phone. Support for this feature requires:

- Version 5.5 or later of the Telephony Server and the VTel client, with appropriate settings in the client configuration file.
- Release level 6 or later of the Avaya Definity G3 switch.
- Version G3v6 and GEv7 support CallVisor LAN Client software 6.1.4, 6.1.5 or 6.1.6. Version G3v7 requires CallVisor LAN Client software 6.1.6.

Reason Code Requirements

The Avaya Definity G3 switch provides the ability for a call center agent to enter a numeric code indicating the reason for certain requests. Supported requests are to log out or to change work mode to "auxwork."

Use of reason codes requires the following:

- Version 5.5 or later of the Telephony Server and the VTel client, with appropriate settings in the client configuration file.

- Version 6 or later of the Avaya Definity G3 switch, which must be running in EAS mode. The Enable Reason Codes parameter must be activated on the switch.
- Version G3v6 and GEv7 support CallVisor LAN Client software 6.1.4, 6.1.5 or 6.1.6. Version G3v7 requires CallVisor LAN Client software 6.1.6.

In addition, the TS must have the following configuration parameters set:

- pbxver set to 6 (use "6" for higher PBX versions also)
- pbx_reasoncodes set to true
- deflogout_reasoncode and defaux_reasoncode can be set to supply an alternate value if the TS detects an invalid reason code.

MAPD Requirements

Use of the MAP-D gateway requires version 6.1.6 of CV/LAN if the telephony server is running on Windows NT / Windows 2000 or SCO Unixware, or version 6.1.6 if the telephony server is running on Solaris/SPARC. CV/PC is not supported.

Quintus Telephony Server Behavior During Hardphone/ Softphone Interaction

The following table depicts those interactions between hard and soft phone sets. The hard phone set is the physical teleset sitting on an agent's desk, while the soft phone is the software-implemented phone depicted by a GUI on an agent's computer screen.

This interaction in a CTI environment is strongly dependent on events being passed to the telephony server from the PBX, and the telephony server in turn passing those events through to the client, to keep both phone sets in synch. As such, certain limitations are imposed by the hardware involved, and not on the software implementation. Additionally, certain keys available on the hard phone set are there as controls specific only to that particular phone set. Some of these keys may be function programming keys, dial last number keys, etc. These keys do not require PBX interaction, and so in turn do not generate events, and are completely transparent to the CTI software.

The following interaction table attempts to identify those telephony features that eContact Telephony supports and does not support today, for supported telephone switches. In the table, those event results listed under "manual" indicate support through the telephony server when those features are evoked through the hard phone set. Those listed under the "QCTI" heading indicate support when the features are evoked through the soft phone.

Action	Definity	
	manual	QCTI
alternate call	yes	future
answer call, connect	yes	yes
auto ready (auto-in)	no	yes
blind transfer	yes	yes
bridge two calls	yes	future
busy (aux-work)	no	yes
conference call	yes	yes
consultative transfer	yes	yes
consultative transfer one-step	no	no
forward calls, cancel forward calls	no	yes
hang up call	yes	yes
hold call	yes	yes
listen disconnect/reconnect (mute)	no	future
login	future	yes
logout	future	yes
make call	yes	yes
make call predictive	no	future
message waiting, cancel msg wait	no	future
monitor and control DN	no	yes
monitor DN	no	no

Action	Definity manual	Definity QCTI
monitor queue	no	future
network take back and transfer	yes	future
off hook	no	no
on hook	no	no
ready (in service/manual-in)	no	yes
reconnect held call	yes	yes
redirect alerting call	no	future
route call	no	yes
send all calls, cancel send all calls	no	yes
send DTMF tones	no	yes
single step conference	yes	future
third party drop	yes	yes
walk away, return from walk away	no	no
wrap up (after-call-wk)	no	yes

Troubleshooting notes

Overall if an operation cannot be performed via the hard phone it can not be executed by the TS. Therefore, attempt the operation manually, if it fails, then the problem is on the PBX configuration.

Events are associated with the Agent Group an agent belongs to. If the incorrect Agent Group is present in the TS configuration, VTEL will login the agent, and it will work correctly for some operations, but not all, and not consistently.

The PBX initiates the communication between the PBX and the host application (the TS). Furthermore, the PBX uses Data Sys Access field on the Data Interlink Record to identify the host application it is talking to. Make sure this value is correctly entered in the TS configuration.

CHAPTER 2
TRACKING CALL AND AGENT DATA

The Telephony Server (TS) automatically tracks two types of call data: the names of interested clients and call end point information. In addition, the TS tracks various types of agent activity for reporting purposes.

How Call Information is Tracked

eContact Telephony stores all information related to a telephone call in a call detail record known as an **eContact Data Unit (eDU)**. eDUs are maintained by the eDU Server. The TS ensures that the eDU follows its call as the call moves through the system. This prevents customers from having to repeat their name and other information, and eliminates redundant database queries.

eDU information can be organized into **call containers**. A call container is a grouping of values under a common name. Call containers are hierarchical structures of data within an eDU.

For more information about eDUs and call containers, see the *eDU Server Programmer's Guide*.

The various parties involved in a call, such as agents, callers, and VRUs, are referred to as end points. Each time a call is connected to a new party, information is stored about the new end point. To manage this, the TS creates a container for each end point. All eDU data relating to call end points (events and name/value pairs) are stored in call containers.

The first end point of a call is named voice.1, the second is voice.2, the third is voice.3 and so on. Information about each end point is stored in a subordinate structure (voice.2.connect, voice.2.transfer, voice.2.queue, etc.). A new end point is created each time a new client becomes associated with the eDU.

For performance reasons, the TS accumulates information about each end point until the connection is terminated. When the connection is terminated, the information is sent to the eDU Server.

Creation of call containers is controlled by the Create Containers (tscon) configuration parameter.

Note: Additional information about telephone call connections and end points should be stored in separate data structures, not in call containers.

Call Container Contents

The table below describes a call container in which end point events and attributes are stored. X represents the unique identification number for each end point. Y and Z represent sequence numbers within each end point's activities.

Note: The call container information presented in this section is new as of Quintus eContact version 5.5. Call container functionality for QeS 5.1 is still supported, and is documented in Appendix B of this manual.

Name	Value	Explanation
voice.X	Delta time	For voice.1, this is base time (should be zero). For other cases, it is elapsed time in seconds since 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 item provides additional details.

Name	Value	Explanation
voice.X.conferencedest.Z	phone number	The destination phone number (Z) of a conference call, where Z = 1, 2, 3, ...
voice.X.connect	delta time	The elapsed time in seconds between the creation of the eDU and the time the call was connected.
voice.X.destination	phone number	Phone number of client at end point X.
voice.X.exit_reason	Reason for exit: "normal," "transfer," "abandon," "other."	The switch supplies a reason for the termination of a call end point.
voice.X.holdtime.Y	time in seconds	Time spent on hold during hold instance Y.
voice.X.leg_id	unique id (UUID)	Unique leg_id of the current leg of the call.
voice.X.loginid	loginid	Login id of client at end point X.
voice.X.origin	phone number	Phone number (ANI) at other end of end point X.
voice.X.queue	delta time	The elapsed time in seconds between the creation of an eDU and the time at which the switch reports the call as queued.
voice.X.queue_number	queue number	Queue number configured within the CallCenter switch as an Application Group.
voice.X.queuetime	time in seconds	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. **Note:** Currently, this value is always zero (0). Subsequent releases of the telephony server will supply more meaningful values.
voice.X.ringtime	time in seconds	Time reported by the switch between agent phone starting to ring and agent answering the phone.

Name	Value	Explanation
voice.X.talktime	time in seconds	Time reported by the switch between agent answering the phone and agent hanging up the phone.
voice.X.transfer	phone number	The phone number to which the call has been transferred.

About Clock Synchronization

If the Telephony Server and eDU Server are running on different machines, you must keep the clocks on the two machines synchronized. Several tools are available for synchronizing clocks, including the UNIX timed (time daemon) process, which synchronizes a group of UNIX servers to a master clock. See the documentation for your operating system for more information about clock synchronization tools.

Reporting from Call Containers

State durations (connect duration, talk time, hold time, etc.) and event counts (number of times a call was placed on hold, etc.) are not calculated by the TS, but can be derived from the time stamps and other information in a call container.

The following table describes use of the process measurement data in a call container.

Note: The information presented in this section applies to call container contents for Quintus eContact 5.5 and newer. For information on reporting from QeS 5.1 call containers, see Appendix B.

Measurement	Definition	Derivation
Hold Time	Time spent on hold during hold instance Y.	TS calculates this value based on the value stored in voice.X.holdtime.Y.
Queue Time	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.	TS calculates this value based on the value stored in voice.X.queuetime. **Note:** Not yet implemented.
Ring Time	Time reported by the switch between agent phone starting to ring and agent answering the phone.	TS calculates this value based on the value stored in voice.X.ringtime.
Talk Time	The length of time that each agent was connected to the telephone call (including time on hold).	TS calculates this value based on the value stored in voice.X.talktime.
Hold Count	The total number of times a call was placed on hold.	Total number of voice.X.holdtime.Y events for a given instance of X.

Agent Containers

As a call moves between agents in a call center, each agent (whether it is a person or a voice response unit) may wish to store information about the call in a way that will not interfere with the next or last agent who handled the call. To manage this, the TS automatically creates an agent container.

Each agent involved in the call takes a private subtree of the agent container for his or her own use. The first uses the name agent.1, and creates names below it — agent.1.callduration, agent.1.reasoncode, and so on. The next uses the name agent.2 and sets his own values in that subtree — agent.2.callduration, and so on.

The agent container is used in a cooperative way by the TS and other applications, such as VTel. The sequence of events is as follows:

1. The TS creates an eDU for a call, determines the login ID (or UUID) of the agent that will first handle the call, and sets the name agent.+loginid to a value of loginid.

