Avaya Media Processing Server SIP 6.3 Features Manual

3.5
NN44100-130
October 2011

© 2011 Avaya Inc.

All Rights Reserved.

Notice

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer

“Documentation” means information published by Avaya in varying mediums which may include product information, operating instructions and performance specifications that Avaya generally makes available to users of its products. Documentation does not include marketing materials. Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of documentation unless such modifications, additions, or deletions were performed by Avaya. End User agrees to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

Link disclaimer

Avaya is not responsible for the contents or reliability of any linked Web sites referenced within this site or documentation provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty

Avaya provides a limited warranty on its Hardware and Software (“Product(s)”). Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya’s standard warranty language, as well as information regarding support for this Product while under warranty is available to Avaya customers and other parties through the Avaya Support Web site: http://support.avaya.com. Please note that if you acquired the Product(s) from an authorized Avaya reseller outside of the United States and Canada, the warranty is provided to you by said Avaya reseller and not by Avaya.

Licenses

THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE, HTTP://SUPPORT.AVAYA.COM/LICENSEINFO/ ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AUTHORIZED AVAYA RESELLER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AUTHORIZED AVAYA RESELLER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA AUTHORIZED RESELLER; AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS “YOU” AND “END USER”), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE (“AVAYA”).

Copyright

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation, Software, or Hardware provided by Avaya. All content on this site, the documentation and the Product provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software unless expressly authorized by Avaya. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil offense under the applicable law.

Third-party components

Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements (“Third Party Components”), which may contain terms that expand or limit rights to use certain portions of the Product (“Third Party Terms”). Information regarding distributed Linux OS source code (for those Products that have distributed the Linux OS source code), and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site: http://support.avaya.com/Copyright.

Trademarks

The trademarks, logos and service marks (“Marks”) displayed in this site, the Documentation and Product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the Documentation and Product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party.

Avaya is a registered trademark of Avaya Inc.

All non-Avaya trademarks are the property of their respective owners, and “Linux” is a registered trademark of Linus Torvalds.

Downloading Documentation

For the most current versions of Documentation, see the Avaya Support Web site: http://support.avaya.com.

Contact Avaya Support

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

Contents

Chapter 1: New in this release 9
- Navigation 9
- Features 9
- CCSS SIP Integration with MPS N+1 redundancy 9
- Other changes 10
- Revision history 10

Chapter 2: Preface 13
- Scope 13
- Intended Audience 13
- Customer service 13
- Getting technical documentation 13
- Getting product training 14
- Getting help from a distributor or reseller 14
- Getting technical support from the Avaya Web site 14
- How to Use This Manual 14
- Organization of this Manual 15
- Conventions Used in This Manual 16
- Solaris and Windows Conventions 17
- Two-Button (Windows) versus Three-Button (Solaris) Mouse 17
- Trademark Conventions 18
- Product Nomenclature Changes 18

Chapter 3: Introduction 21
- Common Channel Signaling Server (CCSS) 21
- CCSS Architectural Overview 21
- System Software 23
- Ethernet Connections 24
- Session Initiation Protocol (SIP) 24
- MPS and CCSS 6.3 SIP Key Features 25
- CCSS 6.3 SIP Message Support 26
- MPS SIP Overview 27
- Packet Voice Connectivity 29
- MPS SIP Server 30
- SIP Operational Modes 32
- MIXED Mode Overview 35
- MPS SIP Implementation Notes 36
- SIP QoS Implementation Notes 37
- SIP/MGCP QOS Configuration 38
- RTP QOS Configuration 39
- SIP/MPS N+1 cluster redundancy feature implementation notes 39
- Network Load Balancing Implementation Notes 40

Chapter 4: Solaris Software Installation and Updates 41
- SIP Server Software Installation 41
- Removal of Previous Version 41
- CCSS Installation for Solaris Systems 42

CCSS Installation for Windows Systems 46
Software Revisions and Patches 51
Chapter 5: Windows Software Installation and Updates 53
SIP Server Software Installation 53
Removal of Previous Version 53
Installation 54
Software Revisions and Patches 54
Chapter 6: Configuration 57
Configuration 57
sip.conf 58
sip.conf Syntax 58
Configuration for MPS/SIP N+1 redundancy in ASE Mode 59
Command Sections 60
CCSS SIP Configuration 79
CCSS SIP configuration for MIXED mode 80
Configuration for SIP Network Load Balancing 82
Round-Robin Algorithm 82
Relative Ratio Algorithm 83
Weightage Algorithm 83
General MPS Configuration Files 84
gen.cfg 84
tms.cfg 85
Chapter 7: Operations, Maintenance, and Debugging 87
System Startup and Shutdown 87
Backup and Restore 89
UNIX cp (Copy) Command 90
The TAR Utility 90
SIP Server Backup and Restore 92
Diagnostics and Debugging 93
DEBUG:root/mps 94
DEBUG:root/mps/cc 101
DEBUG:root/mps/register 103
DEBUG:root/mps/mg 104
procmon 105
Using Snoop to Verify SIP Packets 106
Chapter 8: Application Data Structures 109
Application Data Structures 109
MPS Developer SIP Data Structure 109
SIPconst 111
SIPproto 112
GSipHeader 113
GSipCcData 115
GsSipCcData 115
Media Control Event Data 123
Application API Primitives 125
Global Primitives 126
Call Control Primitives 127

Media Control Primitives ... 130
Additional folders for TMS Integrated Mode ... 134
GSipVarData ... 135
GnHeader ... 135
Chapter 9: Application Programming ... 137
ASE Mode ... 137
Resource Operations ... 137
Condition Data Structure ... 138
SIPEventData ... 138
Sending Messages ... 139
Receiving Messages ... 141
Basic Call Operations ... 142
Media Operations ... 144
Advanced Operations ... 145
Incoming SIP Calls ... 148
Outgoing SIP Calls ... 156
TMS Integrated Mode ... 161
Outgoing SIP Calls ... 161
Incoming SIP Calls ... 165
TMS SIP Feature Message Programming ... 166
Chapter 10: Application Services Registration ... 173
Application Services Registration ... 173
MPS SIP Server Register Function ... 174
Registration ... 176
Multiple Contacts ... 177
Application Pools and Allocation ... 178
Authentication Support ... 179
Authentication Background ... 179
MPS SIP Server Authentication Support ... 180
Registration Configuration ... 180
Registration Configuration ... 180
Application Group Configuration ... 181
Line Configuration ... 181
Command, Control, and Status ... 182
Message Sequence Diagrams ... 183
Registration Success ... 183
Registration Failure ... 183
Deregistration (Cancel Registration) ... 184
Registration Renewal ... 184
Registration Retry for 4xx Responses ... 185
Application Call Flows ... 187
Call Flow Diagrams ... 187
Incoming Call Flow (Single Leg) ... 187
Waiting for CN ... 188
Application Releases ... 189
Application Terminates ... 190
Outbound Call ... 191

Hold and Retrieve ... 191
Prepaid Call ... 192
Voice Bridge ... 194
Re-origination ... 195
Mid-Call Card Status ... 196
Application Receives Refer ... 197
Application Sends Refer ... 198
Inbound Call on ASE Mode using Create Connection Request ... 199
TMS Integrated Mode Message Flows ... 199
Initial Client Connectivity ... 199
Inbound Call Setup ... 201
Outbound Call Setup ... 202
Network Disconnect ... 203
Application Disconnect ... 203
Outgoing Call - No Answer ... 204
Outgoing Call - No Response, Setup Reject ... 205
Outgoing Call - Application Drops Call before Far End Answers ... 206
Incoming Call - No Application Available ... 206
Incoming Call - Rejected by TIM ... 207
Incoming Call - Not Answered ... 207
Incoming Call - Slow Start ... 209
Error Sending CcExt Message to DSSI ... 210
UI Resets ... 210
Inbound Call on TMS Mode using Create Connection Request ... 211
Configuration Information Worksheets ... 213
CCSS SIP ... 213
SIP Network Servers ... 213
SIP URL Registration Information ... 213
TPM VoIP Gateway Module ... 214
TMS/TPM ... 214
MPS Information ... 214
Alarms ... 215
Alarms ... 215
Configuration ... 215
TMS Integrated Mode Diagnostics ... 230
Alarms ... 230
Alarm Logs ... 235
Advanced Configuration ... 237
Advanced Configuration ... 237
sipdefault.properties file configuration ... 243
Configuring .ini File Parameters ... 247
Confirm TPM is in service ... 251
Troubleshooting the TPM card ... 252
Enabling JVM debug options on window machine ... 253
Glossary ... 0
(SIP) transaction ... 0
back-to-back user agent ... 0

call 0
call leg 0
call stateful 0
CCSS 0
CN 0
conference 0
CPS 0
DCC 0
downstream 0
DSSI 0
DTC 0
final response 0
FT 0
initiator, calling party, caller 0
invitation 0
invitee, invited user, called party, callee 0
isomorphic request or response 0
ITU-T 0
location server 0
location service 0
MGCP 0
MPS 0
NAT 0
outbound proxy 0
parallel search 0
provisional response 0
proxy, proxy server 0
redirect server 0
registrar 0
regular transaction 0
ringback 0
SCN 0
SCTP 0
Session Description Protocol (SDP) 0
session 0
Session Initiation Protocol (SIP) 0
SIP INFO 0
SIP MESSAGE 0
spiral 0
SS7 0
stateful proxy 0
stateless proxy 0
TMS 0
TPM 0

upstream 0
URL-encoded 0
user agent (UA) 0
user agent client (UAC) 0
user agent server (UAS) 0
VPS, VPS/is 0
Index 257

Chapter 1: New in this release

The following section details what is new in Avaya Media Processing Server SIP 6.3 Features Manual for Media Processing Server Release 3.5.

Navigation

- Features on page 9
- Other changes on page 10

Features

The following features are new in this document:

CCSS SIP Integration with MPS N+1 redundancy on page 9

CCSS SIP Integration with MPS N+1 redundancy

This feature enables CCSS SIP server to support MPS N+1 AP redundancy in ASE/PROXY mode. This feature affects the following sections:

- SIP/MPS N+1 cluster redundancy feature implementation notes on page 39
- Configuration for MPS/SIP N+1 redundancy in ASE Mode on page 59
- gen.cfg on page 84
- DEBUG:root/mps on page 94
- Configuration on page 215
- sipdefault.properties file configuration on page 243

Other changes

The following are the changes made in the document:

Revision history

June 2009	Standard 01.14. This document is up-issued to add technical content in the following sections, • Section Initiation Protocol • Configuration for MPS/SIP N+1 redundancy in ASE Mode • Configuration • Diagnostics and Debugging • Appendix C Alarms • Configuring .ini File Parameters
June 2009	Standard 02.01. This document is up-issued to support Media Processing Server (MPS) 3.5.
February 2009	Standard 01.13. This is up-issued to add technical content in the section Session Initiation Protocol (SIP) of chapter Introduction.
January 2009	Standard 01.12. This document is up-issued to support Media Processing Server (MPS) 3.0.
April 2008	Standard 01.11. This document is up-issued to add additional alarm information to Appendix C: Alarms.
March 2008	Standard 01.10. This document is up-issued to add gen.cfg information for connecting with the CCSS SIP server.
January 2008	Standard 01.09. This document is up-issued to update the formatting for the ccssinstall script information in the Advanced Configuration appendix.
January 2008	Standard 01.08. This document is up-issued to update the ccssinstall script information in the Advanced Configuration appendix.
November 2007	Standard 01.07. This document is up-issued to correct arguments in the reset, trace, block, and unblock commands.
August 2007	Standard 01.06. This document is up-issued to add a TPM troubleshooting section.

August 2007	Standard 01.05. This document is up-issued to add TPM1100 information.
July 2007	Standard 01.04. This document is up-issued to update SIP information.
June 2007	Standard 01.03. This document is up-issued to support Media Processing Server (MPS) 3.0 Windows GA Release.
August 2006	Standard 01.02. This document is up-issued to support Media Processing Server (MPS) 3.0 Windows GA Release.
January 2006	Standard 01.0. This document is up-issued to support Media Processing Server (MPS) 3.0.

Chapter 2: Preface

Scope

The Avaya Media Processing Server SIP 6.3 Features Manual provides information about the CCSS (Common Channel Signaling Server) and SIP (Session Initiation Protocol) when used with Avaya Media Processing Server (MPS) systems. Step-by-step instructions are provided regarding installing and configuring the various files and features.

Intended Audience

This document is targeted for any user of the CCSS (Common Channel Signaling Server). That is, any person responsible for using CCSS to interact with the Avaya Media Processing Server system will find important information in this manual.

Customer service

Visit the Avaya Web site to access the complete range of services and support that Avaya provides. Go to www.avaya.com or go to one of the pages listed in the following sections.

Navigation

- Getting technical documentation on page 13
- Getting product training on page 14
- Getting help from a distributor or reseller on page 14
- Getting technical support from the Avaya Web site on page 14

Getting technical documentation

To download and print selected technical publications and release notes directly from the Internet, go to www.avaya.com/support.

Getting product training

Ongoing product training is available. For more information or to register, you can access the Web site at www.avaya.com/support. From this Web site, you can locate the Training contacts link on the left-hand navigation pane.

Getting help from a distributor or reseller

If you purchased a service contract for your Avaya product from a distributor or authorized reseller, contact the technical support staff for that distributor or reseller for assistance.

Getting technical support from the Avaya Web site

The easiest and most effective way to get technical support for Avaya products is from the Avaya Technical Support Web site at www.avaya.com/support.

How to Use This Manual

This manual uses many standard terms relating to computer systems, software application functions, and the Internet. However, it contains some terminology that can be explained only in the context of the MPS Series. Refer to the Glossary of Avaya Media Processing Server Series Terminology for definitions of MPS Series specific terms.

Read this manual from start to finish at least once. When you are familiar with the document, you can use the Table of Contents to locate topics of interest for reference and review.

If you are reading this document online, use the cross-reference links (shown in blue) to quickly locate related topics.

Position your cursor over the cross-reference link and click once. Click any point in a Table of Contents entry to move to that topic. Click the page number of any Index entry to access that topic page.

Familiarize yourself with various specialized textual references within the manual, see Conventions Used in This Manual on page 16.

Note:

Periphonics is now part of Avaya. The name Periphonics, and variations thereof, appear in this manual only in reference to a product (for example, the PERImps package, the perirev command, and so on).

Organization of this Manual

- Chapter 1 — Introduction General characteristics of the SIP signaling system. The key features and software components are overviewed.
- Chapter 2 — Solaris Software Installation and Updates Installation procedures for the Solaris system software.
- Chapter 3 — Windows Software Installation and Updates Installation procedures for the Windows system software.
- Chapter 4 — Configuration Setup information for various system configurations.
- Chapter 5 — Operations, Maintenance, and Debugging Startup, shutdown, backup, and debugging of the system.
- Chapter 6 — Application Data Structures Application data structures information for both ASE and TMS modes.
- Chapter 7 — Application Programming Application programming and MPS Developer information for both ASE and TMS modes.
- Chapter 8 — Application Services Registration Application registration information.
- Appendix A — Application Call Flows Various diagrams representing application states and call flows for common application scenarios.
- Appendix B — Configuration Information Worksheets Blank worksheets for recording configuration information.
- Appendix C — Alarms The various CCME, ISUP, MTP2, MTP3, TCU, TCAP, and SCCP alarms that can be generated.
- Appendix D — Advanced Configuration Provides manufacturing procedures for configuring TPM cards.
- Appendix E — Glossary Definitions of technical terms and abbreviations used in this document.

Conventions Used in This Manual

This manual uses different fonts and symbols to differentiate between document elements and types of information. These conventions are summarized in the following table.

Table 1: Conventions Used in This Manual

Notation	Description
Normal text	Normal text font is used for most of the document.
important term	The Italics font introduces new terms, highlights meaningful words or phrases, or distinguishes specific terms from nearby text.
`system command`	This font indicates a system command or its arguments. Enter such keywords exactly as shown (that is, do not fill in your own values).
`command`, `condition` and `alarm`	Command, Condition and Alarm references appear on the screen in magenta text and reference the Command Reference Manual, the MPS Developer User Guide, or the Alarm Reference Manual, respectively. Refer to these documents for detailed information about Commands, Conditions, and Alarms.
`file name / directory`	This font highlights the names of disk directories, files, and extensions for file names. It also shows what is displayed on a text-based screen (for example, to show the contents of a file.)
on-screen field	This font indicates field labels, on-screen menu buttons, and action buttons.
<KEY NAME>	A term that appears within angled brackets denotes a terminal keyboard key, a telephone keypad button, or a system mouse button.
Book Reference	This font indicates the names of other publications referenced within the document.
cross-reference	A cross-reference appears on the screen in blue. Click the cross- reference to access the referenced location. A cross-reference that refers to a section name accesses the first page of that section.
	The Note icon identifies notes, important facts, and other keys to understanding.

Notation	Description
	The Caution icon identifies procedures or events that require special attention. The icon indicates a warning that serious problems may arise if the stated instructions are not followed implicitly.
	The flying Window icon identifies procedures or events that apply to the Windows operating system only. (1)
	The Solaris icon identifies procedures or events that apply to the Solaris operating system only. (2)

(1):Windows and the flying Window logo are either trademarks or registered trademarks of Microsoft Corporation.
(2):Solaris® is a registered trademark of The Open Group in the U.S. and other countries.

Solaris and Windows Conventions

This manual depicts examples (command line syntax, configuration files, and screen shots) in Solaris format. Windows-specific commands, procedures, or screen shots are shown when required. The following table lists general operating system conventions used with either the Solaris or Windows operating system.

	Solaris	Windows
Environment	`$MPSHOME`	`%MPSHOME%`
Paths	`$MPSHOME/common/etc`	`%MPSHOME%\common\etc`
Command	`<command> &`	`start /b <command>`

Two-Button (Windows) versus Three-Button (Solaris) Mouse

<SELECT>	Left button	<SELECT>	Left button

<ADJUST>	Left and Right together	<ADJUST>	Middle button
<MENU>	Right button	<MENU>	Right button





Trademark Conventions

The following trademark information is presented here and applies throughout for third party products discussed within this manual. Trademarking information is not repeated hereafter.

Solaris® and Motif® are registered trademarks of The Open Group in the U.S. and other countries.

Solaris, SunOS, OpenWindows, SPARC, and UltraSPARC are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Microsoft, MSSQL, Windows, Internet Explorer, and the Flying Windows logo are either trademarks or registered trademarks of Microsoft Corporation.

Oracle® is a registered trademark of Oracle Corporation.

Sybase™ and SYBASE™ are trademarks of Sybase, Inc. or its subsidiaries.

Informix® and INFORMIX® are registered trademarks of Informix Corporation or its affiliates.

Java and the Java Coffee Cup Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Product Nomenclature Changes

The following product names changed with the latest Avaya MPS software release. All other references to the former name with respect to environment variables, directory paths, software package names, and so on remain the same. For example, the PeriProducer product is now referred to as the Media Processing Server Developer; however, its package name remains PERIppro.

Former Product Name	New Product Name
IVR Software	Media Processing Server Release x.x

Former Product Name	New Product Name
PeriProducer	Media Processing Server Developer
PeriView	Media Processing Server Manager
PeriStudio	Media Processing Server Studio
PeriReporter	Media Processing Server Reporter
PeriSQL	Media Processing Server RDB
PeriVXML	Media Processing Server VXML Browser
CTI Suite	Communications Control Toolkit (CCT)
Open Signal Computing and Analysis Resource (OSCAR)	Speech Server

Chapter 3: Introduction

This chapter covers:

1. Introduction to SIP
2. Key Features of the CCSS and SIP Systems
3. Integration with the MPS

Common Channel Signaling Server (CCSS)

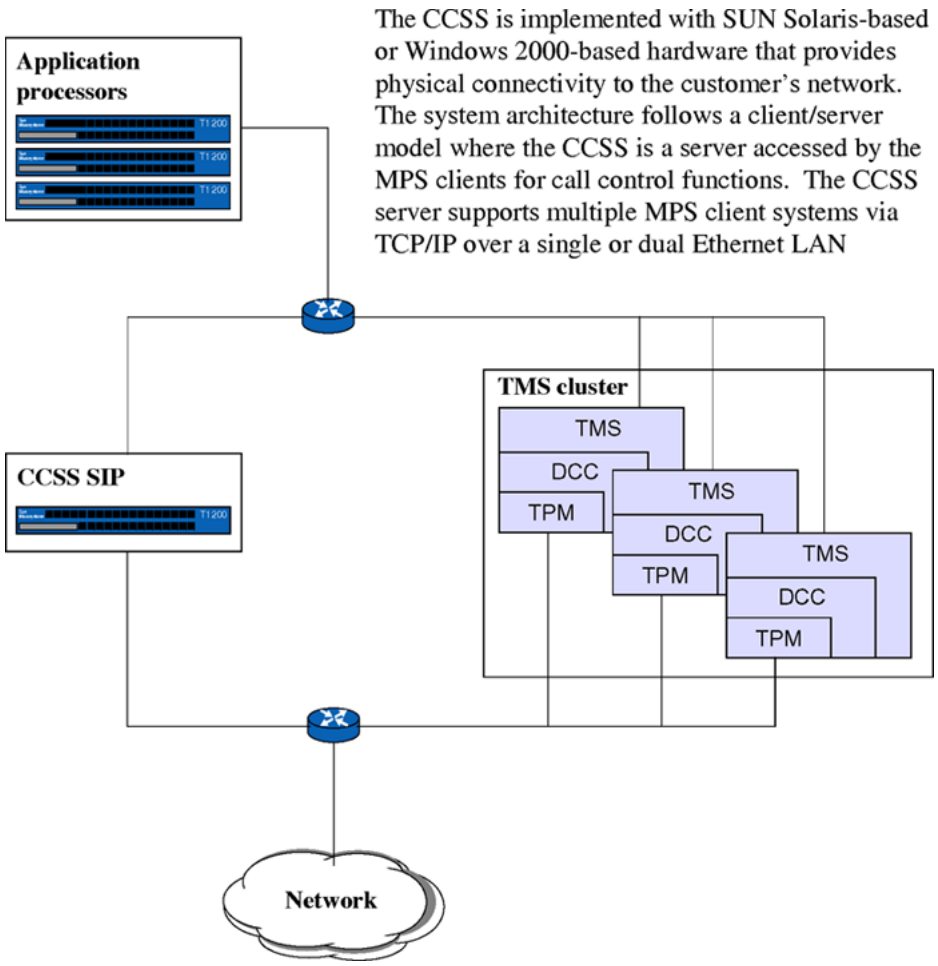
The Common Channel Signaling Server (CCSS) system is an adjunct product to the MPS. The CCSS provides MPS applications with access to several signaling protocols, including SIP and SS7 (ISUP, TCAP, and IUP). This manual documents the use of the CCSS system to provide SIP capabilities. See the publication CCSS 1000/2000 SS7 Feature Documentation for information about the SS7 protocol.

Note:

The SS7 portion of the CCSS product will not be ported to the Windows system. However, the combination of Windows application processor with Solaris CCSS-SS7 are included.

CCSS Architectural Overview

The following is a block diagram of the basic system layout:



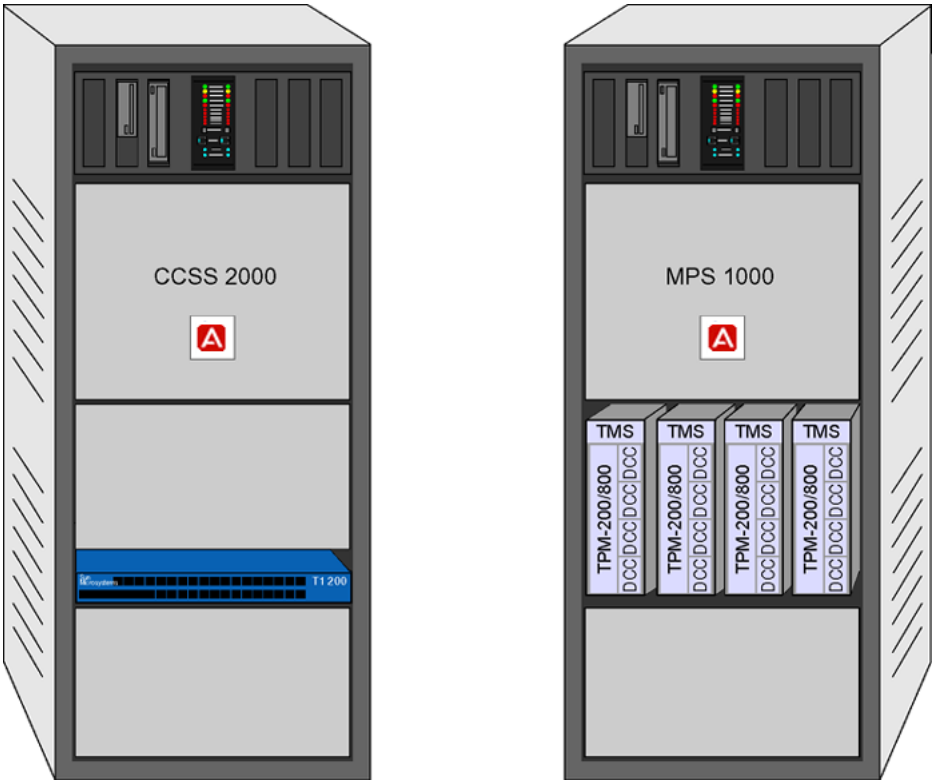
All MPS client applications must include the programming for communicating with CCSS and the signaling network.

The CCSS is housed within the standard Avaya MPS equipment chassis and cabinet. A 48V DC power supply is optionally available for Telco environments. The CCSS is available in two configurations. The CCSS 1000 is an economy system that uses a single Solaris or Windows CPU for both protocol messaging and application functions. The CCSS 2000 has separate processors for protocol interaction and applications.

Note:

SIP functionality is supported only on the CCSS 2000 system. There must be sufficient memory in the system for all the programs to be memory resident (that is, CPU utilization must never reach 100%).

The following diagram shows the basic cabinet design of the CCSS 6.3 SIP system.



The CCSS 2000 has the following standard features:

- Standalone server configuration.
- Dual LAN connectivity to MPS systems.
- Application call control for incoming and outgoing calls.
- Application programmable SIP message parameters.
- Alarm logging.
- Client software compatible with MPS Release 2.1 and higher.
- Example MPS Developer application software.
- UNIX/Solaris or Windows 2000 Operating System Environment.

System Software

The CCSS 2000 server software consists of multiple processes that communicate with each other, the protocol stack, and their clients.

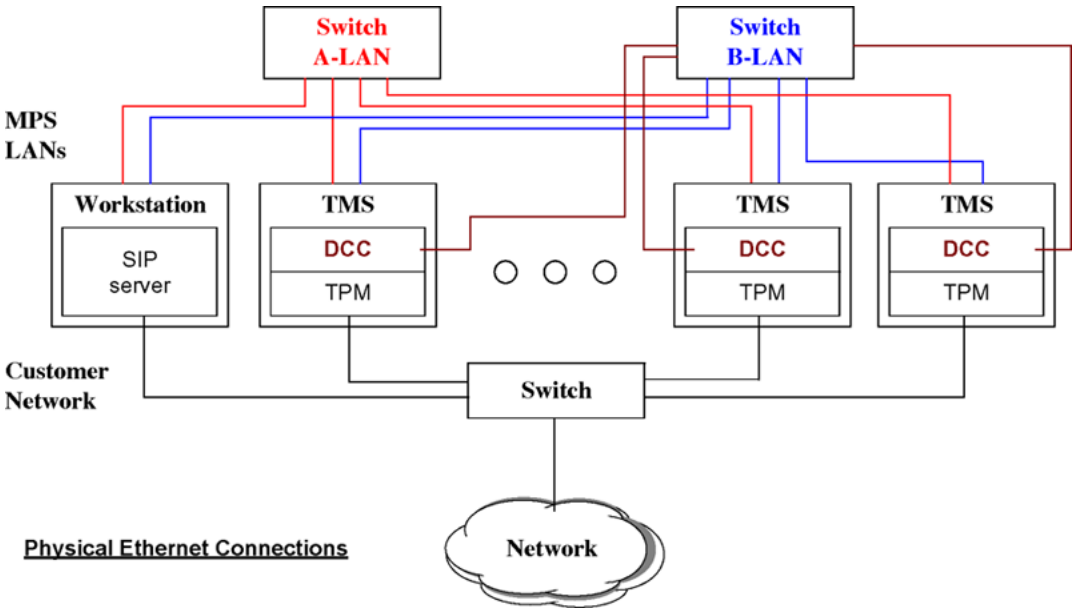
The client-side software provides a conduit for the application to send messages to the server. The MPS clients provides the local resource agent, which is a resource that interfaces with applications and with the CCSS over single or dual Ethernet LAN. It provides broadcast,

message discrimination and keepalive functionality. The following software versions are required on the CCSS 2000 and the MPS systems:

	Required Software
SIP Server	Solaris 2.8 Operating System or Windows 2000 System PERIccss Release: ccss2000 package PERIfw Release: 1 PERIglobl Release: globl2.2.0 PERIperl Release: perl1.0.1
MPS	MPS release 2.1 or higher CCSS release 6.3, with the latest patchbundle.

Ethernet Connections

The Ethernet connections for an MPS SIP network are shown in the following diagram.



Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) on page 0 is a protocol used for initiating and managing communications between users. SIP is a message request-response protocol, in which a message is either a request from a client to a server, or a response from a server to a client. In SIP, the clients endpoints are referred to as user agent (UA) on page 0 s.

SIP uses the Session Description Protocol (SDP) on page 0 for media descriptions to set up the media connections over a packet network. A SIP user agent endpoint can initiate a communication session to another SIP endpoint directly or through a set of intermediate

servers. These intermediate entities can be proxy servers, redirect servers, or back-to-back user agents. Primarily, the intermediate entities are used to locate participants, enforce policy, and perform call routing and billing.

SIP addresses are formatted as URLs, very similar to the common mailto URL. SIP URLs are used to set up sessions in such a way that personal mobility is handled by the use of one identifier to distinguish between users. Personal mobility is the ability of the network to identify users as they change location, allowing them to originate and receive calls at any point on the network. Personal mobility is provided through name mapping and binding of the location. The mechanism to bind a SIP URL to the actual location of a user is the SIP Registration function. A SIP endpoint can send a REGISTER request, associating its SIP URL and contact information, to a server referred to as the registrar on page 0 . This registrar server places the information it receives into the location service for the domain it handles.

In the Avaya CCSS6.3-SIP package, MPS applications act as a SIP user agents. After applications are registered to the network, they can initiate and terminate SIP sessions with other endpoints.

MPS and CCSS 6.3 SIP Key Features

The MPS CCSS6.3-SIP implementation has the following key features:

- Initiate, terminate, and manipulate SIP sessions with or without media connections.
- Fast/slow start support for both inbound and outbound calls.
- Dynamic media port attachment/detachment of an existing session.
- Multiple SIP session support for a single application instance.
- Call hold/retrieve.
- Call transfer using refer/notify or bye-also.
- Voice redirection.
- 3XX redirect support.
- Avaya proprietary webpush, out-of-band digitstring, and digitmap using SIP INFO method.
- Programming interface: standard MPS toolkit (for TMS integrated mode), ASE mode.
- Support for inbound call queuing.
- Programmable SIP headers and MIME body.
- Codec negotiation based on application preferences.
- Configurable IP Quality of Service (QoS) options: IP precedent and IP type of service fields, and IP DiffServe field in the IP headers.
- Automatic registration of one or more URLs with a SIP proxy/registrar server.

- Digest authentication for both REGISTER and INVITE.
- Digest-based message integrity protection.
- Support for SIP Network Load Balancing (NLB).

CCSS 6.3 SIP Message Support

CCSS 6.3-SIP has the ability to send and receive most messages to/from its application service environment. In the following table, "M" indicates that support for the message is mandatory. "O" indicates that the message is optional. Shaded boxes indicate which messages are not supported by CCSS 6.3 release. Some responses like 4xx, 5xx, or 6xx, can be specific to a particular application service. Other times, the SIP protocol interface automates responses.

Message Type	Message	Tx	Rx
Request methods	`INVITE`	M	M
	`Ack`	M	M
	`Bye`	M	M
	`Cancel`	M	M
	`INFO`	O	M
	`REFER`	O	O
	`REGISTER`	O	O
	`SUBSCRIBE`	O	O
	`NOTIFY`	O	O
	`MESSAGE`	O	O
	`PRACK`	O	O
	`OPTIONS`	O	M
	`UPDATE`	O	M

Message Type	Message	Tx	Rx
Responses	`100 Trying`	M	M
	`180 Ringing`	O	M
	`183 Session Progress`	O	M
	`200`	M	M
	`201`	O	O
	`202`	O	O
	`3xx`	O	M
	`4xx`	M	M
	`5xx`	M	M
	`6xx`	M	M

MPS SIP Overview

The MPS with CCSS SIP is designed to interwork with other SIP networks servers such as SIP proxies, soft switches, SIP gateways, and IP-PBXs, and to provide application services with media resource functions. CCSS SIP relies on these network servers to perform SIP URL or telephony-based routing and switching functions for both inbound and outbound calls.

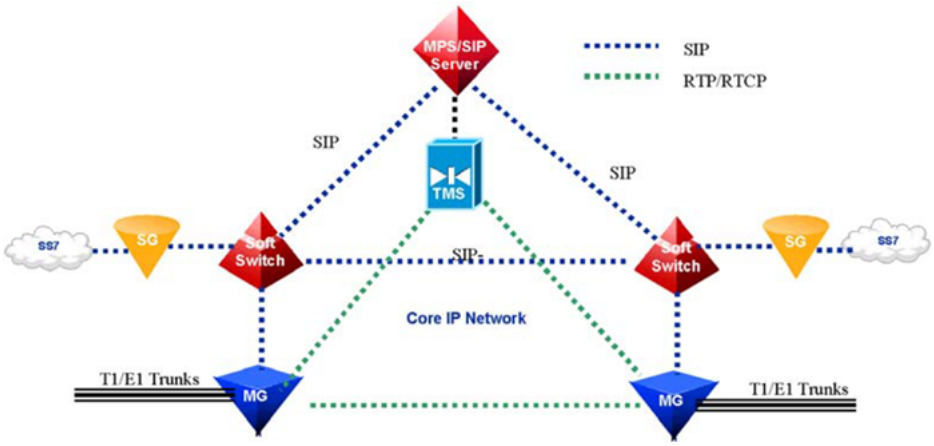
For network servers that support Registrar functions, the CCSS SIP server can send REGISTER requests on behalf of applications to make their presence known to the network. Registration is optional. When registration is not available, the SIP servers are often configured through static provisioning to the network server.

A single CCSS SIP server can support up to 16 network SIP servers in the configuration. The CCSS SIP server can also support direct client access using IP addresses for soft phones that are not managed by any domain server as a configurable option.

In the soft switch environment, the MPS with CCSS SIP is typically viewed as application and media server, or as a feature node. Calls can be routed to the MPS based on their SIP URLs, phone numbers, or as predefined feature service nodes known by the soft switches. The MPS can also originate, bridge, or transfer calls through a set of known soft switches. If SIP

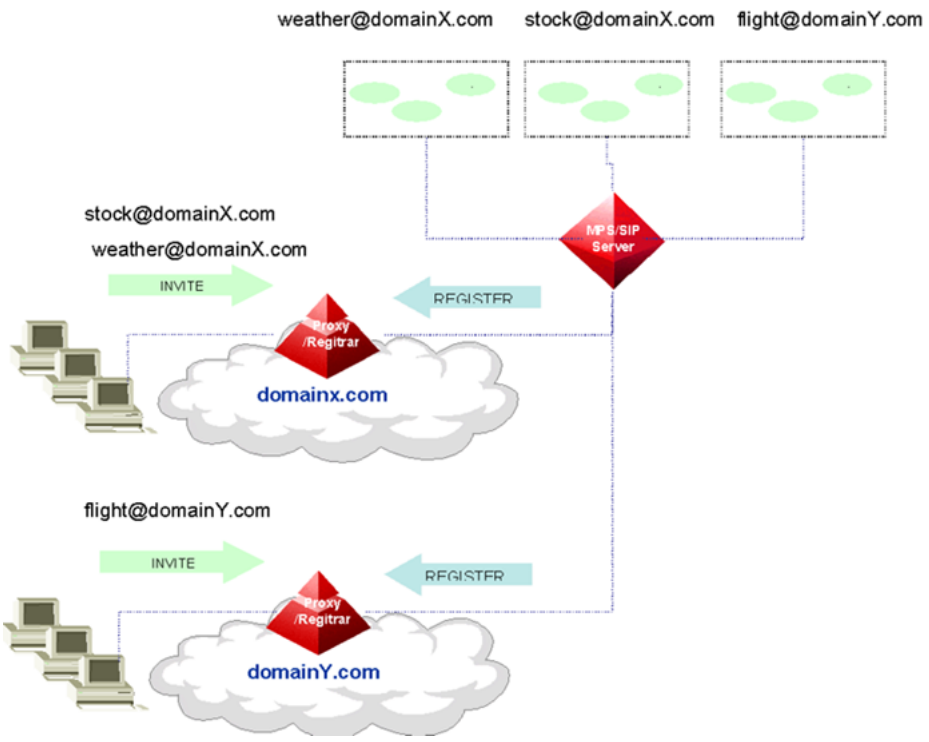
registration is supported, an MPS SIP server can notify its availability by sending REGISTER to the network servers with a SIP URL that identifies itself as a feature node.

The following diagram shows the relationship between MPS/CCSS to the soft switches.



Different types of applications can be hosted on a single MPS/CCSS SIP server. In a multiapplication environment, each application type is associated with its own SIP URL. When registration is available, the CCSS SIP server sends REGISTER to the proxy/registrar on behalf of the applications, once that type of application becomes available. Multiple domains can be supported if the IP addresses are in the same LAN segment.

As an example, an MPS server has three sets of applications: weather, stock, and flight. Each can register with a qualified URL to the corresponding SIP proxy/registrar that manages that domain. A SIP client in the domain can access the services provided by the MPS through the SIP proxy by using these well known URLs.



Packet Voice Connectivity

The MPS SIP software uses packet voice connections through its Telephony Media Server (TMS) and TPM gateway module. The TPM module converts packet voice streams to and from the TMS PCM highways, which can be managed by the MPS to perform media processing (such as playing prompts, collecting caller-entered digits, performing text-to-speech conversions, and so on) or gateway functions to the PSTN trunks. On the packet side, the TPM module performs voice compression, packetization, echo cancellation, and silence suppression. The compressed signal is encapsulated into RTP (Real-Time Protocol) packets sent out through an Ethernet interface to the network.

The MPS with CCSS/SIP supports both the TPM-200 and TPM-800 modules. The TPM-200 has a capacity of 60 voice streams. The TPM-800 has a capacity of 240 voice streams. The default payload size (ptime) is 20ms for all codecs supporting a frame size that is a multiple of 20ms. G.723.1 for example has a default payload size of 30 ms since it has a frame size of 30ms. The CCSS SIP server controls the TPM module through the Media Gateway Control Protocol (MGCP).

The following table highlights the TPM gateway module capabilities.

	TPM-200/800 gateway module
Capacity	TPM-200: 60 media ports. TPM-800: 240 media ports.