2. An agent who has already assigned using agent.*:loginid will get a watch event, because agent.1 has been created with a matching value. From this watch event he 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 *eDU Server Programmer's Guide* for a complete description of the * token and other container naming conventions.)

3. The agent handles the eDU, probably storing data in agent.!, which expands to agent.1, and eventually decides to move the call to a different agent.

4. The TS recognizes the transfer and again marks the eDU with an agent.+newloginid and a value of newloginid. (This becomes agent.2.)

5. A different agent, who has also assigned with agent.*:newloginid, will see this new eDU. In the meantime, the old agent also still sees it, since his criteria still match (agent.1 still contains his login ID).

6. The first agent continues to add data to the eDU, doing his call wrapup. The second agent's client sees these changes in real time.

7. The first agent may decide he no longer wants to get events about the eDU changes, so he sets agent.! to an empty string. This sets agent.1 and he gets a drop event, because his assign criteria no longer match — agent.1 is empty and agent.2 is not his own login ID.

8. At some point, the first agent does a VDU.Terminate, which tells the eDU Server that the eDU does not have to be kept active for this agent.

Reporting From Agent Containers

Whenever a contact is terminated, the TS updates information in agent container queues. This information is then used to generate reports about agent activity.

Data Recorded	Calculation
Number of contacts offered	A tally of items posted to a specific ADU.
Number of contacts handled	Derived from TS 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 (e.g., two seconds.)
Number of contacts abandoned while in queue, ringing, or on hold	Derived from TS container end points that include voice.abandon items. Data is recorded for each agent.
Average call handling time	The sum of all talk time, ring time, and queue time divided by the number of contacts handled.
Average talk time per contact	Total talk time divided by the number of contacts handled. Since the switch reports talk time from its own perspective, talk time includes the period in which a contact is on hold; to exclude time on hold, 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 TS 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.
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 call, 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. So, 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 calls by the total number of calls.

Data Recorded	Calculation
Average answer time	Total ring time for the number of contacts offered, divided by the number of contacts.
Average time before abandoning a call	Raw data is derived from TS container end points that include voice.abandon items. Data is recorded for each agent. Average time before abandoning a call is the sum of ring time, queue time, and talk time for all abandoned calls, divided by the number of abandoned calls.
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.

CHAPTER 3
TELEPHONY SERVER CONFIGURATION

This chapter contains information about configuring the Telephony Server as implemented for the Avaya Definity G3.

Configuration Parameters

The following table lists configuration parameters set using eContact Manager. The table lists the label as it appears in eContact Manager, followed in parentheses by the parameter name as it is required internally by the TS.

For information on using eContact Manager, see the e*Contact Manager User's Guide*.

Label and Internal Name	Description
Minimum Extensions (callplan)	Minimum number of digits to be entered for a valid extension number. Default is 1.
PBX Link (link1)	Device through which the Telephony Server communicates with the switch, such as */dev/asai/asai*. If you are using MAPD, enter the IP address of the MAPD card set.
Signal Number (node1)	When configuring multiple Telephony Servers on a single machine, specify the signal number for the ASAI line associated with each Telephony Server in the format signal*nn*. If not specified, default is signal01.

Label and Internal Name	Description
Create Containers (tscon)	If checked, the TS will create a TS container to store information about calls for reporting purposes (for more information, see Chapter 2).
Default ANI (defani)	Changes the default ANI used when no explicit ANI is known. Default is 5556666. Maximum length is 32.
PBX Version (pbxver)	A digit identifying the version of the Definity switch (6 for version 6 or higher). For MAPD, default is 6.
Call Timeout (calltimeout)	The number of seconds used to define an old call. The default is 7200 seconds (two hours). Minimum value is 60 seconds. The age of calls is checked every five minutes and old call information is eliminated.
Route Timeout (routetimeout)	The number of seconds used to define an old route request. The default is 120 seconds (two minutes). Minimum value is 60 seconds. The age of route requests is checked every five minutes and old route information is eliminated.
Call Control (callcontrol)	Specifies whether the TS can take control of a call. If unchecked, the TS cannot issue a Third Party Call Control command to the PBX. Since only one entity can control any given call, this allows the TS to cooperate with other entities that must control calls. The default is checked.
Reason Codes Are Enabled (pbx_reasoncodes)	If the PBX is configured for reason codes and an application plans to use reason codes for agent logouts and agent changeState to Busy, this should be enabled. Default is disabled.
Default Logout Reason Code (deflogout_reasoncode)	If reason codes are enabled, this code will be used if an agent does not enter a code for logout. Default is 1. This field may be required depending on PBX configuration.
Default AUX Reason Code (defaux_reasoncode)	If reason codes are enabled, this code will be used if an agent does not enter a code when changing his or her state to Busy. Default is 1. This field may be required depending on PBX configuration.
Heart Beat (heartbeat_frequency)	If the switch is not configured to initiate a heartbeat handshake, this parameter will cause the TS to be the initiator. Default is 60 seconds, minimum is 15, maximum is 240 seconds (4 minutes).

Label and Internal Name	Description
Process Prefix (pbx_prefix)	If checked, makecall destination numbers are internally tracked by the TS without a leading expected "1". The default is unchecked.
ds_timeout	Req'd = no Data Type = integer Unit = seconds Min/Max Value = NA Default Value = 32 Description = Allows you to override the default time session associated with a DS request
Link Failure (asai_link_failures)	Type: integer Default: zero Allowable Settings: 0,1,2,3,...,50 Description: This number represents that number of failed "reads" and "writes" the TS will process from the switch in a row before shutting down. If this number is set to zero, the TS will use its default settings, which is to wait for 2 failed heartbeats before shutting down.
Transfer Type (wait_for_event)	Type: Boolean Default: TRUE Allowable Settings: TRUE, FALSE Description: When "wait_for_event" is set to TRUE, the TS will wait for an event before completing a transfer. When "wait_for_event" is set to FALSE the TS will not wait for an event before completing a transfer, it will wait for the number of milliseconds defined by "merge_call_wait_time" and then complete the transfer.
Transfer Wait Time (merge_call_wait_time)	Type: integer Default: zero Allowable Settings: 0,1,2,3,...,50 Description:: The parameter "wait_for_event" must be set to FALSE in order for this parameter to be used. The value of this parameter is the number of milliseconds the TS will wait before completing a transfer. The maximum value that is allowed is 50.

Agent Configuration

The following agent configuration parameters are set with the eContact Manager. (For information on configuring agents in hunt groups, refer to "Defining Hunt and Skill Groups," on page 41.) The table lists the label used when configuring with eContact Manager, followed in parentheses by the parameter name as it is required internally by the 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)	Phone number of teleset. If used, the agent does not have to enter an equipment number when logging in. This number can be overridden at log in time.

CHAPTER 4
AVAYA DEFINITY G3 CONFIGURATION

This chapter describes how to configure the Avaya Definity G3 for use with the telephony server.

Call Adjunct Routing

Call adjunct routing is the process by which a switch informs eContact Telephony that a call has arrived and eContact Telephony informs the switch that the call should be routed to a particular destination.

There are several steps involved in call adjunct routing, as outlined below. Each step is explained more fully in the following sections.

1. Incoming call is identified by Dialed Number Identification Service (DNIS).
2. DNIS is mapped to a Vector Directory Number (VDN).
3. VDN identifies the vector to be executed.
4. Vector identifies the ASAI link to which the call is to be routed.
5. Vector executes an adjunct route on the ASAI link and waits for a return.
6. eContact Telephony responds with a route request.

The following sections trace a call from its arrival at the switch to its eventual transfer to an agent.

Note: The interaction of eContact Telephony and the Avaya Definity G3 switch can be affected by many switch parameter settings that are not described in this chapter. For example, when configuring for outbound dialing, you may have to set the "Answer Supervision by Call Class" parameter to Y. Refer to your Avaya Definity G3 documentation for a complete description of switch parameters.

Defining Vector Directory Numbers (VDNs)

When a call arrives at the switch, the DNIS (Dialed Number Identification Service) is converted to a Vector Directory Number (VDN).

The VDN table associates each VDN with a vector (script) to be executed when a call arrives on that VDN. The following illustrates a VDN table. In this example, a call arriving on VDN 26001 would cause Vector 1 to be executed.

```
Allow VDN              Vec              Orig     Event        Skills
Ext   Name           Override  COR TN  Num  Meas  Annc   Notif Adj  1st

26001 route to ASAI     n       1   1    1   none
26002 Route to LAN      n       1   1    2   none
26003 Rt LAN thn ASAI   n       1   1    3   none
```

Defining Vectors

Vectors are scripts that contain a series of steps to be executed when the vector is run.

The following illustrates the definition of vector 1 (which, following the previous example, is the vector executed when a call arrives on VDN 26001). In this example, vector 1 causes calls to be adjunct routed to station 24961. (Stations are described in "Defining Stations," on page 39.) If the vector were not defining an adjunct route request, line 01 would contain the phrase 'route to' rather than 'adjunct.'

The Avaya Definity G3 will execute step 01, and then will pause until one of the following occurs:

1. eContact Telephony requests a transfer. The Avaya Definity G3 would then leave the vector and make the transfer.
2. More than 4 seconds pass. The Avaya Definity G3 would then execute the next command in the vector, in this case, routing to station 24183.
3. If the ASAI link is not operational, the Avaya Definity G3 would automatically proceed to step 3 in the vector without waiting the 4 seconds.

```
 Number: 1                    Name VECTOR 1
Basic? y   EAS? y    G3V4 Enhanced? n   ANI/II-Digits? n   ASAI Routing? y
Prompting? y   LAI? y      G3V4 Adv Route? n
01 adjunct     routing link 24961
02 wait-time   4   secs hearing ringback
03 route-to    number 24183          with cov n if unconditionally
04
05
```

Defining Stations

A station is any physical device, such as a teleset or telephony link, that has an extension number and has a physical connection to a port on a telephony card.