TPM-200/800 gateway module		
Voice compression	G.711	PCM (µ/A law) at 64 kbps.
	G.723.1	MP-MLQ at 6.3 kbps, ACELP at 5.3 kbps.
	G.726	ADPCM at 16-40 kbps.
	G.727	E-ADPCM at 16-40 kbps.
	G.728	LD-CELP at 16 kbps.
	G.729A	CS-ACELP at 8 kbps
Silence suppression	G.723.1	Annex A
	G.729	Annex B.
Echo Cancellation	G.168	2000 compliant
VoIP standards compliance	DTMF: mute, transfer in-band or in RTP relay per RFC-2833. RTP/RTCP per RFC 1889/1890.	
Dynamic jitter buffer	±150 msec	

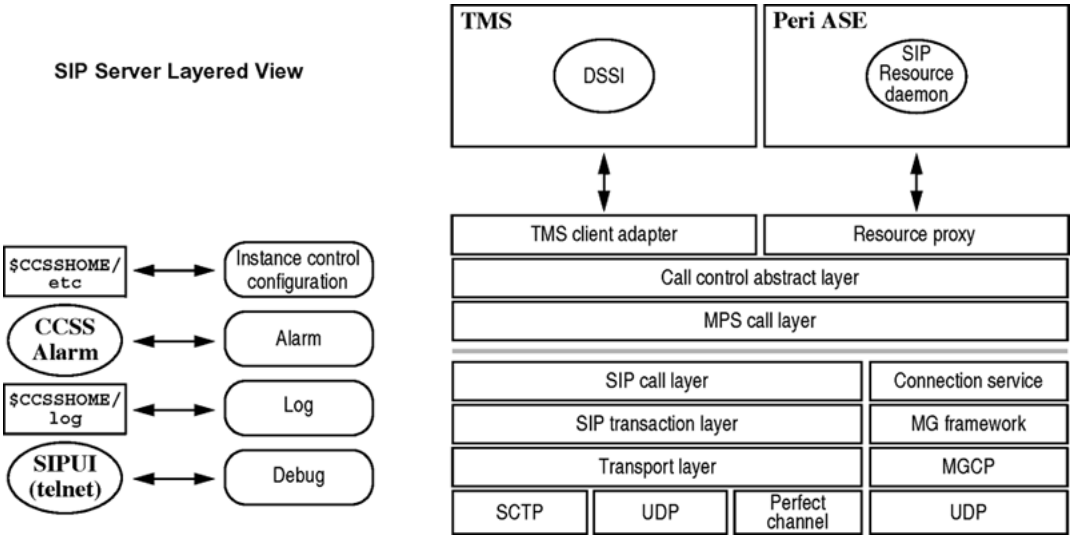
Note:

The following is important information:

- When using speech recognition, to ensure the highest level of quality, it is recommended that the audio be set to an uncompressed G.711 stream.
- For systems equipped with TPM-200 modules, the spanning tree has to be turned off at the switch for the particular port through which the TPM-200 is connected. If the spanning tree is on, the TPM-200 times out and does not load its image file. This restriction does not apply to TPM-800 systems.
- See the MPS Telephony Reference Manual for information about configuring the TPM card for use with MPS systems.

MPS SIP Server

The CCSS SIP server software is implemented in Java. The following diagram illustrates the SIP software in layered view. The call control type adaptation through the abstract layer architecture ensures that different programming models can be adapted to the CCSS-SIP server and provide full system functionality.



Component	Description
Transport layer	Establishes and maintains the connection to the other peer node. A peer node can be a Proxy or a Soft Switch back-to-back user agent (BBUA).
SIP Transaction Layer	Maintains client or server transactions within a call.
SIP call layer	Maintains the state transition of a call and the transactions associated with the call.
MPS call layer	Maintains the state transition of a call and the association of media endpoint states.
Call control abstract layer	Provides a generic interface to various call control interfaces that can be adapted.
Connection service	Triggered by the MPS call layer to create, modify, and delete a connection on a media endpoint.
Media Gateway framework	Maintains the media endpoint connection state, audit, and media gateway failure detection.
Instance controller and configuration module	Reads the configuration and controls the activation/deactivation of each software component in the system.
Alarm and log module	Writes alarm and log entries to files and also relays to the CCSS-ALARM process.
Debug (sipui)	Allows direct user interaction with the CCSS-SIP server through a Telnet port.

SIP Operational Modes

The MPS SIP server implementation is derived from Avaya NetFX software. Software configuration is based on NetFX SIP protocol state machine, MPS User Agent session control, and FPF basic call service combination. Other SIP components such as proxy, redirect, and location database services are external to the MPS.

The CCSS-SIP server provides support for various MPS programming models. The models are selected by enabling one of the four different modes of operation.

Mode	Media Port Binding	Call Control Path	Media Port Status and Control
ASE Mode MPS Developer resource programming paradigm that provides more SIP-specific event control and handling.	Applications can attach and detach a media port to a SIP session during a call. Detached media ports can be reused by other application instances.	Call control signaling data is passed between applications and the CCSS server through the SIP resource daemon. CCSS SIP is accessed using the MPS Developer External Resource block. SIP call control events are received through SIP-specific conditions. Condition data can be examined and manipulated through the condition data folders.	Media ports are maintained in the MPS as pools. Media ports can be acquired and released by applications through the getline and freeline operations.
TMS Integrated Mode Standard MPS unified programming environment. It includes MPS Developer toolkit programming and C-based MediaXact (MX)	Associations from TMS and CCSS configuration create static bindings between SIP sessions and media ports. Application have the ability to detach a media port, but it cannot be reused by other application instances.	Call control signaling data is passed between applications and the CCSS server through the TMS. The call control interface between TMS and the CCSS SIP server is called CCEXT (Call Control Extension). (This is also used by the SS7 protocol.) SIP-specific data can be manipulated	Media ports are allocated when a SIP call is made. The association between a session and the media end point is statically bound throughout the call.

Mode	Media Port Binding	Call Control Path	Media Port Status and Control
		through the use of the MPS Extended Protocol Data toolkit.	

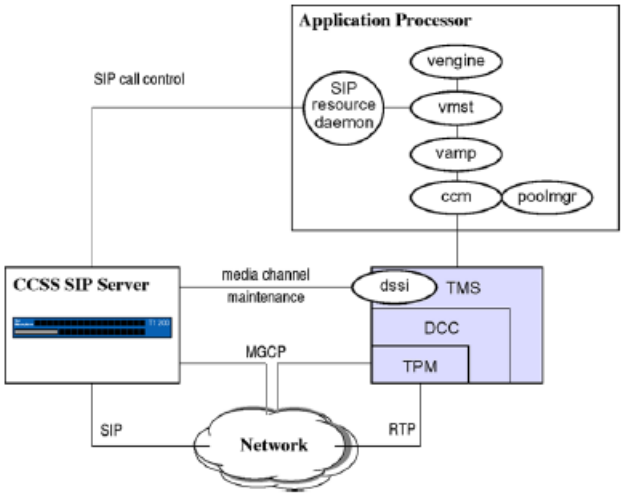
Note:

In both modes, media ports are controlled by the CCSS SIP server through its MGCP on page 0 interface. The CCSS indicates the maintenance state of media ports and the state of the TPM gateway module by communicating with the TMS.

ASE Mode Overview

In the ASE mode of operation, the MPS application processor treats the CCSS SIP signaling server as an external shared resource. The name of the CCSS-SIP call control resource is SIP. An application obtains a SIP resource when it is ready to handle a call. It processes the signaling and frees the resource when the call is terminated.

The SIP resource daemon handles message transmission and conversion between the CCSS SIP server and the applications. The SIP resource daemon executable is installed as part of the PERIase package. The following diagram shows the relationship between components.



In the ASE mode of operation, a media port and the SIP signaling session is loosely coupled. The port can be dynamically attached and detached during a call. For this reason, media ports are normally configured as MPS line pools. An application acquires a port from the MPS pool using the "get Line" MPS Developer block. The get line operation returns the identifier of the media port, which is used by the SIP session media attach operation. When a media port is detached, it can be released using the "free line" operation. This makes the port available for use by other applications.

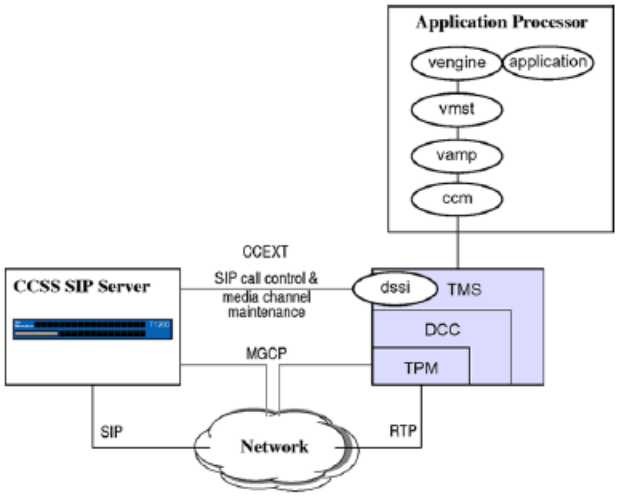
This feature is particularly useful for services like prepaid calling card applications. A prepaid calling application requires media ports only during user interaction. The actual conversation between the prepaid user and the called party (which occupies most of the call) does not require user interaction from the MPS media resources. Using SIP, a prepaid application has the ability to redirect the voice streams between the prepaid user and the called party without

going through the MPS. As a result, the prepaid application is able to handle more calls than the total number of media ports available on a system.

TMS Integrated Mode Overview

In the TMS integrated mode of operation, calls are presented to applications using the unified call model. The unified call model allows the same applications to work with different protocols, with only minor modifications.

In the MPS architecture, all call control signaling protocols are communicated through the TMS. The CCSS-SIP server functions as an external signaling server. It provides integrated SIP signaling to the TMS using an interface called the Call Control Extension (CCEXT). The task in the TMS that handles the conversion between CCEXT and TMS internal signals is called DSSI.



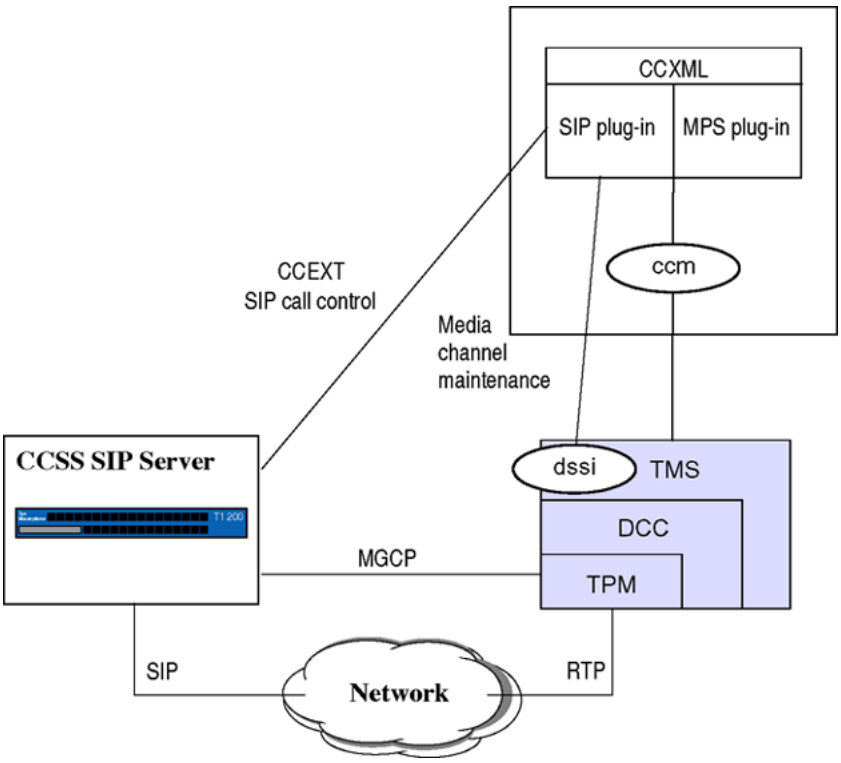
In this mode of operation, the relationship between a media port and a SIP signaling session is tightly coupled. Although a media port can be dynamically detached and reattached after a call is connected, the media port is statically bound to that session. Applications that do not require loosely coupled media ports can benefit from the simplicity of the unified programming model provided by the TMS integrated mode.

CCXML-SIP

Call Control Extensible Markup Language (CCXML) is an XML based language that controls the setup, monitoring, and tear down of telephone calls. CCXML is supported by Avaya MPS systems. For more information about CCXML, refer to the CCXML User Guide.

In the application processor, the CCXML-SIP plug-in is the interface to the SIP platform. The plug-in enables CCXML for the SIP Server and can be used to receive calls or originate calls from the SIP platform.

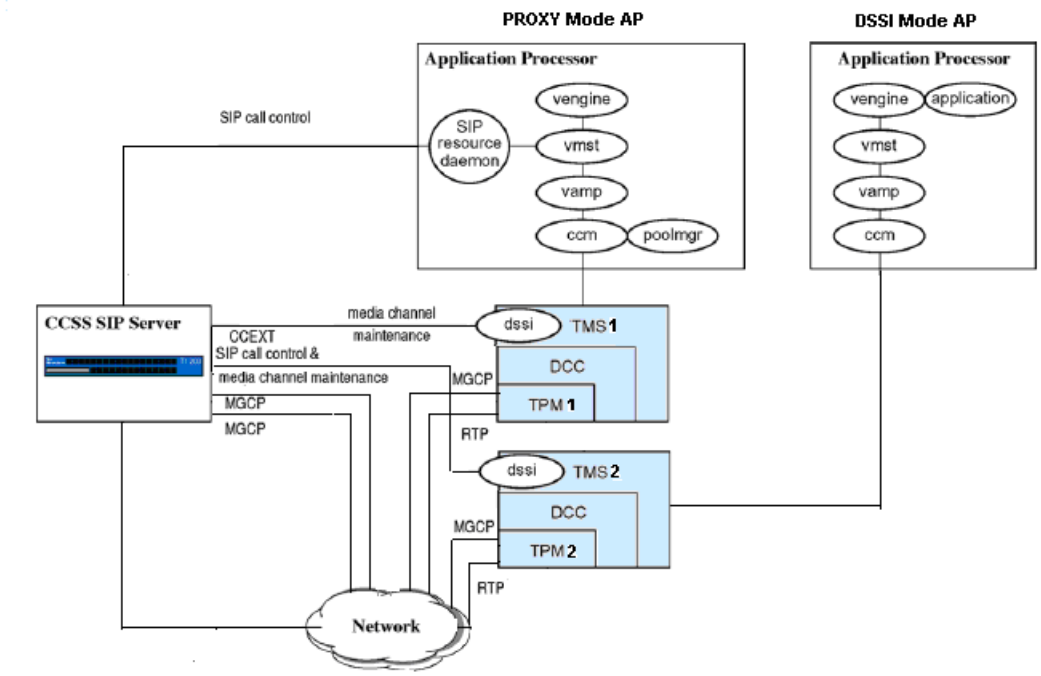
When the SIP Server is configured for CCXML, the SIP server connects to the AP that the CCXML-SIP plug-in resides on using TMS Integrated mode. There is no media or call control signal separation. The following diagram describes the CCXML-SIP configuration.



In the sip.conf file, TMS _HOST_PROFILE is used for CCXML-SIP configuration. For more information, see #unique_46/unique_46_Connect_42_DCAM-3428854.

MIXED Mode Overview

In the MIXED mode of operation, the CCSS SIP Signaling server concurrently is exhibited by serving calls for both ASE and TMS integrated mode of call-control programming. For specific call-control programming in ASE and DSSI mode in details, refer section ASE Mode Overview and TMS Integrated Mode Overview.



MPS SIP Implementation Notes

This section discusses important details of the MPS SIP implementation:

- The SIP protocol is configured using the setup_protocol utility. (See setup_protocol.)
- SIP cannot be used simultaneously with the SS7 protocol on the same CCSS server.
- All SIP server configuration is done in the sip.conf file, which resides in the $CCSSHOME/etc directory in Solaris and the %ccssN\etc directory in Windows This file contains commands for specifying values for timers related to the SIP and MGCP protocols. (See sip.conf on page 58.)
- The SIP server maintains application pools. These pools group classes of applications together, assigns them to a URL (through which users can reach the application group), and associates them with MPS phone lines. When a call arrives from the network, the desired destination is matched with an application from one of the pools and the call is delivered to that application.
- All applications can be assigned to a single pool that receives all calls. A dedicated application can then be used for all call routing.

- Alarms generated during operations are integrated with the CCSS alarm mechanisms. Alarms are stored into the $CCSSHOME/Alarms directory in Solaris and the %common/log/alarm.ccss.N.log directory in Windows.
- The CCSSALARM program forwards all alarms to the ALARMD process and to a MPS Manager workstation if one is available and configured.



The Windows version of CCSS-SIP sends alarms directly to the IPC subsystem.

- The MPS SIP contains a text-based user interface. (A GUI is not available for this release.) The user interface is accessed by connecting a Telnet session to a port specified by the SIP software. The Telnet session can be initiated by invoking the sipui command. For more information, see Diagnostics and Debugging on page 93.
- The SIP server can generate debug output to a log file. This can be enabled or disabled using commands in the text-based user interface. For more information, see Diagnostics and Debugging on page 93.

SIP QoS Implementation Notes

This section discusses important details on the SIP QoS implementation:

- The QoS implementation helps in expediting large volumes of network traffic.
- This QoS is implemented using Type of Service (ToS) bytes in IPv4 header through layer 2. ToS bits within IP headers provide rudimentary signal to network component, so that certain packets are given higher priority over other packets.



Note:

Must Be Zero (MBZ).

Inside IPV4 header the significance of the Bits are as follows:

Table 2: BITS 0-2: Defines Precedence associated with packets

1	1	1	Network Control
1	1	0	Inter Network Control
1	0	1	Critics
1	0	0	Flash Override
0	1	1	Flash

0	1	0	Immediate
0	0	1	Priority
0	0	0	Routine

Table 3: BITS 3-6: Defines TOS Values

1	0	0	0	Minimize Delay
0	1	0	0	Maximize Throughput
0	0	1	0	Maximize Monetary Cost
0	0	0	1	Minimize Monetary Cost
0	0	0	0	Normal Service

BIT 7: Reserved For Future use.

- The desired precedence and the desired ToS values can be assigned to their respective bit positions, and then their decimal ToS value can be found by a simple binary to decimal conversion.
- The ToS value is then assigned to the ip.tos.value in sipdefault.properties Advanced Configuration on page 237
- The ip.tos.value should be within the range 0 – 254. If the ip.tos.value is not in the given range, then the ToS value will be set to the default ToS value (0) in the IPv4 header.
- The same ToS values will apply to the IPv4 header for MGCP protocol also.

SIP/MGCP QOS Configuration

This section discusses details on the SIP/MGCP QOS Configuration:

sipdefault.properties Entry

SIP property entry ip.tos.value is assigned the ToS value between 0 – 254. The Differential Services field in IPv4 header will be set to 0 if the ToS value is not in the above range.

To have the entry with appropriate ToS value, all sites need to update sipdefault.properties file located at:

- $CCSSHOME/etc for Solaris
- %MPSHOME%\ccss1\etc for Windows

For example:

mps.api.header.all=true #mps.proxy.ping=true #mps.proxy.ping.p1=10 ip.tos.value = 112 mg.codec.selection=first mps.sdptransparency.enable=true

Important:
The modification is supported only for the UDP packets sent for SIP(5060) and MGCP(2427) protocols and does not affect the RTP packets.

RTP QOS Configuration

The differential services value can be assigned by ini file parameters in the tp1100.ini file present in tftpboot directory in Solaris and Windows OS.

tftpboot is located at:

- /tftpboot for Solaris
- C:\tftpboot\ for Windows

The value of the Media Path is set by the ini file parameter IPDiffServ. For example: IPDiffServ=21

The range of the Differential Services parameters is set between 0 – 63.

Important:
Changing the value requires the Gateway to be reset.

SIP/MPS N+1 cluster redundancy feature implementation notes

This section discusses details on CCSS SIP integration with MPS N+1 AP cluster redundancy feature implementation.

- This feature enables the CCSS-SIP server to support MPS N+1 AP redundancy architecture in ASE/PROXY mode.
- For SIP configuration, see sip.conf on page 58. To enable CCSS SIP integration with MPS N+1 AP, see sipdefault.properties file configuration on page 243.
- To integrate AP N+1 into CCSS SIP, see General MPS Configuration Files on page 84.
- This feature doesn't protect established calls on failover because there is no CallProtect feature in CCSS SIP ASE mode. Only new calls are allowed to progress once failover is completed.

Network Load Balancing Implementation Notes

This section discusses details of SIP Network Load Balancing (NLB) feature implementation.

- This feature enables the Common Channel Signaling Server (CCSS) SIP to support load balancing across MPS's within a service using different load balancing algorithms detailed in Chapter 4.
- CCSS SIP enables Network Load Balancing statically through "sipdefault.properties" file configuration.
- CCSS SIP enables Network Load Balancing during run time through the sipui utility.

For more information on enabling Network Load Balancing see "mps.sip.nlb" in see sipdefault.properties file configuration on page 243

Chapter 4: Solaris Software Installation and Updates

This chapter covers:

1. Logging In
2. Removal of Previous Software
3. Installation of New Software
4. Updates to System Software

SIP Server Software Installation

The following instructions are for installing the system software on the CCSS 2000 server. First, fill out the information worksheets in the Appendix Configuration Information Worksheets. (See Configuration Information Worksheets on page 213.) Use this information during the installation and configuration.

Removal of Previous Version

If there is a previous version of the server software currently installed, it will have to be removed before installing the new software. To remove the old installation packages:

1. Log in as superuser.

   peri@ccss {1} su Password: ***** #

2. Execute pkgrm to remove the old installation. Respond to all prompts.

Note:

The removal of the PERIccss package does not affect the files present in $CCSSHOME/etc directory.

pkgrm PERIccss

root@tcmps01 {26} pkgrm

```
 The following packages are available:
1 ANCrules Adrian's Rules & Tools
(all) 2.1 (pre-fcs)
2 PERIase PERI ase Distribution
(SPARC) ase5.0.0
3 PERIbrdge PERI Bridge Distribution
(SPARC) brdge2.1.0
4 PERIccss CCSS Server Distribution
(SPARC) ccss6.3-012003
5 PERIcmpat PERI Global1.x Compatibility
(SPARC) cmpat1.1.1
6 PERIdist PERI Distribute
(SPARC) dist1.0.1
7 PERIfw Cross Platform Frameworks
(SPARC) 1
8 PERIglobl PERI Global Module Distribution
(SPARC) globl2.2.0
9 PERIhostp Host Protocols Distribution
(SPARC) hostp2.1.0
10 PERImps MPS Distribution
(SPARC) mps2.1.0
```

... 430 more menu choices to follow; <RETURN> for more choices, <CTRL-D> to stop display:^ D Select package(s) you wish to process (or 'all' to process all packages). (default: all) [?,??,q]: 4

```
The following package is currently installed:
PERIccss CCSS Server Distribution
(SPARC) ccss6.3-012003
```

Do you want to remove this package? y

```
## Removing installed package instance <PERIccss>
This package contains scripts which will be executed with
super-user permission during the process of removing this package.
```

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

```
## Verifying package dependencies.
## Processing package information.
## Executing preremove script.
## Removing pathnames in class <ccss>
## Executing postremove script.
Removing /etc/rc3.d/S50ccss
Removing /etc/ccsshome
## Updating system information
Removal of <PERIccss> was successful.
```

CCSS Installation for Solaris Systems

1. If not already logged in, log in as superuser.

peri@ccss {1} su Password: ***** #

2. Execute pkgadd -d . to install the new version. Respond to all prompts.

```
pkgadd -d .
```

root@tcmps01 {27} pwd /vps/build/ccss6.3/distribution/packages/ccss6.3-011303-ccss root@tcmps01 {28} pkgadd -d .

```
The following packages are available:
1 PERIccss CCSS Server Distribution
(SPARC) ccss6.3-032803
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1
Processing package instance <PERIccss> from </vps/build/ccss6.3/
distribution/packages/ccss6.3-032803-ccss>
CCSS Server Distribution
(SPARC) ccss6.3-032803
Copyright (c) 2003 Avaya Incorporated.
All Rights Reserved.
CONFIDENTIALITY AND LIMITED USE
This software, including any software of third parties embodied
herein, contains information and concepts which are confidential
to Avaya and such third parties. This software is licensed for
use solely in accordance with the terms and conditions of the
applicable license agreement with Avaya or its authorized
distributor.
```

```
Specify the base directory under which to install the CCSS
Distribution
Base directory [/opt/vps]:
=== Checking Java Environment Settings ===
```

```
Java is needed to run the SIP Server
If java is not installed, the Java Runtime Environment 1.4.1
can be installed now
Would you like to install JRE 1.4.1 [y,n]?
```

y

Where do you want to install the Java Runtime Environment? Java Base directory [/opt]: <Return>

WARNING: /opt/vps/packages/PERIccss link exists, do you want to upgrade to ccss6.3-032803 [y,n,?,q] y

```
upgrading to ccss6.3-032803 distribution. Please wait....
Using </opt/vps> as the package base directory.
## Processing package information.
## Processing system information.
4 package pathnames are already properly installed.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.
```

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

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

```
Installing CCSS Server Distribution as <PERIccss>

## Installing part 1 of 1.
/opt/vps/packages/ccss6.3-032803/bin/S50ccss
/opt/vps/packages/ccss6.3-032803/bin/alm_off
/opt/vps/packages/ccss6.3-032803/bin/alm_on
.
.
.
/opt/vps/packages/ccss6.3-032803/samples/vram/vr_inbound.cob
/opt/vps/packages/ccss6.3-032803/samples/vram/vr_outbound.cob
[ verifying class <ccss> ]

## Executing postinstall script.
Creating /etc/ccsshome

Installing Java Runtime Environment 1.4.1

Sun Microsystems, Inc. Binary Code License Agreement  READ THE
TERMS OF THIS AGREEMENT AND ANY PROVIDED  SUPPLEMENTAL LICENSE
TERMS (COLLECTIVELY "AGREEMENT")  CAREFULLY BEFORE OPENING THE
SOFTWARE MEDIA PACKAGE. BY  OPENING THE SOFTWARE MEDIA PACKAGE,
YOU AGREE TO THE TERMS  OF THIS AGREEMENT. IF YOU ARE ACCESSING
THE SOFTWARE  ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF THESE
TERMS BY  SELECTING THE "ACCEPT" BUTTON AT THE END OF THIS
AGREEMENT.  IF YOU DO NOT AGREE TO ALL THESE TERMS, PROMPTLY
RETURN THE  UNUSED SOFTWARE TO YOUR PLACE OF PURCHASE FOR A
REFUND OR,  IF THE SOFTWARE IS ACCESSED ELECTRONICALLY, SELECT
THE  "DECLINE" BUTTON AT THE END OF THIS AGREEMENT.

1. LICENSE TO USE. Sun grants you a non-exclusive and non-
transferable license for the internal use only of the
accompanying software and documentation and any error corrections
provided by Sun (collectively "Software"), by the number of users
and the class of computer hardware for which the corresponding
fee has been paid.

2. RESTRICTIONS. Software is confidential and copyrighted. Title
to Software and all associated intellectual property rights is
retained by Sun and/or its licensors. Except as specifically
authorized in any Supplemental License Terms, you may not make
copies of Software, other than a single copy of Software for
archival purposes. Unless enforcement is prohibited by applicable
law, you may not modify, decompile, or reverse engineer Software.
Licensee acknowledges that Licensed Software is not designed or
intended for use in the design, construction, operation or
maintenance of any nuclear facility. Sun Microsystems, Inc.
disclaims any express or implied warranty of fitness for such
uses. No right, title or interest in or to any trademark, service
mark, logo or trade name of Sun or its licensors is granted under
this Agreement.

3. LIMITED WARRANTY. Sun warrants to you that for a period of
```

```
ninety (90) days from the date of purchase, as evidenced by a
copy of the receipt, the media on which Software is furnished (if
any) will be free of defects in materials and workmanship under
normal use. Except for the foregoing, Software is provided "AS
IS". Your exclusive remedy and Sun's entire liability under this
limited warranty will be at Sun's option to replace Software
media or refund the fee paid for Software.


4. DISCLAIMER OF WARRANTY. UNLESS SPECIFIED IN THIS AGREEMENT,
ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND
WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT ARE
DISCLAIMED, EXCEPT TO THE EXTENT THAT THESE DISCLAIMERS ARE HELD
TO BE LEGALLY INVALID.


Do you agree to the above license terms? [yes or no]
```

yes

```
Unpacking...
Checksumming...
0
0
Extracting...
Archive: ./install.sfx.8172
creating: j2re1.4.1_02/
creating: j2re1.4.1_02/bin/
.
.
.
inflating: j2re1.4.1_02/javaws-1_2_0_02-solaris-sparc-i.zip
Done.


Modifying .cshrc of peri.........done
Creating /etc/rc3.d/S50ccss
Creating /opt/ccss/etc/perirc.csh
```

Please select a protocol from the following: 1. isup-tcap 2. iup 3. btoss 4. sip (Enter protocol as a single number) Protocol:==> 4 The systems is currently configured for ISUP validating current setup OK Setting up SIP Setup of SIP complete Installing MTP2 software into /tftpboot Cannot find perirev or PERIglobl. Using ccssalarm.nois Installation log is at /opt/ccss/etc/ccss.install.log Installation of <PERIccss> was successful. The following packages are available: 1 PERIccss CCSS Server Distribution (SPARC) ccss6.3-032803 Select package(s) you wish to process (or 'all' to process all packages). (default: all) [?,??,q]: q root@tcccss15 {110} perirev
==========
---------- SunOS tcccss15 5.8 Generic_108528-17 sun4u sparc SUNW,UltraSPARC-IIi-Engine ---------- Hostname: tcccss15 Linked packages:

```
MPS Manager Workstation
PERIccss Release: ccss6.3-032803
PERIfw Release: 1
PERIglobl Release: globl2.2.0
PERIperl Release: perl1.0.1
```

```
==============================================
Installed Packages
----------------------------------------------
PERIccss Release: ccss6.3-032803
PERIfw Release: 1
PERIglobl Release: globl2.2.0
PERIperl Release: perl1.0.1
--
```

Change Protocol

To change the protocol use the setup_protocol command.

setup_protocol
root@tcccss12 {1} setup_protocol Please select a protocol from the following: 1. isup-tcap 2. iup 3. btoss 4. SIP (Enter protocol as a single number) Protocol:==> 4 The systems is currently configured for SIP validating current setup OK Setting up SIP Setup of SIP complete

CCSS Monitoring by MPS Manager

The CCSS server can be monitored by MPS Manager. A CCSS icon depicting the state of the CCSS server appears in the MPS Manager GUI monitoring tool running on the MPS server. The CCSS server actively reports its status to the MPS server.

The CCSS icon changes color to depict the state of the CCSS server. When the CCSS server is up and running, the CCSS icon is green. If the CCSS server is shut down or if a fault occurs, the icon changes color to red. If the CCSS server is inaccessible to MPS Manager, the CCSS icon is black.

To enable CCSS server monitoring by MPS Manager, the environment variables CCSSSTARTSCRIPT and CCSSSTOPSCRIPT have to be set as shown.

setenv CCSSSTARTSCRIPT/CCSSSTOPSCRIPT
root@tcccss12 {1} setenv CCSSSTARTSCRIPT $CCSSHOME/bin/ vos_start : root@tcccss12 {1} setenv CCSSSTOPSCRIPT $CCSSHOME/bin/ vos_stop

CCSS Installation for Windows Systems

The CCSS 2000 (SIP Standalone) configuration is installed by Avaya_SelfService_3.5.0 in the interactive mode. The software resides on the Avaya_SelfService CD-ROM, which contains the Avaya_SelfService_3.5.0_[#].[#].msi file and an InteractiveInstall.bat file.

Note:

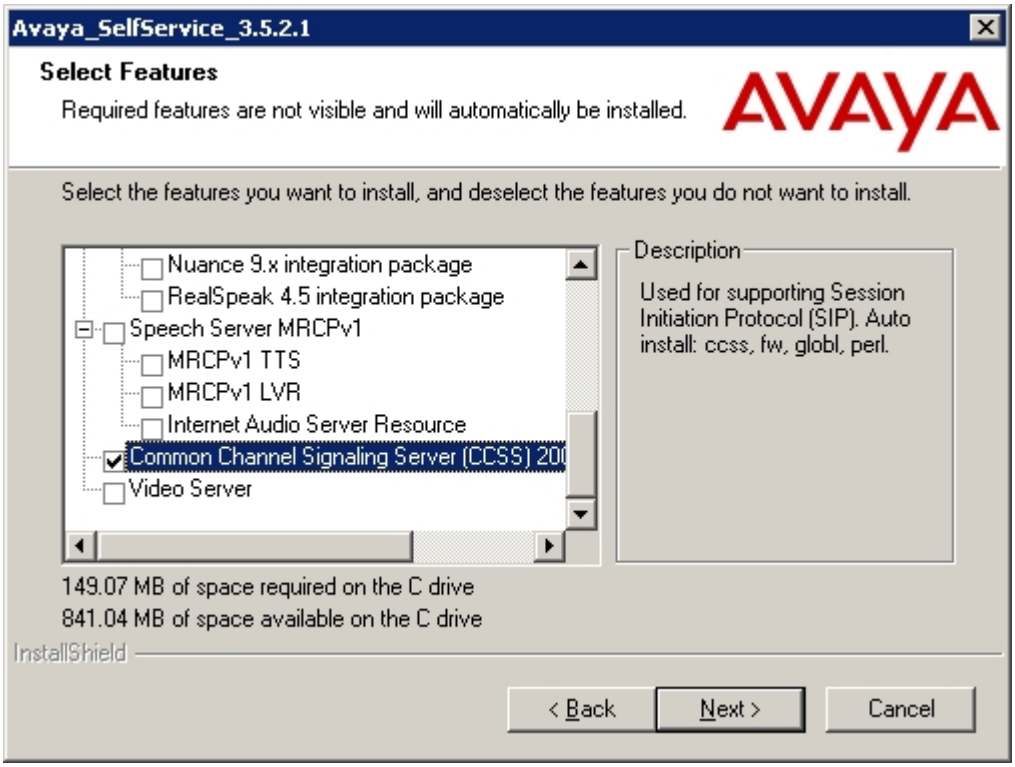
For more information about Avaya_SelfService_3.5.0, refer to Installing MPS Software on the MPS Platform.

Use the following instructions to install a CCSS 2000 (SIP Standalone) configuration on a Windows node in the Interactive mode.

1. Double-click the Avaya_SelfService_3.5.[#].[#].exe file to issue the interactive installation.
2. The Windows Installer window appears and a message about preparing to install the software is displayed.
3. At the License Agreement window, select I accept the terms in the license agreement. Click Next.
4. The selected configuration and any optional packages are installed to the MPSHOME directory. The MPSHOME environment variable is used if it is detected. If the MPSHOME environment variable is not detected, you are prompted to confirm the default location (%program files%\Avaya\SelfService) or specify a new location.

   For information about the optional packages that are associated with a particular configuration, refer to Installing MPS Software on the Windows Platform. You must choose a custom installation to select optional packages.

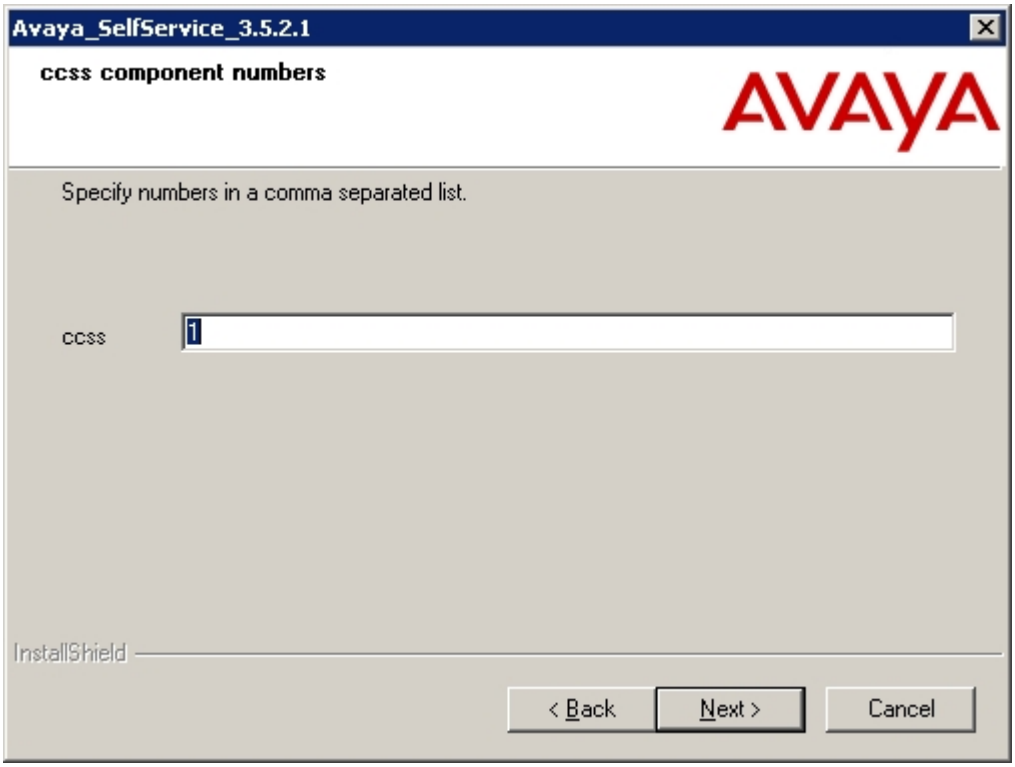
   A message may appear that indicates that you must restart the node if certain environment variable are being defined or if files that need to be updated were in use. If you receive this message, restart the node now.
5. The Configuration window appears and the list of configurations you can install is displayed. Select **Common Channel Signalling Server (CCSS) 2000 (SIP Standalone)**. Click **Next**.



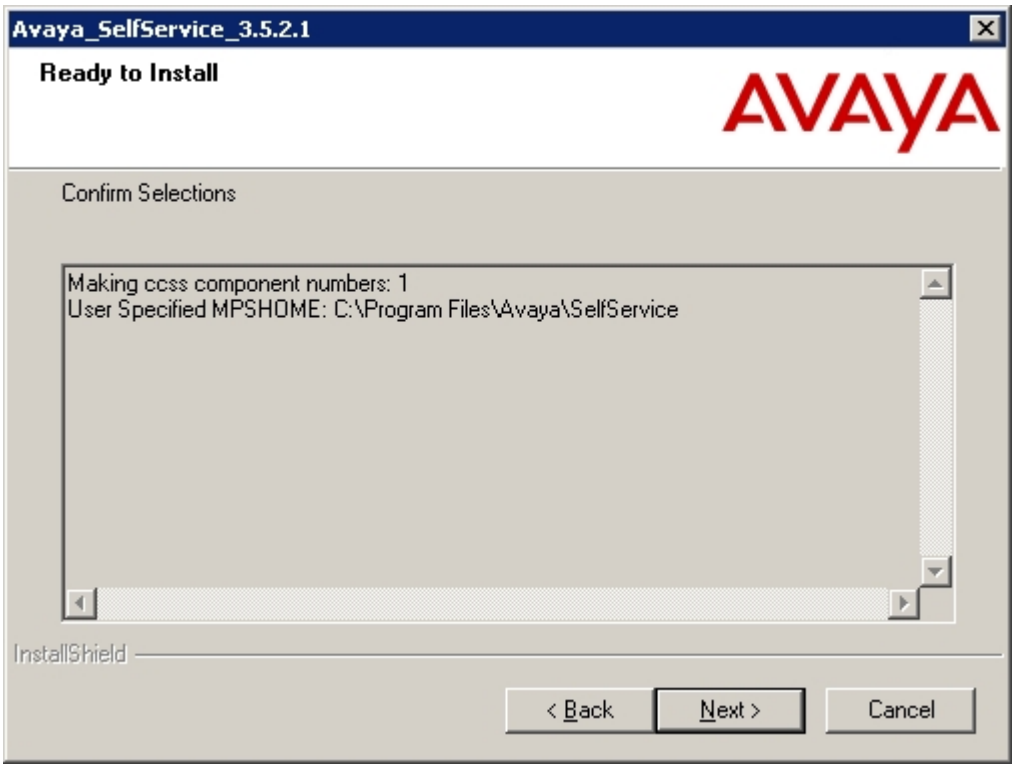
Note:

If you install the CCSS configuration, the directory `C:\tftpboot` is created and contains TPM card images.

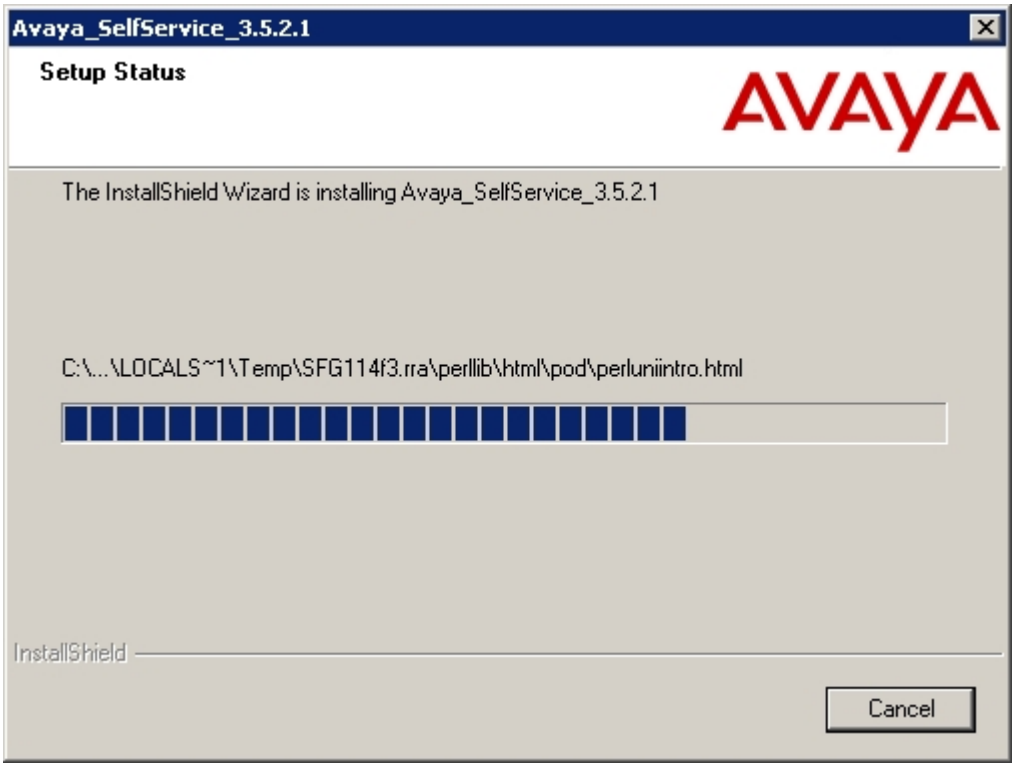
6. Select a component number and click **Next**.



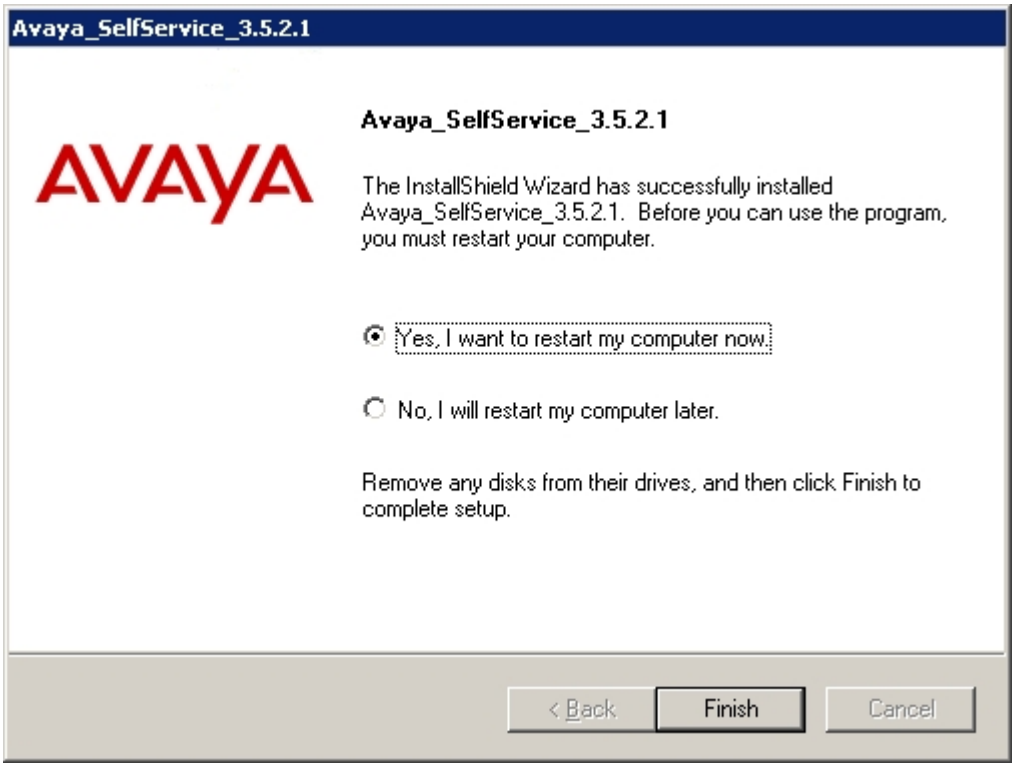
7. At the Ready to Install window, click **Next**.



8. A progress bar displays the progress of the installation.



9. When the installation is complete, select to restart the node and click **Finish**.



Software Revisions and Patches

Major updates to the CCSS and SIP software are provided in the form of a software revision with a complete re-release of all software. Minor updates to the current revision are provided in the form of software patches.

The installation of new and upgraded software should be done only during a designated maintenance window. This should be a time interval that occurs during a period of minimal system activity, when the system can be taken offline. (For many MPS sites, the ideal times are early morning hours, such as between 1:00 AM and 4:00 AM.)

The installation of patches to the system software use the standard MPS mechanisms. Instructions on how to add patches are documented in the publication Installing MPS Software on the Solaris Platform.

Chapter 5: Windows Software Installation and Updates

This chapter covers:

1. Removal of Previous Software
2. Installation of New Software
3. Updates to System Software

SIP Server Software Installation

The following instructions are for installing the system software on the CCSS 2000 server. First, fill out the information worksheets in the Appendix Configuration Information Worksheets. (See Configuration Information Worksheets on page 213.) Use this information during the installation and configuration.

Removal of Previous Version

If there is a previous version of the server software currently installed, it will have to be removed before installing the new software. To remove the old installation packages:

1. Follow the menu path Start > Control Panel > Add or Remove Programs.
2. Click Avaya_SelfService_3.0.
3. Click Remove.

When the previous version is removed, the system will automatically reboot.

Note:

The removal of the PERIccss package does not affect the files present in %ccssN\etc directory.

Installation

Set the MPSHOME variable to the c:\program files\Avaya directory

1. Insert the installation CD-ROM.
2. Click InteractiveInstall.bat.
3. Select CCSS for config type.
4. Select Typical installation.
5. Enter the component number.
6. Click Install.
7. When the installation is complete, click Finish.

The following files are installed:

- PERIperl
- PERIglobl
- PERIfw
- PERIccss

In the root directory of the installation CD-ROM, there are additional software patches that are required. Each software patch has a unique install.exe file associated with it that will need to be run in order to be installed.

CCSS Monitoring by MPS Manager

The CCSS server can be monitored by MPS Manager. A CCSS icon depicting the state of the CCSS server appears in the MPS Manager GUI monitoring tool running on the MPS server. The CCSS server actively reports its status to the MPS server.

The CCSS icon changes color to depict the state of the CCSS server. When the CCSS server is up and running, the CCSS icon is green. If the CCSS server is shut down or if a fault occurs, the icon changes color to red. If the CCSS server is inaccessible to MPS Manager, the CCSS icon is black.

Software Revisions and Patches

Major updates to the CCSS and SIP software are provided in the form of a software revision with a complete re-release of all software. Minor updates to the current revision are provided in the form of software patches.

The installation of new and upgraded software should be done only during a designated maintenance window. This should be a time interval that occurs during a period of minimal

system activity, when the system can be taken offline. (For many MPS sites, the ideal times are early morning hours, such as between 1 AM and 4 AM.)

The installation of patches to the system software use the standard MPS mechanisms. Instructions on how to add patches are documented in the publication Installing MPS Software on the Solaris Platform.

Chapter 6: Configuration

This chapter covers:

1. Configuration Files
2. sip.conf Configuration Parameters
3. tms.cfg
4. gen.cfg

Configuration

Beyond standard MPS configuration procedures, SIP configuration requires having appropriate command and parameter entries in the sip.conf (see sip.conf on page 58) and tms.cfg (see tms.cfg on page 85) files.

General MPS configuration files that require entries for SIP and the TPM-200/800 audio card include the following:

- gen.cfg
- vos.cfg
- voip.cfg
- etc/hosts

Refer to the MPS Telephony Reference Manual for documentation about these files.

Note:

SIP configuration also requires the file sipdefault.properties (See Advanced Configuration on page 237). It contains internal configuration parameters that set global properties for debugging and task-oriented operations on a system-wide basis. This file should only be modified by Avaya personnel.

sip.conf

The sip.conf file is stored in the $CCSSHOME/etc directory in Solaris and the %ccssN\etc in Windows. It contains all available configuration parameters for adjusting how applications use the protocol.

sip.conf Syntax

The sip.conf file stores text-based configuration entries formatted as key=value pairs. A key is a string that identifies a value by name. Keys are associated with values by the equal sign (=). For example, KEY = value.

A value is identified by a key. A key can be of type string, integer, boolean, or array of strings. Values can contain the equal sign (=), but the first equal sign (=) encountered in an entry is treated as the key=value separator.

Many of the value specifications in the sip.conf file are of type array of strings. String arrays require special formatting. The entire set of string value elements is enclosed within curly braces. The opening brace ("{") follows immediately after the equal sign (=). The set of values are then specified. Each value must be on its own logical line. A logical line that is terminated with a carriage return character. In some text editors, lines can wrap prematurely. A line that is terminated with a wrap instead of a return is continued to the next physical line. Each logical line after the opening brace is treated as an array string element until a line with a closing curly brace is found.

In addition to key=value pairs, the files can contain comments. A comment is any line that begins with an octothorp (#) or exclamation point (!).

Table 4: Configuration File Syntax Examples

comments	``` ############################ # # These are comments. Each line # begins with an # octothorp. # ############################ ! ! This comment uses the alternative syntax. ! ```
simple key value pairs	``` aBooleanKey = true anIntegerKey = 356 aStringKey = string value ```

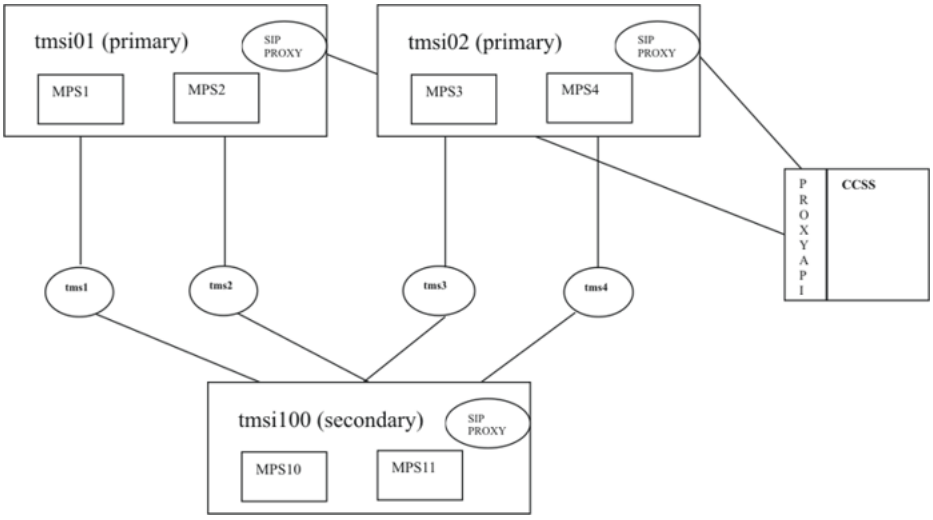
array key value pairs	anArray = {
	value 1
	value 2
	value 3
	}
	Carriage returns must exist in these locations following the last character in the value.

Configuration for MPS/SIP N+1 redundancy in ASE Mode

In an N+1 redundant MPS system, CCSS SIP PROXY (ASE) call control identifies active VOS component and route the Application messages received from and to CCSS. For MPS N+1 redundancy, a secondary node serves as a backup for multiple primary nodes. Once a failover occurred, the secondary node loads the VOS, GEN, and ASE process configurations of the failed primary node to become Active.

When N+1 redundancy is configured on an MPS system, the secondary and primary nodes share a single TMS or multiple TMSs. The MPS component attempts to resume the resources that it provides and applications that it runs. The CCSS must be configured to identify primary and secondary nodes and the line numbers on each VOS component associated with the MPS components on Primary MPS Nodes.

The group of primary components and their backup components are referred as a cluster. The following diagram identifies a basic MPS N+1 redundancy system configuration that is integrated with a CCSS SIP ASE mode configuration. MPS N+1 redundancy is designed to protect against MPS node function failures. MPS SIP N+1 redundancy does not protect against TMS hardware failures.



Command Sections

The sip.conf configuration file consists of six sections:

- default section
- system description
- SIP network configuration
- call control
- registration control
- media gateway configuration

Except for the Default Section, a section consists of a section name followed by a set of related key=value pairs. A section name is a string defined within square braces (for example, [SYSTEM]). All key=value pairs in the sip.conf file must be organized into sections.

The Default Section (see #DCAM-3428847/DCAM-3428848) is the only section that is not explicitly identified with a section name.

Default Section

In SIP release 6.3, the default section points to the location of the sipdefault.properties sipdefault.properties file configuration on page 243 file and specifies the version of the SIP software.

default	See The configuration of the default section.
Specifies global SIP configuration properties. Parameter	
`sip.default.property.file=sipdefault.properties`	
Specifies the location of the file for storing global properties.	
`sip.version=1.0`	
Specifies the version number of the MPS SIP software. MPS SIP server codec list	

`codec.list.1=0`
`codec.list.2=8`
`codec.list.3=18`	

Specifies the codecs supported.	
`codec.list.1=0`	

identifies G.711 uLaw codec

```
codec.list.2=8
```

identifies G.711 Alaw codec

```
codec.list.3=18
```

identifies G.729 codec

Optional Parameters for Pinging Hosts

```
mps.proxy.ping=false
mps.proxy.ping.p1=5
mps.proxy.ping.p2=10
```

Note:

Only add these parameters to the default section of the sip.conf file if values different from these are desired.

These parameters enable pinging and ping intervals of all the hosts configured in the NETWORK_HOSTS section of the sip.conf file.

When mps.proxy.ping is set to true, it enables the sip server to send "OPTIONS" periodically every P1 seconds, to all configured sip proxy/registrar in NETWORK_HOSTS within sip.conf. When there is no reply for two consecutive "OPTIONS" from a proxy/registrar, it is considered to be unreachable.

A MAJOR alarm occurs the first time an unreachable situation occurs. Pinging continues every P2 seconds to that proxy and no alarm is thrown for later failures once it is found unreachable. A MAJOR alarm is thrown when the proxy is available again. If there is any successful SIP registration with that proxy, they will have to be reregistered.

System Description Section

The system section of the sip.conf file is identified by the string [SYSTEM].

[SYSTEM]	See SYSTEM.

Identifies the system section.

```
[SYSTEM]
SYSTEM = {...}
```

SYSTEM	See SYSTEM.

Specifies system parameters.

```
SYSTEM =    name / sip server host name / redund enable / redund
            primary / ft sig port / peer ftsig host name }
```

Parameter	Type	Description
`name`	string	Name of the SIP server.

`sip server host name`	string	SIP server host name. It can also be specified as an IP address string in #.#.#.# format.
`redund enable`	false	Must be specified as false. (Intended for future system enhancement.)
`redund primary`	-	Must be specified as a dash (-). Intended for future system enhancement.
`ft sig port`	-	Must be specified a dash (-). Intended for future system enhancement.
`peer ftserver host name`	-	Must be specified a dash (-). Intended for future system enhancement.

SIP Network Configuration Section

The network section of the sip.conf file is identified by the string [NETWORK].

[NETWORK]	See NETWORK.

Identifies the network section.

```
[NETWORK]
CLIENT_ACCESS     = {...}
NETWORK_HOSTS     = {...}
SIP_TIMER_PROFILE = {...}
```

CLIENT_ACCESS	See CLIENT_ACCESS.

Specifies the client access mode, without using the soft switch.

```
CLIENT_ACCESS = { name / type / local port / timer profile }
```

Parameter	Type	Description
`name`	string	Name of the client.
`type`	string	Specify UDP for all configurations.
`local port`	int	UDP port number.
`timer profile`	string	Name of this timer profile.

Note:
The value of parameter "name" should always be CLIENT.

NETWORK_HOSTS	See NETWORK_HOSTS = {.

Specifies characteristics of the remote host mode (using the soft switch).

`NETWORK_HOSTS =`	`name / type / sip domain name / primary host name / secondary host name / port / ssa port / timer profile / registrar flag }`

Parameter	Type	Description
`name`	string	Name of the network host.
`sip domain name`	string	SIP domain name of this node.
`primary host name`	string	For UDP, this is the IP identifier of this node (soft switch). For perfect channel, this is the IP identifier of the primary host. This can be specified as a host name or as an IP address string in #.#.#.# format.
`secondary host name`	string	For UDP, this must be specified as a dash (-). For perfect channel, this is the peer's IP identifier. It can be specified as a host name or as an IP address string in #.#.#.# format.
`port`	int	Signaling port for UDP or Perfect Channel.
`ssa port`	int	Softswitch advertisement port. Heartbeat monitoring is performed using this port.
`timer profile`	string	Name of the timer profile to use.
`registrar flag`	string	Specify true if this is a registrar, and false in all other cases.

SIP_TIMER_PROFILE	See SIP_TIMER_PROFILE.

Specifies the transaction layer timer profile value range.

`SIP_TIMER_PROFILE =`	`name / inv2prv / inv2fin / inf2fin / fin2ack / prv2prk / prk2fin / bye2fin / t1 / t2 }`

Parameter	Type	Description

`name`	string	Name of this timer profile.
`inv2prv`	int	Timer value (in milliseconds) for the receipt of a Provisional response to an Invite message.
`inv2fin`	int	Timer value (in milliseconds) for the receipt of a Final response to an Invite message.
`inf2fin`	int	Timer value (in milliseconds) for the receipt of a Final response to an Info message.
`fin2ack`	int	Timer value (in milliseconds) for the receipt of an Ack to a Final response.
`prv2prk`	int	Timer value (in milliseconds) for the receipt of a Prack response to a Provisional message.
`prk2fin`	int	Timer value (in milliseconds) for the receipt of a Final response to a Prack message.
`bye2fin`	int	Timer value (in milliseconds) for the receipt of a Final response to a Bye message.
`t1`	int	Time delay (in milliseconds) between sending retransmissions over UDP. It has an exponential backoff to T2.
`t2`	int	The cap on delays (in milliseconds) between sending retransmissions over UDP.

Media Gateway Configuration Section

The media gateway section of the sip.conf file is identified by the string [MEDIA_GATEWAY].

[MEDIA_GATEWAY]	See MEDIA_GATEWAY.

Identifies the media gateway section.

```
[MEDIA_GATEWAY]
MG_LIST          = {...}
MG_PROFILE       = {...}
MG_TIMER_PROFILE = {...}
```

MG_LIST	See MG_LIST.

Specifies parameters for the media gateway used as the MPS SIP server.

`MG_LIST =`	`name / domain / host name / mg port / mg type / chassis ID / slot ID / default codec /  tms_host profile name}`

Parameter	Type	Description
`name`	string	Name of this MG_LIST entry.

`domain`	string	Domain name of the media gateway.
`host name`	string	Media card's host name. It can be specified as a host name or as an IP address string in #.#.#.# format.
`mg port`	int	Gateway transport port number. This should be specified as 2427.
`mg type`	string	MG_PROFILE type for this entry. (This is defined in MG_PROFILE.) The choices are tpm200 and tpm800.
`chassis ID`	int	Chassis number.
`slot ID`	int	Slot number.
`default codec`	string	Default payload Codec. It is one of the following: G711, G723, G726, or G729.
`tms_host profile name`	string	Specifies the profile name of the TMS host that controls the TPM card for this entry. This profile name must also be specified in TMS_HOST_PROFILE.

MG_PROFILE	See MG_PROFILE.

Specifies the media card profile.

`MG_PROFILE =` `{name / ep prefix / num of ep group / num ports per group / start}`

Parameter	Type	Description
`name`	string	Name of this MG_PROFILE entry.
`ep prefix`	string	Prefix for endpoint.
`num of ep group`	int	Number of endpoint groups associated with this type of media card.
`num ports per group`	int	Number of endpoints associated with each endpoint group.
`start`	int	Start number used to name the endpoint.

MG_TIMER_PROFILE	See MG_TIMER_PROFILE.

Specifies the timer profile to be used for MGCP.

`MG_TIMER_PROFILE =`	`name / crcx / mdcx / dlcx / rqnt / ntfy / auep/ beat }`

Parameter	Type	Description
`name`	string	Name of this timer profile.
`crcx`	int	Time out value (in seconds) for create connection.
`mdcx`	int	Time out value (in seconds) for modify connection.
`dlcx`	int	Time out value (in seconds) for delete connection.
`rqnt`	int	Time out value (in seconds) for request notify.
`ntfy`	int	Time out value (in seconds) for notify.
`auep`	int	Time out value (in seconds) for audit endpoint.
`beat`	int	Time out value (in seconds) auepbeat.

Call Control Section

The call control section of the sip.conf file is identified by the string [CALL_CONTROL].

[CALL_CONTROL]	See CALL_CONTROL.

Identifies the call control section.

```
[CALL_CONTROL]
ASE_HOST_PROFILE   = {...}
TMS_HOST_PROFILE   = {...}
CC_HOSTS           = {...}
CC_LINEMAP         = {...}
```

ASE_HOST_PROFILE TMS_HOST_PROFILE	See ASE_HOST_PROFILE. See TMS_HOST_PROFILE.

Specifies ASE or TMS host profile information. If there is no host available for the type (ASE or TMS), that profile must be specified with the parameters commented out.

```
ASE_HOST_PROFILE   = {profile name / primary host / secondary host / port }
TMS_HOST_PROFILE   = {profile name / primary host / secondary host / port }
```

Parameter	Type	Description

`profile name`	string	Name of this profile.
`primary host`	string	ASE: Primary host name or IP address of this (VAS) host. TMS: Primary host name or IP address of this (TMS) host.
`secondary host`	string	ASE: Secondary host name or IP address of this (VAS) host. TMS: Secondary host name or IP address of this (TMS) host.
`port`	int	Port number for ASE or TMS connection.

CC_LINEMAP	See CC_LINEMAP.

Call control line map. The applications on the specified lines are placed into the ready state.

```
CC_LINEMAP = {host id / comp id / line range }
```

Parameter	Type	Description
`host id`	string	Host ID.
`comp id`	string	TMS component ID.
`line range`	int:int	Starting line number to ending line number.

CC_HOSTS	See CC_HOSTS.

Specifies the call control host for the MPS SIP server.

```
CC_HOSTS =      host id / host profile name }
```

Parameter	Type	Description
`host id`	int	Host ID.
`host profile name`	string	Host profile name defined in one of these commands: ASE_HOST_PROFILE or TMS_HOST_PROFILE.

Registration Control Section

The registration control section of the sip.conf file is identified by the string [REGISTRATION_CONTROL].

[REGISTRATION_CONTROL]	See REGISTRATION_CONTROL.

Identifies the registration control section.

```
[REGISTRATION_CONTROL]
REGISTRATION_SERVICE = {...}
CONTACT_PROFILE      = {...}
REG_TIMER_PROFILE    = {...}
APPSVC_GROUP         = {...}
REG_LINE_CONFIG      = {...}
```

REGISTRATION_SERVICE	See REGISTRATION_SERVICE.

Specifies the registration service for MPS SIP server.

```
REGISTRATION_SERVICE =          {service name / domain name / expires /
contact profile / keep registered }
```

Parameter	Type	Description
`service name`	string	Name of service provided by the application. The last entry is always for the default, which is specified as default.
`domain name`	string	Domain of the location service for which the registration is intended.
`expires`	int	Registration expiry value (in seconds) for contacts that do not have the EXPIRES parameter. The value of this field is specified as an integer decimal number between 0 and (2^{31})-1. If neither mechanism for expressing a suggested expiration time is present in a REGISTER entry, a default suggestion of one hour is used.
`contact profile`	string	Name of the contact profile for this service. If there is more than one, use commas to separate the profiles (for example, CONTACT_1,CONTACT_2).
`keep registered`	bool	Specifies whether or not registration is to be always maintained for this service. Specify "Y" to maintain registration. Specify "N" to for short-lived registration.

CONTACT_PROFILE	See CONTACT_PROFILE.

Information about this contact profile used in the contact header fields of the request.

```
CONTACT_PROFILE =          profile name / user name / transport / port /
expires}
```

Parameter	Type	Description

`profile name`	string	Name of this profile.
`user name`	string	Name of the service.
`transport`	string	Transportation type. This is specified as UDP for all configurations.
`port`	int	Contact port number.
`expires`	int	The number of seconds that the UA expects the binding to be valid. If this parameter is not provided, the value of the EXPIRES header field is used.

REG_TIMER_PROFILE	See REG_TIMER_PROFILE.

Information about this registrar profile.

```
REG_TIMER_PROFILE = { no app / retry }
```

Parameter	Type	Description
`noapp`	int	After REGISTER is sent out, if no applications are available for the specified time (in seconds), send DEREGISTER.
`retry`	int	Retry time period (in second), if REGISTER with all registrars fails.

APPSVC_GROUP	See APPSVC_GROUP.

Specifies information about application services groups.

```
appsvc_group = { group name / service list }
```

Parameter	Type	Description
`group name`	string	Name of this group.
`service list`	string	List of services supported by this group.

REG_LINE_CONFIG	See REG_LINE_CONFIG.

Specifies information about line configuration for each service.

`REG_LINE_CONFIG =`	`service group name / host ID / comp ID / line range }`

Parameter	Type	Description

service group name	string	Name of this group.
host ID	int	Host ID. This must be the same host id as used in the command. (See CC_HOSTS.)
comp ID	int	Component ID number.
line range	string	Lines configured for this service. This is specified as a comma-separated list, which may contain range indicators (for example, 1,2,5-10 is the same as specifying 1,2,5,6,7,8,9,10).

Configuration for CCXML-SIP Plug-in

The MG_LIST specifies the media gateways used by the SIP server. MG_LIST has a TMS HOST NAME field that indicates the TMS host that controls the TPM. This field is used to indicate the CCXML profile name as shown the following example.

Sample MG_LIST

```
MG_LIST = {
!            ------------  ----------------------  ----------------------  ----------  ----------
                                                                            --          --
!            NAME          DOMAIN                  HOST                    MG          MG
                                                    NAME                    PORT        TYPE
!            ------------  ----------------------  ----------------------  ----------  ----------
                                                                            --          --
             tpm1-1        us.avaya.com            47.185.16.242           2427        tpm800
!            ------------  ----------------------  ----------------------  ----------
                                                                            --
!!!          CHASSIS       SLOT                    DEFAULT                 TMS
             ID            ID                      CODEC                   HOST
                                                                           NAME
!            ------------  ----------------------  ----------------------  ----------
                                                                            --
             1             1                       G711                    G711
```

Enter the IP address and port number in the TMS_HOST_PROFILE section for the CCXML-SIP plug-in.

Sample TMS_HOST_PROFILE

```
TMS_HOST_PROFILE = {
```

Sample TMS_HOST_PROFILE

!	----------	----------	----------	----------
!	PROFILE NAME	PRIMARY HOSTNAME	SECONDARY HOSTNAME	PORT
!	----------	----------	----------	----------
	Ccxml1	tms1-1a	-	5314

Sample sip.conf File

The following is a sample configuration file for a standalone MPS SIP server system.

Sample sip.conf File ()

```
############################################################
#
# File:
#       sip.conf Sample File
#
# Description:
#
#  - A comment is any line prefaced by an octothorp (#) or
#    an exclamation point(!).
#
#
############################################################
############################################################
#
# The configuration of the default section.
# Please note that the default section does
# not have a section name.
#
# The property defined in this section is mainly
# used internally, not for customer.
############################################################
# The properties of this default section are
# defined in the following property file
# under the same dir with sip.conf
sip.default.property.file=sipdefault.properties
sip.version=1.0
codec.list.1=0
codec.list.2=8
codec.list.3=18
```

```
!*********************************************************
!
! S Y S T E M   D E S C R I P T I O N   S E C T I O N
!
!*****************************************************************
!
! This section specifies the system param definition.
!
#
# NAME:            The name of the sip server.
# SIP Server HOST NAME: The valid host name or
#                  IP address of this sip server
# REDUND ENABLE:   True or false, shows if
#                  the system is redundant enabled
# REDUND PRIMARY:  For redundant system show if this
#                  instance is the primary instance
#                  For stand along system, put "-".
```

```
# FT SIG PORT:     The port number used to communicate
#                  between primary and redundant pair
#                  For stand alone system, put "-".
# PEER FTSever HOST NAME: Valid sip server HOST Name
#                  or IP address of the redundant peer.
#                  For stand alone system, put "-"
#
#################################################################
```

```
[SYSTEM]
SYSTEM
 = {
! ------------------------------------------------------------------
!  NAME SIP Server         REDUND REDUND  FT SIG PEER FTServer
!       HOST NAME          ENABLE PRIMARY  PORT   HOST NAME
! ------------------------------------------------------------------
mps1 mps1.us.avaya.com   false  -      -         -
}
!*****************************************************************
!
!  S I P  N E T W O R K  C O N F I G U R A T I O N  S E C T I O N
!
!*****************************************************************
!
! This section specifies the network parameters definition.
!
! The system can configure CLIENT ACCESS mode or
! NETWORK_HOSTS mode, or Both.
!
!
```

```
[NETWORK]

#################################################################
#
# The configuration of the CLIENT_ACCESS
# NAME:          the name of this client access.
# TYPE:          Type of the transportation, for now
#                only UDP is supported
# LOCAL PORT:    UDP port number
# TIMER PROFILE: name of timer profile, it has to be
#                defined in SIP_TIMER_PROFILE.
#
#################################################################
```

```
CLIENT_ACCESS
 = {
! ------ ------- ----- ---------
! NAME   TYPE    LOCAL TIMER
!                PORT  PROFILE
! ------ ------- ----- ---------
CLIENT  UDP      5060  SIP_TMR1
}
```

```
#################################################################
#
# The configuration of the NETWORK_HOSTS
# NAME: the name of this network host.
# TYPE: Type of the transportation, for now only "UDP"
# and "PERFECT" is supported
# SIP DOMAIN NAME:     Sip domain name of this node.
# PRIMARY HOST NAME:   For UDP, it the IP Address for
#                      this node ( soft switch),
#                      for perfect channel this is the IP
#                      Address for the primary host,
#                      it can be host name for this IP Address or
```

```
#                       IP address string in "%d.%d.%d.%d" format.
# SECONDARY HOST NAME: For UDP, it has to be "-".
#                       For Perfect channel,
#                       this is the peer's IP Address,
#                       it can be host name for this IP Address or
#                       IP address string in "%d.%d.%d.%d" format.
# PORT:                 port number for UDP or Perfect Channel
# SSA PORT:             softswitch advertisement port,
#                       heartbeat is monitored on this port.
# TIMER PROFILE:        name of timer profile, it has to
#                       be defined in SIP_TIMER_PROFILE.
# REGISTRAR FLAG:       "true" if this is a registar, "false" otherwise.
#
################################################################
NETWORK_HOSTS = {
! ------------------------------------------------------
! NAME  TYPE    SIP DOMAIN NAME     PRIMARY
!                                   HOST NAME
! ------------------------------------------------------
SS1   UDP     avaya.com  47.104.12.12
SS2   UDP     avaya.com  47.104.12.13
! SS3  PERFECT  avaya.com  tcmps01.us.avaya.com
! SS4  PERFECT  avaya.com  tcmps01.us.avaya.com

! ------------------------------------------------------
! SECONDARY             PORT   SSA     TIMER    REGISTRAR
! HOST NAME             PORT   PROFILE FLAG
! ------------------------------------------------------
-                   5060     -     SIP_TMR1   true
-                   5060     -     SIP_TMR1   true
! tcmps02.us.avaya.com 20000  20005   SIP_TMR1   true
!   -                  20000  20005   SIP_TMR2   false
}
```

```
################################################################
#
# The definition of the sip timer
# NAME:    the name of this timer profile.
# INV2PRV: timer for the receipt of a Provisional response
#          to an Invite message ( in milliseconds)
# INV2FIN: timer for the receipt of a Final response to
#          an Invite message ( in milliseconds)
# INF2FIN: timer for the receipt of a Final response to
#          an Info message( in milliseconds)
# FIN2ACK: timer for the receipt of an Ack
#          to a Final response( in milliseconds)
# PRV2PRK: timer for the receipt of a Prack response
#          to an Provisional message( in milliseconds)
# PRK2FIN: timer for the receipt of a Final response
#          to an Prack message( in milliseconds)
# BYE2FIN: timer for the receipt of a Final response
#          to a Bye message( in milliseconds)
# T1:      represents the time delay between sending
#          retransmissions over UDP.
#          It has an exponential backoff to
#          T2( in milliseconds)
# T2:      represents the cap on delays between
#          sending retransmissions over
#          UDP( in milliseconds)
#
################################################################
SIP_TIMER_PROFILE
 = {
! ----------------------------------------
! NAME      INV2PRV INV2FIN  INF2FIN FIN2ACK
```

```
! ----------------------------------------
SIP_TMR1  5000    60000    5000    5000
SIP_TMR2  7000    60000    5000    5000
! ----------------------------------------
! PRV2PRK PRK2FIN BYE2FIN   T1    T2
! ----------------------------------------
3000    3000    3000    3000  2000
3000    3000    3000    3000  3000
}
!*****************************************************************
!
!  M E D I A  G A T E W A Y  C O N F I G U R A T I O N  S E C T I O N
!
!*****************************************************************
```

```
[MEDIA_GATEWAY]
```

```
################################################################
#
# The configuration of the MG_LIST, the media gateway
# configured for sip server
# NAME:          name of this MG_LIST entry.
# DOMAIN:        domain name of the media gateway.
# HOST Name:     Audiocode card's IP Host Name,
#                it can be IP address string
#                in "%d.%d.%d.%d" format.
# MG PORT:       gateway transport port number
#                use "2427" for standard,
#                or the number configured for this card.
# MG TYPE:       MG_Profile type for this entry, it
#                has to be defined in the MG_PROFILE.
#                Only tpm200, tpm800 is implemented for now.
# CHASSIS ID:    chassis number.
# SLOT ID:       slot number.
# DEFAULT CODEC: default payload Codec
#                "G711"|"G723"|"G726"|"G729".
# TMS_HOST NAME: Profile name of the tms host
#                which controls this TPM card, it must be one
#                of the PROFILE NAME entry of
#                TMS_HOST_PROFILE in CALL_CONTROL section.
#
#
################################################################
MG_LIST
 = {
! ------------------------------------------------
!  NAME     DOMAIN        HOST Name             MG
!                                                PORT
! ------------------------------------------------
tctpm01 us.avaya.com  tpm01.us.avaya.com   2427



! ------------------------------------------------
!  MG      CHASSIS   SLOT   DEFAULT   TMS_HOST
!  TYPE    ID        ID     CODEC     PROFILE NAME
! ------------------------------------------------
tpm800   1         1     G711      tms1

}
################################################################
#
# The configuration for each type of audiocode card
# NAME:             the name of this mg profile.
# EP PREFIX:        prefix for endpoint.
```

```
# NUM OF EP GROUP:  number of endpoint group this type of media card has
# NUM OF PORTS PER GROUP: number of endpoint each endpoint group has
# START:            The start number used to name the endpoint.
#
#################################################################
```

```
MG_PROFILE
 = {
! ------ ---------- -------- ------------- ------
! NAME   EP PREFIX  NUM OF   NUM OF PORTS   START
!                   EP GROUP PER GROUP
! ------ ---------- -------- ------------- ------
tpm200   ACGW        2          30             0
tpm800   ACGW        8          30             0
}
```

```
################################################################
#
# The definition of the mgcp timer
# NAME: the name of this timer profile.
# CRCX: time out for create connection( in seconds)
# MDCX: time out for modify connection( in seconds)
# DLCX: time out for delete connection( in seconds)
# RQNT: time out for request notify( in seconds)
# NTFY: time out for notify( in seconds)
# AUEP: time out for audit endpoint( in seconds)
# BEAT: time out for auepbeat( in seconds)
#
################################################################
MG_TIMER_PROFILE
 = {
! ----------  ------- ------- ------- ------- ------- ------- -------
! NAME         CRCX    MDCX    DLCX    RQNT    NTFY    AUEP    BEAT
! ----------  ------- ------- ------- ------- ------- ------- -------
MGCP_TMR1     1       1       1       1       1       1       1
}
!*****************************************************************
!
!  C A L L  C O N T R O L  I N T E R F A C E  S E C T I O N
!
!*****************************************************************
```

```
[CALL_CONTROL]

################################################################
#
# Specify the ASE host profile information for MPS SIP Server
#
# PROFILE NAME: name of this profile.
# PRIMARY HOSTNAME: Primary Host name or IP
# Address of this host (Vas host name).
# SECONDARY HOSTNAME: Secondary Host name or
# IP Address of this host (Vas host name).
# PORT: port number for proxy connection.
# if no host available, leave it empty.
#
################################################################
```

```
ASE_HOST_PROFILE = {
! --------  -------------- ------------- -----
! PROFILE     PRIMARY       SECONDARY
! NAME       HOSTNAME       HOSTNAME     PORT
! --------  -------------- ------------- -----
ase1       asehost1         -          9893
```

```
#  ase2       asehost2         -          9893
}
```

```
###################################################################
#
# Specify the TMS host profile information for MPS SIP Server
# PROFILE NAME: name of this profile.
# PRIMARY HOSTNAME: Primary Host name or IP
# Address of this host (Tms host name).
# SECONDARY HOSTNAME: Secondary Host name
# or IP Address of this host (Tms host name).
# PORT: port number for tms connection.
#
# if no host available, leave it empty.
#
###################################################################

TMS_HOST_PROFILE ={
! --------  -------------- ------------- -----
! PROFILE     PRIMARY       SECONDARY
! NAME       HOSTNAME       HOSTNAME     PORT
! --------  -------------- ------------- -----
   tms1       tmshost1         -          5314
#  tms2       tmshost2         -          5314
}
```

```
###################################################################
#
# The definition of the call control host for MPS SIP Server
# HOST_ID:           host id.
# HOST PROFILE NAME: The profile name of the host used for call control.
#                    This file has to be defined in one
#                    of the above profile entries.
# #
###################################################################
CC_HOSTS
 = {
! ----  -------------
! HOST  HOST
! ID    PROFILE NAME
! ----  -------------
0       ase1
}
```

```
###################################################################
#
# The definition of the CC_LINEMAP
# HOST ID:     host id, it has to match with
#              one of the entry in CC_HOSTS
# COMP ID:     Component id( vms id for ASE
#              mode and tms id of Span Id
#              for DSSI mode)
# LINE RANGE from:to:  Starting line number
#              to ending line number
#
###################################################################
CC_LINEMAP
= {
! ---- ----- ----------
! HOST COMP  LINE RANGE
! ID   ID    from:to
! ---- ----- ----------
0     1      1:30
}
!*******************************************************************
!
```

```
!  R E G I S T R A T I O N  C O N T R O L  S E C T I O N
!
!****************************************************************
!  This section defines Registration service for MPS SIP Server
!

[REGISTRATION_CONTROL]

################################################################
#
# The definition of the Registration service
# SERVICE NAME:    The service provided by the application.
#                  E.g. "prepaid" "stock"
#                  The last one is always for the default if there any,
#                  the service name for the default is "default".
# DOMAIN NAME:     The domain of the location service for
#                  which the registration is meant.
# EXPIRES:         Registration expiry value for the
#                  contacts that do not have the
#                  "expires" parameter.
#                  The value of this field is an integral
#                  number of seconds(in decimal)between
#                  0 and (2**31)-1.
#                  If neither mechanism for expressing
#                  a suggested expiration time is
#                  present in a REGISTER,
#                  a default suggestion of one
#                  hour SHOULD be assumed.
# CONTACT PROFILE: Name of the contact profile for
#                  this service, if more than one
#                  then use "," to separate each
#                  profile, E.g. CONTACT_1,CONTACT_2,
#                  each of the contact profile
#                  has to be defined in CONTACT_PROFILE.
# KEEP REGISTERED: A flag used to specify if this
#                  service will be kept registered always.
#
################################################################
REGISTRATION_SERVICE
 = {
! -------- ------------------ ------------- ------------ -------------
! SERVICE       DOMAIN                          CONTACT       KEEP
! NAME           NAME            EXPIRES       PROFILE       REGISTERED
! -------- ------------------ ------------- ------------ -------------
prepaid   avaya.com    60000        CONTACT_1        Y
stock     avaya.com    60000        CONTACT_2        Y
default   avaya.com    3600         CONTACT_DFT      N
}

################################################################
#
# The definition of the contact profile
# PROFILE NAME: Name of this profile.
# USER NAME:    Name of the service.
# TRANSPORT:    Transportation type.
# PORT :        Port number used
# EXPIRES:      The "expires" parameter of the contact
#               field indicates how long the UA would
#               like the binding to be valid.  The
#               value is a number indicating seconds.
#               If this parameter is not provided, the
#               value of the Expires header field is
#               used instead.
#
################################################################
CONTACT_PROFILE
```

```
 = {
! ---------- ---------- ---------- ------ ---------
! PROFILE      USER
! NAME         NAME      TRANSPORT  PORT   EXPIRES
! ---------- ---------- ---------- ------ ---------
CONTACT_1   prepaid      UDP      5060   60000
CONTACT_2   stock        UDP      5060   60000
CONTACT_DFT default      UDP      5060   40
}
```

```
###################################################################
#
# The definition of the Register timer.
# NOAPP: After register is sent out, if no applications available
#        for the time( in second) specified in this
#        timer, send deregister.
# RETRY: If register with all the registrars failed,
#        retry after the time
#        period(in second)specified in this timer.
#
###################################################################
REG_TIMER_PROFILE
 = {
!   ------- -------
!    NOAPP   RETRY
!   ------- -------
30      40
}
```

```
###################################################################
#
# The definition of application service group
# GROUP NAME:   Name of this group.
# SERVICE LIST: List of service supports by this group.
#
###################################################################
APPSVC_GROUP
 = {
! ---------- ---------------
! GROUP       SERVICE
! NAME         LIST
! ---------- ---------------
groupA     prepaid,stock
}
```

```
###################################################################
#
# The definition of line configuration for each service
# SERVICE GROUP NAME: Name of this group.
# HOST NAME:          List of service supports by this group.
# COMP ID:            Component id number.
# LINE RANGE :        Line that configured for this service,
#                     E.g. 1, 2, 5-10 means line 1, 2, 5, 6, 7, 8, 9,
#                     10 are configured for this service.
#
###################################################################
REG_LINE_CONFIG
 = {
! ----------------- -------- ------ ----------
!  SERVICE GROUP      HOST     COMP   LINE
!  NAME                ID       ID    RANGE
! ----------------- -------- ------ ----------
prepaid            tcmps02    1    1,10,20-30
stock              tcmps02    1    5-8
}
###################################################################
```

```
# $Id$ sip.conf.sample,v 1.6 2003/03/17
##################################################################
```

CCSS SIP Configuration

Example: 2+1 MPS SIP redundancy configuration (here N=2)

1. The sip.conf configuration file modified to include Secondary/Standby MPS in ASE_HOST_PROFILE of call control configuration section:

   Example: Node ase3 profile mapped to tmsi100 which is a secondary/standby AP Host serving ase1 profile for primary MPS Node tmsi01 and ase2 profile for another primary MPS Node tmsi02.