The following illustrates a list of stations on a Avaya Definity G3. In this example, the 8410D telephone sets are digital telephones, the 602A1 are CallMaster sets, the 8510T telesets are ISDN phones, the ASAI LINE 1 is the telephony link (physical) to the eContact Telephony server, and the ASAI LAN 01 is the LAN gateway which replaces the ASAI direct-connect link with an ethernet connection.

You must configure either an ASAI line or an ASAI LAN for an eContact Telephony connection.

Following the example, calls routed to station 24961 are routed to ASAI line 1.

```
                                        Data  Cv   COR/
Ext     Port       Type   Name       Room    Ext  Pth   COS  Cable  Jack


24141   01A0803   8410D   STATION 24141                  1  1
24142   01A0804   8410D   STATION 24142                  1  1
24181   01A0801   602A1   AGENT 1                        1  1
24182   01A0802   602A1   AGENT 2                        1  1
24183   01A0402   8510T   STATION 24183                  1  1
24184   01A0403   8510T   STATION 24184                  1  1
24185   01A0805   602A1   Agent 5                        1  1
24186   01A0806   602A1   Agent 6                        1  1
24187   01A0807   602A1   Agent 7                        1  1
24188   01A0808   602A1   Agent 8                        1  1
24961   01A0401   ASAI    ASAI LINE 1                    1  1
24962   01A1501   ASAI    ASAI LAN 01                    1  1
```

Involvement of the Telephony Server

Continuing with the above example:

1. The Telephony Server will be notified of a new call arrival and will request that an eDU be created by the VDU Server.
2. The Telephony Server will pass the call to the Workflow server to determine the call's final destination.
3. The Workflow server will make a request of the Telephony Server (TS.Route) to transfer the call to a station or queue.
4. When the Telephony Server gets the request, it will send the commands to the Avaya Definity G3 switch over the ASAI line to transfer the call.

Defining Hunt and Skill Groups

The Avaya Definity G3 system supports two environments for defining agents and assigning them to queues: EAS (Expert Agent Selection) and non-EAS.

EAS agents are associated with skill groups. If a caller needs to speak with an agent having a specific skill, such as understanding a foreign language or having knowledge of a specific procedure, s/he will be routed to a queue associated with the required skill. The next available agent with the appropriate skill will take the call and eContact Telephony will be notified that the call has been transferred to an agent.

In a non-EAS environment (also referred to simply as an ACD environment), agents are defined in hunt groups. When agents log in they must identify the queue with which they are associated. They are not affiliated with a specific skill. The Auto Available Split (AAS) parameter in the Hunt Group must be set to N.

Depending on your environment, you must identify agents by either skill groups or hunt groups.

```
Grp Grp          Grp  ACD/          Que No. Cov   Notif/      Dom   Message
No. Ext.  Name   Typ  MEAS Vec MCH Siz Mem Path Ctg Adj   Ctrl     Center

1   24301  Skill 1 ucd y/N  SK  none  5  0         n                  n

2   24302  Skill 2 ucd y/N  SK  none  5  0         n                  n

16  24316  Skill 16 ucd y/N SK none   5  0         n                  n

17  24317  Skill 17 ucd y/N SK none   5  0         n                  n
```

Summary of Adjunct Routing Requirements

The following summarizes the items that must be configured for adjunct routing to take place.

1. Stations (phones connected and configured, ASAI Link or ASAI LAN gateway configured to a station ID).
2. Network DNIS mapped to a VDN on the switch.
3. VDN assigned to a vector number.
4. Vector script performing adjunct route to the ASAI station ID.

5 Agent logins defined with associated skills.

6 Skill queues or hunt groups defined to route calls from eContact Telephony.

7 In the definition of the station link, event minimization must be set to N.

8 To allow calls to be sent to an “agentid”, the “Direct Agent Calling” parameter of the “Class of Restriction” for both the sending and receiving parties (i.e., the Agent ID and the incoming VDN) must be set to “Y”. If not set to “Y” the call could be delivered to the agent as a direct call rather than an ACD call.

Note: An agent's profile can be set to either Auto In mode or Manual In mode. This setting affects VTel's ability to control the agent's availability. Refer to the *VTel Programmer's Guide* for additional information on this setting.

Configuring Redundant CTI Links

Installing two or more Telephony Servers allows for failover in the event of a malfunction of the server or ISDN line.

There are two approaches available for configuring redundant ASAI links:

- Configure each link to connect with a different physical computer, each with its own Telephony Server.
- Configure each link to connect with a single computer running multiple Telephony Server processes on that machine. (Remember to set the “node1” configuration parameter to specify signal numbers for the ASAI links.)

Avaya Definity G3 Setup

Regardless of whether the primary and backup Telephony Servers are on the same or different hardware, two separate VDNs and vectors are required in the Avaya Definity G3, one for each Telephony Server. The division of work between ASAI links (and thus servers) is determined by the number of calls that are processed by each vector.

The following illustrates vectors that are configured to fail over to alternate ASAI lines and finally, if both links fail, to a default routing. If either Telephony Server is disabled or the ISDN connection is broken, subsequent calls to the corresponding vector are immediately channeled through the alternate ASAI link to the working Telephony Server. When the Telephony Server is reenabled (or the ISDN connection is restored) subsequent calls to that vector are processed via their usual path with no operator intervention.

```
display vector 7
Page   1 of 2
                                          CALL VECTOR

Number: 7                   Name Troy link 1
ASAI Routing? y             Basic? y        Prompting? y
EAS? y

01 adjunct     routing link extension 5100
02 wait        time 4   secs hearing ringback
03 adjunct     routing link extension 5113
04 wait        time 4   secs hearing ringback
05 route       to number 4000             if
unconditionally


display vector 13
Page   1 of 2
                                          CALL VECTOR

Number: 13                 Name Troy link   2
ASAI Routing? y            Basic? y                  Prompting? y
EAS? y

01 adjunct      routing link extension 5113
02 wait          time 4   secs hearing ringback
03 adjunct     routing link extension 5100
04 wait         time 4   secs hearing ringback
05 route        to number 4000             if
unconditionally
```

The wait time (in steps 02 and 04 of the sample vectors) should be configured to allow adequate time for the adjunct route to process normally when the link is functioning properly. If the link fails because the physical link, the Telephony Server, or the Workflow server fails, the vector will skip the "wait" and immediately execute the next step in the vector. The time-out will occur only if the link, Telephony Server, and Workflow server are up and assigned, but network traffic or database access is so slow that the route does not reach the switch in time.

Configuring the Switch in an ACD Environment

When configuring the switch for non-EAS agents, certain parameters must be set. Note that the default setting of these parameters, and even whether or not they are displayed, varies depending on whether the switch was ever set up for EAS use.

The following notes are specific to the menus displayed by the Avaya Definity G3 version 6 or higher. The wording and location of these parameters may vary slightly for other versions.

1. Log on to the Definity console using an account with appropriate security, such as the init account.
2. Enter the Change System Customer Options command.
3. On page 3 of the menu, the BCMS/VuStats loginIDs parameter should be set to N and the EAS parameter must be set to N. Note that, in some circumstances, the BCMS parameter can be set to Y. The switch will display a message indicating if it must be changed.
4. Log off for the changes to take effect.
5. Log on again (using the init account is optional for the following command).
6. Enter the Change System Parameter Features command.
7. On page 7 of the menu, the EAS Enabled parameter must be set to N.
8. On page 8 of the menu, Auxwork Reason Code and Logout Reason Code must both be set to None.

When defining hunt groups, note that the Auto Available Split (AAS) parameter must be set to "N" and the skill for the hunt group (if this option is displayed) must also be set to "N". Agent phone extensions must be included in the hunt group as group members.

CHAPTER 5
INTERFACE DEFINITION LANGUAGE (IDL)

This chapter contains the IDL for the Telephony Server, a list of all methods, and a description of each method.

IDL

```
interface TS : General {
    ORBStatus AnswerVDU( in VDU_ID vduid );

    ORBStatus Assign( in string criteria );

    ORBStatus Busy();
    ORBStatus BusyTerminate();
    ORBStatus BusyWithReason( in string reasoncode );

    ORBStatus ConferenceCancelVDU( in VDU_ID vduid );
    ORBStatus ConferenceCompleteVDU( in VDU_ID vduid );
    ORBStatus ConferenceInitVDU( in VDU_ID vduid, in string dest );

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

    ORBStatus FindVduFromAni(in string ANI, in SeqString WantVDUData,
                             out VDU_ID VDU_ID, out SeqCouple VDUData);
    void Command( in string command, in string opt1,in string opt2 );

    ORBStatus GetPBXTime( out string time );
    ORBStatus GetPhoneInfo( in string selection, out SeqCouple info );
    ORBStatus GetQueueInfo( in string queue, in string selection,
                            out SeqCouple info );
```

```
    ORBStatus HangupVDU( in VDU_ID vduid );

    ORBStatus HoldReconnectVDU( in VDU_ID vduid );
    ORBStatus HoldVDU( in VDU_ID vduid );

    ORBStatus Login( in string login, in string password,
                     in string queue, in string ext );
    ORBStatus Logout( in string queue, in string ext );
    ORBStatus LogoutWithReason( in string queue, in string ext,
                                in string reasoncode );

    ORBStatus MakeCallSetVDU( in VDU_ID vduid, in string dest );
    ORBStatus MakeCallVDU( in string dest, out VDU_ID vduid );

    ORBStatus MakePredictiveCallSetVDU( in VDU_ID vduid, in string dest,
                                  in string rings, in string allocation );
    ORBStatus MakePredictiveCallVDU( in string dest, in string rings,
                                 in string allocation, out VDU_ID vduid );

    ORBStatus Ready();
    ORBStatus ReadyAuto();

    ORBStatus Route( in VDU_ID vduid, in string dest );
    ORBStatus RouteWithInfo( in VDU_ID vduid, in string dest,
                             in string info );

    ORBStatus SendDTMFtonesVDU( in VDU_ID vduid, in string tones );

    ORBStatus TransferCancelVDU( in VDU_ID vduid );
    ORBStatus TransferCompleteVDU( in VDU_ID vduid );
    ORBStatus TransferInitVDU( in VDU_ID vduid, in string dest );
    ORBStatus TransferVDU( in VDU_ID vduid, in string dest );

    ORBStatus WrapUp();
};
```

Method Overview

The following are brief descriptions of the methods available for use with the Telephony Server as implemented for the Avaya Definity G3. Detailed descriptions begin on page 51.