   ASE_HOST_PROFILE = { ! -------- ---------- ---------- ---------- ! PROFILE PRIMARY SECONDARY PORT ! NAME HOSTNAME HOSTNAME ! -------- ---------- ---------- ---------- ase1 tmsi01 - 9893 ase2 tmsi02 - 9893 ase3 tmsi100 - 9893 }

2. HostId entry for Primary/Active & Secondary/Standby MPS nodes are in CC_HOSTS in sip.conf:

   Example: ase3 profile assigned host id 3 for Secondary/Standby MPS node.

   CC_HOSTS = { ! ---- ---------- ! HOST HOST ! ID PROFILE NAME ! ---- ---------- 1 ase1 2 ase2 3 ase3 }

3. No change to CC_LINEMAP section:

   CC_LINEMAP = { ! ---- ---------- ! HOST COMP LINE RANGE ! ID ID from: to ! ---- ---------- 1 1 1:240 1 2 1:240 2 3 1:240 2 4 1:240 }

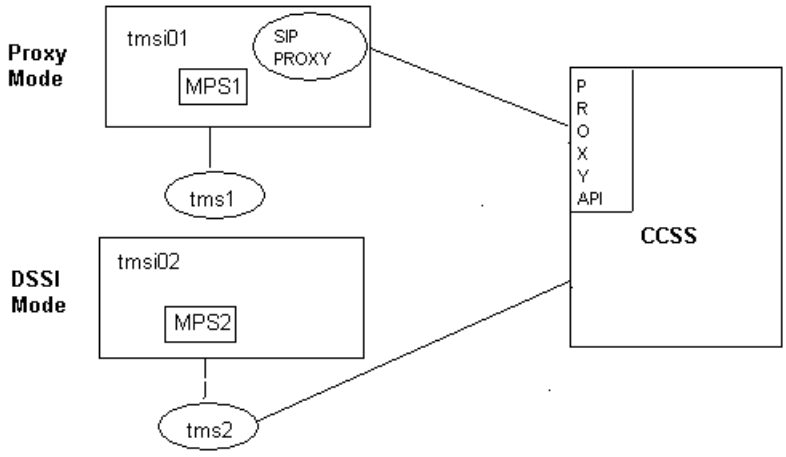


Note:

- No entry for CC_LINEMAP section for secondary/standby MPS components.
- Changes are only for ASE mode MPS SIP N+1 Configuration. No configuration changes in DSSI mode to support MPS N+1.

CCSS SIP configuration for MIXED mode

In MIXED mode CCSS SIP Server supports configuring and running both proxy / dssi mode call-control programming applications concurrently.



Example: Different modes of MPS components configuration

- The sip.conf configuration file modified to include proxy mode mps component in ASE_HOST_PROFILE and dssi mode component in TMS_HOST_PROFILE of the call control configuration section

  For example: Node ase1 profile mapped to tmsi01 and tms1 profile mapped to tmsi02.

```
ASE_HOST_PROFILE = {
! -------- -------------- ------------- ------------------------------
! PROFILE PRIMARY SECONDARY PORT
! NAME HOSTNAME HOSTNAME
! -------- -------------- ------------- ------------------------------
ase1 tmsi01 - 9893
}

TMS_HOST_PROFILE = {
! -------- -------------- ------------- ------------------------------
! PROFILE PRIMARY SECONDARY PORT
! NAME HOSTNAME HOSTNAME
! -------- -------------- ------------- ------------------------------
```

```
tms1 tmsi02 - 5314
}
```

- HostId entries for both ASE profile entry and TMS profile entry are in CC_HOSTS in sip.conf:

  Example: ase1 profile is assigned with host id 1 and tms1 profile is assigned with host id 2.

```
CC_HOSTS = {
! ---- ------------------------------
! HOST HOST
! ID PROFILE NAME
! ---- ------------------------------
1 ase1
2 tms1
}
```

- NO Change to CC_LINEMAP section

```
CC_LINEMAP = {
! ---- ------------------------------------
! HOST COMP LINE RANGE
! ID ID from: to
! ---- ------------------------------------
1 1 1:240
2 1 1:30
2 2 1:30
2 3 1:30
2 4 1:30
2 5 1:30
2 6 1:30
2 7 1:30
2 8 1:30
}
```

Note:

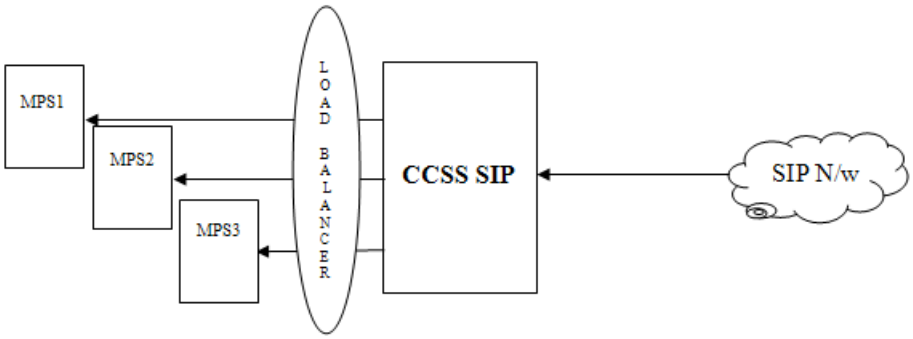
To enable the CCSS server in MIXED mode; at least two independent call-control host profiles configured in CC_HOSTS section.

Configuration for SIP Network Load Balancing

Load Balancing distributes processing and communications activity evenly across a clustered network so that no single device is overwhelmed. This is of huge importance in a network where there is an unpredictable flow of traffic or requests to the server.

- To maximize the utilization of the entire available resources in a distributed system environment
- To avoid acquiring new resources without fully using the available resources.

This makes load balancing the most efficient way to achieve high availability and high performance. The MPS SIP balances the inbound traffic within each service across MPS Application Processors.



CCSS SIP allows the system to be enabling Network Load Balancing in 3 different modes of algorithm:

- Round Robin algorithm – Load sharing within a single service-name irrespective of configured SIP lines within MPS's.
- Relative Ratio algorithm – Load sharing within a single service-name by load ratio on the available SIP lines using "divide by minimum" method.
- Weightage algorithm – Load sharing within a single service-name by weightage ratio on the available SIP lines within each MPS Component using "median" method.

Round-Robin Algorithm

Incoming SIP traffic is load balanced within the service by allocating one available resource from each AP (MPS component) at a time. This algorithm is best suited and recommended

where a service with various components configured with similar number of lines. For example, component #1 configured with 90 lines, component #2 configured with 90 lines and component #3 configured with 90 lines.

Relative Ratio Algorithm

Incoming SIP traffic is load balanced within the service by allocating available resources from each AP (MPS component) based on the relative ratio obtained by Divide by Min Algorithm. This algorithm is best suited and recommended for varied line configuration for various components for a particular service. For example component #1 configured with 30 lines, component #2 configured with 60 lines and component #3 configured with 90 lines.

Weightage Algorithm

Incoming SIP traffic is load balanced within the service by allocating available resources from each AP (MPS component) based on the weightage ratio obtained by Median Algorithm. This algorithm is best suited and recommended, when lines configured for various components for any particular service are varied and in extreme range. For example, component #1 configured with 10 lines, component #2 configured with 70 lines and component #3 configured with 240 lines.

This table details the lines picked for various components configured for a service based on different NLB algorithms enabled on MPS SIP:

SIP Service Name	Comp No	Line Configured	Round Robin	Relative Ratio	Weightage
any-inbound	101	30	1	1	1
	102	90	1	3	1
	102	240	1	8	2

General MPS Configuration Files

gen.cfg

The following command is inserted into the $VPSHOME/common/etc/gen.cfg ($MPSHOME/common/etc/gen.cfg in Windows) file during software installation. No user intervention is required.

sipproxy	
This command configures the SIP resource daemon executable (sipproxy). The SIP resource daemon performs the message transmission and conversion between application and sip server.	
`sipproxy -S resource -s service -E -k -n [-L destination] [-l logging] [-X a | i | o | n] -p -d sipproxy -H | -h sipproxy -v sipproxy -V`	
`-S`	Sets the VVP resource name. This should always be set to sip.
`-s`	Sets the TCP service name or port.
`-E`	Disables use of environment conditions.
`-k`	Sets the keepalive interval.
`-n`	Sets the redundancy mode. -n implies only one LAN is in use. Calls may be lost if the LAN fails. No -n implies that redundancy is active, and this may increase system load.
`-L`	Sets the destination for logging messages to one of the following values: console for main console window/screen or file for the log file name.
`-l`	Sets the logging level to one of the following values: emerg, alert, critical, error, warning, notice, info, deb1, deb2, or deb3.
`-p`	Enables ptimer child debugging. This is for use by Avaya only.
`-d`	Enables plib debugging. This is for use by Avaya only.
`-t`	Enables terse logging. Logging messages will not have process ID or timestamp information. This is for use by Avaya only.
`-h -H`	Displays the options list.
`-v`	Prints the sipproxy version number and exits.

sipproxy	
-V	Indicates the number of local vmst component it has to connect. For example, –V <#componentNo>. This option is not recommended when MPS SIP N+1 redundancy feature is enabled through -c option.
-c	Enables SIP resource proxy to support MPS AP N+1 cluster redundancy feature.

Note:

The command is set up as follows. Do not alter these parameters.

```
sipproxy -S sip -s 9893 –V 1 -L cons -l info -k 10 –n
```

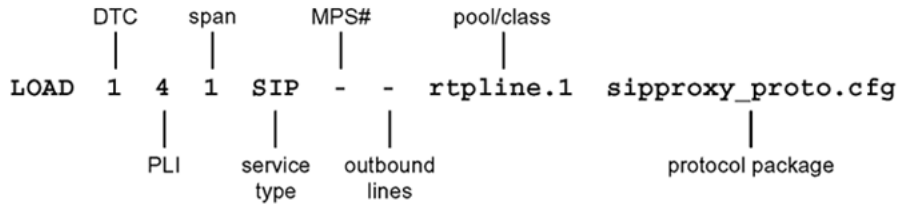
tms.cfg

The $MPSHOME/common/etc/tms/tms.cfg file in Solaris (%MPSHOME%/common/etc/tms.cfg in Windows) stores configuration parameters for the TMS component of the MPS. To configure a SIP system for either ASE or TMS mode, the tms.cfg file requires minor modifications. Telephone lines are pooled and configured as inbound or outbound in this file.

The tms.cfg file includes a set of LOAD commands, which are used to specify the particular protocol packages to be loaded into the system's line cards. For DCC cards, there is a protocol package specified for span of the DCC card.

To configure any SIP system, in the LOAD command, specify the service type as SIP. For ASE mode, specify the protocol configuration as sipproxy_proto.cfg. For TMS integrated mode, specify the protocol configuration as sipdssi_proto.cfg. Example LOAD commands are shown in the following:

ASE Mode:



TMS Integrated Mode:

```
LOAD  1  4  1  SIP  -  -  rtpline.1  sipdssi_proto.cfg
```

Parameter	Description
DTC	DTX number. This is taken from the [DTCMAP] section of the tms.cfg file.
PLI	DTC slot to which the card is connected.

Parameter	Description
span	Span number of the DCC card. For analog cards, this is not applicable. For analog, this should be specified as a dash (-).
service type	ASCII string returned to applications in responses to GetInCall and GetOutLine containers.
MPS #	ID number of the MPS.
outbound line	Span/cards that have the outbound phone lines to be placed into the pool. This parameter can be specified as a single line, range of lines, all lines (*), or as an ALI (-) card. Syntax: {<line> | <startline>-<endline>} | * | -} single line | line range | all | ALI
pool/class	Class name for creating the resource pool.
protocol package	Name of the file having DTC resources needed to support the requested protocol.

Chapter 7: Operations, Maintenance, and Debugging

This chapter covers:

1. System Startup and Shutdown Procedures
2. Backup and Restore Procedures
3. System Status and Statistics
4. Diagnostics
5. Debugging Tools

System Startup and Shutdown

To start the CCSS 2000 and SIP software:

Solaris / UNIX System:



1. Turn on the main power switch. Wait for the login prompt.
2. Log in as root. (Enter root; as the password.)

   If the server software hasn't been installed yet, follow the procedures in the Installation and Configuration section. If the software has been installed, all processes will start automatically.

   To start the SIP software manually, enter ccss_start at the command line.

Windows 2000 System:



Example

The Windows version of CCSS uses SRP to start. The Startup Service is installed with the PERIglobl package and should be loaded during bootup, along with other required subsystems.

To manually start SRP:

1. Click Start > Settings > Control Panel
2. Double-click Services
3. Highlight Avaya Startup Service and click Start

Note:
You must have administrative permissions to perform these actions. Once the system is started, the system status can be queried through vsh srp command.

To shut down the CCSS 2000 and SIP software:

Solaris / UNIX System:



1. Enter super user mode by issuing su at the system prompt.
2. Enter the ccss_stop command to shut down the SIP processes. Any applications that are running when ccss_stop is executed are brought down immediately.
3. Stop the system by entering halt at the system prompt.

ccss_start	Starts the SIP stack and Call Control processes.
This command is put into the /etc/rc3.d/S50ccss file during software installation and configuration. S50ccss is executed during system startup. It is normally not necessary to use ccss_start from the command line. The only time this is valid is when the system was brought down manually and has to be restarted without rebooting.	

ccss_stop	Stops the Call Control and SIP stack processes.
The ccss_stop command terminates all call control and SIP stack processes. All calls in progress are lost when ccss_stop is executed.	

Windows 2000 System:



1. Click Start > Settings > Control Panel
2. Double-Click Services
3. Highlight Avaya Startup Service and click Stop

Note:
You must have administrative permissions to perform these actions. Once the system is started, the system status can be queried through vsh srp command.

Backup and Restore

The backup and restore procedures in this section assume that you have the original software installation and update (patch) tapes (including the operating system, in the event of a total disk failure). In this case, backup involves only making copies of those files that have been changed since the initial installation and any custom application files (if applicable). Restoring the system depends on how much needs to be restored. In the event of a total disk failure, the OS must be installed first on the new disk, followed by the SIP software (with any updates) and the custom files. To restore to an earlier software version, reinstall the desired SIP software and any custom files.

There are several choices for backup devices. The MPS can be equipped with one or more 1/4 inch tape drives or internal hard drives. Each device has a unique logical name.

Device	Name	Capacity	Usage
cartridge tape	`/dev/rmt/0`	250 MB	Weekly backup of all system and application files
cartridge tape	`/dev/rmt/0`	1 GB	Weekly backup of all system and application files
internal drive	varies with system configuration	varies with system configuration	Daily or weekly backup of all system and application files

Note:
Cartridge tapes have a write-protect feature. If there is difficulty backing up to a tape, check that write-protect is off. For details, see the instructions included with the tape.

There are two different backup methods:

Method	Description
`tar`	Transfers files to a tape or internal drive. Use tar to archive files to a single tape archive (TAR) file.
`cp`	The copy command backs up individual files or groups of files to other internal drives or directories.

The following are some useful recommendations:

- Keep a supply of sequenced and labeled tapes on hand.
- Store backup media in a safe place. Handle media carefully.
- Label each tape with the date of backup and the names of the included files.
- Keep a log of backup filenames and dates.
- Follow all instructions included with the media.
- Write protect the tapes after the backup.

UNIX cp (Copy) Command

The UNIX cp command copies individual files or groups of files. Use this command to back up crucial files to alternative directories and to copy files between directories.

See the Solaris system documentation for details and examples about using cp.



To copy an individual file in Windows, highlight the file in the directory and select Edit > Copy from the main menu. Highlight the directory to move the file to and select Edit > Paste from the main menu.

The TAR Utility

Backing Up

The tar command combines and saves files to a tape or an internal drive.

To backup to a tape, label it before inserting into the tape drive. The tar command overwrites existing files. Therefore, in general, it is best to not backup to a tape that already contains useful files.

Note:

The tar command does not support multivolume backups. Use df to display the amount of disk space used by the file system. If there is need to back up more than the capacity of

your tape drive (250 MB or 1 G), split the job into several backups (that is, back up individual partitions).

The following describes how to use tar to make backups.

1. Access a UNIX command shell:
    a. Click <MENU> on the desktop.
    b. In the Workspace pop-up menu, click <MENU> on Programs.
    c. From the Programs menu, select Command Tool....
2. Move to the directory where the files are located (that is, the source directory).
3. Enter the tar command.
    - To backup to a tape, use the syntax tar cvf /dev/rmt/0 <sourcefiles>
    - To backup to disc, first backup to the present working directory, tar cvf <archivefile.tar> <sourcefiles>

      then use cp to place the tar file in an alternate directory or drive.

Note:

This command combines and saves the source files in the archive file <archivefile.tar>. Specify either individual files or entire directories to be backed up. Use a descriptive file name with the .tar extension for <archivefile.tar> (for example, oldvocabs.tar).

Backups can also be made directly to an alternative directory or internal drive. See the MPS System Reference Manual for details on the tar command, and for details and examples on using cp.



In a Windows environment, equivalent tools are Winzip or tar.exe. Both tools are available on any public domain.

Verifying a Backup File

The tar command can be used to verify each file stored in the archive file by displaying its name and size on the screen. Display the table of contents for each backup file to ensure that it is readable and complete.

To display the contents of a tape-based backup file, insert the tape into the drive before proceeding.

1. Access a UNIX command shell:
    a. Click <MENU> on the desktop.
    b. In the Workspace pop-up menu, click <MENU> on Programs.
    c. From the Programs menu, select Command Tool....
2. Enter the tar command.
    - To verify the contents of a tape, enter tar tvf /dev/rmt/0.

- To verify the contents of a tar file on the internal drive, first move to the directory containing the file. Then, enter the command tar tvf <archivefile.tar> specifying the name of the tar file.

Restoring a Backup File

Use the tar command to restore selected files or the entire backup.

To restore the contents of a tape-based backup file, insert the tape into the drive before proceeding.

1. Access a UNIX command shell:
    a. Click <MENU> on the desktop.
    b. In the Workspace pop-up menu, click <MENU> on Programs.
    c. From the Programs menu, select Command Tool....
2. Move to the directory where the files are to be located (that is, the destination directory).
3. Enter the tar command.
    - To restore files from a tape, enter tar xvf /dev/rmt/0.
    - To restore files from an internal drive, use cp to place the tar file in the destination directory. Then, enter the command tar xvf <archivefile.tar> specifying the name of the tar file.

SIP Server Backup and Restore

The file $CCSSHOME/etc/sip.conf (Solaris) or %ccssN/etc/sip.conf (Windows), which is required for all MPS SIP configurations, must be included in backup operations:

Other files, on a site-specific basis, may require backing up. In general, backups should be made of any files residing on the SIP server that can be modified by users or applications.

To use tar to backup the SIP server, perform the following steps:

1. Insert a blank tape into the tape drive.
2. Enter the following command on the command line:

```
tar cvf /dev/rmt/0 <file list>
```

Note:

The <file list> should include the sip.conf file and any other customized files. The tar command must be entered on one line.

To restore the SIP server from a tar file, perform the following steps:

1. Install the Solaris operating system (if required). Refer to the Solaris system documentation for information. A standard peri account is required.
2. Insert the SIP server software distribution tape into the tape drive and install the server software and any updates.
3. Insert the backup tape into the tape drive and enter the following to restore the modified/custom files:

   ```
   tar xvf /dev/rmt/0
   ```

4. Enter the following commands in the order shown here to use the restored software:

   ccss_stop ccss_start



This is done in Windows by starting and stopping the SRP service.

Diagnostics and Debugging

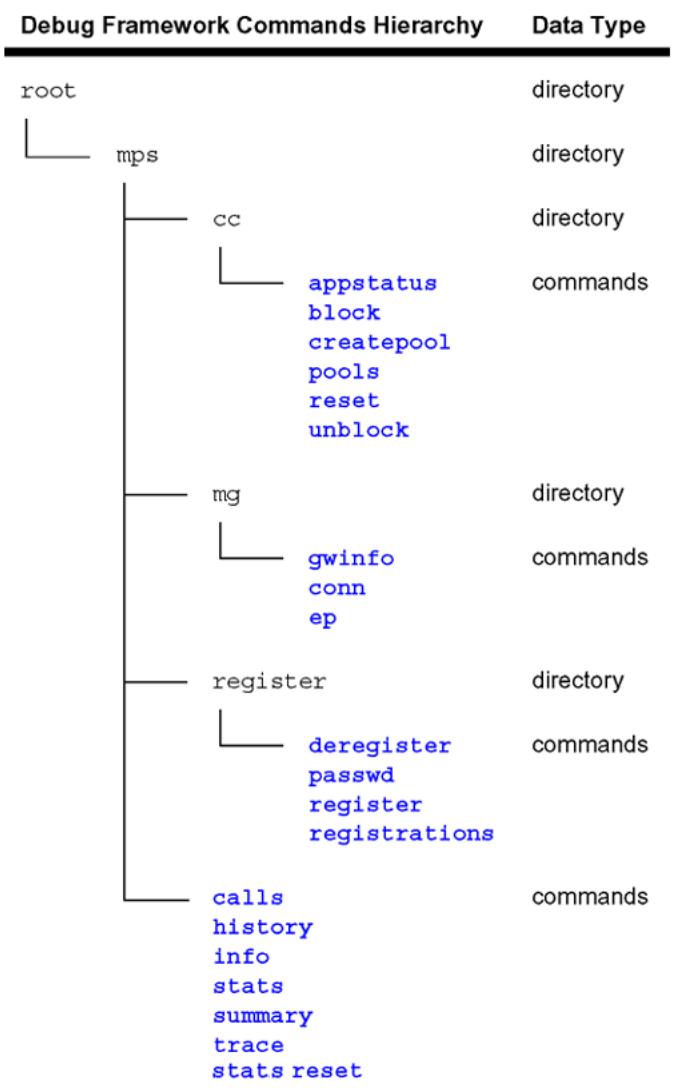
The MPS SIP debug framework contains various commands organized into directories. To enter the debugging environment, issue the command sipui at the command line. This places the current user position in the DEBUG:root directory. All commands relevant to MPS SIP debugging are stored in the directory DEBUG:root/mps or in its subdirectories.



The sipui script in Solaris version uses telnet utility to establish connection to SIP server. To avoid incompatible CR-LF interpretation from Windows telnet utility, a simple sipui.exe utility is provided for command user interface access.

A debug command performs some action on application objects. The command set includes status-reporting commands (such us for the status of each application, the status of the proxy connection, and so on), control and operational commands for system operations and configuration, statistics commands (such as for acquiring the general statistics of the call model), and tracing commands. Online help screens are available for all system and debugging commands. To access a help screen, type help <command>, specifying the name of the command.

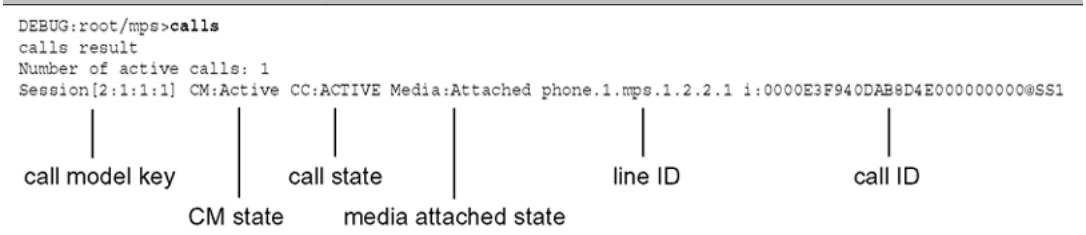
Commands are issued using a command prompt. The command prompt string is the word DEBUG: followed by a denotation of the current directory.



DEBUG:root/mps

The commands in this section are stored in DEBUG:root/mps directory.

calls	Displays details about active calls.
Example:	

calls	Displays details about active calls.
	
call model key	Call model session ID.
CM state	Call model state.
call state	Call state.
media attached state	Shows whether or not media is attached.
line ID	Private media endpoint name.
call ID	Call ID.
history	Traces the log history for the specified host, component, and line.
`history on`	Enables the history.
`history off`	Disables history.
`history clear`	Clears all App trace events that logged to history lines.
`history <HostID> <CompID> <line>`	
`HostID`	Host identifier number.
`CompID`	VPS component ID number.
`line`	Phone lines to be traced specified as a range. A range is sequence of consecutive numbers. It can be only one number, representing a single phone line number, or two numbers separated by a dash to denote a sequence of lines.
info	Shows TMS or ASE connection details.

Example for TMS mode:
`DEBUG:root/mps> info System Started at: 1/01/03`
`12:31 PM DSSI Mode`
`TMS Connection Count: 1`
`==========`
`HostID TmsID IP Address Port Span Span`
`Count Status`
`==========`
`1 0 47.185.24.134 5314 2 STATE_CONNECTED`
`==========`	

Example for ASE mode:

```
DEBUG:root/mps> info System Started at: 1/01/03
 12:32 PM ASE Mode
================================================================
Host Vps    IP Address      Port   Line        Span
ID   ID                             Range       Status
================================================================
VPS Connection Count: 1
0   1      47.185.16.71    9893   1-30        STATE_CONNECTED
================================================================
```

`HostID`	Host ID
`TmsID/ Vps ID`	TMS or VPS ID
`IP Address`	IP address
`Port`	Port number
`Span Count`	Total number of spans (TMS mode only)
`Line Range`	Phone line range (ASE mode only)
`Span Status`	One of the following: STATE_DISCONNECTED, STATE_CONNECTING, STATE_CONNECTED, STATE_STANDBY

Example for ASE mode when Primary is active

```
DEBUG:root/mps>info
System Started at: 1/01/03 12:32 PM ASE Mode
================================================================
HostID  VpsID     IP Address     Port    line       Span
range    Status
================================================================

VPS Connection Count: 1
0      1     47.185.16.71  9893     1-30    STATE_CONNECTED
================================================================
```

Example for ASE mode when secondary/standby is active:

```
DEBUG:root/mps>info
System Started at: 1/01/03 12:32 PM ASE Mode
Failover at: 1/01/03 12:45 PM ASE Mode
================================================================
Host Vps IP Address Port Line Secondary Active Active Span Status

ID   ID                    Range  IP Address Host ID VPS ID
================================================================
Configured VPS Connection Count: 2
```

```
1  1      47.185.16.73  9893   1-30  47.185.16.132  2  3  STATE_CONNECTED
============================================================
```

`HostID`	Host ID
`TmsID/ Vps ID`	TMS or VPS ID
`IP Address`	Primary AP IP address
`Port`	Port number
`Span Count`	Total number of spans (TMS mode only)
`Line Range`	Phone line range (ASE mode only)
`Secondary IP Address`	IP address of the secondary/active AP (Application processor)
`Active Host ID`	Host ID of the Secondary/active AP (Application processor)
`Active Vps ID`	VPS ID of the secondary/active vps component
`Span Status`	One of the following: STATE_DISCONNECTED, STATE_CONNECTING, STATE_CONNECTED, STATE_STANDBY
`info`	Shows TMS or ASE connection details

Example for MIXED mode
DEBUG: root/mps/cc/>info System Started at: 12/16/09 10:32 AM MIXED Mode
TMS Connection Count: 1
==========
HostID TmsID IP Address Port Span Count Span Status
==========
1 0 10.7.3.1 5314 8 STATE_CONNECTED
==========
==========
Host ID Vps ID IP Address Port Line Range Span Status
==========
Configured VPS Connection Count:
2 2 47.185.16.26 9893 1-240 STATE_CONNECTED

HostID	Host ID.
TmsID/ Vps ID	TMS or VPS ID.
IP Address	Primary AP IP address

Port	Port number
Span Count	Total number of spans (TMS mode only)
Line Range	Phone line range (ASE mode only)
Secondary IP Address	IP address of the secondary/active AP (Application Processor).
Active HostID	Host ID of the Secondary/active AP (Application Processor).
Active Vps ID	VPS ID of the secondary/active VPScomponent
Span Status	One of the following: STATE_DISCONNECTED, STATE_CONNECTING, STATE_CONNECTED, STATE_STANDBY.
Info	Shows TMS or ASE or MIXED connection details.

nlb	Sets different methods of load sharing for received SIP traffic across configured MPS components.					
Defaults {defaultls roundrobinls relativels weightagels stats }						
	defaultls	No load-sharing across MPS, sets as-is functionality of sequential acceptance of inbound traffic.				
	roundrobinls	Sets load sharing across configured MPS's.				
	relativels	Sets relative ratio method for load balancing - divide by minimum method.				
	weightagels	Sets weightage based method for load balancing - median method.				
	stats	Shows statistics of a service in terms of available lines to load share across MPS's.				
Example: DEBUG:root/mps/cc/> nlb stats NLB Algorithm enabled : relativels at 12/7/09 9:52 AM						
	ServicePool	HostID:Vpsl	NumActive	NumReady	Total Calls	Weightage
	registrar postpaid any-inbound outbound prepaid					

Comments? infodev@avaya.com

		1:1	0	2	15	1
		1:2	0	2	15	1
HostID	Host ID					
Vps ID	VPS ID.					
ServicePool	Pool containing the applications for particular service.					
NumActive	Number of applications that are in the active state.					
NumReady	Number of applications that are in the ready state.					
Weightage	Mean ratio on available SIP lines within each MPS component.					
nlb stats	Shows statistics of a service in terms of available lines to load share across MPS components.					

stats	Displays MPS statistics for MPS_CALL, SIP_CALL, and MPS_ERR.

Example:

```
DEBUG:root/mps> stats
================================================================
<MPS_CALL>02-04-2003,15:01:37 on tcccss14:0
================================================================
I_ATMPTS  I_ANSWR  I_RJCT  I_TIMEOUT  I_FAIL  O_ATMPTS  O_ANSWR
MPSCC        0        0         0         0         0      30756    30724
O_RJCT  O_TIMEOUT  O_FAIL  Q_QUEUED  Q_SUCC  Q_CANCEL
MPSCC        0         0          0         0         0         0
================================================================
<SIP_CALL>02-04-2003,15:01:37 on tcccss14:0
================================================================
I_ATMPTS    O_ATMPTS    T_ATMPTS     I_ANSWR     O_ANSWR
SS1         0          30756       30756         0          30724
CLIENT      0            0           0           0            0
================================================================
<MPS_ERR>02-04-2003,15:01:37 on tcccss14:0
================================================================
TASKQFULL  RTNOTFOUND  TIMEREXHST  UNEXPMSG  ENCODERR
MPSCC         0          0           0           0         0
NOAPP  MAFAILED  MMFAILED  MDFAILED   MGFAILED
MPSCC         0          0           0           0         0
```

stats reset	Resets MPS statistics for MPS_CALL, SIP_CALL, and MPS_ERR.

Example:

```
DEBUG:root/mps> stats reset
================================================================
<MPS_CALL>02-04-2003,15:01:37 on tcccss14:0
================================================================
I_ATMPTS  I_ANSWR  I_RJCT  I_TIMEOUT  I_FAIL  O_ATMPTS  O_ANSWR
```

```
MPSCC       0         0         0         0         0      0    0
O_RJCT  O_TIMEOUT  O_FAIL  Q_QUEUED  Q_SUCC  Q_CANCEL
MPSCC       0         0          0         0         0         0
================================================================
<SIP_CALL>02-04-2003,15:01:37 on tcccss14:0
================================================================
I_ATMPTS    O_ATMPTS    T_ATMPTS     I_ANSWR     O_ANSWR
SS1         0         0         0          0          0
CLIENT      0           0           0           0           0
================================================================
<MPS_ERR>02-04-2003,15:01:37 on tcccss14:0
================================================================
TASKQFULL  RTNOTFOUND  TIMEREXHST  UNEXPMSG  ENCODERR
MPSCC        0           0           0           0        0
NOAPP  MAFAILED  MMFAILED  MDFAILED   MGFAILED
MPSCC        0           0           0           0        0
```

Component	Field	Description
MPS_CALL / SIP_CALL	I_ATMPTS I_ANSWER I_RJCT I_TIMEOUT I_FAIL O_ATMPTS O_ANSWER O_RJCT O_TIMEOUT O_FAIL Q_QUEUED Q_SUCC Q_CANCEL T_ATMPTS	Number of inbound attempts Number of inbound calls answered Number of inbound calls rejected Number of inbound calls timeout Number of inbound calls failed Number of outbound attempts Number of outbound calls answered Number of outbound calls rejected Number of outbound calls timeout Number of outbound calls failed Number of calls queued Number of calls succeed Number of calls canceled Number of total attempts
MPS_ERR	TASKQFULL RTNOTFOUND TIMEREXHST UNEXPMSG ENCODERR NOAPP MAFAILED MMFAILED MDFAILED MGFAILED	Number of task queue full error Number of route not found Number of timer expired Number of unexpected message Number of encoding error Number of no app available Number of media attach failed Number of media modify failed Number of media delete failed Number of media gateway failed

summary — Displays the last reported call connected timestamp.

```
DEBUG:root/mps/> summary
(1:1:19)<last conn>10:18:05.405 <last trace>10:18:12.340 In
CcApp::setAppState -- current state is act new state is ins
(1:1:20)<last conn>10:18:05.405 <last trace>10:18:10.232 In
CcApp::setAppState -- current state is act new state is ins
(1:1:21)<last conn>10:18:09.350 <last trace>10:18:10.703 In
CcApp::setAppState -- current state is ins new state is ins
(1:1:22)<last conn>10:18:09.351 <last trace>10:18:10.711 In
CcApp::setAppState -- current state is ins new state is ins
(1:1:23)<last conn>10:18:07.854 <last trace>10:18:09.174 In
CcApp::setAppState -- current state is act new state is ins
```

```
(1:1:24)<last conn>10:18:07.851 <last trace>10:18:09.595 In
CcApp::setAppState -- current state is act new state is ins
(1:1:25)<last conn>10:18:00.908 <last trace>10:18:06.907 In
CcApp::setAppState -- current state is act new state is ins
```

trace	Enables tracing of call events on the specified lines.
`trace <HostID> <mpsComponentNo/tmsSpanNo> <lineno/linerange>`	
`HostID`	Host identifier number.
`mpsComponentNo/ tmsSpanNo`	MPS component number for ASE Mode. TMS span number for DSSI Mode.
`lineno/linerange`	Phone lines to be traced are specified as individual lines or a range. A linerange is a sequence of consecutive numbers. Lineno represents a single phone line number. Linerange denotes a sequence of lines and is represented as two numbers separated by a dash.
	Note: Trace can be performed on events flowing between different tasks based on their Host, span, and channel IDs. The tracing of events can be viewed from a Telnet session on SIP server.

DEBUG:root/mps/cc

The commands in this section are stored in DEBUG:root/mps/cc directory. These commands are for application status and control.

appstatus	Displays application details.
`appstatus <HostID> {<MPS_comp#> | <TMS_span#>} <lines>`	
`HostID`	Host identifier number.
`MPS_comp#/ TMS_span#`	MPS component number (for ASE mode) or TMS span number (for TMS mode).
`lines`	Phone lines specified as a range. A range is sequence of consecutive numbers. It may be only one number, representing a single phone line number, or two numbers separated by a dash to denote a sequence of lines.
Examples	DEBUG:root/mps/cc> appstatus 0 1 1 (apps on MPS component 1, line 1, with host 0) DEBUG:root/mps/cc> appstatus 1 1 1-10 (apps on MPS component 1, lines 1-10, with host 1)

	DEBUG:root/mps/cc> appstatus 1 (apps on all lines, with host 1)

```
DEBUG:root/mps/cc/> appstatus 1
11:14:01.184
=========================================
MPS component 1 on HostId:1
=========================================
Total number of Lines: 30
0 act    1 act    2 act    3 ins    4 ins    5 ins    6 ins    7 ins
8 ins    9 ins   10 ins   11 ins   12 ins   13 ins   14 ins   15 ins
16 ins   17 ins   18 ins   19 ins   20 ins   21 ins   22 ins   23 ins
24 ins   25 ins   26 ins   27 ins   28 ins   29 ins
=========================================
MPS component 2 on HostId:1
=========================================
Total number of Lines: 30
0 ins    1 ins    2 ins    3 ins    4 ins    5 ins    6 ins    7 ins
8 ins    9 ins   10 ins   11 ins   12 ins   13 ins   14 ins   15 ins
16 ins   17 ins   18 ins   19 ins   20 ins   21 ins   22 ins   23 ins
24 ins   25 ins   26 ins   27 oos   28 oos   29 oos
```

	Application status is indicated on a per-line basis. The status of each line is one of the following: ins – In Service rdy – Ready to take call acp – Active Pending act – Active oos – Out of Service shd – Shutdown
block unblock	Enables or disables the receiving of calls in the TMS mode environment on the specified phone lines. This command is only for TMS mode.
`block/unblock <line-description> <line-description> = <HostID> <tmsSpanNo> <lineno/linerange>`	
`HostID`	Host identifier number.
`tmsSpanNo`	TMS span number for DSSI Mode.
`lineno/linerange`	Phone lines to be traced are specified as individual lines or a range. A linerange is a sequence of consecutive numbers. Lineno represents a single phone line number. Linerange denotes a sequence of lines and is represented as two numbers separated by a dash.
Examples	DEBUG:root/mps/cc> block 1 1 1-20 (Blocks lines 1-20 on TMS 1 of component 1.) DEBUG:root/mps/cc> unblock 1 1 1-20 (Unblocks the previously blocked lines.)
createpool	Creates the specified application pool.
`createpool <pool>`	
`pool`	Name of pool to be created.
pools	Displays the names of available application pools.

pools {list | group | details <pool>}	
list	Displays pool names.
group	Displays the number of active, activepending, inservice, and ready applications by group name.
details	Displays details about the specified pool.

Example:

```
DEBUG:root/mps/cc> pools group
service  numInService numReady numActivePending numactive
prepaid      3           1           1             3
outbound     4           1           2             5
```

numInService	Number of applications that are in the in-service state.
numReady	Number of applications that are in the ready state.
numActive Pending	Number of applications that are in the active pending state.
numActive	Number of applications that are in the active state.

reset	Resets the connections on the specified lines.
reset <HostID> <mpsComponentNo/tmsSpanNo> <lineno/linerange>	
HostID	Host identifier number.
mpsComponentNo/ tmsSpanNo	MPS component number for ASE Mode. TMS span number for DSSI Mode.
lineno/linerange	Phone lines to be traced are specified as individual lines or a range. A linerange is a sequence of consecutive numbers. Lineno represents a single phone line number. Linerange denotes a sequence of lines and is represented as two numbers separated by a dash.

DEBUG:root/mps/register

The commands in this section are stored in DEBUG:root/mps/register directory. These commands perform operations related to SIP registration.

register deregister	Issues a SIP register or deregister request for the specified service.

`{ register | deregister} <service>`	
`service`	Service to be registered or deregistered.

passwd	Performs register password maintenance.
`passwd {new | modify | delete}`	
`new`	Creates a new password entry.
`modify`	Modifies an existing password entry.
`delete`	Deletes an existing password entry.

registrations	Lists the currently registered SIP services.

Examples	`DEBUG:root/mps/register/> registrations`
	`==========`
	`From:mps5@avaya.com`
	`Address Of Record:mps5@avaya.com`
	`Contacts:`
	`m:<sip:mps5@47.185.24.181:5060>;expires=0*`
	`Expires:0`
	`Registrar : SS1`
	`Register Session State : Register Null`
	`Register Status : DeRegistration Success`
	`Listers : 1`
	`==========`
	`From:mps1@avaya.com`
	`Address Of Record:mps1@avaya.com`
	`Contacts:`
	`m:<sip:mps1@47.185.24.181:5060>;expires=60000*`
	`Expires:60000`
	`Registrar : SS1`
	`Register Session State : Register Success`
	`Register Status : Registration Success`
	`Listers : 1`

Note:

See Command, Control, and Status on page 182 and Message Sequence Diagrams on page 183 for more information about application service registration.

DEBUG:root/mps/mg

The commands in this section are stored in the DEBUG:root/mps/mg directory. These commands provide status information related to the gateways.

gwinfo	Displays information for the configured gateways.

Displays the gwid (gateway ID), host, port, protocol, chassisID.slotID, numconn (number of connections). This command does not take any arguments.	
conn	Displays information about the connections for the configured gateways.
Displays GatewayID, EndPointID, State (connection state), GlobalCallID, and ConnectionID.	
`conn {list | callid <call_id> | id <gw_id> <ep_id>}`	
`list`	Displays all details of all connections.
`callid`	Displays connection information for the specified callid.
`id`	Displays the details for the connection specified by the included gateway ID and endpoint name.
ep	Displays information about the endpoints of the configured gateways.
Displays GatewayID, EndpointID, state (inservice, out of service), and HasConnection (true, false).	
`ep {list | id <gw_id> <ep_id>}`	
`list`	Displays status information for all endpoints.
`id`	Displays status information for the connection specified by the included gateway ID and endpoint name.

procmon

The process manager on the CCSS system is PROCMON. It is responsible for monitoring the starting and stopping of all software processes. In cases where a software crash occurs, procmon restarts that process.

The PROCMON process checks whether the environment variable CCSSNOLOOP exists and if the file that it references is executable. If the file is executable, upon termination of the process, procmon runs the file.

If CCSSNOLOOP is not set or if the file is not executable, the normal PROCMON functionality is invoked, where a limited number of process restarts will take place in any hour. If the process fails more than the number of times set by the MAX_RESTARTS parameter (by default, set to 20), it will not be restarted.

Note:

If the process to be monitored is ccss_stat, the CCSSNOLOOP check is not made and the normal functionality of procmon is invoked.

Example of normal start and stop of process

```
[01/29/03
 08:47:21]procmon.14715: INIT
[01/29/03
 08:47:21]procmon.14716: START [/opt/java/bin/java]
[01/29/03
 08:47:54]procmon.14715: KILLING [/opt/java/bin/java]
[01/29/03
 08:47:54]procmon.14715: EXIT
```

Example of crash and restart of process

```
[01/29/03
 10:16:11]procmon.15327: INIT
[01/29/03
 10:16:11]procmon.15328: START [/opt/java/bin/java]
[01/29/03
 10:31:51]procmon.15327: ABNORMAL EXIT [/opt/java/bin/java] signal 11
core: N
[01/29/03
 10:31:51]procmon.15367: RESTART [/opt/java/bin/java]
```



In a Windows 2000 system, the user will use the SRP process to start and stop the service.

Using Snoop to Verify SIP Packets

The snoop utility is an interactive program by Sun Microsystems for capturing network packets. Packet data can be displayed in real time or saved to a file. The data can be captured as one-line summaries or with verbose details. The summaries show only the data for the highest level protocol. If one of the verbose options is used, the underlying RPC, UDP, IP, and Ethernet frame information is also displayed.

See the Sun Microsystems documentation for complete information about snoop.

Sample snoop sessions

Sample session using port 5060:

```
root@tcctx01 {68} ifconfig -a
```

Sample snoop sessions

```
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
inet 192.168.101.200 netmask ffffff00 broadcast 192.168.101.255
ether 8:0:20:a7:3b:8c
qfe2: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
inet 47.185.16.26 netmask ffffff00 broadcast 47.185.16.255
ether 8:0:20:a7:3b:8c
qfe3: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
inet 192.84.103.200 netmask ffffff00 broadcast 192.84.103.255
ether 8:0:20:a7:3b:8c
ipdptp0: flags=10008d0<POINTOPOINT,RUNNING,NOARP,MULTICAST,IPv4> mtu 8232
index 5
inet 192.84.100.1 --> 192.84.100.2 netmask ffffff00
ether 0:0:0:0:0:0
root@tcctx01 {69} snoop -d qfe2 -x 42 udp port 5060
Using device /dev/qfe (promiscuous mode)
tcctx01 -> chunyang-2.us.avaya.com UDP D=5060 S=5060 LEN=430
0: 494e 5649 5445 2073 6970 3a63 6875 6e79    INVITE sip:chuny
16: 616e 6740 3437 2e31 3834 2e32 3331 2e32    ang@47.184.231.2
32: 3338 3a35 3036 3020 5349 502f 322e 300d    38:5060 SIP/2.0.
48: 0a76 3a53 4950 2f32 2e30 2f55 4450 2034    .v:SIP/2.0/UDP 4
64: 372e 3138 352e 3136 2e32 363a 3530 3630    7.185.16.26:5060
80: 0d0a 663a 3c73 6970 3a6d 7073 3140 6e6f    ..f:<sip:mps1@no
96: 7274 656c 6e65 7477 6f72 6b73 2e63 6f6d    rtelnetworks.com
112: 3a35 3036 303e 3b74 6167 3d68 3031 6730    :5060>;tag=h01g0
128: 316c 3030 6330 310d 0a74 3a3c 7369 703a    1l00c01..t:<sip:
144: 6368 756e 7961 6e67 406e 6f72 7465 6c6e    chunyang@avaya
160: 6574 776f 726b 732e 636f 6d3a 3530 3630    .com:5060
176: 3e0d 0a69 3a30 3030 3030 3030 3035 3936    >..i:00000000596
192: 4534 4131 4343 3030 3030 3030 3030 3040    E4A1CC000000000@
208: 434c 4945 4e54 0d0a 4353 6571 3a31 2049    CLIENT..CSeq:1 I
224: 4e56 4954 450d 0a6d 6178 2d66 6f72 7761    NVITE..max-forwa
240: 7264 733a 3230 0d0a 633a 6170 706c 6963    rds:20..c:applic
256: 6174 696f 6e2f 7364 700d 0a6d 3a3c 7369    ation/sdp..m:<si
272: 703a 3437 2e31 3835 2e31 362e 3236 3a35    p:47.185.16.26:5
288: 3036 303e 0d0a 6c3a 3131 390d 0a0d 0a76    060>..l:119....v
304: 3d30 0d0a 6f3d 7463 6374 7830 3120 3020    =0..o=tcctx01 0
320: 3020 494e 2049 5034 2074 6363 7478 3031    0 IN IP4 tcctx01
336: 2f34 372e 3138 352e 3136 2e32 360d 0a73    /47.185.16.26..s
352: 3d73 6573 7369 6f6e 0d0a 633d 494e 2049    =session..c=IN I
368: 5034 2034 372e 3138 352e 3136 2e32 3431    P4 47.185.16.241
384: 0d0a 6d3d 6175 6469 6f20 3430 3030 2052    ..m=audio 4000 R
400: 5450 2f41 5650 2030 0d0a 613d 7074 696d    TP/AVP 0..a=ptim
416: 653a 3330 0d0a e:30..
```

To capture SIP packets into a file: root@tcctx01 {69} snoop -d qfe2 -o /tmp/snooptrace udp port 5060

To capture SIP details into a file: root@tcctx01 {69} snoop -i /tmp/snooptrace -x 42



The freeware utility ethereal can also be used to read snoop-formatted data. (See http://www.ethereal.com.)

Chapter 8: Application Data Structures

This chapter covers:

1. MPS Developer Datacards and Folders
2. Data Structures for ASE and TMS Modes
3. Condition Data

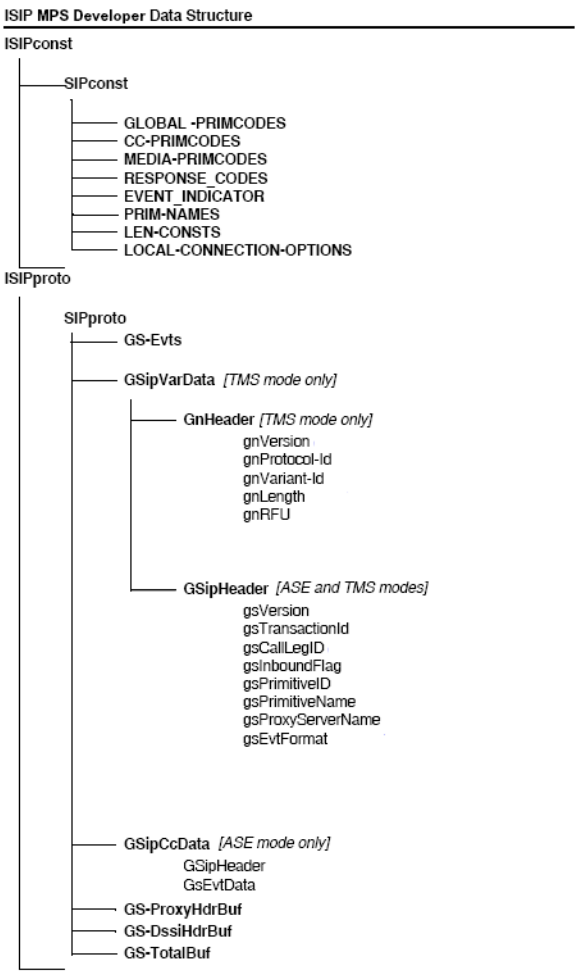
Application Data Structures

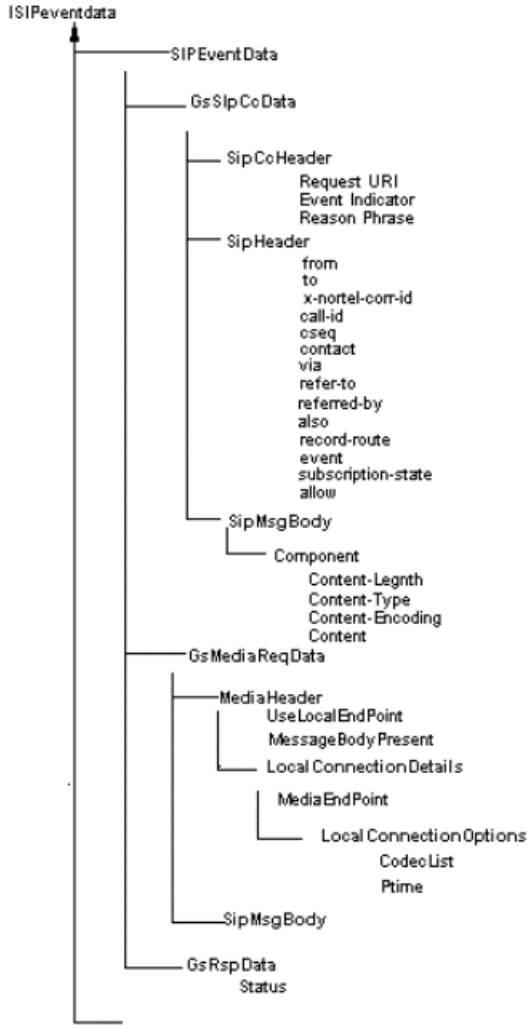
This chapter discusses the MPS Developer folders and datacards used by SIP applications. Most data structures are common to both the ASE and TMS integrated modes, and these are documented early in this chapter. See Additional folders for TMS Integrated Mode on page 134 for discussions of folders and data cards specific to the TMS mode.

ASE versus. TMS mode	ASE Mode	TMS Integrated Mode
MPS Developer programming	SIP toolkit for call control. MPS toolkit for media processing.	Standard MPS toolkit.
MediaXact C API	Not supported	Supported
Media channel binding	Dynamic	Static
Protocol-specific data	SIP folder	Var_data

MPS Developer SIP Data Structure

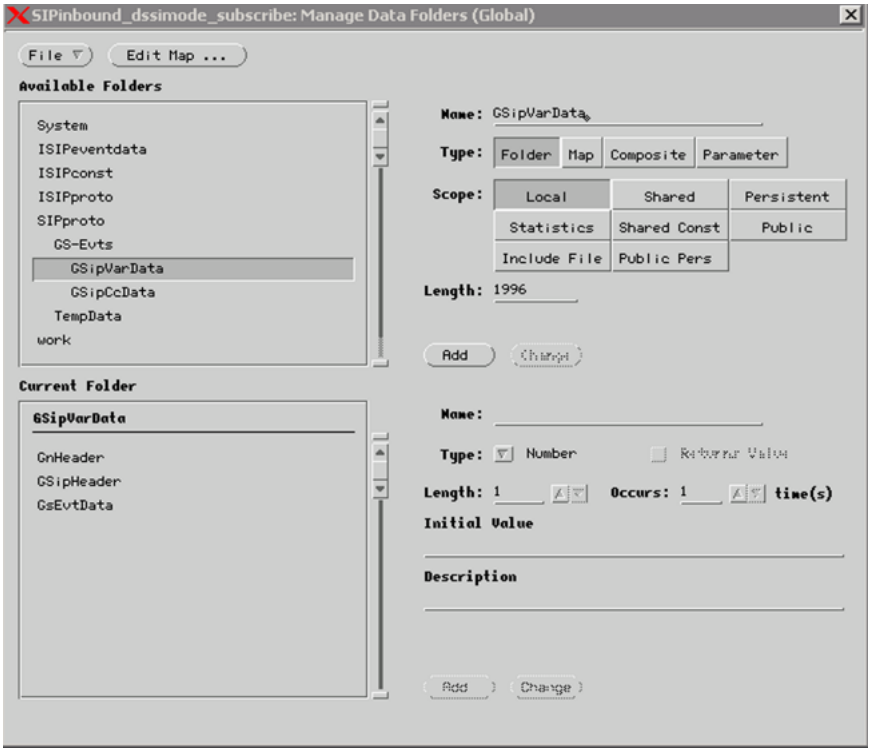
All programming data related to SIP applications functions are stored in subfolders beneath the ISIPconst, ISIPproto, and ISIPeventdata folders.


ISIP MPS Developer Data Structure
ISIPconst
SIPconst
GLOBAL -PRIMCODES
CC-PRIMCODES
MEDIA-PRIMCODES
RESPONSE_CODES
EVENT_INDICATOR
PRIM-NAMES
LEN-CONSTS
LOCAL-CONNECTION-OPTIONS
ISIPproto
SIPproto
GS-Evts
GSipVarData [TMS mode only]
GnHeader [TMS mode only]
gnVersion
gnProtocol-Id
gnVariant-Id
gnLength
gnRFU
GSipHeader [ASE and TMS modes]
gsVersion
gsTransactionId
gsCallLegID
gsInboundFlag
gsPrimitiveID
gsPrimitiveName
gsProxyServerName
gsEvtFormat
GSipCcData [ASE mode only]
GSipHeader
GsEvtData
GS-ProxyHdrBuf
GS-DssiHdrBuf
GS-TotalBuf



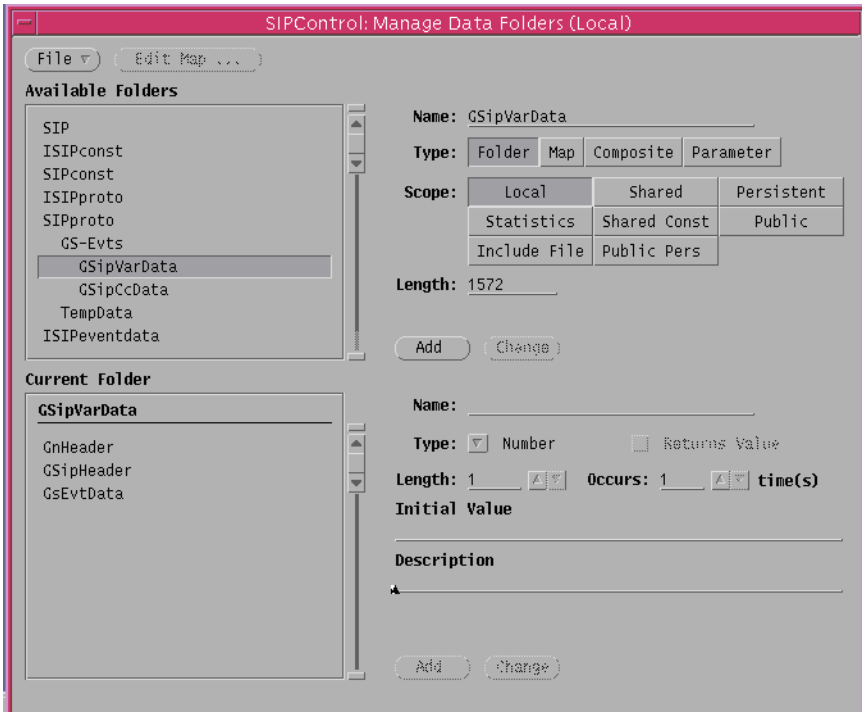
SIPconst

The ISIPconst SIP toolkit include folder, when included in the application, loads the SIPconst folder. The SIPconst folder consists of subfolders that contain data cards with constant values. These constants are used to represent primitive codes, data lengths, and the default values for the local connection options.



SIPproto

The ISIPproto SIP toolkit include folder , when included in the application, loads the SIPproto folder. The SIPproto folder contains the GS-Evts folder. The GS-Evts folder is a composite folder that contains the data received as condition data or vardata over the network, which have a header and data part.



GSipHeader

The SIP message header part contains the SIP primitive type, version, and the message body. The fields in the SIP message header are shown in the following table.

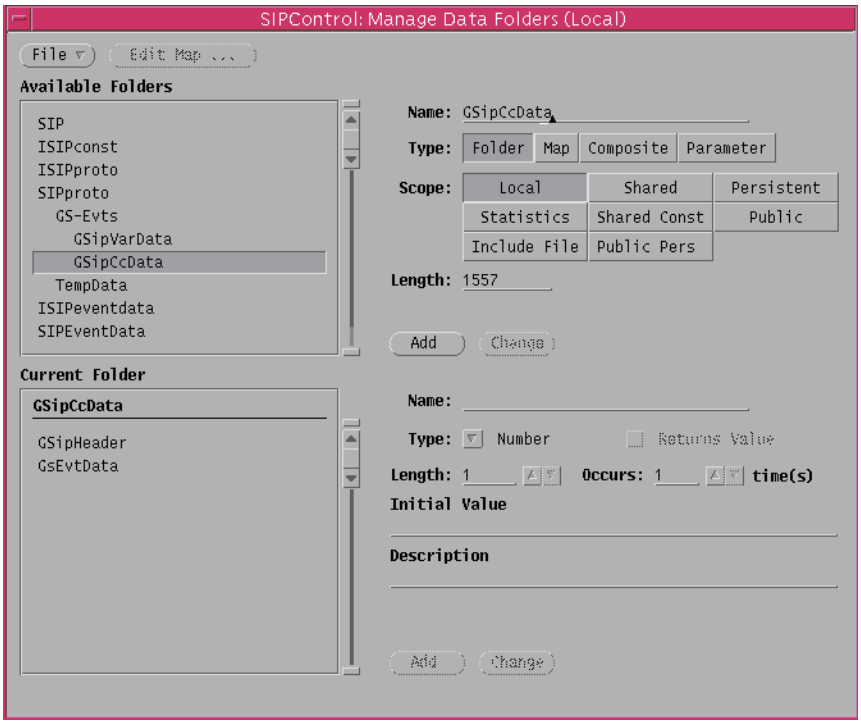
Folder Name	Description	Type	Length
gsVersion	Version of this data structure. "1" is the current version.	N	2
gsTransactionId	Transaction ID. Send Resource requests and OC conditions include a transaction ID. An OC condition delivered to the application in response to a SndRsc request has the same transaction ID as that SndRsc request. Otherwise the transaction ID is 0.	N	4
gsCallLegID	Call leg Identifier (00 to 99). 00 is reserved for call leg independent events.	N	2
gsInboundFlag	Used for getcallleg in ASE mode.	N	1
gsPrimitiveID	Specifies the code for the particular SIP message (Register, Connect, ConnectAck, Progress, Release,	N	3

Folder Name	Description	Type	Length
	ReleaseAck, MediaAttachReq and so on.) See Application API Primitives on page 125 for more information.		
gsPrimitiveName	Primitive name.		
gsProxyServerName	Proxy server name, to which the message is intended.	C	25
gsEvtFormat	Event Data Format (0 – AVString)		
gsEvtDataLength	EventDataLength		
gsEvtData	EventData. This holds either The Call control Event Data or Media Event Data depending on the PrimitiveId.		

SIP protocol-specific data is maintained in folders and datacards of the ISIPeventdata SIP toolkit include folder. This folder loads the SIPEventData folder that contains the following folders and datacards . The information in the folder is converted to AVS format before passing it through vardata.

SIP-Specific Folder/Datacard	Length
SIPEventData	
GsSipCcData	5630
SipCcHeader	515
RequestURI EventIndicator ReasonPhrase	512 3 64
SipHeader	3884
from to via call-id cseq contact refer-to referred-by also x-nortel-corr-ID allow record-route event subscription-state	512 512 512 100 100 512 512 512 512 100 128 512 128 128
SipMsgBody	1229
Component	805
content-length content-type content-encoding content	5 100 100 1024

GSipCcData



GsSipCcData

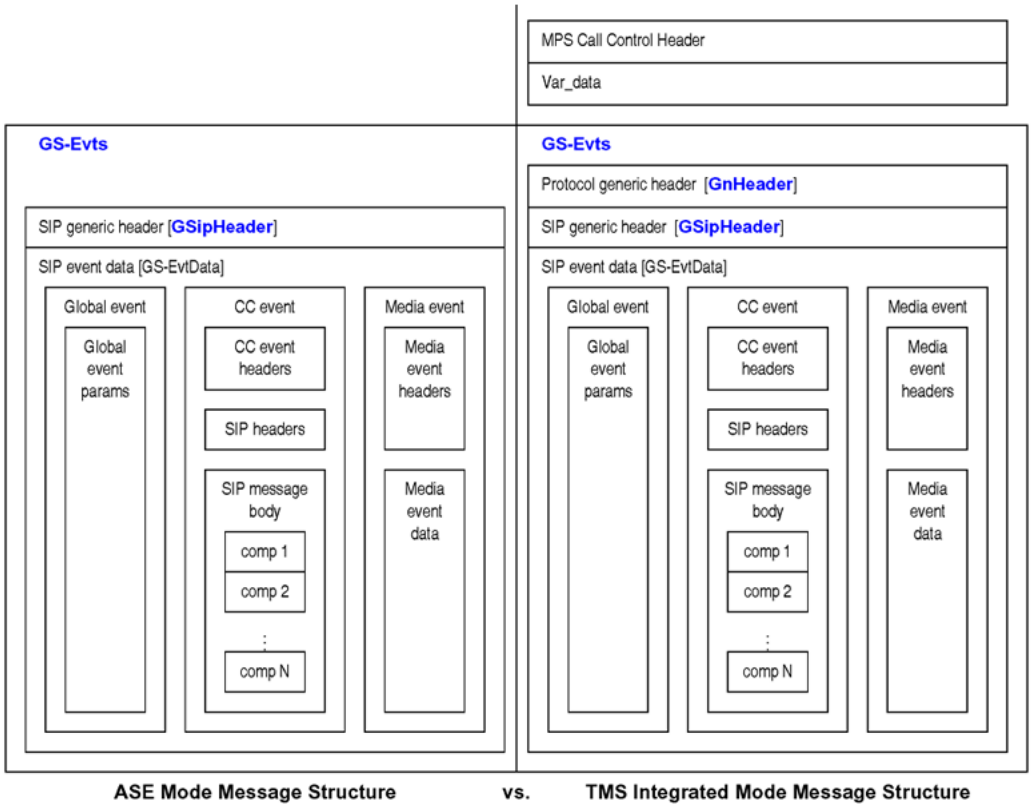
A SIP message is either a request from a client to a server, or a response from a server to a client. Both request and response messages use the generic-message format, which consists of a start-line, one or more header fields (referred to as headers), an empty line (that is, a line with nothing preceding the carriage-return/line-feed indicating the end of the header fields), and an optional message-body.

There are three types of SIP event data delivered to applications:

- Call Control Data – Call Control primitives carry this data.
- Media Data – Media control primitives carry this data.
- Response Data – Returns the media response with operational failure/success indications using the SIPOC and SIPOF conditions.

Each of these event types has a unique format and data structure as shown in the following diagram, and discussed on the next set of pages.

In TMS mode, the protocol generic header identifies the protocol (and variant) in the var_data of generic events. (ASE mode does not use the protocol generic header.)



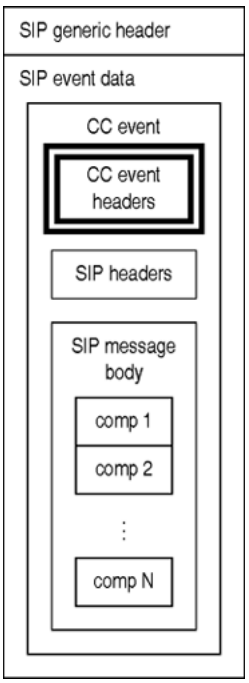
Note:

Currently, AVS is the only available format.

See #DCAM-3428884/DCAM-3428888 for additional information and an example message.

SipCcHeader

For both the TMS and ASE modes, SIP Call Control data headers are used in both request and response messages. The SIP CC Event Header part carries the SIP message start line, which contains the RequestURI field for the requests and EventIndicator status code for the response messages.



Field Name	Description	Type	Length
RequestURI	Specifies the user or service to which the request is intended. Typically, this matches the value in the To field, but unlike the To field, the RequestURI may be rewritten by the proxies. For example: `Sip: 974658365638@dal001.dal; user=phone`	C	50
EventIndicator	One of the following values: 200,1xx, 4xx, 5xx, 6xx. This is based on the following table. 000 is used for not applicable cases (for example, ReleaseAck/Setup will have 000 in the EventIndicator).	N	3
ReasonPhrase	Specifies the reason phrase for response messages. Supported response messages are 1xx,2xx,3xx,4xx & 5xx. Example: SIP/ 2.0 403 Forbidden.	C	64

An overview of the status codes is provided in the following table. The first digit of the status code defines the class of response. The last two digits do not have any categorization role. SIP allows six values for the first digit.

Status Codes	Class of Response	Description
1xx	Informational	Request received. Continuing to process the request.
2xx	Success	The action was successfully received, understood, and accepted.
3xx	Redirection	Further action has to be taken to complete the request.
4xx	Client Error	The request contains bad syntax or cannot be fulfilled at this server.
5xx	Server Error	The server failed to fulfill an apparently valid request.
6xx	Global Failure	The request cannot be fulfilled at any server.

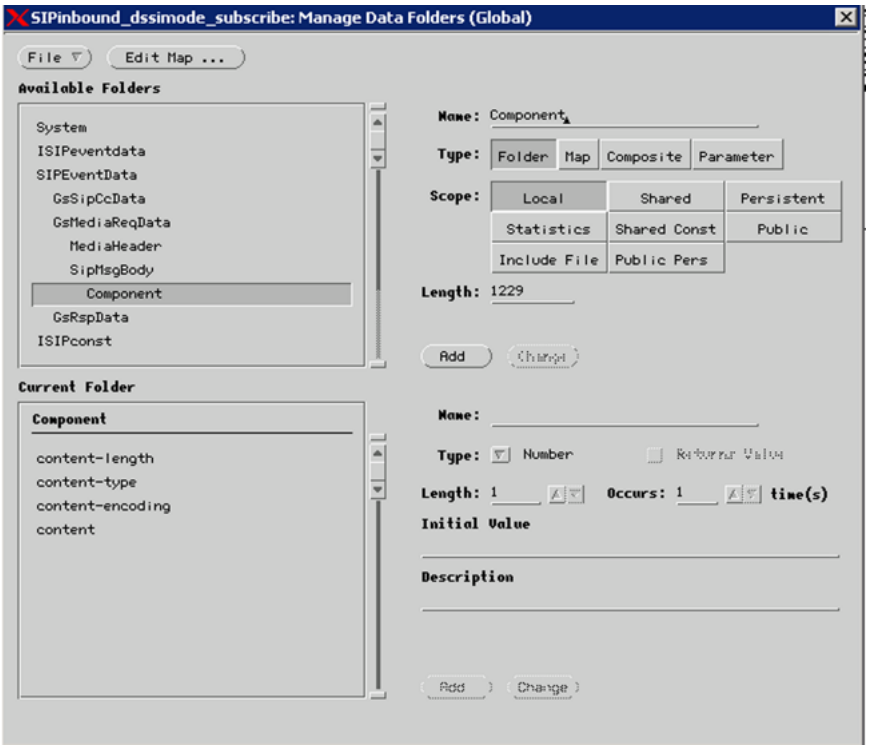
SipHeader

The SipHeader data folder contains data cards used to provide header information for call control messages.

SIP header field values are stored in the data cards From and To as a whole string value. Application should parse the text to access the values.

Similarly for outgoing calls fields (such as From, To, and so on) accept values as string and sent the SIP server.

The following image shows the GsSipCcData and the message body definition in the folder format.



The SIP headers part of the call control event data has certain fields that are the same as the "longer form" for the SIP header field name. Some of the fields are managed and controlled

by the SIP server and cannot be manipulated by applications. The following table identifies the fields in the SIP header structure.

Header Field	Description

From	Address of the request originator. This field is present in all SIP request and response messages.
	`<sip:bob@avaya.com>;tag=1`
	`<sip:2398230948@dal001.dal;user=phone>`

To	Intended destination of a request. This field is present in all SIP request and response messages. Typically, this value matches that of the Request URI. `<Sip:974658365638@dal001.dal;user=phone>`
x-nortel-corr-id	Enables correlation of billing information. `x-nortel-corr-id="x- nortel-corr-id":"bill-track-id bill-track-id=1*alphanum` 000000002D075BCD15P000A0001
Via	Records the route of a request to allow an intermediary SIP server to forward the replies along the same path. `SIP/2.0/UDP here.com:5060`
Call-Id	Identifies the call. A new call ID is generated for each call. `12345601@here.com`
Cseq	Unsigned sequence number and method name. Within a call, this sequence number is incriminated for each new request. Every request is required to have this field. `1 INVITE`
Contact	General header field providing a URL where the user can be reached for further communications. `"Mr. Watson"< sip:watson@worcester.bell-telephone.com> ;q=0.7`
referred-by	The referred-by field is optional and it is used to indicate the URL of requester. It is used in Consult Transfer. `<sip:mps2@avaya.com>`
also	Used to provide bye-also functionality for a blind transfer where an outbound call is attempted to the URL in the also field. `<sip:mps1@avaya.com>`

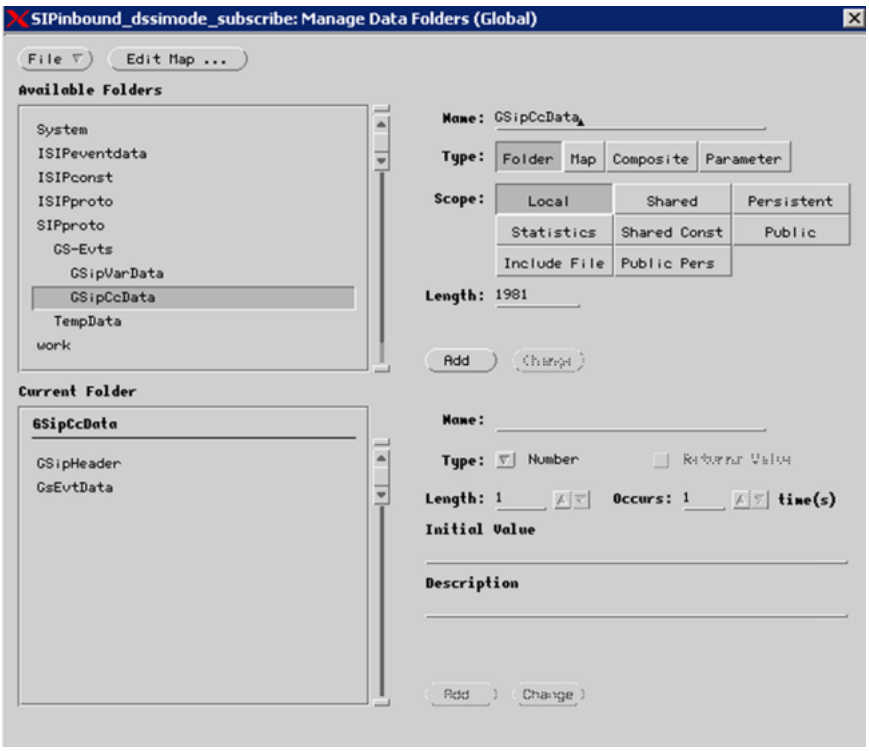
Header Field	Description
refer-to	The refer-to request header field, which only appears in a REFER request, provides a URL to reference. It is used in Consult Transfer. `<sip:mps2@nt.com>`
record-route	Routing header field contains an URI to resolve hostmname and IP address of proxy. This field may appear in INVITE receilved from SIP Proxy. `<sip:1234@47.185.16.74:5060;tag=0000000002CE979EED000000000>`
event	Indicates the type of state for which a subscription is requested. SUBSCRIBE request must include one Event header.. `"message-summary"`
subscription-state	Identifies the subscription state on a successful authentication for a specific event. `"active"`
allow	The Allow header field lists the set of methods supported by the UA generating the message. `INVITE, ACK, CANCEL, BYE`

SipMsgBody

Some request messages include a message body. (The CANCEL request, for example, must not contain a message body.) Response messages may include a body.

The SIP message body carries only text-based content. Binary data is not supported.