In the descriptions, the term "first party" signifies the client invoking the method, "second party" is the party to whom the first party is connected, and "third party" is the party brought into an existing call.

Method	Description
AnswerVDU	Answer the phone.
Assign	Create a session with the Telephony Server.
Busy	Make the phone busy.
BusyTerminate	Make the phone available to receive calls.
BusyWithReason	Change an agent's work mode to AUX WORK.
ConferenceCancelVDU	Cancel conference, hang up third party, retrieve second party from hold.
ConferenceCompleteVDU	Retrieve second party from hold and join conference.
ConferenceInitVDU	Put second party on hold and call third party.
DropVDU	Drop a party from a conference.
FindVduFromAni	Server-to-server method.
GetPBXTime	Get the current time from the PBX.
GetPhoneInfo	Get the current state of a phone.
GetQueueInfo	Get the current state of a queue.
HangupVDU	Hang up a phone.
HoldReconnectVDU	Retrieve a party from hold.
HoldVDU	Put a party on hold.
Login	Log onto an ACD phone.
Logout	Log off of an ACD phone
LogoutWithReason	Log off an agent with a specific reason code.
MakeCallSetVDU	Initiate a call with a defined eDU
MakeCallVDU	Initiate a call.
MakePredictiveCallSetVDU	*Reserved for future use.*

Method	Description
MakePredictiveCallVDU	*Reserved for future use.*
Ready	ACD agent ready.
ReadyAuto	ACD agent auto-ready.
Route	Route a call.
RouteWithInfo	Route a call and pass information that has been collected.
SendDTMFtonesVDU	Send DTMF tones from a soft phone to the switch.
TransferCancelVDU	Cancel a transfer, hang up third party, retrieve second party from hold.
TransferCompleteVDU	Hang up the first party in a transfer.
TransferInitVDU	Put second party phone on hold and call third party (consultative transfer).
TransferVDU	Transfer, momentarily placing second party on hold (blind transfer).
WrapUp	ACD agent enter wrap-up state.

Adding a Client to an eDU's List of Interested Parties

The eDU Server maintains a list of clients interested in each eDU. When a client is no longer interested in the eDU, the client invokes the VDU.Terminate() method, which removes the client's name from the list. When the list is empty, the eDU is terminated.

A successful TransferVDU, TransferInitVDU, or ConferenceInitVDU method invocation adds the third party (the party to whom the call is being transferred or who is being included in a conference) to the list of interested parties for the eDU. That client must eventually invoke a VDU.Terminate() method to remove the client's name from the list of interested parties. (Another process may issue the Terminate on a client's behalf.)

If the third party is not logged in at the time of the Transfer or Conference, the TS cancels the request and the client's name is not added to the list. However, if the client is logged in, regardless of whether or not the request is subsequently canceled, the list is updated.

A general rule to follow is that if the client receives a TS.IncomingCall.event, the client must eventually invoke a VDU.Terminate method.

If a Terminate is not invoked, the eDU will eventually time out and be terminated automatically

Method Descriptions

All TS methods are described on the following pages.

TS.AnswerVDU

Syntax `ORBStatus AnswerVDU( <in VDU_ID vduid> );`

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

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.

Value	Description
VESP_FAILURE	Request has failed; possible internal protocol problems.
VESP_RESOURCE_NOT_AVAILABLE	No call to answer.
VESP_SERVICE_NOT_AVAILABLE	Switch does not have this function.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
Environment ev;
status = Vesp_Request( "TS.AnswerVDU", callback, user_data, session,
                                          vduid );
```

TS.Assign

Syntax `ORBStatus Assign( <in string criteria> );`

Description Create a session with the Telephony Server. Once a session is created, events will be sent to the client that corresponds to the monitoring criteria.

If the monitoring criteria contains a login ID and the equipment field in the directory is not set, the assign will be deferred until the agent logs in. The assign will return a success, but control of the phone will not take effect until after login.

If a client has assigned with the *r criteria, subsequent attempts by other clients to assign with *r will be rejected.

Parameters

Value	Description
criteria	Device to monitor or control with this session. Monitoring criteria are as follows: number — To monitor and control a phone station. Contains either an extension number or, in an EAS system, an agent ID. login_id — To monitor a person. *r — To control all call routing for an ASAI link.

Returns

Value	Description
VESP_SUCCESS	Request was successful.

Exceptions

Value	Description
VESP_BAD_SESSION	Client ID is invalid.
VESP_ASSIGN_FAILURE	Assign has failed.

Example

```
status = Vesp_Assign_Request ( "TS.Assign", &ev, callback, user_data,
                                                event_callback, session, "5112" );
```

TS.Busy

Syntax `ORBStatus Busy( void );`

Description Put an ACD or EAS phone in the "AuxWork" state. No calls will be received until a BusyTerminate or Ready is received. ACD calls will be blocked; direct calls will not be blocked.

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.BusyTerminate

Syntax `ORBStatus BusyTerminate( void );`

Description Make an ACD phone available to receive calls. The phone would be in the Ready 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 problems.
VESP_SERVICE_NOT_AVAILABLE	Service not available.

Example

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

TS.BusyWithReason

Syntax `ORBStatus BusyWithReason( <in string reasoncode> );`

Description This function changes an agent's work mode to "AuxWork."

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 function cancels a conference begun with the ConferenceInit() function.

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 problems.
VESP_ILLEGAL_STATE	State of the phone is not compatible with operation.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.ConferenceCancelVDU", callback, user_data,
                                                    session, vduid );
```

TS.Deassign

IDL Syntax `oneway void Deassign( void );`

Description Terminate a session with the Telephony Server. Once a session is terminated, the flow of events from the Telephony Server to the client will cease.

By default the device associated with this session will be logout of the switch, and the session will be terminated with the Telephony Server. Once a session is terminated, the flow of events from the Telephony Server to the client will cease. It is possible to configure the TS not to logout a device during a TS.Deassign.

Parameters

Value	Description
logout_on_deassign	True (default) issues a logout logout during a TS.Deassign. False

If you do not want to have to TS issue a logout during a TS.Deassign set it to false. However, if it is set to true, the TS will only try to log you out if it thinks that you are logged in. So in most cases, it will not try to log you out during a deassign.

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.ConferenceCompleteVDU

Syntax

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

Description This function completes the conference initiated with the initiate conference function. The party on hold is joined to the other calls.

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	No call to conference.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.ConferenceCompleteVDU", callback,
                                          user_data, session, vdu_id );
```

TS.ConferenceInitVDU

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

Description This function places a party on hold and dials a third party. If this function fails, every effort is made to retrieve the party on hold automatically. The eDUID is passed in the incoming call event the end point receives.

If successful, this function places the third party on the eDU's list of interested parties. Refer to"Adding a Client to an eDU's List of Interested Parties," on page 50 for more information.

Parameters

Value	Description
vduid	eDUID of the call
dest	Can be one of the following: extension number — The specified number is included in the conference. name — A logical phone number or a login ID, which could represent a user or a queue. The name is looked up in the eContact Telephony Directory. If the telephone number field associated with the name contains an extension number, that extension will be included in the conference.

Returns

Value	Description
VESP_SUCCESS	Request was successful

Exceptions

Value	Description
VESP_BAD_SESSION	Session is invalid.
VESP_BUSY	Destination was busy
VESP_FAILURE	Request has failed; possible internal protocol problems.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.ConferenceInitVDU", callback, user_data,
                                          session, vdu_id, 5551234 );
```

TS.DropVDU

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

Description This function is used to drop a party from a conference.

Parameters

Value	Description
vduid	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,
                                                vdu_id, dest );
```

TS.FindVduFromAni

Syntax

```
ORBStatus FindVduFromAni( <in string ANI>, <in SeqString WantVDUData>,
                                <out VDU_ID vdu_id>, <out SeqCouple VDUData> );
void Command( <in string command>, <in string opt1>, <in string opt2> );
```

Description This is a server-to-server method. Any server running in a VRU (hetero-switch transfer) environment must implement this method, even if that server is not configured to execute it. Library calls will invoke this method even if the method returns a "not found here" response.

Parameters

Value	Description
WantVDUData	Data that the invoking server wants the TS to put into the eDUID for this call.
ANI	Automatic Number Identification service; in this case, this is the reserved phone number responsible for this call's arrival at the TS.
vdu_id	eDUID for this call.
VDUData	Data requested via the WantVDUData input parameter.

Returns

Value	Description
VESP_SUCCESS	An eDUID was located, mapped to this ANI, and all data items specified in the WantVDUData parameter were located.
VESP_PARTIAL_SUCCESS	Similar to VESP_SUCCESS, but not all data items specified in the WantVDUData parameter were found.

Exceptions

Value	Description
VESP_NOT_FOUND	Library lookup indicated there was an eDUID mapped to an ANI, but the function did not return the eDUID.

TS.GetPBXTime

Syntax `ORBStatus GetPBXTime( <out string time> );`

Description This function returns a string consisting of the current day, month, year, and hour.

Parameters

Value	Description
time	A string containing the date and time.

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.GetPBXTime", callback, user_data, session,
                                        &time );
```

TS.GetPhoneInfo

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

Description Returns switch-dependent information about the state of a phone. If the client is an ACD or EAS agent, the mode is also returned.