For the simple content, the content details type, length information, and the content is available in the data structure is shown in the following image.

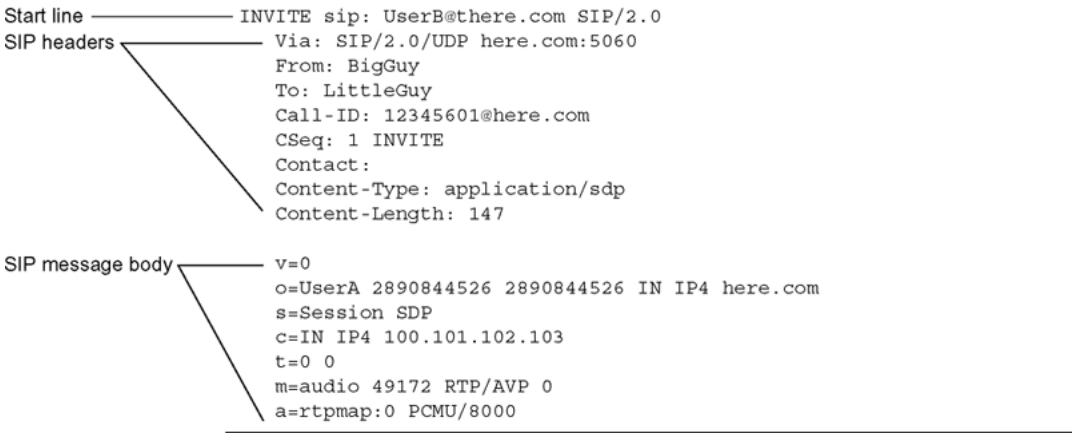


Component Header Field	Description
Content-Type	Describes the media type of the message body (for example, SDP, text).
Content-Encoding	Indicates the type of encoding for the message body. This field does not exist unless the body has been encoded.
Content-Length	Number of octets of the message body.
Content	Message body content (SDB byte stream).

Simple Content

The content type of the SIP message body can be either simple or multipart content. Simple content has one set of data as shown in the following sample SIP invite message:

Sample SIP Invite Message



Multipart Content

For multipart content, the data structure for the message body is repeated (that is, it occurs) multiple times for each component, such as in the following example.

Sample Multipart Message

```
From: Nathaniel Borenstein
To: Ned Freed
Date: Mon, 22 Mar 1993 09:41:09 -0800 (PST)
Subject: Formatted text mail
MIME-Version: 1.0
Content-Type: multipart/alternative; boundary=boundary42
--boundary42
Content-Type: text/plain; charset=us-ascii
```

[plain text version of message]

```
--boundary42
Content-Type: text/enriched
```

[RFC 1896 text/enriched version of same message]

```
--boundary42
Content-Type: application/x-whatever
```

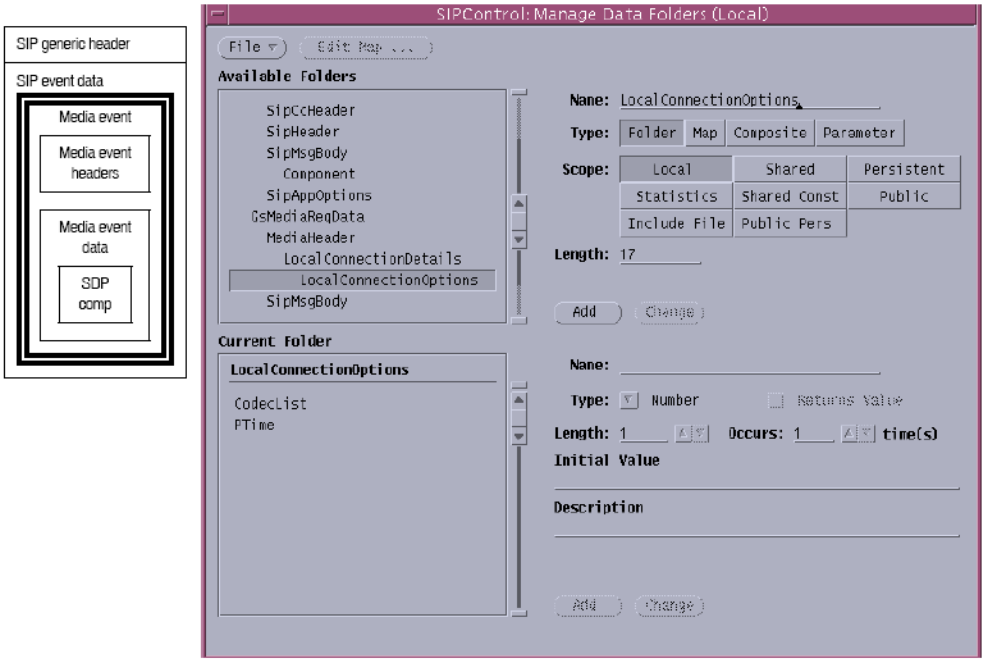
[fanciest version of same message goes here]

```
--boundary42--
```

Note:

Support for multipart content is a future enhancement. It is not available as of this writing.

Media Control Event Data



The following table describes the folders and fields for Media Request operation data.

Field Name	Description	Type	Length
MediaHeader	Specifies local connection information. See the following table.	F	
UseLocalEndPoint	Specifies whether the media command uses the local connection information or external SDP in the Media event. True if Local Media endpoint is used for the operation. False to use the SDP specified in the media event data.	T/F	1
MessageBodyPresent	Indicates whether or not the body is included.	T/F	1
LocalConnection-Details	Describes the local connection details.	F	
MediaEndPoint	LineID of the acquired MediaChannel	C	40

Field Name	Description	Type	Length
	(phone.<line#>.mps.<comp#>.<chassis>.<slot>. <rset>).		
LocalConnection-Options	Folder that contains options pertaining to the local connection. These are the CodecList and Ptime. See the following table.	F	
CodecList	Codec choice of the local connection. The RTP Payload number identifies the codec. The GS_CONST folder defines the default list that are supported.	C	15
Ptime	Packetization time.	N	2
MessageBody	Stores the remote connection information. Same as SIP message body (see #unique_103/unique_103_Connect_42_DCAM-3428887).	F	

SipMsgBody

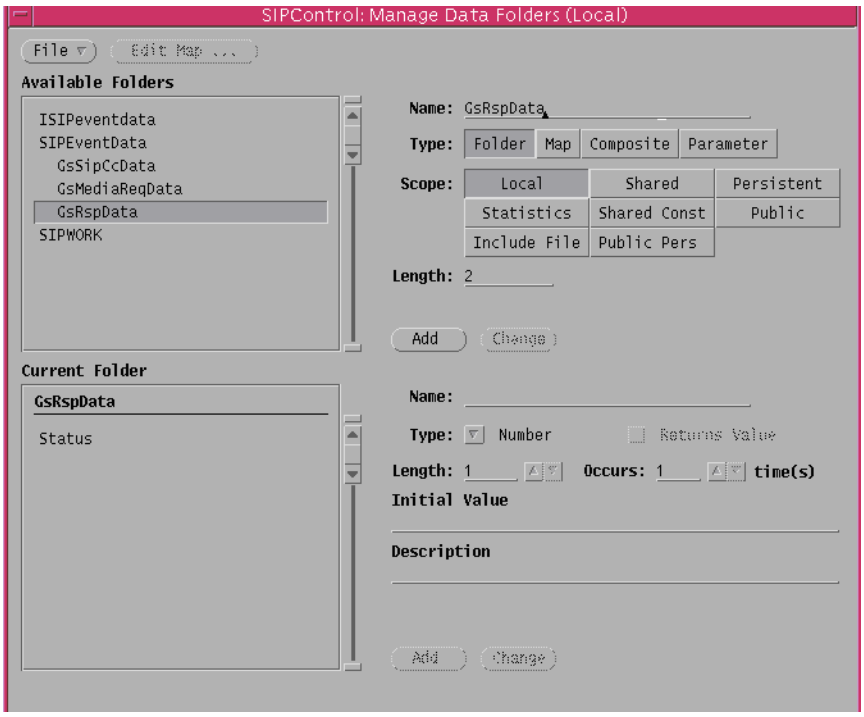
The message body for media requests stores the remote SDP details. It is similar to the SIP message body.

Note:

Component fields for the message body have to be specified for Media Control messages. See Component Header Field Component Header Field.

GsRspData

The response data carries the message transmission status for the messages sent with the Send Resource block.



The OF/SIPOF condition data contains the cause code for the failure. The following table describes these cause codes.

Field Name	Description	Type	Length
Status	Success -- 00 Failed – 01 Rejected – 02 Message transmitted OK – 03 Avs Parsing Error – 04 Invalid call leg Id – 05 Invalid Primitive – 06 Invalid_State_Operation-07 Duplicate_TransId -08	N	2

Application API Primitives

There are three different types of primitives: global, call control, and media control.

Note:

All of the primitives shown in the tables in this section are used in the ASE mode. If a primitive is supported for TMS integrated mode, it is indicated under the heading of TMS.

Global Primitives

Global primitives are issued by applications on an application group basis.

Code	Global Primitive	SIP Command	Description / Direction	Type	TMS
001	[Register]	Register	Registers the application's address/ services to the SIP proxy. *	2	N
002	GenerateCallLeg	n/a	Application requests the SIP server to establish the call leg before making an outbound call or receiving an inbound call. The result returns the generated call leg ID. App → SIP server	2	N
003	GetRegister	n/a	Application request SIP Server to request incoming register. Application uses "registrar" as pool name. Upon receiving SIPOC, Application becomes READY to receive incoming "Register"		N
004	RegisterSuccess	n/a	Application sends Success for incoming register		N
005	RegisterFailure	n/a	Application sends Failure for incoming register		N
Type2:	If waitflag is set, OC/OF is provided to the application with condition data including a status/failure code to indicate the message transmission. Otherwise, SIPOC/ SIPOF is provided to the application with condition data indicating the status/ failure code.				

Note:

* The SIP REGISTER request informs the SIP proxy or SIP redirect server about the address(es) that can be used to reach client SIP call processing applications. This request can also install call handling features on the server. In the current version of CCSS 6.3 SIP,

Register does not support sending register from the application. GetRegister is used to receive the register.

To use an application to register the set of applications in its pool, use IssueGet with the pool name specified in the Mode folder. This causes the SIP server to issue a SIP Register command for the set of applications in the pool. The default pool is any-inservice and for outbound it is outbound.

The GenerateCallLeg function is used to generate a SIP call leg. Applications are capable of having more than one call leg, and operations like bridge, hold, retrieve, and transfer, can be applied to the call legs. A unique ID is used to identify each call leg. This ID is generated by and maintained on the SIP server. Applications intended to make/receive SIP call legs, should generate the call leg first. The following fields are the minimum required fields for generating a call leg.

SIPproto.GS-Evts.GSipCcData.GSipHeader.gsTransactionID <- 123 SIPproto.GS-Evts.GSipCcData.GSipHeader.gsCallLegID <- 0 SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.GLOBAL-PRIMCODES.GETCALLLEGID SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveName < SIPconst.GLOBAL-PRIMNAMES.GETCALLLEGID SIPproto.GS-Evts.GSipCcData.GSipHeader.gsProxyServerName <- SS0 SIPproto.GS-Evts.GSipCcData.GSipHeader.gsInboundFlag <- 1 SIPproto.GS-Evts.GSipCcData.GSipHeader.gsEvtDataLength <- 0

If a callLegID is successfully generated by the SIP server, it is returned to the application as condition data associated with the SIPOC/OC condition.

Call Control Primitives

Call control primitives are used by applications to establish calls and track their progress on a call-leg basis. These primitives perform connection and release operations for inbound, outbound, and transferred calls. Messages that use these primitives internally carry the SIP request and response messages.

Code	Call Control Primitive	SIP Message	Description / Direction	Type	TMS
101	Setup (invite)	INVITE in	Incoming SIP call to the application SIP server → App	1	N
	Setup (Invite)	INVITE out	Outgoing SIP call App → SIP server	2	N
102	SetupRej (4xx, 5xx, 6xx)	SIP response message with	Call rejected App ↔ SIP server	3	N

Code	Call Control Primitive	SIP Message	Description / Direction	Type	TMS
		4xx, 5xx, 6xx status code			
103	Info (Info)	INFO in/out	SIP Info message to/from the Application App ↔ SIP server	3	Y
104	Progress - in /out (180/181/182/183)	180/181/182/183 in/out	Call Progress indication App ↔ SIP server	3	N
105	Connect in/out (OK)	200 OK	Call connected App ↔ SIP server	3	N
106	ConnectAck in/out (Ack)	ACK in/out	Acknowledge the call connection App ↔ SIP server	3	N
107	Release (BYE, 4xx, 5xx, 6xx) in/out	BYE in/out, or 4xx, 5xx, 6xx	Disconnect/release the call App ↔ SIP server	3	N
108	Redirect in/out (3xx Messages)	3xx message	Redirection indications App ↔ SIP server	3	Y
109	Hold	Re-invite with SDP 0.0.0.0	Call leg on hold App ↔ SIP server	3	Y
110	HoldSuccess	200 OK	Response for re-invite with SDP 0.0.0.0 App ↔ SIP server	3	Y
111	HoldAck	ACK	Ack for the re-invite with SDP 0.0.0.0 App ↔ SIP server	3	Y
112	HoldFailed	Reinvite with SDP 0.0.0.0 failure with (4xx,5xx,6xx)	Hold failure App ↔ SIP server	3	Y
113	Retrieve	Invite with local SDP	Retrieve the call leg back to the TPM (from hold or bridge mode) App ↔ SIP server	3	Y
114	RetrieveSuccess	200 OK	Response for invite App ↔ SIP server	3	Y
115	RetrieveAck	ACK	Ack for the Invite App ↔ SIP server	3	Y
116	RetrieveFailure	Reinvite with local SDP	Retrieve failure App ↔ SIP server	3	Y

Code	Call Control Primitive	SIP Message	Description / Direction	Type	TMS
		failure with (4xx,5xx,6xx)			
117	VoiceReroute	Re-Invite with other call legs SDP.	Bridge the voice in the network App ↔ SIP server	3	Y
118	VoiceReroute-Success	200 OK	Response for invite App SIP server	3	Y
119	VoiceReroute-Ack	ACK	Ack for the Invite App ↔ SIP server	3	Y
120	VoiceReroute-Failure	Re-invite failed with response code 4xx, 5xx, 6xx	Voice reroute failure App ↔ SIP server	3	Y
121	DigitMapRequest	Info with digitmap	Arm request to detect a tone App → SIP server	3	Y
122	Digitmap-Success	200	Indicate Arm request success SIP server → App	1	Y
123	DigitmapFailed	5xx/INFO with status failure	Indicate Arm request failure SIP server → App	1	Y
124	DigitmapCancel-Req	INFO with package type cancel in vnd	Disarm request to cancel previous digitmap request SIP server → App	3	Y
125	Digitmap-Cancelled	200	Indicate disarm success SIP server → App	1	Y
126	DigitsNotify	Info with Digits	Carry DTMF digits App ↔ SIP server	3	Y
127	Refer	SIP REFER	Contact a third party App ↔ SIP server	3	Y
128	ReferSucc	-	Indicate refer operation success App ↔ SIP server	3	Y
129	ReferFailed	-	Indicate refer operation failure App ↔ SIP server	3	Y
130	Notify	SIP Notify	Provide status to application sending the REFER. App ↔ SIP server	3	Y
131	NotifySuccess	-	Notify operation success App ↔ SIP server	3	Y
132	NotifyFailure	-	Notify operation failure App ↔ SIP server	3	Y

Code	Call Control Primitive	SIP Message	Description / Direction	Type	TMS
133	Subscribe	SIP SUBSCRIBE	Subscribe for any event in Active Call. App ↔ SIP server	3	Y
134	SubscribeSucc	-	Subscribe operation success. App ↔ SIP server	3	Y
135	SubscribeFail	-	Subscribe operation fail. App ↔ SIP server	3	Y
136	NonCallSubscrib	-	Subscribe in non call scenario. App ↔ SIP server	3	Y
139	Message	SIP MESSAGE	Send Instant Messagell. App ↔ SIP server	3	Y
140	MessageSucc	-	Message operation success. App ↔ SIP server	3	Y
141	MessageFail	-	Message operation fail. App ↔ SIP server	3	Y
Type1:	SIPCC is provided to the application with appropriate condition data.				
Type2:	If waitflag is set, OC/OF is provided to the application with condition data including a status/failure code to indicate the message transmission. Otherwise, SIPOC/SIPOF is provided to the application with condition data indicating the status/failure code.				
Type3:	SIPCC with condition data is provided to the application. If waitflag is set, OC/OF with condition data including a status/failure code to indicate the message transmission. Otherwise, the application receives SIPOC/SIPOF with condition data providing the status/failure code.				

Media Control Primitives

Media Control primitives are used by applications to make requests of the SIP server to establish, close, and modify media paths. These primitives are issued by the application on a call-leg basis.

The application acquires the media channel from the pool. The LineID is retrieved from the condition data of the GETLINECMP condition. The application passes the LineID in the media primitive to the SIP server to create, delete, or modify the media connection.

MGCP Background

The SIP server interfaces with the Media (VoIP) gateway of the system using the Media Gateway Controller Protocol (MGCP). Call control data is managed outside the Media gateways and is handled by the SIP server and the applications.

MGCP uses the Session Description Protocol (SDP) internally to describe the various parameters to establish the connections between two media channels. MGCP has set of commands to control the Media gateway.

MGCP Commands

CreateConnection

This command is creates a connection between two endpoints. A connection is defined by its endpoints. The input parameters in CreateConnection provide the data necessary to build a gateway's "view" of a connection. The application has to specify the media channel it acquired to CreateConnection and LocalConnectionOptions.

LocalConnectionOptions is a parameter used to direct the handling of the connection by the gateway. The fields contained in LocalConnectionOptions are as follows:

- Encoding Method
- Packetization period
- Bandwidth
- Type of Service
- Usage of echo cancellation
- Usage of silence suppression or voice activity detection
- Usage of signal level adaptation and noise level reduction
- Gain control
- Usage of reservation service
- Usage of RTP security
- Type of network used to carry the connection.

Some of the details in the local connection options are defined in the application as folder to select. The remaining details are using the default values in the SIP server.

ModifyConnection

This command is used to modify the data maintained by the VoIP gateway for a connection. This includes both the local and remote connection details.

The ModifyConnection command can do the following:

- Provide the gateway with information about the other end of the connection to activate the connection.
- Activate or deactivate the connection.
- Change the sending parameters of the connection, such as switching coding schemes, changing the packetization period, or modifying the handling of echo cancellation.

DeleteConnection

This command is used to terminate a connection.

Media Control Primitives List

The following table displays the media control primitives for the application. These primitives trigger the SIP server to issue the appropriate media actions. These primitives are not mapped to the MGCP commands.

New primitives (codes 210-215) were added to support the TPM-MRS (Media Relay Server) connection and relay.

The difference between Media Attach Request and Create Connection Request is that Media Attach Request is used for SIP Audio calls. With a typical audio-only call, the SIP server creates a bidirectional relay between the TPM card and the remote by creating TPM and RTP media groups, respectively. The audio coming from the remote is relayed through the RTP media group to the TPM media group and then out to the TPM card. The relay in the opposite direction is used for the audio coming from the TPM card and going out to the remote.

For calls that include video, the remote also has its video media terminate on the same RTP media group. The direction of the video media also applies, but does not derive from the TPM card. Instead video comes from a file, live feed, and so on, and goes out to the remote. Likewise, video from the remote can be recorded out to a file.

When a call arrives at the SIP server, the application exchanges the remote SDP and MRS SDP using MRS API. Also, it is required that the SIP server enables the media connection between the TPM card and the MRS TPM media group audio RTP port. This is required to enable the DTMF detection and any speech recognition operations in MPS Developer applications during the video call.

Create and Delete connection primitives are introduced to perform the media connection between the TPM and MRS TPM media group audio RTP port (or any other SDP provided from the application).

Code	Media Primitive	Description / Direction	Type	TMS
201	MediaAttachReq	Media attach request. The application acquires a media channel from getOutLine and issues this command to create a media path connection. Internally, the MGCP command CreateConnection is sent by the SIP server to the Media gateway to create the connection. The application specifies the local connection options. The rest of the data is supplied by the SIP server using default values. App → SIP server	2	Y
202	MediaAttachSuccess	Media attach operation success. SIP server → App	1	Y
203	MediaAttachFailed	Media attach operation failure. SIP server → App	1	Y
204	MediaDetachReq	Releases a previously acquired media channel. The MGCP command	2	Y

Code	Media Primitive	Description / Direction	Type	TMS
		DeleteConnection is sent by the ConnSvc to the Media Gateway to delete the connection. App → SIP server		
205	MediaDetachSuccess	Media detach operation success. SIP server → App	1	Y
206	MediaDetachFailed	Media detach operation failure. SIP server → App	1	Y
207	MediaModifyReq	Change local connection properties. App → SIP server	2	Y
208	MediaModifySuccess	Media modify operation success. SIP server → App	1	Y
209	MediaModifyFailed	Media modify operation failure. SIP server → App	1	Y
210	CreateConnection-Req	Create connection request. Application acquires a media channel from getLine and issues this command with a SDP and media line, to create a media connection between these two. Internally, the MGCP command, CreateConnection, is sent by the SIP server to the Media Gateway to create the connection. App → SIP server	2	Y
211	CreateConnection-Success	Indicates create connection operation success. SIP server → App	1	Y
212	CreateConnection-Failed	Indicates create connection operation failed. SIP server → App	1	Y
213	deleteConnectionReq	Delete connection request. The MGCP command DeleteConnection is sent by the SIP server to the Media Gateway to delete the connection. App → SIP server	2	Y
214	deleteConnection-Success	Indicates delete connection operation succeeded. SIP server → App	1	Y
215	deleteConnection-Failed	Indicates delete connection operation failed. SIP server → App	1	Y
Type1:	SIPCC is provided to the application with appropriate condition data.			
Type2:	If waitflag is set, OC/OF is provided to the application with condition data including a status/failure code to indicate the message transmission. Otherwise, SIPOC/ SIPOF is provided to the application with condition data indicating the status/ failure code.			

Additional folders for TMS Integrated Mode

In TMS integrated mode, SIP protocol-specific data is passed in the vardata of all indication messages to the DSSI task of the TMS. From there, the data is tunneled to the applications. For incoming calls, the From and To SIP fields are assigned values for ANI and DNIS taken from the setup indication message.

Note:

The ANI and DNIS fields are not URL friendly. These fields support only 36 characters, comprised from the character set shown in the following table.

Example

Character	Description
`0 1 2 3 4 5 6 7 8 9 * #`	Standard keypad digits
`A B C D`	Extended keypad digits
`,`	Long pause
`.`	Short pause
`!`	Hookflash
`W`	Wait for dial tone
`T`	Tone dialing
`P`	Pulse dialing

GSipVarData

SIP protocol data is maintained by a generic header and corresponding folder structure. These are tunneled through vardata to the applications. The generic header data is as follows.

Example

Generic Folder/Datacard		Length
GSipVarData	1996	
GnHeader	15	
gnVersion		2
gnProtocolId		2
gnVariantId		2
gnLength		5
gnRFU		4

GnHeader

The generic protocol header part is used only with TMS integrated mode vardata. The fields in the generic protocol header part are shown in the following table.

Folder Name	Description	Type	Length
gnVersion	Version of this header: 1 - current version	N	2
gnProtocol-Id	Protocol Id: 0 - Unknown and data is not available. 1 - FGD 2 - ISUP 3 - IUP 4 - ISDN 5 - SIP 6 - H.323 7 - R2	N	2
gnVariant-Id	Variation ID of the protocol. Values depend on the protocol ID. • FGD Protocol: 1 - EANA 2 - EAIN • ISUP Protocol: 0 - Default or implied protocol 1 - ANSI 2 - ETSI 3 - ITU • IUP Protocol: 0 - Default or implied protocol • ISDN Protocol:	N	2

Folder Name	Description	Type	Length
	0 - Default or implied protocol 1 - ATT 2 - NT 3 - NISDN 4 - DPNSS 5 - 1TR6 6 - NTT 7 - V5.1 8 - NZ • SIP Protocol: 0 - Default or implied protocol • H.323: 0 - Default or implied protocol.		
gnLength	Length of the ObjectData. This is the size of the attached data, which is the number characters following the protocol generic header.	N	5
gnRFU	Reserved for future use.	N	4

Chapter 9: Application Programming

This chapter covers:

1. MPS Developer Blocks and Windows
2. Call Scenarios for the ASE and TMS Integrated Modes

ASE Mode

Resource Operations

In ASE mode, SIP messages are delivered to the applications through the MPS SIP resource daemon. This daemon in the ASE (Application Service Environment) system is a server process that connects the MPS and the SIP client on the SIP server. The following table explains the VMS/MPS resource operation in details.

The following table explains the resource commands sent from MPS applications to the SIP proxy client, the action taken by SIP proxy client, and the result the applications receive. These resource commands are valid in any machine state. However, the particular results that applications receive depends on the particular state.

Resource Command	SIP Server Action	Reply to the Application
GET RESOURCE	SIP server allocates necessary data structure and resources to handle the application.	If successful, the application receives the GOTRES condition. If the application cannot acquire the resource, it receives the GETFAIL condition.
SEND RESOURCE	Application uses SNDRSC to send the messages to the SIP server. The SIP server sends it to the MPS call model.	If sent successfully, the application receives the SIPOC/OC (output complete) condition. Otherwise, the application receives SIPOF/OF (output fail) with an error code.
FREE RESOURCE	The SIP call is finished and the application frees the resource. The	No reply to the application.

Resource Command	SIP Server Action	Reply to the Application
	SIP server takes action to release the call.	

Messages are delivered to the application with the SIPCC condition. The associated condition data carries the SIP event data. The application must acquire the SIP resource to receive SIPCC condition.

Condition Data Structure

All the SIP conditions carry condition data in the same data structure, which is mapped to application folders. The condition data structure is common to both ASE mode condition data and TMS integrated mode vardata. The data has a header part and data part.

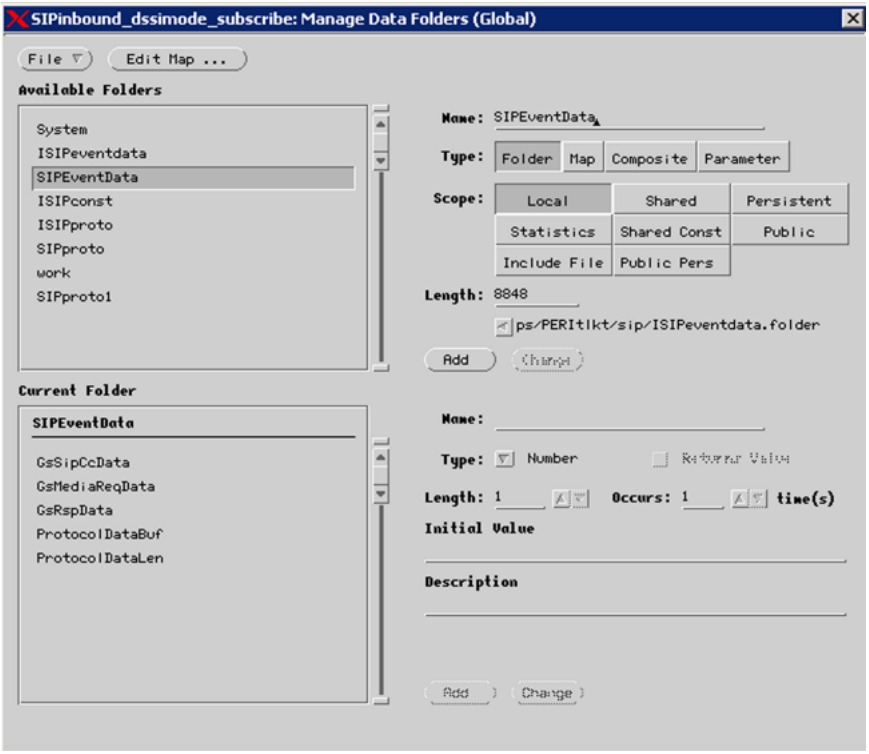
SIP Data Unit	SIP generic header	SIP event data

In the TMS integrated mode, an additional protocol generic header is added to identify the protocol/variant information in the vardata used for generic events.

vardata structure for TMS mode	Protocol generic header	SIP generic header	SIP event data

SIPEventData

The SIPEventData folder has the call control, media, and response data and are shown in the following image.



The application uses the FromAVS operation to parse the AVString and retrieve the values in the folder. The data is sent to the application as AVStrings, as shown in the following table. AttributeValueStrings use the format name: value to pass data to and from folders.

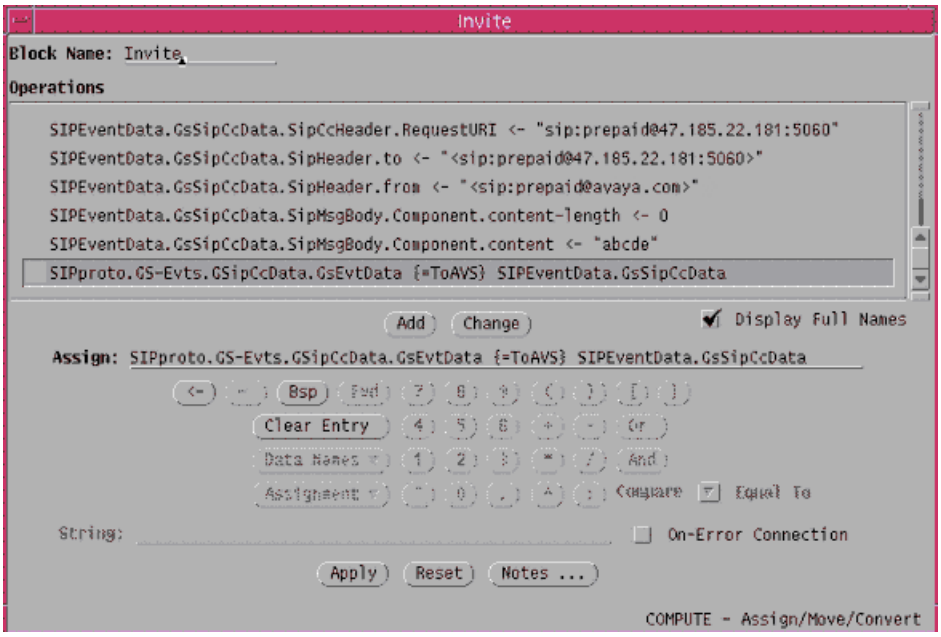
AVString example

```
'gsSipCcData:(SipCcHeader:(RequestURI:"sip:peri@MPS_CC"
EventIndicator:0) SipHeader:(from:"User
Name<sip:user@avaya.com>;tag=1016861634*"
to:"<sip:peri@47.184.236.44>*") SipMsgBody:(Component:(content-length:
230 content:"v=0\0D
o=ramalaks 13845334250949982 13845334250949982 IN IP4
47.184.236.235\0D
s=avaya\0D
p=+1 972 684 1000\0D
c=IN IP4 47.184.236.235\0D
t=0 0\0D
m=audio 50070 RTP/AVP 4 18\0D
a=rtpmap:4 G723/8000\0D
a=rtpmap:18 G729A/8000\0D
a=ptime:30\0D
")))'
```

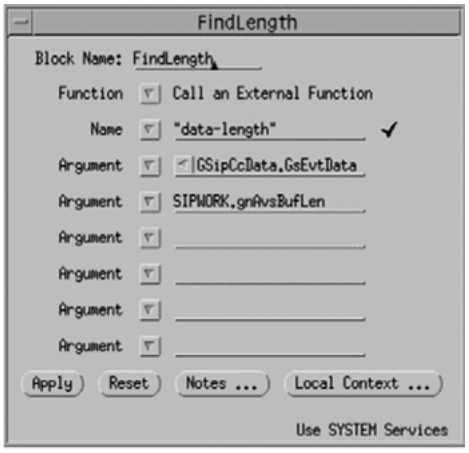
Sending Messages

The following are instructions for sending any message to the SIP server.

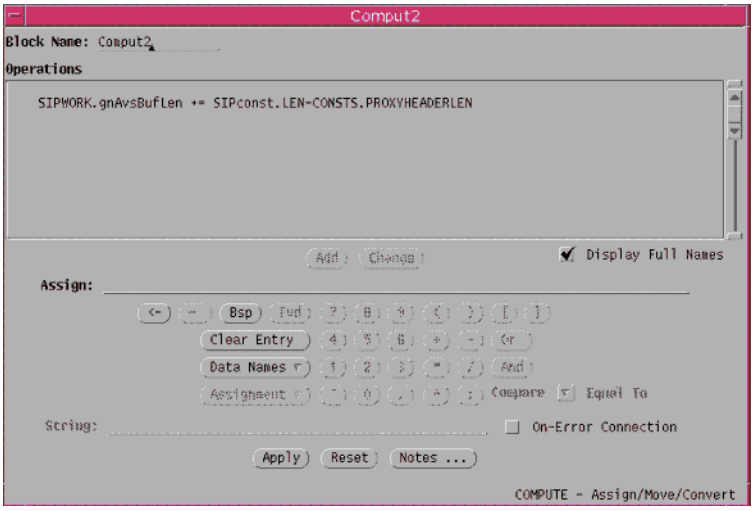
1. Construct the message through the Compute block. Use the ToAvs operation to construct the AVString of the folder SIPEventData.GsSipCcData to the folder GsEvtData.



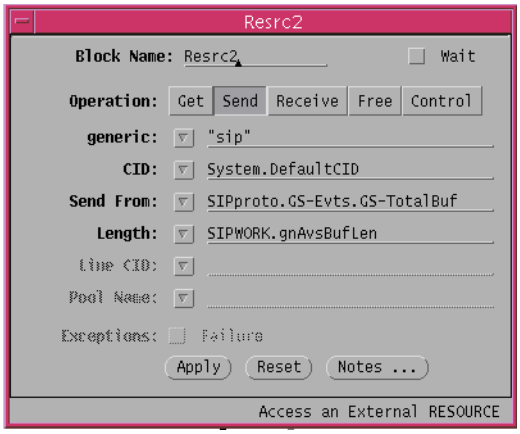
2. Calculate the GsEvtDataLength using the call function data-length.



3. Add the proxy header length to obtain the total message length.

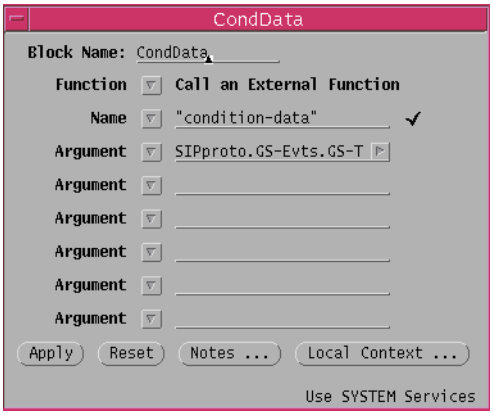


4. Send the message using the SIP proxy. Depending on the setting of the Wait flag, either SIPOC/SIPOF or OC/OF are sent to the application to indicate the message transmission status.



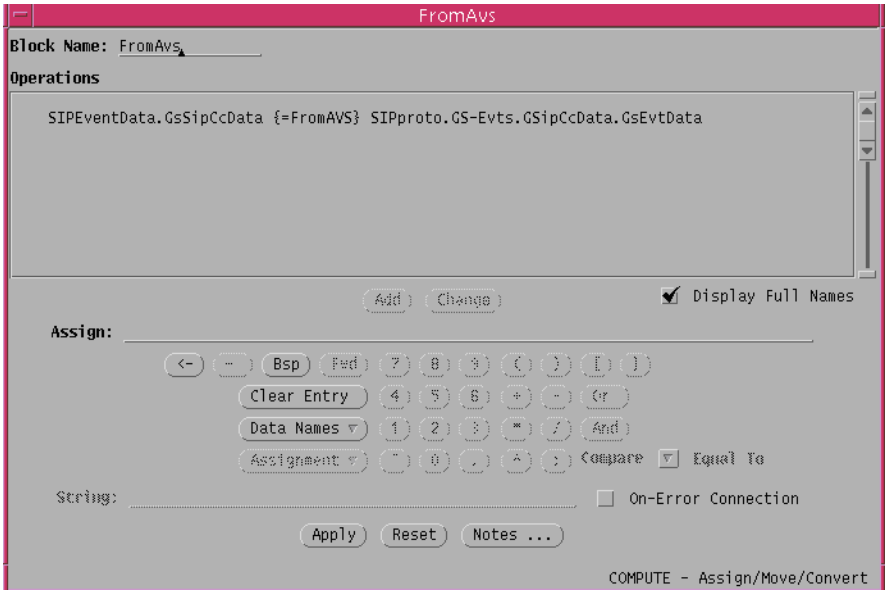
Receiving Messages

Messages are delivered as conditions to the application. The application acquires the condition data when it receives the SIPCC condition.



GS-Evts is a composite folder. The GSipHeader is filled in automatically as this not in AVS format.

The FromAvs function parses the AVString to the folder structure. To obtain SIP message details, FromAvs operations should be performed as shown in the following image.



Basic Call Operations

Make a SIP Call

1. The application acquires a callLeg for making a an outbound call.
2. The application constructs a SETUP message with From, To, RequestURI, and ProxyServiceName fields with proper values and sends the message to the SIP server. The following fields are the minimum required fields used to create the setup from an application.

SIPproto.GS-Evts.GSipCcData.GSipHeader.gsCallLegID <- SIPWORK.CallLegId /* callLegId acquired from the generateCallLeg operation*/ SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.CC-PRIMCODES.SETUP SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveName <- SIPconst.CC-PRIMNAMES.SETUP SIPproto.GS-Evts.GSipCcData.GSipHeader.gsProxyServerName <- "SS1" SIPEventData.GsSipCcData.SipCcHeader.EventIndicator <- 000 /* Not applicable*/ SIPEventData.GsSipCcData.SipCcHeader.RequestURI <- "sip:mps2@avaya.com" SIPEventData.GsSipCcData.SipHeader.to <- "<sip:mps2@avaya.com>" SIPEventData.GsSipCcData.SipHeader.from <-"<sip:peri@47.185.16.72>" SIPproto.GS-Evts.GSipCcData.GsEvtData {=ToAVS} SIPEventData.GsSipCcData

The SIP server constructs the SIP INVITE with proper SDP for the media channel acquired by the application and sends to the SIP Proxy. (See Media Operations on page 144.)

Receive a SIP Call

1. The application acquires a callLeg for receiving an inbound call.
2. The application waits for the SIPCC condition. When received, the application acquires the associated condition data to identify the primitive and decode the event data using the FromAVS operation. (See Receiving Messages on page 141.)

Accepting a SIP call

After receiving SETUP, the application sends CONNECT and waits for ConnectAck to accept a call.

SIPproto.GS-Evts.GSipCcData.GSipHeader.gsCallLegID <- SIPWORK.CallLegId /*callLegId acquired from the generateCallLeg operation*/ SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.CC-PRIMCODES.CONNECT SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveName <- SIPconst.CC-PRIMNAMES.CONNECT SIPEventData.GsSipCcData.SipCcHeader.EventIndicator <- 200

Rejecting a SIP call

The application can reject a call by sending SetupRej. The EventIndicator is used to indicate the reason code.

SIPproto.GS-Evts.GSipCcData.GSipHeader.gsCallLegID <- SIPWORK.CallLegId /* callLegId acquired from the generateCallLeg operation*/ SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.CC-PRIMCODES.SETUPREJ SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveName <- SIPconst.CC-PRIMNAMES.SETUPREJ SIPEventData.GsSipCcData.SipCcHeader.EventIndicator <- 603 /* 4xx,5xx,6xx*/ SIPproto.GS-Evts.GSipCcData.GsEvtData {=ToAVS} SIPEventData.GsSipCcData

Disconnecting the SIP Call

The primitive release is used to disconnect any SIP call leg.

SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.CC-PRIMCODES.RELEASE SIPproto.GS-Evts.GSipCcData.GSipHeader.gsCallLegID <- 1

ISSUE FREE from the application causes the release of all its SIP call legs.

Media Operations

Attaching a Media Channel

As explained earlier, in the ASE Mode, the media channel is acquired and freed dynamically as needed. The applications uses the MPS Developer operations GetLine to acquire the media channel, and FreeLine to free the channel. The MediaAttach primitive is used to attach a media channel to the call leg. MediaAttachRequest requires the LineID of the media line and local connection options (Codec list, Ptime) as input.

The LineID is specified in the format phone.<line#>.mps.<comp#>.<chassis>.<slot>.<rset>, which can be acquired from the condition data associated with the GETLINECMP condition.

The SIPconst folder defines the valid values of the local connection options for the application to use

The following is the sequence for media attach operations.

1. The application acquires a media channel from the pool manager using the getLine function. The LINEID of the media channel is acquired using the condition-data call function.
2. The application constructs the MediaAttach primitive as shown by the following:

   SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.MEDIA-PRIMCODES.MEDIA_ATTACH_REQ SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveName <- SIPconst.MEDIA-PRIMNAMES.MEDIA_ATTACH_REQ SIPEventData.GsMediaReqData.MediaHeader.LocalConnectionDetails.MediaEndpoint LINEID /*eg:LINEID= phone.22.mps.1.1.1.22*/ SIPEventData.GsMediaReqData.MediaHeader.UseLocalEndPoint <- System.True SIPEventData.GsMediaReqData.MediaHeader.LocalConnectionDetails.LocalConnectionOptions.CodecList <- SIPconst.LOCAL-CONNECTION-OPTIONS.CodecList SIPproto.GS-Evts.GSipCcData.GsEvtData {=ToAVS} SIPEventData.GsMediaReqData
3. Upon receiving the MediaAttach information, the SIP server sends MGCP CRCX to create the connection on the media channel specified by the application and maintains the SDP.

Detaching a Media Channel

The MediaDetach function is used to by the applications to request the SIP Server to delete the connection on any media channel. The MediaDetachRequest also requires the LineID as input as shown by the following example:

SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.MEDIA-PRIMCODES.MEDIA_DETACH_REQ SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveName <- SIPconst.MEDIA-PRIMNAMES.MEDIA_DETACH_REQ SIPEventData.GsMediaReqData.MediaHeader.LocalConnectionDetails.MediaEndpoint

LINEID /*eg:LINEID= phone.22.mps.1.1.1.22*/
SIPEventData.GsMediaReqData.MediaHeader.UseLocalEndPoint <- System.True
SIPproto.GS-Evts.GSipCcData.GsEvtData {=ToAVS} SIPEventData.GsMediaReqData

Advanced Operations

Hold/Retrieve a Call Leg

Hold is performed by re-INVITEs with an SDP of zero. Retrieve is performed by re-INVITE with a local SDP. To put a call leg on hold and retrieve it, applications just have to specify the callLegId and the operation code as shown by the following example:

SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.CC-PRIMCODES.HOLD SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveName <- SIPconst.CC-PRIMNAMES.HOLD SIPproto.GS-Evts.GSipCcData.GSipHeader.gsCallLegID <- 1

The SIP server performs the re-invite with SDP 0.0.0.0 for Hold. HoldSuccess and HoldFailure primitives indicate the hold operation's success or failure. The SIP ACK for the re-INVITE is sent to the application as HoldAck.

SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.CC-PRIMCODES.RETRIEVE SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveName <- SIPconst.CC-PRIMNAMES.RETRIEVE SIPproto.GS-Evts.GSipCcData.GSipHeader.gsCallLegID <- 1

The SIP server performs the re-invite with local SDP for retrieving the call leg. RetrieveSuccess and RetrieveFailure primitives indicate the retrieve operation's success or failure. The SIP ACK for the re-INVITE is sent to the application as RetrieveAck.

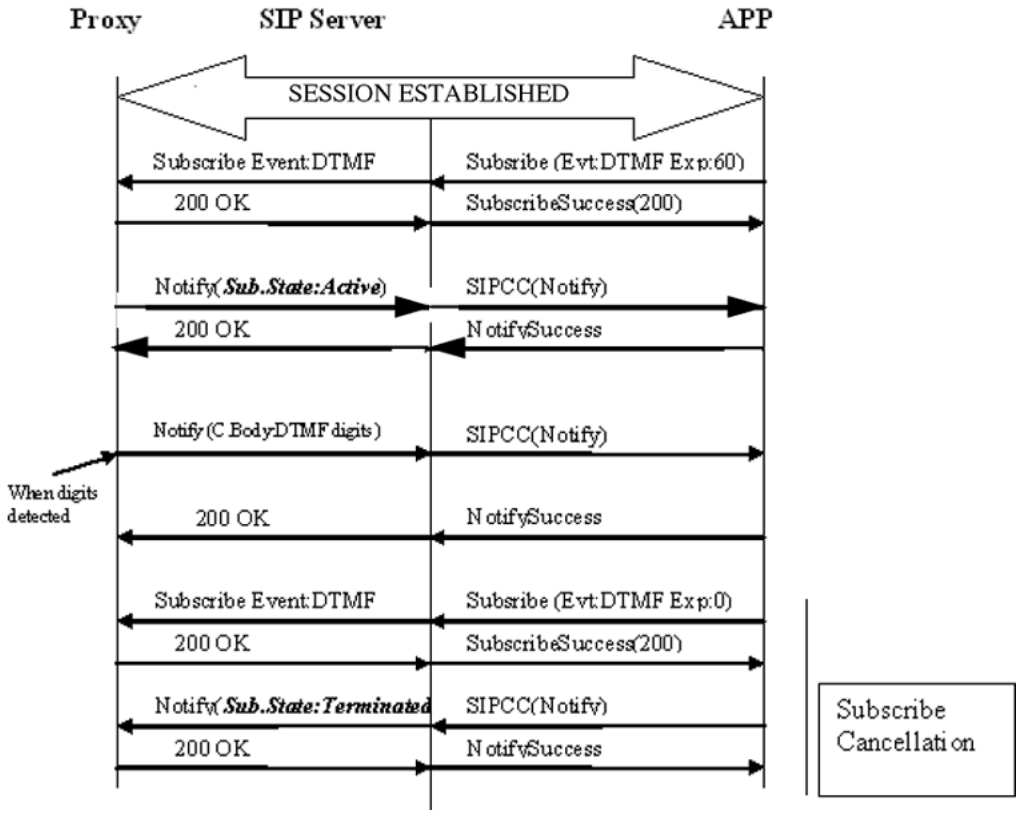
See Hold and Retrieve on page 191 for a call-flow depiction of these operations.

Subscribe/Notify

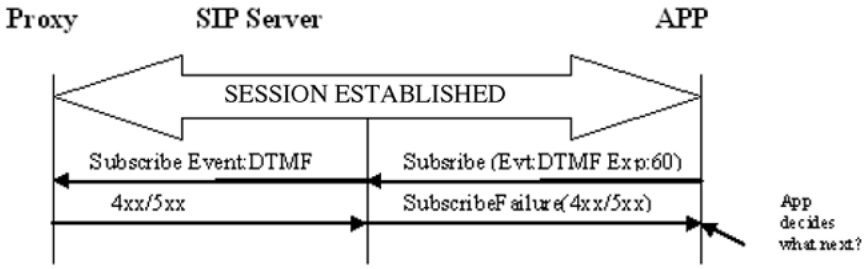
The application subscribes for event notification when DTMF event is detected during the call. Application receives notification when DTMF is detected. Application cancels the subscription when notifications are no longer needed. This functionality is required for prepaid networking with CS2K as the digits are notified with notify method.

The call processing application sends "SUBSCRIBE" and receive "NOTIFY" to detect the DTMF events. A new set of primitives are introduced for the application programming API to subscribe, unsubscribe and receive notify, and to indicate the operation success/failure status. See the following call flow diagrams.

Subscribe/Notify Success Call Flow



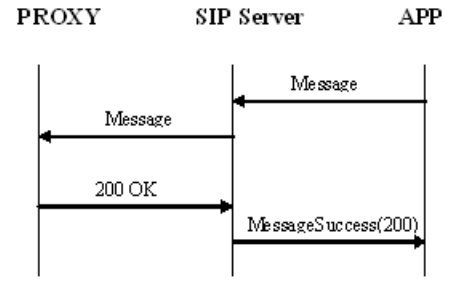
Subscribe Failure Call Flow



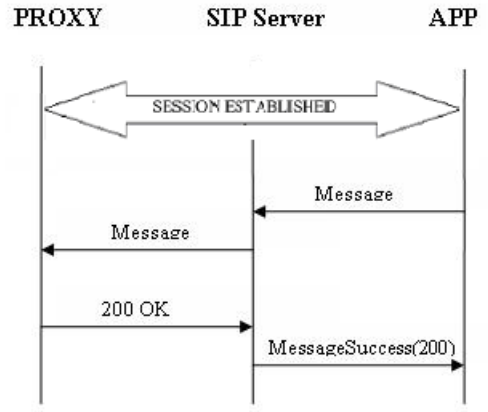
MESSAGE Call flow

MESSAGE is an SIP extension method that allows the transfer of Instant Messages (IM). The MESSAGE request inherits all the request routing and security features of base SIP.

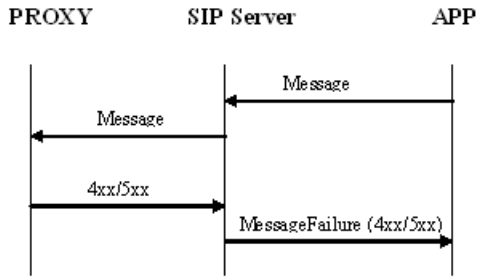
MESSAGE Success Call flow (Non Call Scenario)



MESSAGE Success Call flow (Active Call Scenario)



MESSAGE Failure Call flow



Transfers

There are two types of transfers.

1. In the Blind transfer, the party that requests the transfer is not interested in the outcome of the transfer. This is achieved by sending the BYE (Release) with an Also field indicating the URL of the referred party. Applications that receive a RELEASE with Also field in the SIP header will attempt to make an outbound call to the URL specified in the Also field. Applications can also send RELEASE with Also field to terminate a session having a blind transfer request.

SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.CC-PRIMCODES.RELEASE " SIPproto.GS-Evts.GSipCcData.GSipHeader.gsCallLegID <- 1 SIPEventData.GsSipCcData.SipCcHeader.EventIndicator <- 200 SIPEventData.GsSipCcData.SipHeader.also <- "<sip:mps2@nt.com>"

2. In the Consult transfer, SIP provides a method called REFER to perform the call transfer. The REFER method indicates that the recipient (identified by the Request-URI) should contact a third party using the contact information provided in the request. This also provides a mechanism allowing the party sending the REFER to be notified of the outcome of the referenced request.

The refer-to request header field, which only appears in a REFER request, provides a URL to reference. The referred-by field is optional, and used to indicate the URL of requester.

SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.CC-PRIMCODES.REFER SIPproto.GS-Evts.GSipCcData.GSipHeader.gsCallLegID <- 1 SIPEventData.GsSipCcData.SipCcHeader.EventIndicator <- 000 SIPEventData.GsSipCcData.SipHeader.refer-to <- "<sip:mps2@avaya.com>" SIPEventData.GsSipCcData.SipHeader.referred-by <- "<sip:mps1@avaya.com>"

Notify is used to inform the application sending the REFER request of the status of the reference. The component part of the Notify condition data contains the content type message/sipfrag and the content indicating the status of the refer operation.

See Application Receives Refer on page 197 and Application Sends Refer on page 198 for call-flow depictions of these operations.

Bridge Operations

For an application with two call legs to bridge the voice in the network, the application sends a voice-reroute request with the SDP of the one leg to the other. This triggers the SIP server to send re-INVITEs with the SDP provided by the application.

See Hold and Retrieve on page 191 and Voice Bridge on page 194 for call-flow depictions of these operations.

Info Digits

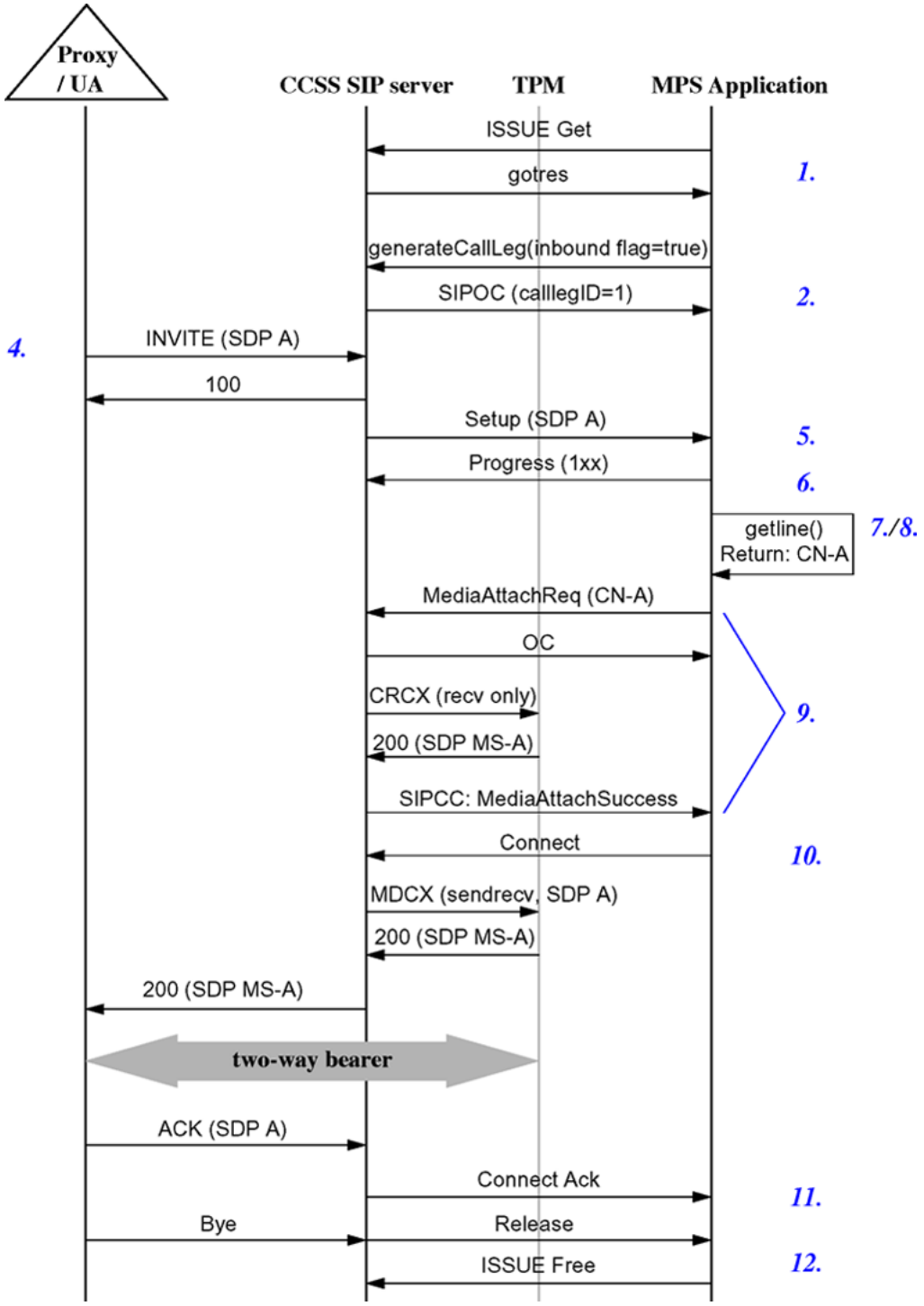
The INFO message is used to carry application-level information along the SIP signaling path. The INFO message can be used for web push, with the URL of the requested page in the contents of the SIP message body, or can be used to send DTMF digits through the signaling path. The DigitNotify primitive is provided for the applications to send SIP INFO with Digits.

SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID <- SIPconst.CC-PRIMCODES.DIGITS_NOTIFY SIPproto.GS-Evts.GSipCcData.GSipHeader.gsCallLegID <- 1 SIPEventData.GsSipCcData.SipMsgBody.Component.content-length <- 5 SIPEventData.GsSipCcData.SipMsgBody.Component.content-type <- "avaya/vnd" SIPEventData.GsSipCcData.SipMsgBody.Component.content <- "12345" SIPproto.GS-Evts.GSipCcData.GsEvtData {=ToAVS} SIPEventData.GsSipCcData

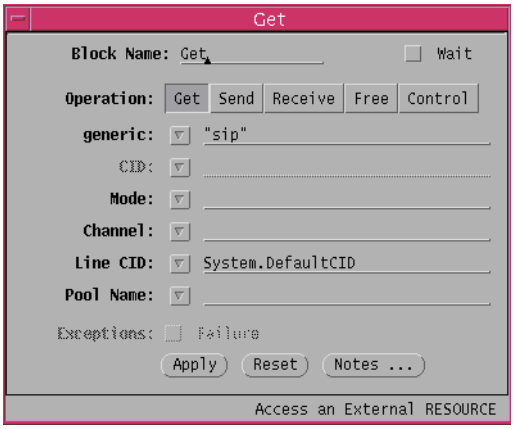
DigitsNotify triggers the SIP server to send SIP INFO with Digits to the remote end.

Incoming SIP Calls

The process for applications to receive incoming SIP calls is discussed in this section. The following diagram shows a depiction of the relevant message exchanges.



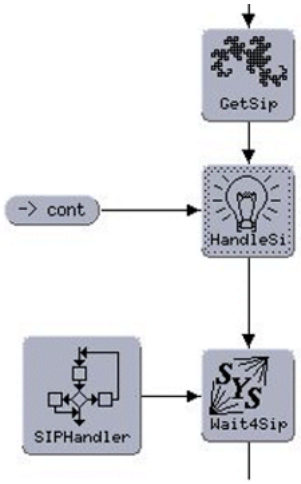
1. The application first acquires the SIP resource daemon for sending and receiving data to and from the SIP server. The application registers its service name in the Mode field. This places the application into the appropriate pool. (Leaving this field empty specifies that it is to receive any-inbound call.) The MPS SIP server in turn registers the application with the SIP proxy.



2. The application sends a getCallLegId message to the MPS SIP server and waits for the SIPOC/OC condition to receive the call leg ID. gsInboundFlag=1 indicates that the application acquires an inbound call leg. This makes the application ready to receive the call.



3. Next, the application waits for a condition to occur.



4. INVITE arrives to the MPS SIP server. The INVITE message contains the dialed number information (phone number or URI) in the RequestURI and the To headers. The From header contains calling party information. INVITE also includes the remote end SDP as shown by the following:

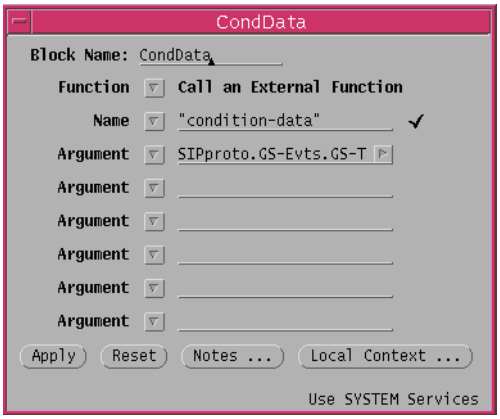
   INVITE sip:mps1@47.184.231.253:5060 SIP/2.0 Via: SIP/2.0/UDP 47.184.231.224:5060 To: sip:mps1@avaya.com From: "Avaya" <sip:avaya@avaya.com> ;tag=788693653 Call-ID: 821971086@avaya.com User-Agent: Avaya IP RAIDer/1.04pb1c Date: Wed, 21 Aug 2002 21:13:45 GMT CSeq: 1 INVITE Accept: application/sdp Expires: 90 Content-Type: application/sdp Content-Length: 230 Contact: <sip:avaya@47.184.231.224:5060;transport=udp>

   v=0 o=avaya 13908065148669574 13908065148669574 IN IP4 47.184.231.224 s=avaya p=+1 972 684 1000 c=IN IP4 47.184.231.224 t=0 0 m=audio 50012 RTP/AVP 18 4 a=rtpmap:18 G729A/8000 a=rtpmap:4 G723/8000 a=ptime:30

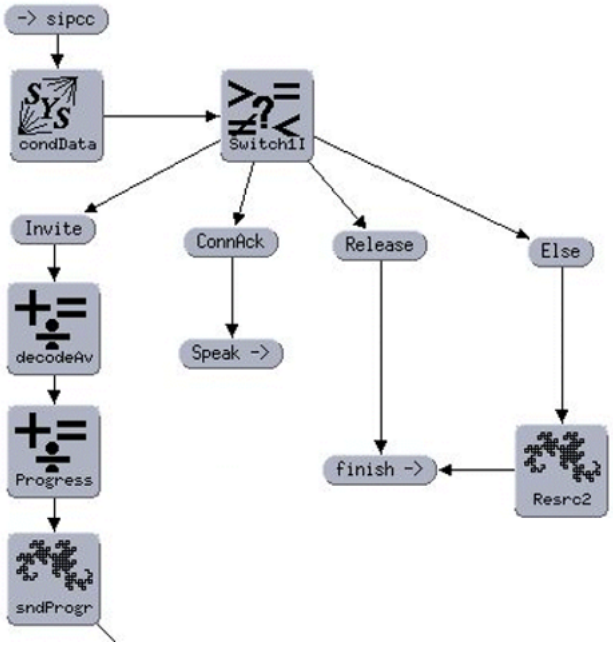
   The MPS SIP server responds with a 100 Trying.

   SIP/2.0 100 v:SIP/2.0/UDP 47.184.231.224:5060 f:"Avaya"<sip:avaya@avaya.com>;tag=788693653 t:<sip: mps1@avaya.com > i: 821971086@avaya.com CSeq:1 INVITE l:0

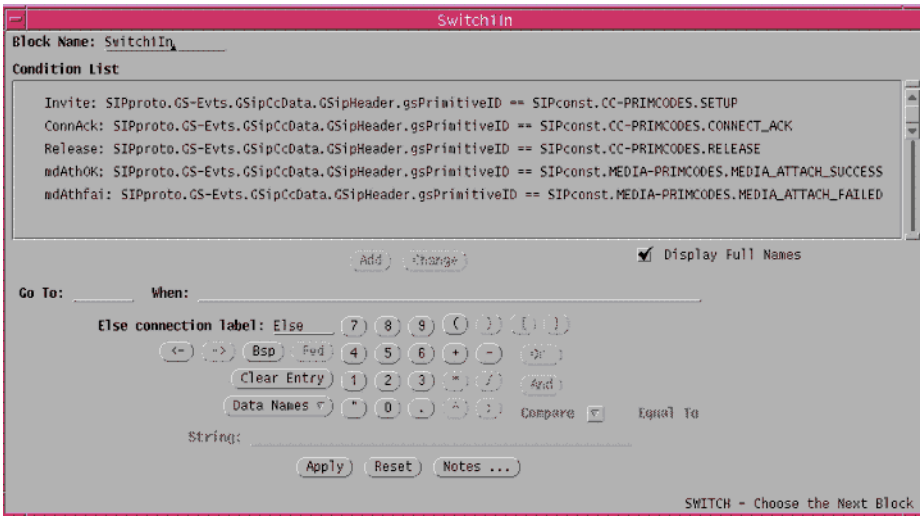
   The MPS SIP server translates the INVITE message to a SETUP message and sends it to the application as the SIPCC condition. The application handles SIPCC and retrieves the condition data to the SIPproto folder as shown here.



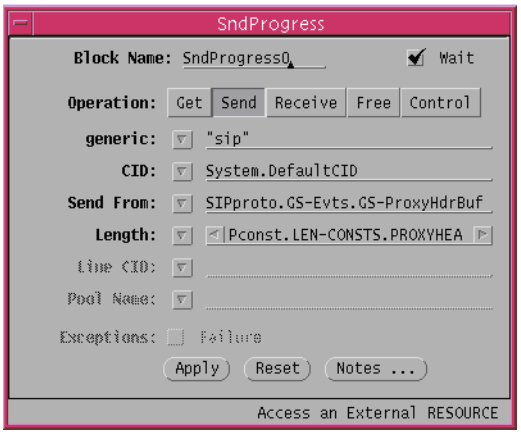
5. Based on the primitive received, the application responds/handles the condition accordingly.



The gsPrimitiveID indicates which primitive was received. The Switch block is shown in the following image.

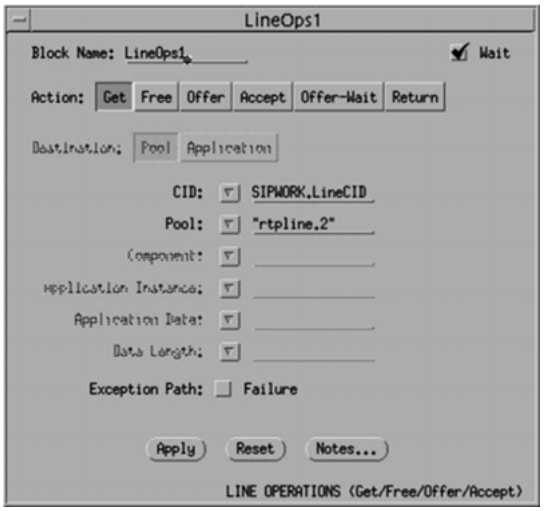


6. The application sends Progress with 180 when the SETUP is received from the MPS SIP server as shown here.



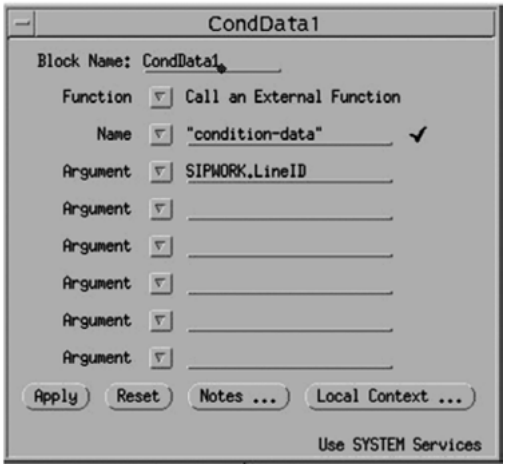
SIP/2.0 180 v:SIP/2.0/UDP 47.184.231.224:5060 f:"Avaya"<sip:avaya@avaya.com>;tag=788693653 t:<sip: mps1@avaya.com > i: 821971086@avaya.com CSeq:1 INVITE l:0

7. The application acquires a line from the RTP pool using LINE OPERATIONS as shown here.

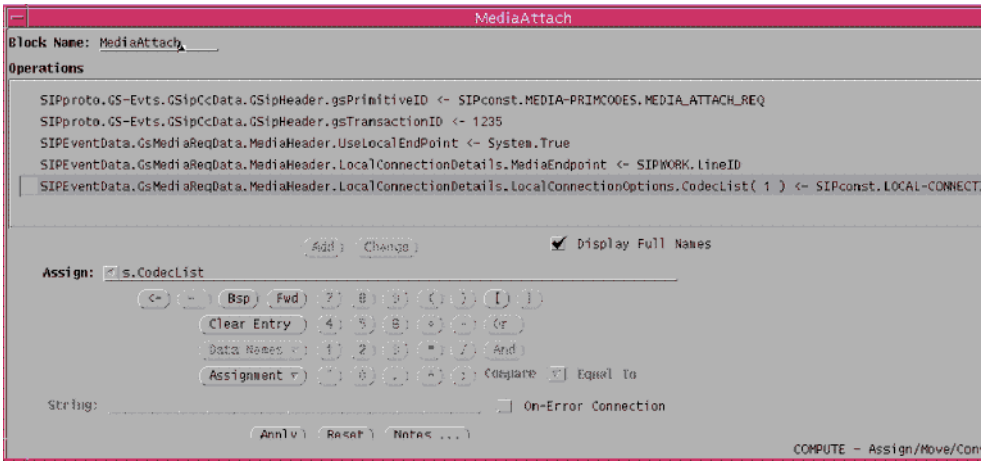


8. If getLine succeeds, the applications receives the GETLINECMP condition, and retrieves the LINEID from the condition data. (See the following CondData1 window.)

   The LineID is specified using the following format:
   phone.<line#>.mps.<comp#>.<chassis>.<slot>.<rset>



9. The application sends MediaAttachReq to create a connection with the MGCP gateway to start the media path. MediaAttachReq specifies the LineID (MediaEndPoint) and codec choice as shown in the following MediaAttach window. The MPS SIP server sends a CRCX to create a connection with TPM.



10. After receiving MediaAttachSuccess, the application sends CONNECT, which is 200 OK in the network as shown by the following:

    SIP/2.0 200 v:SIP/2.0/UDP 47.184.231.224:5060 f:"Avaya"<sip:avaya@avaya.com>;tag=788693653 t:<sip: mps1@avaya.com >;tag=000000002CE979EED000000000 i:821971086@avaya.com CSeq:1 INVITE l:69

    v=0 c=IN IP4 47.161.2.1 m=audio 2001 RTP/AVP 0 a=rtpmap:G.711u 0

    Then, the application waits for ConnectAck.

11. The remote end responds with ACK, which is delivered to the application as ConnectAck.

    ACK sip:mps1@47.184.231.253:5060 SIP/2.0 Via: SIP/2.0/UDP 47.184.231.224:5060 To: <sip: mps1@avaya.com > ;tag=000000002CE979EED000000000 From: "Avaya" <sip:avaya@avaya.com> ;tag=788693653 Call-ID: 821971086@avaya.com User-Agent: Avaya IP RAIDer/1.04pb1c CSeq: 1 ACK Content-Length: 0 Contact: <sip:avaya@47.184.231.224:5060;transport=udp>

    Once the ConnectAck is received in response to the Connect message, the call is connected and RTP voice stream is established. The applications now can speak until a Release comes from either side.

12. The application can send RELEASE and ISSUE FREE to free the SIP resource daemon. The application can also terminate the call with ISSUE FREE, which triggers RELEASE from the MPS SIP server if there is no prior RELEASE message with BYE in the network, as shown in the following image.

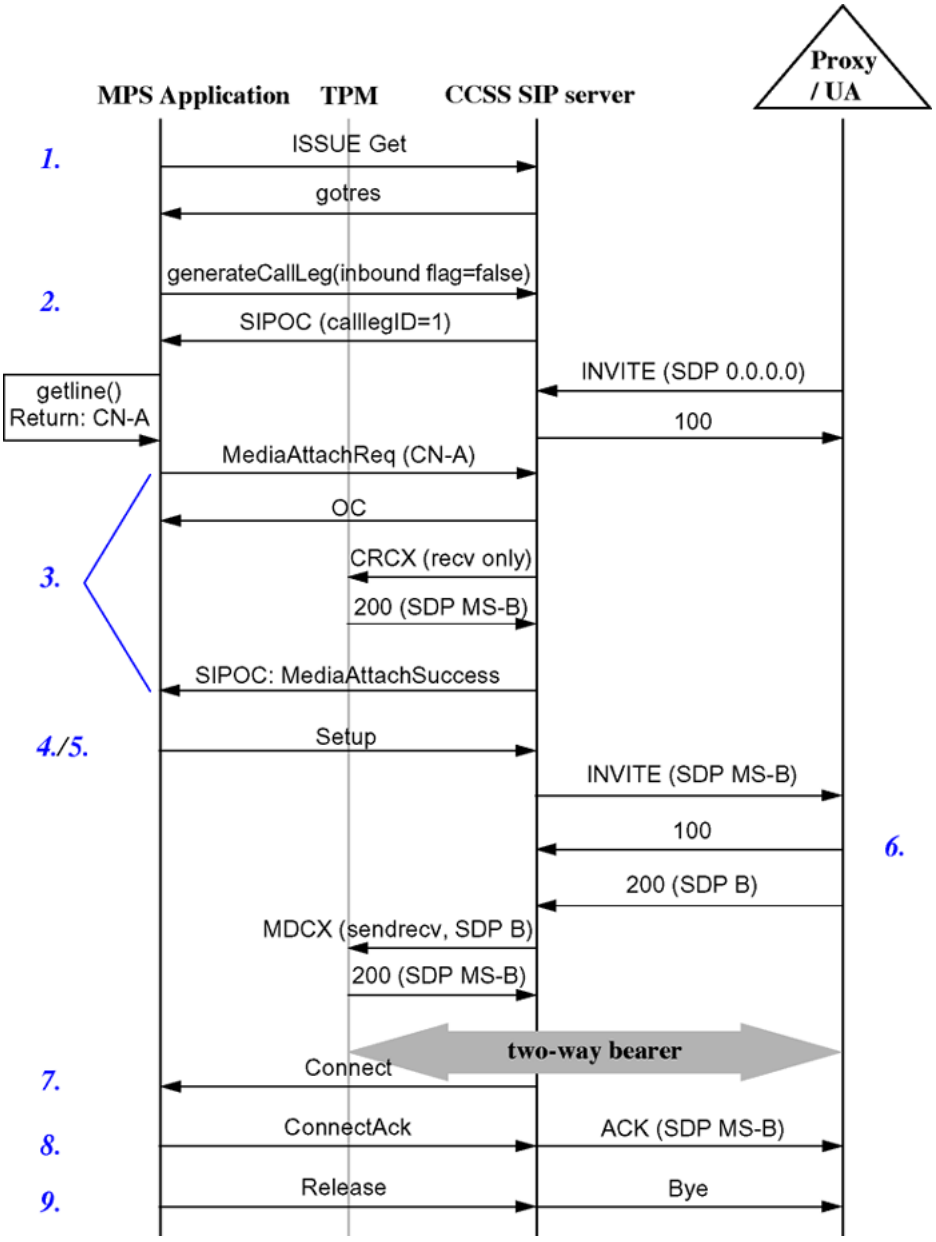


BYE sip:mps1@47.184.231.253:5060 SIP/2.0 Via: SIP/2.0/UDP 47.184.231.224:5060 To: <sip: mps1@avaya.com > ;tag=000000002CE979EED000000000 From: "Avaya" <sip:avaya@avaya.com> ;tag=788693653 Call-ID: 821971086@avaya.com User-Agent: Avaya IP RAIDer/1.04pb1c CSeq: 2 BYE Content-Length: 0

SIP/2.0 200 v:SIP/2.0/UDP 47.184.231.224:5060 f:"Avaya"<sip:avaya@avaya.com>;tag=788693653 t:<sip: mps1@avaya.com >;tag=000000002CE979EED000000000 i:821971086@avaya.com CSeq:2 BYE l: 0

Outgoing SIP Calls

The process for applications to place outbound SIP calls is discussed in this section. The following diagram shows a depiction of the relevant message exchanges.



To make an outgoing call from an application, the From, To, RequestURI and Proxy fields require values. The contents of these fields are converted to AVS format and retrieved by the MPS SIP server to originate a SIP call.

From and To comprise the SIP URL and are assigned in the format shown in the following example. The Proxy field is used to route the outgoing call. The name of the client system or configured proxy/softswitch name (defined in the sip.conf file) has to be specified in the Proxy field. RequestURI should be set to the configured proxy URL (as in the following example). If no proxy is configured, set RequestURI to the same value as the To field.

- In the following example, the To field has the value <sip:AnyUser@47.184.231.91:5060>
- The From field is assigned the value <sip:MPS01@47.185.16.71>

Sample AVS Format String tunneled in *vardata*

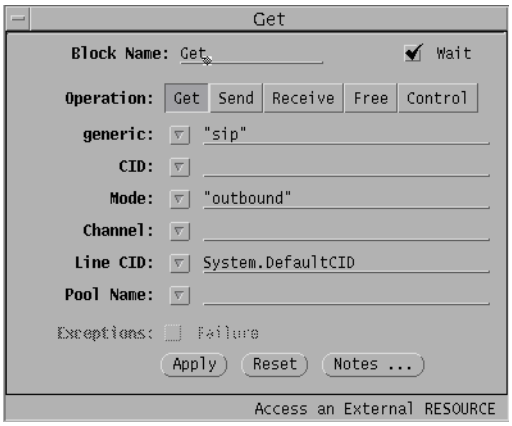
Generic header

SIP folder structure in AVS format

```
01000001000Unknown          00689
GsSipCcData:(SipCcHeader:(RequestURI:"sip:AnyUser@MPS_CC:5060"
EventIndicator:0) SipHeader:(to:"<sip:AnyUser@47.185.16.71:5060>"
expires:90 from:"\"Deepak
N\"<sip:deepakna@avaya.com>;tag=1278019451"
BillTrackID:1965546908@avaya.com date:"Tue, 13 May 2003 18:04:53
GMT" user-agent:"Avaya IP RAIDer/1.05pb4" accept:application/sdp)
SipMsgBody:(Component:(content-length:276 content:"v=0
o=deepakna 13915436523195692 IN IP4 47.184.231.91
s=avaya
p=+1 972 684 1000
c=IN IP4 47.184.231.91
t=0 0
m=audio 50046 RTP/AVP 0 8 18 4
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729A/8000
a=rtpmap:4 G723/8000
a=ptime:30
")))
```

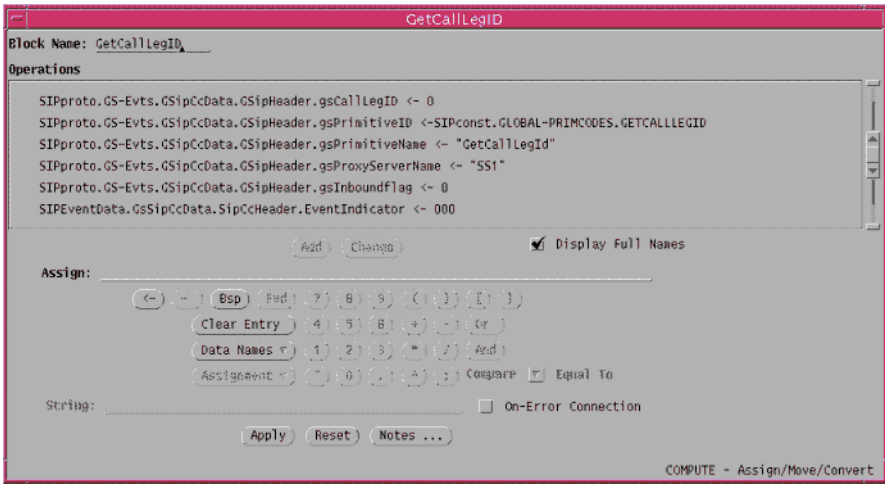
The following steps outline steps for applications to make outgoing calls.