Parameters

Value	Description
selection	Kind of information to be retrieved. Possible values are: "type" — Domain type and extension type of this agent's teleset. "login" — Agent login audit information. "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 and agent mode. State can be "active", "idle", or "null". Agent mode can be "busy", "wrapup", "autoready", "ready", or "null".

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 Returns switch-specific information about the queue.

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.HangupVDU

Syntax

```
ORBStatus HangupVDU( <in VDU_ID vduid> );
```

Description This function hangs up the voice portion of a call. The eDU remains active for any call wrap-up activities required by the application.

Note: To have an agent move to a wrap-up state after hanging up a call, do not program the switch so that the agent is "auto-ready" after hang-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; possible internal protocol problems.
VESP_SERVICE_NOT_AVAILABLE	Service not available

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.HangupVDU", callback, user_data, session,
                                          vdu_id );
```

TS.HoldReconnectVDU

Syntax `ORBStatus HoldReconnectVDU( <in VDU_ID vduid> );`

Description Removes a call from the Hold state.

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 failed.
VESP_RESOURCE_NOT_AVAILABLE	No call on hold.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.HoldReconnectVDU", callback, user_data,
                                              session, vdu_id );
```

TS.HoldVDU

Syntax `ORBStatus HoldVDU( <in VDU_ID vduid> );`

Description This function places the voice portion of a call on hold. The eDU can 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_FAILURE	Request failed.
VESP_RESOURCE_NOT_AVAILABLE	No call to put on hold.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.HoldVDU", callback, user_data, session,
                                          vdu_id );
```

TS.Login

Syntax

```
ORBStatus Login( <in string login>, <in string password>,
                 <in string queue>, <in string ext> );
```

Description This function logs an ACD or feature phone on to the switch. Refer to the Definity documentation to check support and meaning of parameters.

Parameters

Value	Description
login	Logical identification of the agent as defined in the Phone field of the Directory Server.

Value	Description
password	Password as the switch knows it (none if non-EAS).
queue	For non-EAS systems, enter the group extension number as defined in the hunt group. For an EAS system, enter the physical phone equipment number the agent is using. (This duplicates the ext field for EAS agents.)
ext	Extension to log in as. This is the physical equipment number. For non-EAS, this must be included as a group member in the hunt group.

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,
                                          "1234", "5678", "4000", "4009" );
```

TS.Logout

Syntax `ORBStatus Logout( <in string queue>, <in string ext> );`

Description This function logs off an ACD or feature phone from the switch.

Parameters

Value	Description
queue	Queue to log off from.
ext	Extension of agent.

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,
                                                "4000", "4009" );
```

TS.LogoutWithReason

Syntax

```
ORBStatus LogoutWithReason( <in string queue>, <in string ext>,
                            <in string reasoncode> );
```

Description This function logs out an agent. If the session's phone is of type EAS, the TS requests the PBX to logout this agent with the supplied reason code.

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.MakeCallSetVDU
TS.MakeCallVDU

Syntax

```
ORBStatus MakeCallSetVDU( <in VDU_ID vduid>, <in string dest> );

ORBStatus MakeCallVDU( <in string dest>, <out VDU_ID vduid> );
```

Description These functions initiate a call attempt. MakeCallSetVDU uses a specified eDU. MakeCallVDU generates a new eDU and returns it.

The status of the call attempt is reported back to the client.

The MakeCallVDU function fails if it receives any type of busy event, recording the call attempt details and reason for termination.

It is possible that a Busy event will be generated if the destination is busy.

If successful, a Ring event will be generated. If the destination answers, a CallConnect event will be generated.

If the originating caller hangs up before the call is connected, no disconnect event is generated.

Parameters

Value	Description
dest	Destination of the call. Can be one of the following: extension number — The call is made to the specified number. name — A logical phone number or a login ID, which could represent a user or a queue. The name is looked up in the eContact Telephony Directory. If the telephone number field associated with the name contains an extension number, the call will be made to that extension.
vduid	eDUID of the new call.

Returns

Value	Description
VESP_SUCCESS	Request was successful.

Exceptions

Value	Description
VESP_BAD_SESSION	Session is invalid
VESP_BUSY	Destination is busy.
VESP_FAILURE	Request failed.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.MakeCallSetVDU", callback, user_data,
                                              session, vdu_id, "400" );
VDU_ID vduid;
status = Vesp_Request( "TS.MakeCallVDU", &ev, callback, user_data,
                                             session, "400", &vduid );
```

TS.MakePredictiveCallSetVDU

Reserved for future use.

TS.MakePredictiveCallVDU

Reserved for future use.

TS.Ready

Syntax `ORBStatus Ready( void );`

Description Places a telephone set 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 problems.
VESP_SERVICE_NOT_AVAILABLE	Service not available

Example

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

TS.ReadyAuto

Syntax `ORBStatus ReadyAuto( void );`

Description Automatically places a telephone set in the "ready" state in preparation for receiving a telephone call every time the phone is hung up.

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.ReadyAuto", callback, user_data, session );
```

TS.Route

Syntax `ORBStatus Route( <in VDU_ID vduid>, <in string dest> );`

Description This method instructs the switch to route the call to a new destination. Call this method in response to an event, such as TS.IncomingCall.event.

Parameters

Value	Description
vduid	eDUID of the call to be routed.
dest	Destination of the call. Can be one of the following: extension number — The call is made to the specified number. name — A logical phone number or a login ID, which could represent a user or a queue. The name is looked up in the eContact Telephony Directory. If the telephone number field associated with the name contains an extension number, the call will be made to that extension.

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.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.Route", callback, user_data,
                           session, vduid, "5000" );
```

TS.RouteWithInfo

Syntax

```
ORBStatus RouteWithInfo( <in VDU_ID vduid>, <in string dest>,
                         <in string digits> );
```

Description This method requests the switch to route the call to a new destination, passing digits that have been collected, such as an account number or Social Security number. Call this method in response to an event, such as TS.IncomingCall.event.

Parameters

Value	Description
vduid	eDUID of the call to be routed.
dest	Destination of the call. Can be one of the following: extension number — The call is made to the specified number. name — A logical phone number or a login ID, which could represent a user or a queue. The name is looked up in the eContact Telephony Directory. If the telephone number field associated with the name contains an extension number, the call will be made to that extension.
digits	Any digits that have been collected.

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.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.RouteWithInfo", callback, user_data,
                                          session, vduid, "5000", 3989049 );
```

TS.SendDTMFtonesVDU

Syntax `ORBStatus SendDTMFtonesVDU( <in VDU_ID vduid>, <in string tones> );`

Description This function sends DTMF tones from a soft phone to the switch, just as though they had been generated on the phone set keypad.

Parameters

Value	Description
vduid	eDUID of the call.
tones	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, eDU_id, tones );
```

TS.TransferCancelVDU

Syntax `ORBStatus TransferCancelVDU( <in VDU_ID vduid> );`

Description This function cancels a transfer begun with the TransferInitVDU function.

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_ILLEGAL_STATE	State of the phone is not compatible with the operation.
VESP_FAILURE	Request has failed.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.TransferCancelVDU", callback, user_data,
                                              session, vduid );
```

TS.TransferCompleteVDU

Syntax `ORBStatus TransferCompleteVDU( <in VDU_ID vduid> );`

Description This function completes the transfer started with the initiate transfer function. The party on hold is connected with the called third party and the first party 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_ILLEGAL_STATE	State of the phone is not compatible with the operation.
VESP_FAILURE	Request has failed.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.TransferCompleteVDU", callback, user_data,
                                                 session, vduid );
```

TS.TransferInitVDU

Syntax `ORBStatus TransferInitVDU( <in VDU_ID vduid>, <in string dest> );`

Description This function places the call initiator on hold and calls a third party. If this function fails, the call initiator is retrieved from hold. The eDUID is passed in the incoming call event the end point receives.

If successful, this function places the third party on the eDU's list of interested parties. Refer to "Adding a Client to an eDU's List of Interested Parties," on page 50 for more information.

Parameters

Value	Description
vduid	eDUID of the call to be routed.
dest	Destination of the call. Can be one of the following: extension number — The call is made to the specified number. name — A logical phone number or a login ID, which could represent a user or a queue. The name is looked up in the eContact Telephony Directory. If the telephone number field associated with the name contains an extension number, the call will be made to that extension.

Returns

Value	Description
VESP_SUCCESS	Request was successful

Exceptions

Value	Description
VESP_BAD_SESSION	Session is invalid
VESP_BUSY	Destination was busy
VESP_FAILURE	Request has failed; possible internal protocol problems.
VESP_ILLEGAL_STATE	State of the phone is not compatible with operation

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.TransferInitVDU", callback, user_data,
                              session, vduid, "5000" );
```

TS.TransferVDU

Syntax `ORBStatus TransferVDU( <in VDU_ID vduid>, <in string dest> );`

Description This function transfers a call and its eDU to a destination. The second party (the party being transferred) is momentarily placed on hold and the destination receives an incoming call event. This is a blind transfer.

If successful, this function places the third party on the eDU's list of interested parties. Refer to "Adding a Client to an eDU's List of Interested Parties," on page 50 for more information.

Parameters

Value	Description
vduid	eDUID of the call to be routed.
dest	Destination of the call. Can be one of the following: extension number — The call is made to the specified number. name — A logical phone number or a login ID, which could represent a user or a queue. The name is looked up in the eContact Telephony Directory. If the telephone number field associated with the name contains an extension number, the call will be made to that extension.

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.

Value	Description
VESP_ILLEGAL_STATE	State of the phone is not compatible with operation.
VESP_RESOURCE_NOT_AVAILABLE	No call to transfer.

Example

```
VDU_ID vduid = "3016acec000700007800002c1b580002";
status = Vesp_Request( "TS.TransferVDU", callback, user_data, session,
                                vduid, "4500" );
```

TS.WrapUp

Syntax `ORBStatus WrapUp( void );`

Description This function places the phone in a wrap-up state. This method is only useful for ACD phones.

Returns

Value	Description
VESP_SUCCESS	Request was successful.

Exceptions

Value	Description
VESP_BAD_SESSION	Session is invalid.
VESP_FAILURE	Request has failed.
VESP_SERVICE_NOT_AVAILABLE	Service not available

Example

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

CHAPTER 6
EVENTS

This chapter describes the Telephony Server event messages.

List of Events

The following events are generated when a state change has occurred on the switch. It is possible to receive an event before getting a response to the request that initiated the action.

Event	Description
Busy	Destination is busy.
CallOffered	Inbound call has entered a monitored queue.
Conference	Call has been conferenced.
Connect	Call has been connected with an end point.
Disconnect	Call has been terminated.
Drop	Party has dropped from the call.
Hold	Call has been put on hold.
HoldReconnect	Call has been taken off hold.
IncomingCall	A call is present at the device.

Event	Description
Queued	Outbound call has been queued.
Ring	End point is ringing.
ServerFailed	Server has failed.
SessionFailed	PBX has rejected the association with the phone.
Transfer	Call has been transferred.

Event Descriptions

This section describes the Telephony Server events and the fields returned with each event.

Note: This section lists some of the fields returned with each event. Additional information may be returned. The fields are not necessarily listed in the order in which they are returned.

TS.Busy

This event informs the client application that the party called is busy.

Field	Description
called	The number used by the Telephony Server to place the call. This could be an equipment number, a logical ID, or a queue number.
cause	Reason why phone is busy, as reported by the switch.
vdu_id	eDUID of the call

TS.CallOffered

This event informs the client application that an incoming call has entered a monitored queue.

Field	Description
call_ref_id	Call reference ID of call, from the perspective of the switch.

TS.Conference

This event informs a client application that a third party has been added to a call. The Conference event is sent to all parties monitoring the call.

Field	Description
call_ref_id	Previous reference id of call, used for debugging purposes only.
called	The number used by the Telephony Server to place the call. This could be an equipment number, a logical ID, or a queue number.
dest	The physical extension number. (For external conferences, this is the external phone number dialed.) This event can contain multiple dest name/value pairs.
new_call_ref_id	New call reference id.
number_in_call	A digit representing the number of parties involved in the conference.
orig	The equipment number (physical phone number) of the party that initiated the conference.
vdu_id	eDUID of the call

TS.Connect

This event informs a client application that the call has been connected. Connect results from a MakeCallVDU, TransferInitVDU, ConferenceInitVDU, or direct call.

Field	Description
call_ref_id	Call reference id of call, used for debugging purposes only.
called	The number used by the Telephony Server to place the call. This could be an equipment number, a logical ID, or a queue number.
dest	The physical extension number. (For external conferences, this is the external phone number dialed.) This event can contain multiple dest name/value pairs.
iidigits	Information about the originating line, supplied by public network and reported to switch.
orig	The equipment number (physical phone number) of the party that initiated the conference.
vdu_id	eDUID of this call.

TS.Disconnect

This event informs a client application that the phone has been disconnected after connect or queued events.

Field	Description
call_ref_id	Call reference ID of call, used for debugging purposes only.
ctype	Call type, if supplied by switch.
vdu_id	eDUID of this call.

TS.Drop

This event informs the client that a party has been dropped from a call.

Field	Description
call_ref_id	Call reference ID of call, used for debugging purposes only.
dest	Physical number of the dropped party.
vdu_id	eDUID of this call.

TS.Hold

This event informs a client application that a telephone line has been put on hold.

Field	Description
call_ref_id	Call reference ID of call, used for debugging purposes only.
dest	Destination of the call on hold.
vdu_id	eDUID of this call.

TS.HoldReconnect

This event informs a client application that a telephone line has been retrieved from hold.

Field	Description
call_ref_id	Call reference ID of call, used for debugging purposes only.
dest	Destination of the call on hold.
vdu_id	eDUID of this call.

TS.IncomingCall

This event informs a client application that a telephone line for which it is responsible is in the ringing state or has received a call set-up request. In addition to telling the client that its associated telephone number is alerting, it also passes the eDU with the event to the client.

Once a client has received an IncomingCall event, that client is responsible for invoking the VDU.Terminate method when it is no longer interested in the eDU.

Field	Description
ani	Automatic Number Identification. The caller's 10-digit telephone number.
call_ref_id	Call reference ID of call, used for debugging purposes only.
called	The number used by the Telephony Server to place the call. This could be an equipment number, a logical ID, or a queue number.
calltype	Can be either "direct" (a direct call) or "queue" (a queued call).
ctype	Can be "acd" (an ACD call) or "direct" (a direct call).
dest	The actual phone number to which the call was placed by the switch. This is typically an equipment (phone) number.
digits	For a routed call, the digits collected by the PBX, if any.
dnis	Dialed Number Identification Service; the number dialed.
ext	Extension of agent receiving the call.
group	Group ID of agent receiving the call.
iidigits	Information about the originating line, supplied by public network and reported to switch.
loginid	Login ID of agent receiving the call.

Field	Description
orig	The equipment number (physical phone number) of the party that placed the call or initiated the transfer or conference to the party receiving this event.
queue	The number of the queue that routed the call. If only one queue is involved in the call, this will be the same as the "called" field. If a queue overflows, "called" will contain the dialed queue, and "queue" will contain the queue that actually routed the call to the agent.
vdu_id	eDUID of the call.

TS.Queued

This event informs a client application that a call has been queued. When the call is finally answered, a connect event is generated. In addition to telling the client that its associated telephone number is alerting, it also passes the eDU with the event to the client.

Field	Description
call_ref_id	Call reference ID of call, used for debugging purposes only.
number_in_queue	Number of calls in the queue.
queue	Number dialed to access the queue.
vdu_id	eDUID of this call.

TS.Ring

This event informs a client application that a telephone line for which it is responsible is in the ringing state or has received a call set-up request. In addition to telling the client that its associated telephone number is alerting, it also passes the eDU with the event to the client. This event occurs for outbound, transferred, and conferenced calls.

Field	Description
call_ref_id	Call reference ID of call, used for debugging purposes only.
called	Number originally dialed.
dest	Destination of the call. At first this is equivalent to the DNIS, but it changes as the call is routed through the system.
orig	Caller or originator of the transfer or conference.
vdu_id	eDUID of this call.

TS.ServerFailed

This event informs a client application that the server has failed.

TS.SessionFailed

This event informs a client that the PBX has rejected the association with the phone. The client must de-assign and assign again.

TS.Transfer

This event informs a client that a call has been transferred. The transfer event contains the name/value pairs of all parties in the resulting call. The Transfer event is sent to the client initiating the transfer.

Field	Description
vdu_id	VDUID of this call.
call_ref_id	Previous reference id of call, used for debugging purposes only.
new_call_ref_id	New call reference id.
orig	The equipment number (physical phone number) of the party that initiated the transfer.
dest	The physical extension number. (For external transfers, this is the external phone number dialed.) This event can contain multiple dest name/value pairs.
called	The number used by the Telephony Server to place the call. This could be an equipment number, a logical ID, or a queue number.
number_in_call	Number of monitored extensions for this call (if reported by switch).

CHAPTER 7
ALARMS

This chapter lists and describes the alarms generated by the Telephony Server. For each alarm, a cause and recommended remedial action are given.

List of Alarms

Alarm Name	Priority	Description	Cause/Recommended Action
Abort_Association	high	PBX has aborted a device monitor for a TS assigned client	Note alarm details, turn on tracing to gather more information
Bad_Argument	info, low	TS method detected a parameter problem	Varies with the reported parameter
Bad_Configuration	info	TS detected problem with a configuration value	Change value and restart the TS
Bad_Request	high	A response from the PBX arrived after client request has timed out, or client has logged out	Server or PBX could be getting overloaded, or increase timeout configuration value for aging open client requests

Alarm Name	Priority	Description	Cause/Recommended Action
Bad_Response	high	PBX sent a response that TS could not recognize, or the client request did not match the type of the PBX response	Server or PBX could be getting overloaded, or ASAI link is having problems
DirLoadFail	high	Could not load local tables from Directory Server.	Check the Directory Server
DirUpDateFail	high	Could not update local tables from Directory Server.	Check the Directory Server
Duplicate_Element	high	2 call entries were found where only one was expected to be in the list	Save logfiles, report to Quintus
Duplicate_Event	info	2 events arrived from the PBX where one was sufficient	none
Failed_DelSubTree	high	ADU method request to delete subtree failed	Generally a network environment problem or a server bug, capture logfile
Failed_ObjCopy	high	Method request ORB_object_copy_to_string failed	Report to Quintus
Failed_ObjRelease	low	Method request Object_release failed	Some memory not released, TS will expand
Failed_ObjToString	high	Method request ORB_object_to_string failed	Report to Quintus
Failed_SendEvent	high	Method request to send a TS.event to assigned client failed	Generally a network environment problem or a server bug, capture logfile
Failed_SendResponse	high	Method request failed to send a TS.xxx.response to assigned client	Generally a network environment problem or a server bug, capture logfile
Failed_SetAndTrans	high	eDU method request to set values in an eDU and transfer failed	Verify eDU server is running, capture logfiles for eDU and TS

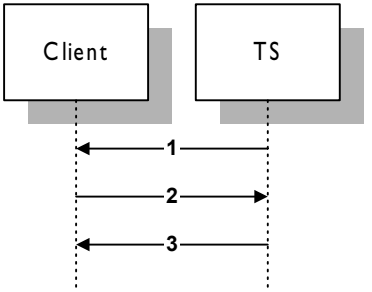
Alarm Name	Priority	Description	Cause/Recommended Action
Failed_SetValues	high	eDU method request to set values in an eDU failed	Verify eDU server is running, capture logfiles for eDU and TS
Failed_TermADU	high	ADU method request to terminate an ADU failed	Generally a network environment problem or a server bug, capture logfile
Failed_VDUAssign	high	eDU method request to assign to eDU failed	Verify eDU server is running, capture logfiles for eDU and TS
Link_Heartbeat	info	ASAI link is up	none
Link_Heartbeat	low	TS could not enable PBX heartbeat	none, TS will initiate heartbeat handshakes with PBX instead
LinkLostAll	emergency	Lost all connections to the switch. This is a catastrophic failure and disconnect events will be sent to the client application.	Check the PBX link. Use ASAI Heartbeat, Tracing, or Link Status to verify
LinkNotDefined	high	Link to the switch is not defined. Server will exit.	Define the link
NIVR_FailedRequest	high	Failed internal NIVR library call	Internal error, save TS and VOX logfiles, report to Quintus
NIVR_MissingData	info	Unable to find all expected data items in a call structure	none
NIVR_Unknown VDUID	high	Unable to find eDU to match to the current call	Internal error, save TS and VOX logfiles, report to Quintus
Read	high	ASAI library call to read socket failed	Note alarm's full details, ASAI link may need to be reset
VDUNotCreated	high	eDU could not be created for call; major system problem.	Check the eDU Server and the system load
Write	high	ASAI library call to write socket failed	Note alarm's full details, ASAI link may need to be reset

APPENDIX A
GENERIC CALL FLOWS

This appendix illustrates and describes some generic call flows.

Route Call

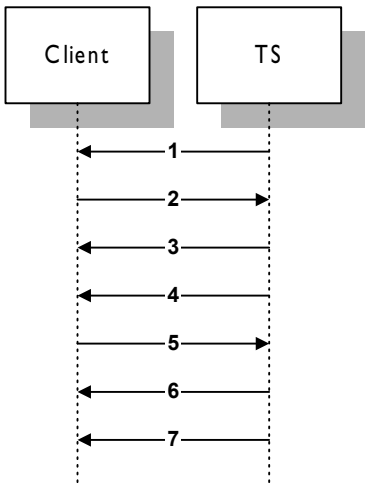
The following diagram illustrates the events and methods that are exchanged when the TS routes a call to a client.



Step	Description
1	TS sends TS.IncomingCall.event(*vdu_id, call_id, ani, dnis, ctype*) to client.
2	Client invokes TS.RouteVDU(*vdu_id, dest*) method on the TS.
3	TS returns TS.RouteVDU.response(*vdu_id, dest*) to the client.

Inbound Call

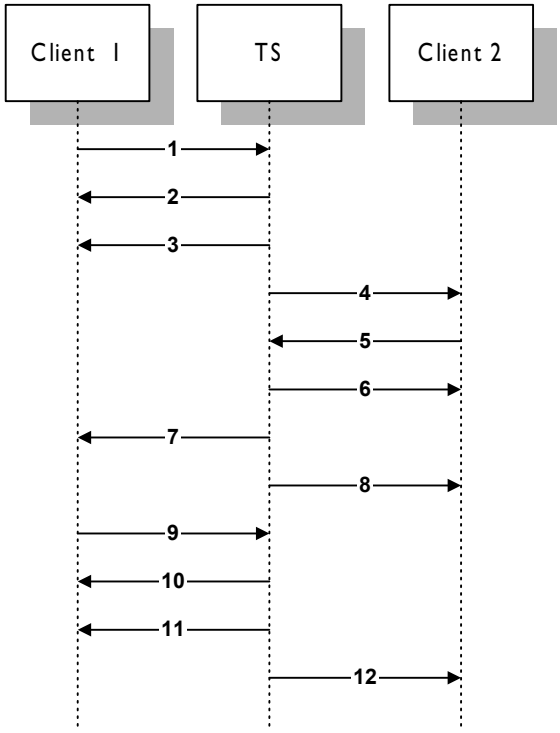
The following diagram illustrates the methods and events that are exchanged after a call arrives at the TS from the PBX or ACD. The sequence ends with the call being terminated by the client.



Step	Description
1	TS sends a TS.Incoming.event(*vdu_id, ani, dnis, call_ref_id*) to the client.
2	Client sends a TS.AnswerVDU(*vdu_id*) to the TS.
3	TS returns a TS.AnswerVDU.response(*vdu_id*) to the client.
4	TS sends a TS.Connect.event(*vdu_id, call_ref_id*) to the client.
5	Client invokes the TS.HangupVDU(*vdu_id*) method on the TS.
6	TS returns a TS.HangupVDU.response(*vdu_id*) to the client.
7	TS sends a TS.Disconnect.event(*vdu_id, call_ref_id*) to the client.

Outbound Call

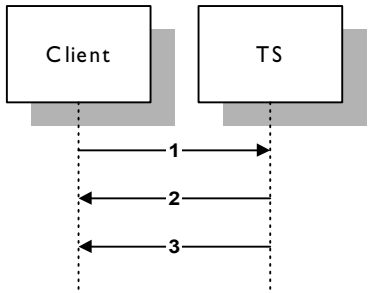
The following diagram illustrates the methods and events that are exchanged during an outbound call. In this example, Client 1 is either an agent or a queue and Client 2 is an automatic dialer. The sequence ends with the call being terminated.



Step	Description
1	Client 1 invokes TS.MakeCallVDU(*dest*) method on the TS.
2	TS returns TS.MakeCallVDU.response(*dest, vdu_id*) to Client 1.
3	TS sends a TS.Ring.event(*vdu_id, call_ref_id*) to Client 1.
4	TS sends a TS.IncomingCall.event(*vdu_id, ani, dnis, call_ref_id*) to Client 2.
5	Client 2 invokes a TS.AnswerVDU(*vdu_id*) method on the TS.
6	TS returns a TS.AnswerVDU.response(*vdu_id*) to Client 2.
7	TS sends a TS.Connect.event(*vdu_id, call_ref_id*) to Client 1.
8	TS sends a TS.Connect.event(*vdu_id, call_ref_id*) to Client 2.
9	Client 1 invokes a TS.HangupVDU(*vdu_id*) on the TS.
10	TS returns a TS.HangupVDU.response(*vdu_id*) to Client 1.
11	TS sends a TS.Disconnect.event(*vdu_id, call_ref_id*) to Client 1.
12	TS sends a TS.Disconnect.event(*vdu_id, call_ref_id*) to Client 2.

Busy Destination

The following diagram illustrates the methods and events that are exchanged when the call destination is busy.



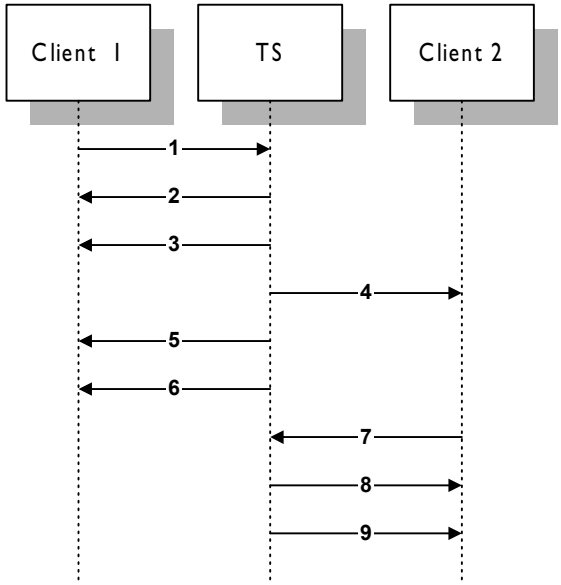
Step	Description
1	Client invokes TS.MakeCallVDU(*dest*) on the TS.
2	TS returns TS.MakeCallVDU.response(*dest, vdu_id*) to the client.
3	TS returns TS.Busy.event(*vdu_id.call_ref_id*) to the client.

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).

ACD sets must be in the Ready state. Calls will be answered automatically if the switch is configured to AutoAnswer or if AutoAnswer is set through VTel.

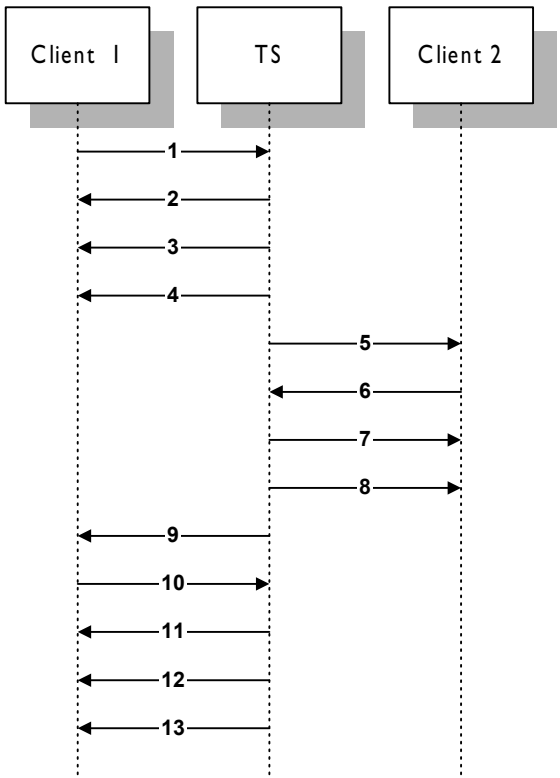
If the call originates within the switch, additional events will be generated that are not indicated in this model.



Step	Description
1	Client 1 invokes a TS.TransferVDU(*vdu_id, dest*) method on the TS.
2	TS sends a TS.Hold.event(*vdu_id, call_ref_id, dest*) to Client 1.
3	TS returns a TS.TransferVDU.response(*dest, vdu_id*) to Client 1.
4	TS sends TS.IncomingCall.event(*vdu_id, call_ref_id, ani, dnis, orig, dest*) to Client 2.
5	TS sends a TS.Transfer.event(*vdu_id, call_ref_id, new_call_ref_id*) to Client 1.
6	TS sends a TS.Disconnect.event(*vdu_id, call_ref_id*) to Client 1.
7	Client 2 invokes a TS.AnswerVDU(*vdu_id*) method on the TS.
8	TS returns a TS.AnswerVDU.response(*vdu_id*) to Client 2.
9	TS sends a TS.Connect.event(*vdu_id, call_ref_id*) to Client 2.

Consultative Transfer

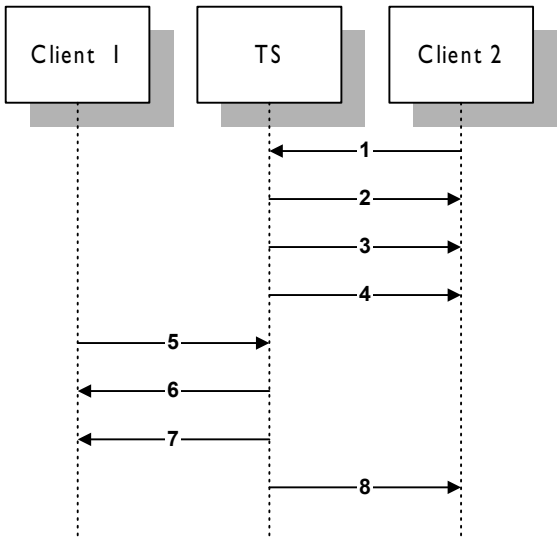
The following diagram illustrates the methods and events that are exchanged during a consultative transfer (a transfer in which the first agent remains on the line until the next agent picks up the call).



Step	Description
1	Client 1 invokes TS.TransferInitVDU(*vdu_id, dest*) method on the TS.
2	TS sends TS.Hold.event(*vdu_id, call_ref_id, dest*) to Client 1.
3	TS returns TS.TransferInitVDU.response(*vdu_id, dest*) to Client 1.
4	TS sends TS.Ring.event(*vdu_id, call_ref_id*) to Client 1.
5	TS sends TS.IncomingCall.event(*vdu_id, call_ref_id, ani, dnis*) to Client 2.
6	Client 2 sends TS.AnswerVDU(*vdu_id*) to TS.
7	TS returns TS.AnswerVDU.response(*vdu_id*) to Client 2.
8	TS sends TS.Connect.event(*vdu_id, call_ref_id*) to Client 2.
9	TS sends TS.Connect.event(*vdu_id, call_ref_id*) to Client 1.
10	Client 1 invokes TS.TransferCompleteVDU (*vdu_id*) method on the TS.
11	TS returns TS.TransferCompleteVDU.response(*vdu_id*) to Client 1.
12	TS sends TS.Transfer.event(*vdu_id, call_ref_id, new_call_ref_id*) to Client 1.
13	TS sends TS.Disconnect.event(*vdu_id, call_ref_id*) to 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.



Step	Description
1	Client 2 invokes TS.MakeCallVDU(*dest*) method on the TS.
2	TS returns TS.MakeCallVDU.response(*vdu_id, dest*) to Client 2.
3	TS sends TS.Queued.event(*vdu_id, call_ref_id*) to Client 2.
4	TS sends TS.IncomingCall.event(*vdu_id, ani, dnis, call_ref_id*) to Client 2.
5	Client 1 invokes TS.AnswerVDU(*vdu_id*) on the TS.

Step	Description
6	TS returns TS.AnswerVDU.response(*vdu_id*) to Client 1.
7	TS sends TS.Connect.event(*vdu_id, call_ref_id*) to Client 1.
8	TS sends TS.Connect.event(*vdu_id, call_ref_id*) to Client 2.

APPENDIX B
QES 5.1 CALL CONTAINERS

This appendix provides information about call containers for QeS 5.1, which are still supported in Quintus eContact 5.6. Information about call containers for Quintus eContact 5.6 is presented in Chapter 2.

QeS 5.1 Call Container Contents

The table below describes a call container in which end point events and attributes are stored. X represents the unique identification number for each end point. Y and Z represent sequence numbers within each end point's activities.

Name	Value	Explanation
ts.X	delta time	For ts.1 this is the base time (should be zero). In all other cases it is the elapsed time in seconds between the creation of the eDU and the connection to the end point.
ts.X.loginid	loginid	Login ID of client.
ts.X.session	session	Session ID of client.
ts.X.UUID	UUID	UUID of a given leg of the call.
ts.X.phone	phone number	Phone number of client.

Name	Value	Explanation
ts.X.ptype	phone type	Phone number type of client.
ts.X.equip	equipment number	Equipment number of client.
ts.X.in	delta time	The elapsed time in seconds between the creation of the eDU and the incoming call event. Only one per end point (at most).
ts.X.in.q	queue number	Incoming call's queue number.
ts.X.in.c	called number	Incoming call's called number (DNIS).
ts.X.in.d	connect number	Number that incoming call connected to (destination).
ts.X.in.o	originating number	Incoming call's originating number (ANI).
ts.X.ri.Y	delta time	The elapsed time in seconds between the creation of the eDU and the alerting event (the telephone starts to ring) for outbound calls and conference/transfer attempts.
ts.X.ri.Y.c	called number	The called (dialed) number for outbound (ringing) call or conference/ transfer attempt.
ts.X.ri.Y.d	connect number	Number that outbound (ringing) call or conference/transfer attempt is actually connecting to (destination).
ts.X.ri.Y.o	originating number	Originating number for outbound (ringing) call or conference/transfer attempt.
ts.X.ri.q	queue number	The ACD queue number on which the ring event arrived.
ts.X.que.Y	delta time	The elapsed time in seconds between the creation of the eDU and placing the call in a queue.
ts.X.que.Y.q	queue number	The queue the call is in before ringing at the agent desktop.

Name	Value	Explanation
ts.X.que.Y.n	number	The number of calls in the queue, including this call.
ts.X.rt	queue	The queue controlling the route.
ts.X.rt.d	destination number	The dialable number of the route destination.
ts.X.rt.o	originating number	The number from which the routing originated (ANI).
ts.X.rt.digits	digits	Digits collected at the switch through which the routing occurs.
ts.X.busy.Y	delta time	The elapsed time in seconds between the creation of the eDU and the busy event (busy signal occurs) for outbound calls and conference/transfer attempts.
ts.X.busy.Y.d	called number	The dialed number for outbound calls and conference/transfer attempts that encounter a busy signal.
ts.X.co.Y	delta time	The elapsed time in seconds between the creation of the eDU and the connection of the call.
ts.X.co.Y.d	connect number	Number that outbound call or conference/transfer attempt actually connected to (destination).
ts.X.co.Y.c	called number	The called (dialed) number for outbound call or conference/transfer attempt that resulted in the current connection.
ts.X.co.Y.o	originating number	The originating number for outbound call or conference/transfer attempt that resulted in the current connection.
ts.X.hold.Y	delta time	The elapsed time in seconds between the creation of the eDU and the placing of the call on hold.

Name	Value	Explanation
ts.X.recon.Y	delta time	The elapsed time in seconds between the creation of the eDU and the reconnection to the call of a party previously placed on hold. (Does not occur if the hold is followed by a transfer (ts.X.tran) or conference (ts.X.con.Y).)
ts.X.tran.Y	delta time	The elapsed time in seconds between the creation of the eDU and the transfer of the call.
ts.X.tran.Y.c	called number	The called number of the transfer destination.
ts.X.con.Y	delta time	The elapsed time in seconds between the creation of the eDU and the time the call is conferenced.
ts.X.con.Y.d.Z	destination	The dialable number of the conference destination. May not be supported on all PBX types.
ts.X.drop.Y	delta time	The elapsed time in seconds between the creation of the eDU and a party being dropped from the call.
ts.X.drop.Y.d	dropped number	The dialable number of the party dropped from a call.
ts.X.disc	delta time	The elapsed time in seconds between the creation of the eDU and the disconnection of the end point. Only one per end point (at most).

Reporting from QeS 5.1 Call Containers

State durations (connect duration, talk time, hold time, etc.) and event counts (number of times a call was placed on hold, etc.) are not calculated by the TS, but can be derived from the time stamps and other information in a TS container.

The following table describes use of the process measurement data in a TS container.

Measurement	Definition	Calculation Technique
Hold Time	The duration of the hold state. (Hold can be terminated by conference or transfer.)	For each hold/reconnect, hold/ conference, hold/transfer pair, subtract the delta time of the hold event from the delta time of the corresponding reconnect, conference, or transfer event.
Talk Time	The length of time that each agent was connected to the telephone call (but not on hold).	For each end point, subtract the delta time when the call was connected from the delta time when the call was dropped, then subtract the total hold time for that end point from the result.
Hold Count	The total number of times that the call was placed on hold.	Count the number of ts.X.hold.Y events.To identify who placed a call on hold, you might count the number of ts.X.hold.Y events for each X.

INDEX