1. The application first acquires the SIP resource to send and receive data to and from the MPS SIP server.



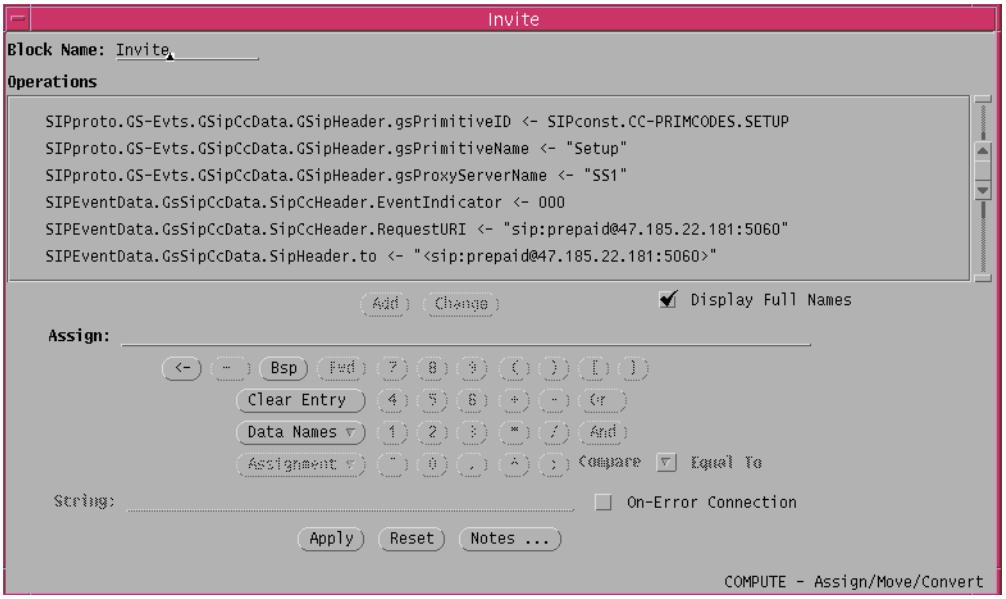
   The mode value outbound indicates that this application is handling outgoing SIP calls.

2. Acquire a call leg for the outbound call. The SIP server allocates the call leg and returns the ID to the application.



After sending getCallLegID, the application should wait for OC or SIPOC to receive the call leg ID. If an OF or SIPOF condition is received, the application cannot make the outbound call.

3. Acquire a media line and attach the application. (See steps 7-9 for information and examples. See 7 on page 153.)
4. After receiving MediaAttachSuccess, construct the Setup (Invite) message, as shown in the following image. For the Setup message, fill in the fields From and To with the Proxy server name, which specifies the remote SIP/Proxy server name to which the call is directed.



5. Send the Setup message, and wait for the condition to occur. The MPS SIP server receives SETUP and sends out INVITE.

   INVITE sip:prepaid@47.185.22.181:5060 SIP/2.0 v: SIP/2.0/UDP 47.184.231.238:5060 t: sip:<prepaid@47.185.22.181:5060> f: sip:prepaid@avaya.com;tag=33932065 i: 1062752400@avaya.com User-Agent:

IMS1.1 FP1 Avaya IP RAIDer/1.1.57pb1 Date: Tue, 11 Mar 2003 18:04:45 GMT CSeq: 1 INVITE Accept: application/sdp Expires: 90 Allow: ACK,BYE,CANCEL,INVITE,OPTIONS,REGISTER,INFO,MESSAGE,REFER,NOTIFY c: application/sdp l: 273 m: <sip:peri@47.185.16.72:5060;transport=udp>

v=0 o=peri 3256412685 3256412685 IN IP4 47.184.231.238 s=avaya p=+1 972 684 1000 t=3256412685 0 m=audio 50002 RTP/AVP 0 8 18 4 c=IN IP4 47.184.231.238 a=rtpmap:0 PCMU/8000 a=rtpmap:8 PCMA/8000 a=rtpmap:18 G729A/8000 a=rtpmap:4 G723/8000 a=ptime:30

The application waits for connect.

6. MPS SIP server receives 100 Trying, which is not sent to the application.

   SIP/2.0 100 Trying v: SIP/2.0/UDP 47.184.231.238:5060 t: sip:<prepaid@47.185.22.181:5060> f: sip:prepaid@avaya.com;tag=33932065 i: 1062752400@avaya.com CSeq: 1 INVITE k: com.avaya.firewall,100rel l: 0 Server: IMS1.1

7. When the MPS SIP server receives 200 OK, connect is sent to the application. The call is now connected and the voice path is established.

   SIP/2.0 200 OK v: SIP/2.0/UDP 47.184.231.238:5060 t: sip:<prepaid@47.185.22.181:5060> f: sip:prepaid@avaya.com;tag=33932065 i: 1062752400@avaya.com CSeq: 2 INVITE k: com.avaya.firewall,100rel l: 142 c: application/sdp m: <sip: 47.103.142.13:5060;nt_end_point=ZaBCAIBBFLAPBEANAN>

   v=0 o=tcctx01 0 0 IN IP4 tcctx01/47.185.16.26 s=session e=unknown@invalid.net c=IN IP4 47.185.16.241 m=audio 4360 RTP/AVP 0 a=ptime:30

8. The application responds with ConnectAck. The MPS SIP server responds back with an ACK.

   ACK sip:47.103.142.13:5060 SIP/2.0 v: SIP/2.0/UDP 47.184.231.238:5060 t: sip:<prepaid@47.185.22.181:5060> f: sip:prepaid@avaya.com;tag=33932065 i: 1062752400@avaya.com User-Agent: IMS1.1 FP1 Avaya IP RAIDer/1.1.57pb1 CSeq: 2 ACK l: 0 m: <sip:peri@47.185.16.72:5060;transport=udp>

9. The application processes the call until Release (BYE) is sent or received.

TMS Integrated Mode

Outgoing SIP Calls

This section describes the outgoing SIP call dial procedure for TMS integrated mode. The dial procedure is required outgoing call applications. There are three steps involved with originating a SIP call:

- prepare sip header (Compute block)
- prepare protocol data (Compute block)
- and originate the call (Originate block)

If necessary, a Handle Exception Conditions block can be inserted before the Originate block to manage call-origination exceptions and conditions.

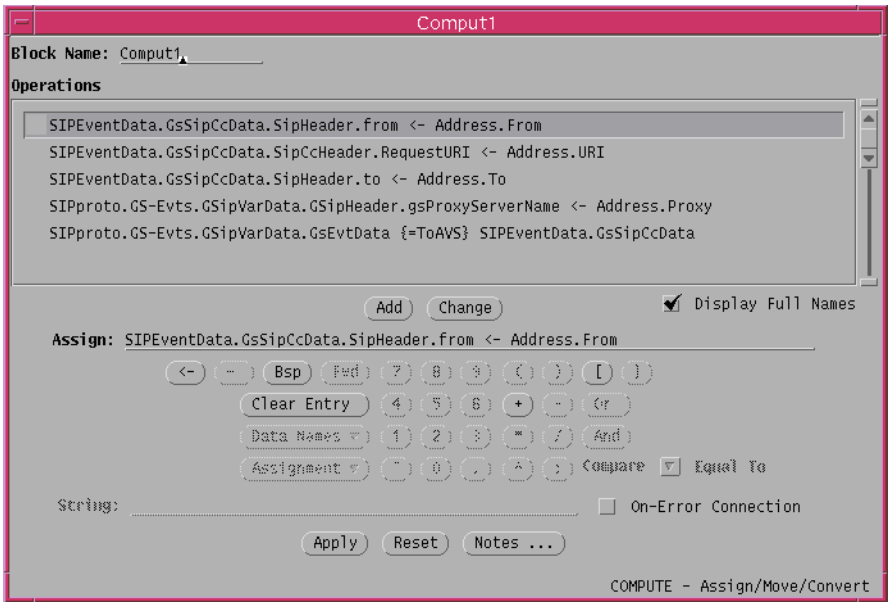


1. Setup the SIP header using a Compute block. Fill in the following mandatory SIP header fields:

   - SIPproto.GS-EvtData.GsSipCcData.SipHeader.to
   - SIPproto.GS-EvtData.GsSipCcData.SipHeader.from
   - SIPproto.GS-EvtData.GsSipCcData.SipCcHeader.RequestURI
   - SIPproto.GS-Evts.GsipVarData.GsipHeader.gsProxyServerName

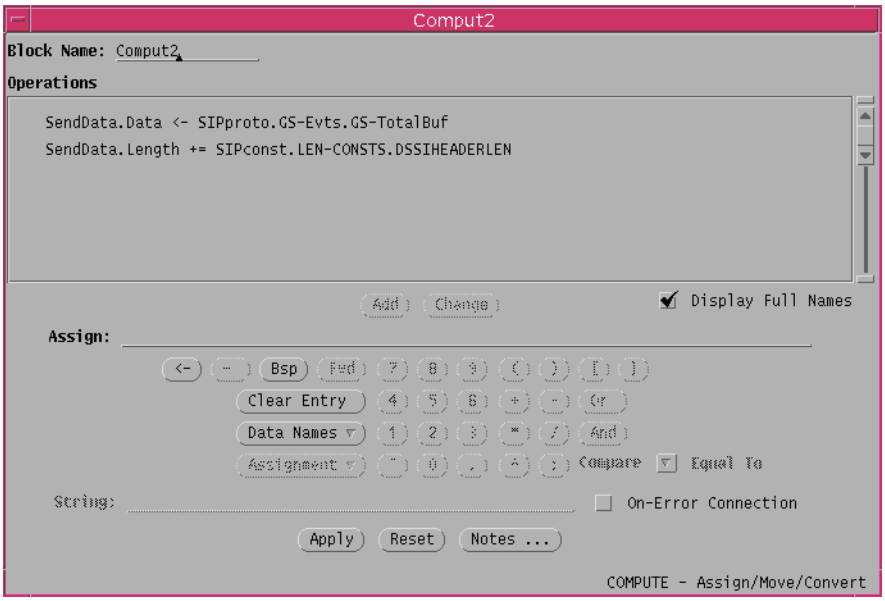
   From and To comprise the SIP URL and are assigned in the format shown in the following image. The Proxy field is used to route the outgoing call. The name of the client system or configured proxy/softswitch name (defined in the sip.conf file) has to be specified in the Proxy field. RequestURI should be set to the configured proxy URL (as in the following example). If no proxy is configured, set RequestURI to the

same value assigned to the To field. Do not use angular brackets while specifying RequestURI. Lastly, SIPEventData.GSSipCcData is converted to an AVS string and put into GsEvtData.

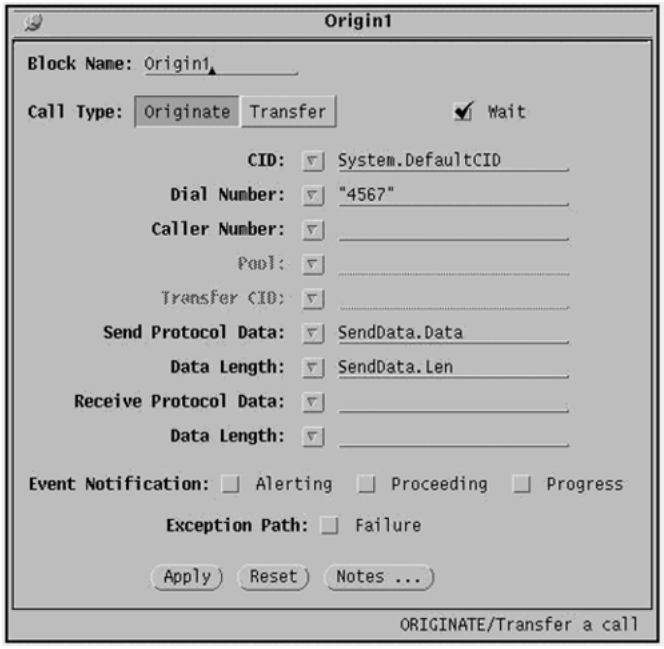


2. Prepare the SIP protocol data using another Compute block. Place the contents of the protocol data buffer into the folder used to send out protocol data.

   Please refer to the sample applications in /opt/vps/PERIccss/samples/sip directory for local data folders definitions.



3. Complete the dial procedure using an Originate block to make the call.

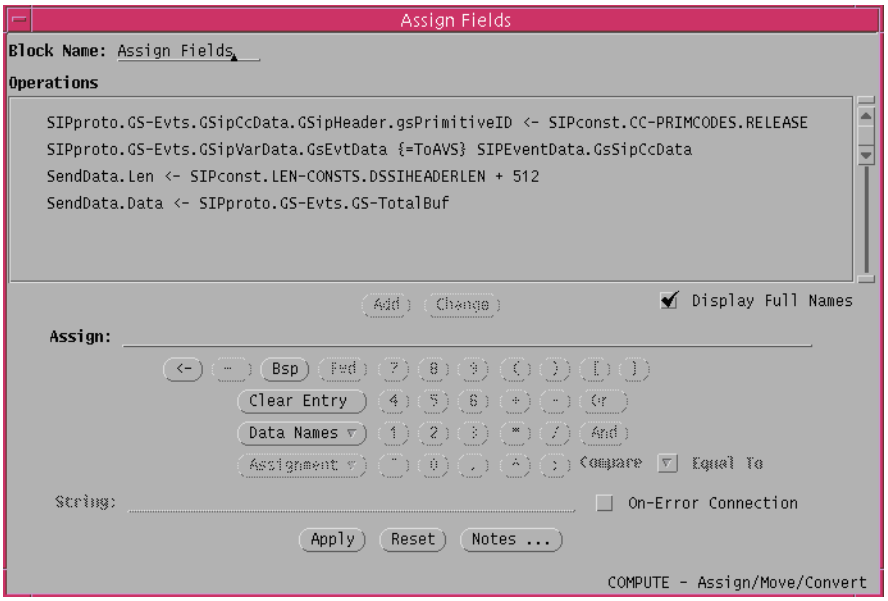


- Set the Call Type to Originate.
- Use System.DefaultCID in CID field.
- Fill Dial Number, a random number can be used here
- Use the protocol data prepared in last step in Send Protocol Data field
- Fill the Data Length field with the length of the protocol data.

4. Release (that is, disconnect) the SIP call.

   First prepare the protocol data using a Compute block, then send the disconnect request with a Disconnect block.

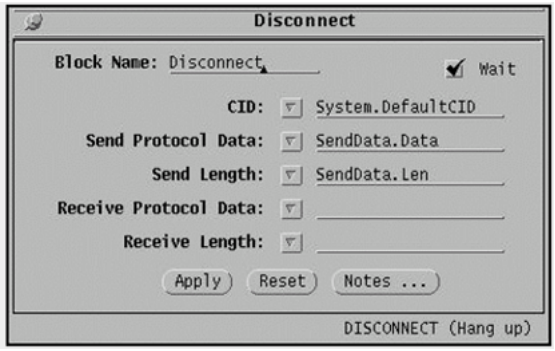




To release the protocol data, it is required that the data field SIPproto.GS-Evts.GSipCcData.GSipHeader.gsPrimitiveID be set to SIPconst.CC-PRIMCODES.RELEASE.

Some other headers are optional, such as SIPEventData.GsSipCcData.SipHeader.also, which provides bye-also functionality.

Finally, end the SIP call by using a Disconnect block to send out the disconnect request.



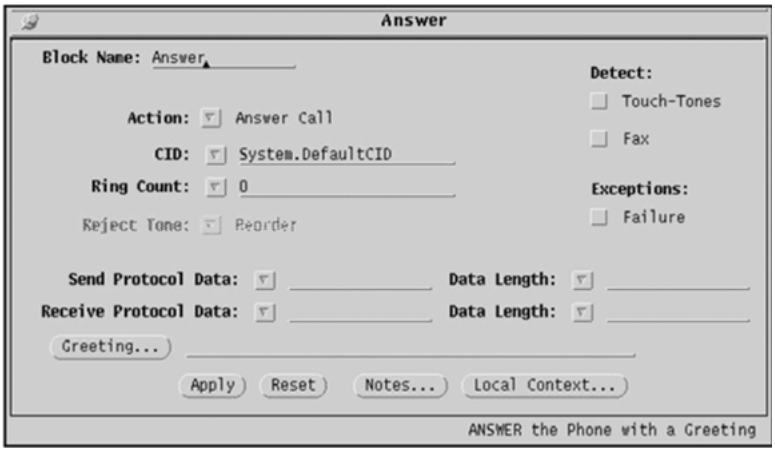
- Set CID to System.DefaultCID.
- Set Send Protocol Data to the data prepared in the previous step.
- Set Send Length to the length of the protocol data.

Incoming SIP Calls

This section describes how to receive an inbound SIP call.



1. Receive the inbound SIP call using an Answer block.



2. Disconnect the call. (See 4 on page 163.)

TMS SIP Feature Message Programming

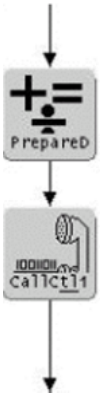
In TMS integrated mode, some call and media control primitives are implemented by sending feature messages. These are identified in the following table. (See Application API Primitives on page 125 for additional information about these primitives.)

Primitive	Code	Special Notes
Info	103	Store info data in SIP message body.
Redirect	108	
Hold	109	
Hold_Success	110	
Hold_Ack	111	
Hold_Fail	112	
Retrieve	113	
Retrieve_Success	114	
Retrieve_Ack	115	
Retrieve_Fail	116	
Voice_Reroute	117	
Voice_Reroute_Success	118	
Voice_Reroute_Ack	119	
Voick_Reroute_Fail	120	
Digit_Map_Request	121	
Digit_Map_Success	122	
Digit_Map_Fail	123	
Digit_Map_Cancel_Request	124	
Digit_Map_Cancel_Success	125	
Digit_Notify	126	Store digits information in SIP message body.
Refer	127	Fill in Refer-to and Refered-by field in SIP header.
Refer_Success	128	
Refer_Fail	129	
Notify	130	

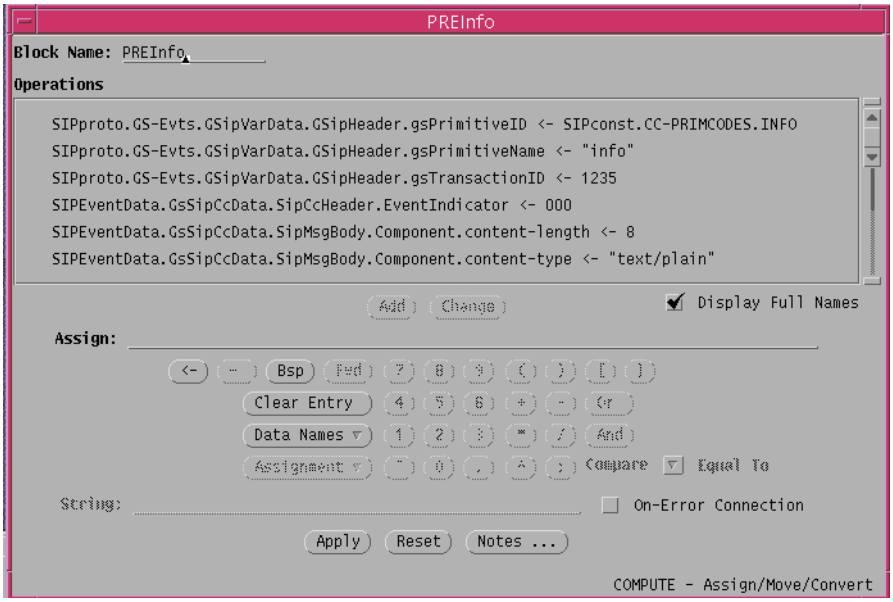
Primitive	Code	Special Notes
Notify_Success	131	
Notify_Fail	132	
Subscribe	133	
Subscribe_Success	134	
Subscribe_Fail	135	
NonCallSubscribe	136	
Message	139	
Message_Success	140	
Message_Fail	141	
Media_Modify_Request	207	
Media_Modify_Success	208	
Media_Modify_Fail	209	

Sending Feature Messages

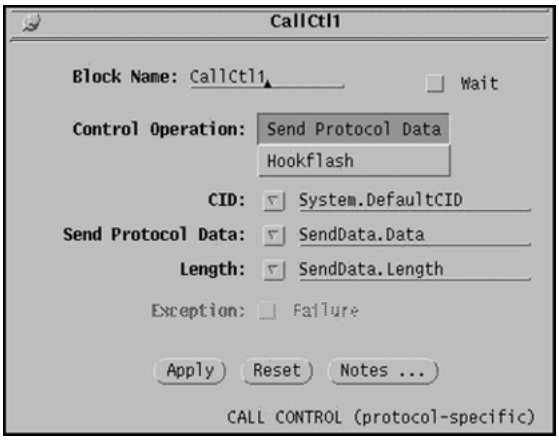
To send a feature message, use a Compute block to prepare the feature protocol data, then use the Callcontrol block to send out the feature protocol data. The following is an example of sending a Digit_Notify feature message.



1. Using a Compute block, prepare the feature message data. In the SIP header, the primitiveID and primitiveName are required. (Other fields are optional.) After the header is filled in, SIPEventData.GsSipCcData is converted to an AVS string and assigned to GS-Evts.GsipVarData.GSEvtData.



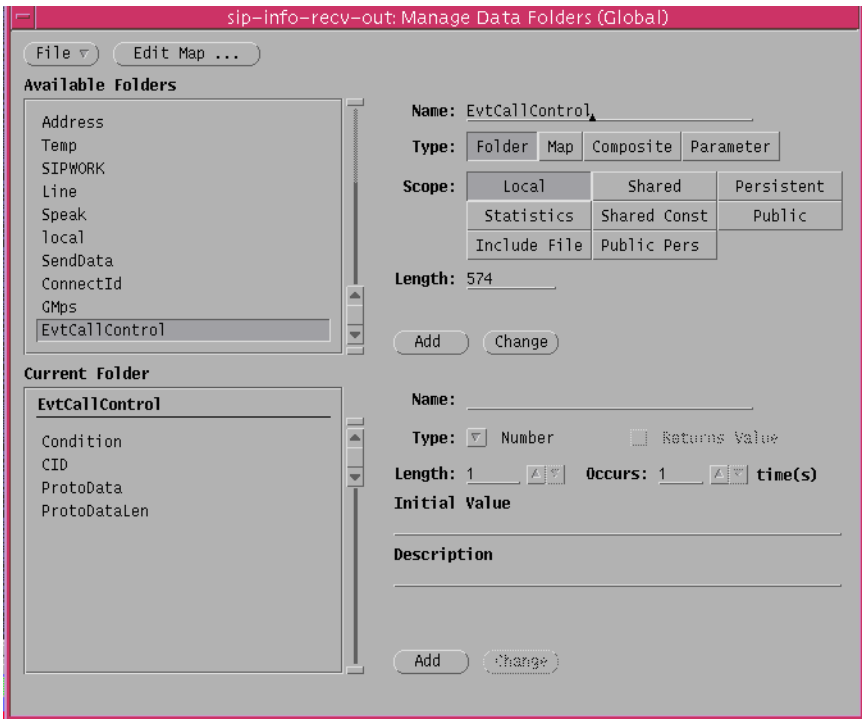
2. Using a Callcontrol block, send out the feature protocol data.



- Set Control Operation to Send Protocol Data.
- Set CID to System.DefaultCID.
- Set Send Protocol Data to the data prepared in the previous step.
- Set Length to the length of the protocol data to be sent.

Receiving Feature Messages

Feature messages are received as condition data with the PROTODATA condition in the Phone Line Messages and Events section of the Handle Exception Conditions block. To receive the condition data correctly, the data folder shown in the following image must be used in the Get Condition Data function of System block.

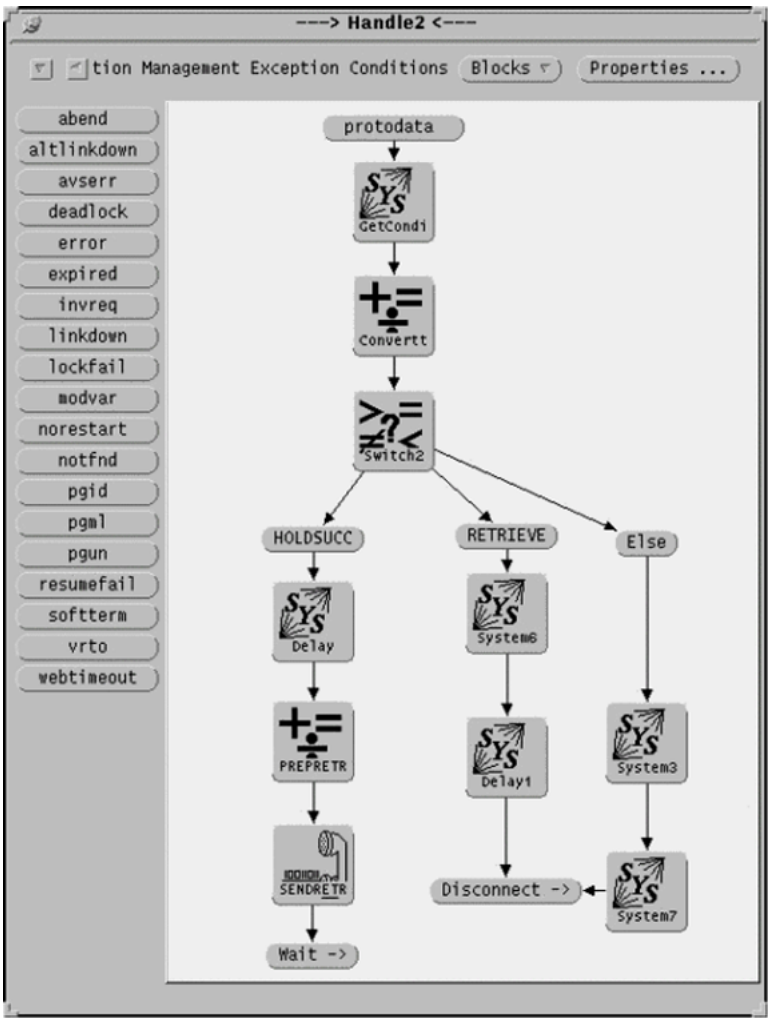


The following is the data folder for receive feature protocol data.

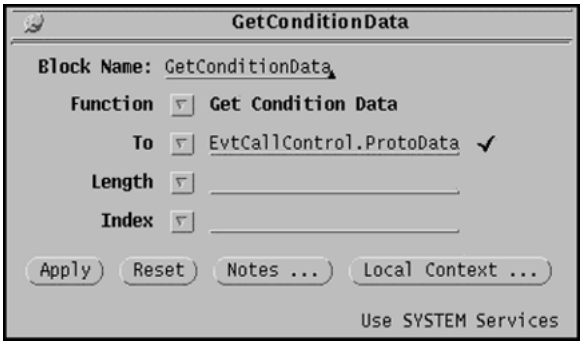
Name	Type	Length	Note
Condition	N	1	
CID	C	40	Indicates which CID caused the condition to occur.
ProtoData	C	512	
ProtoDataLen	N	6	

The following is an example of receiving a feature protocol message. The procedure includes:

- Use of the get condition data function of the System block to store the incoming protocol data in the data folder previously shown.
- Use of a Compute block to convert the protocol data into a SIP folder.
- Use of a Switch block to determine the type of protocol data.
- Appropriate responses in each branch of the Switch block.



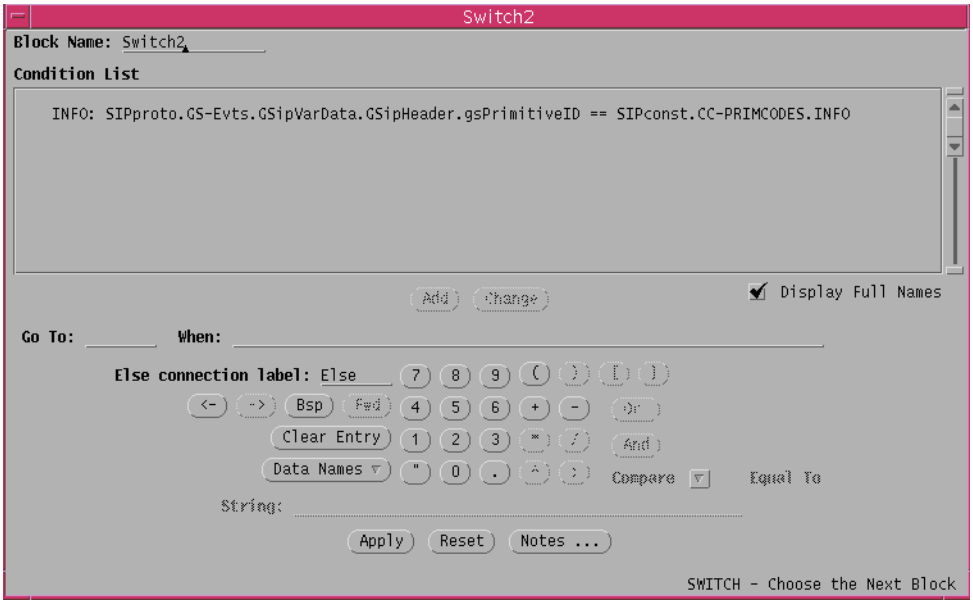
The GetConditionData function takes the protocol data folder as a parameter.



The Compute block converts the protocol data into the SIP data folder.



A Switch block is used to examine the primitiveid.



Chapter 10: Application Services Registration

This chapter covers:

1. SIP Application Registration
2. Registration Methods

Application Services Registration

Registration creates bindings for a particular domain's location service that associate an address-of-record URI with one or more contact addresses. When a proxy for that domain receives a request with a Request-URI matching the address-of-record, the proxy forwards the request to the registered contact address(es).

Registration entails sending a REGISTER request to a special type of service referred to as a SIP registrar. A registrar functions as the front end of the location service, and manages address mappings based on the REGISTER requests. Typically, the proxy server responsible for routing requests for a given domain consults that domain's location service.

Upon initialization, and at periodic intervals, the SIP client sends REGISTER messages to the registrar. The REGISTER messages associate SIP client or SIPS URI (for example, sip:prepaid@avaya.com) with the machine into which the client is currently logged (conveyed as a SIP or SIPS URI in the Contact header field). The registrar writes this association (that is, the binding) to the location service's database, where the proxy domain can use it. Often, a registrar server for a domain is co-located with the proxy for that domain.

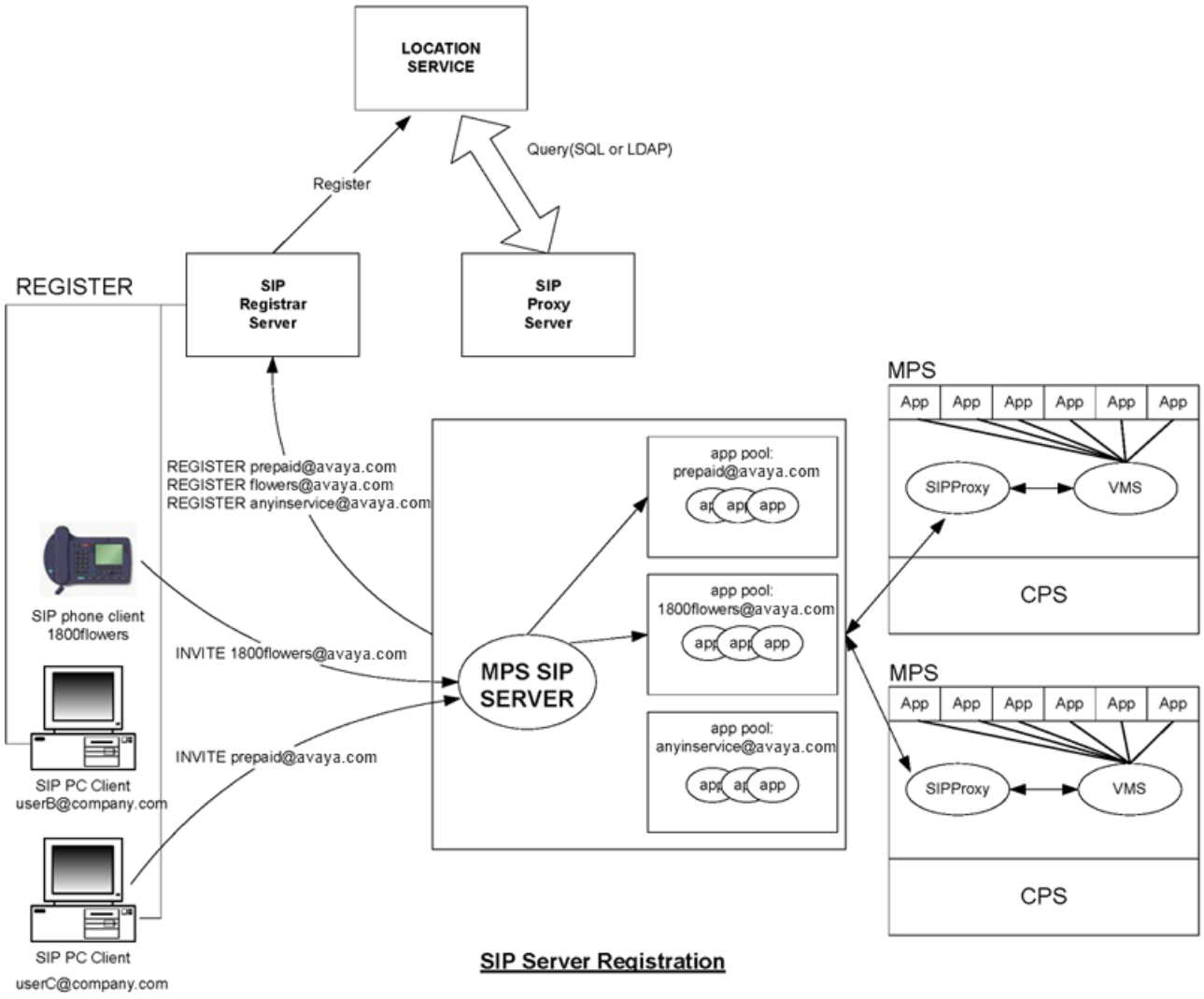
The following is important information:

- For each SIP server, the MPS SIP environment supports only one contact address for each address-of-record. Multiple register requests can be sent from different devices for the same address-of-record. To update the proxy/registrar with multiple contacts, use registrations from different SIP servers for a given user name.
- Multiple users can be registered on a single device at the same time.
- Different types of contacts are not supported. All contacts are of URLAddress type. E-mail and telephone URL is currently not supported.
- The distinction between types of SIP servers is logical, not physical.

- Generally, it is useful to only register an address-of-record in a domain's location service when requests for that address-of-record are to be routed to that domain. In most cases, the domain of the registration has to match the domain in the URI of the address-of-record.
- Registration can be renewed upon expiration and cancelled before expiration.

MPS SIP Server Register Function

MPS applications provide a single service or group of services. The MPS SIP registration function notifies the SIP server about the current locations of the MPS applications/services. This information is used by proxy servers, and used by the SIP server to route incoming calls to the appropriate application services. The SIP server maintains pools of applications based on the services they provide.



SIP Server Registration

The SIP server sends the REGISTER request on behalf of the MPS application groups. During system startup, the SIP server reads the services and registrar details from the sip.conf configuration file and creates the application pools. When the applications are started, to become registered in the appropriate pool, they notify the service by using the Mode field of ISSUE GET.

Registration can be accomplished in three different ways:

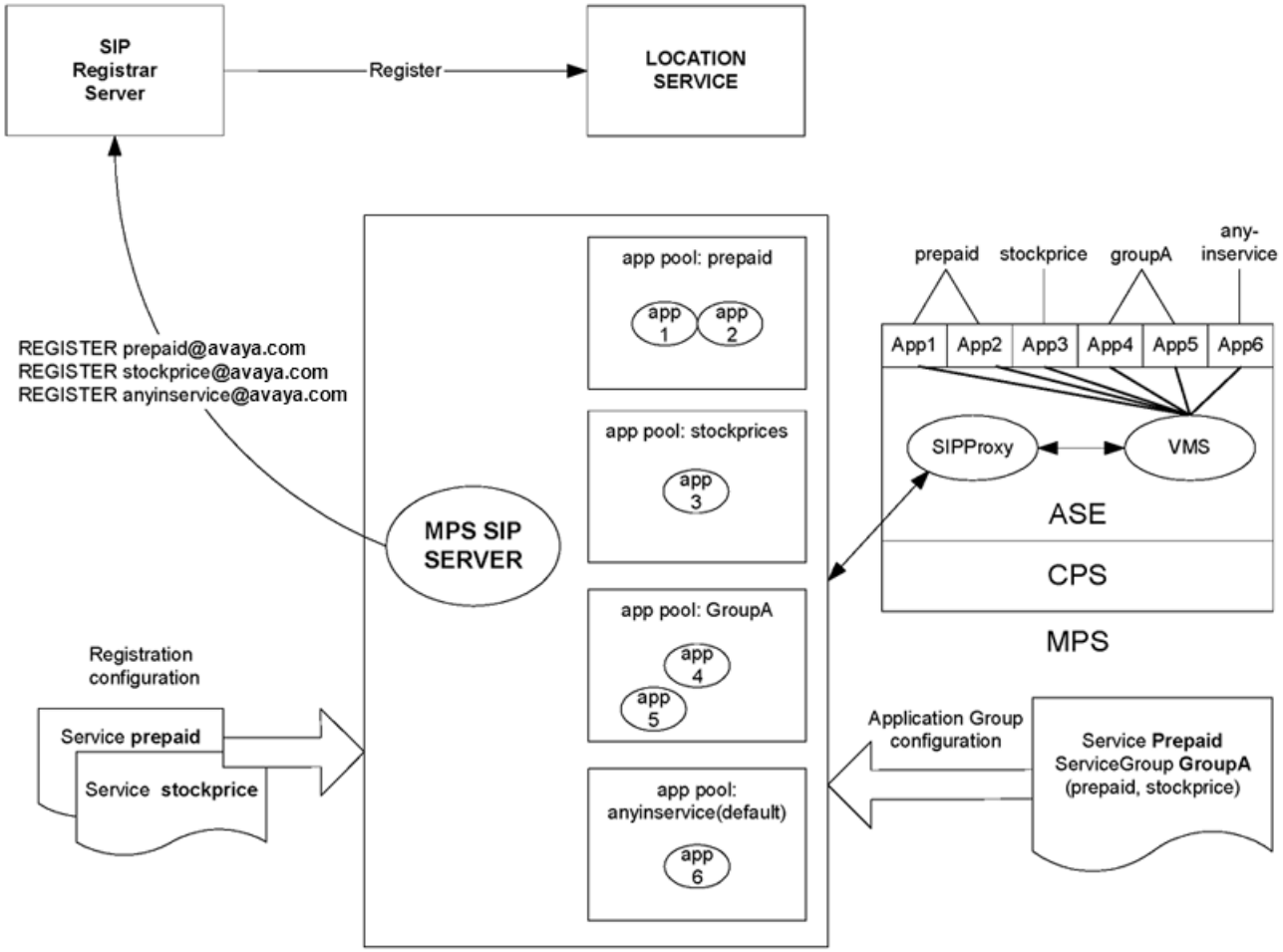
- During system startup by setting the keep registered flag in the sip.conf on page 58 file.
- If the keep registered flag is not set, REGISTER is sent only when there are ready applications available in the pool.
- Using a UI session. (See Command, Control, and Status on page 182.)

When an incoming call arrives, it is routed to the appropriate application based on the value of the To:ServiceURL field.

Registration

The following figure shows a REGISTRATION request going out based on the data in the configuration file. There is one registration record for each service. The registration details come from the [REGISTRATION_CONTROL] section (see [REGISTRATION_CONTROL]) of the sip.conf file (see sip.conf on page 58).

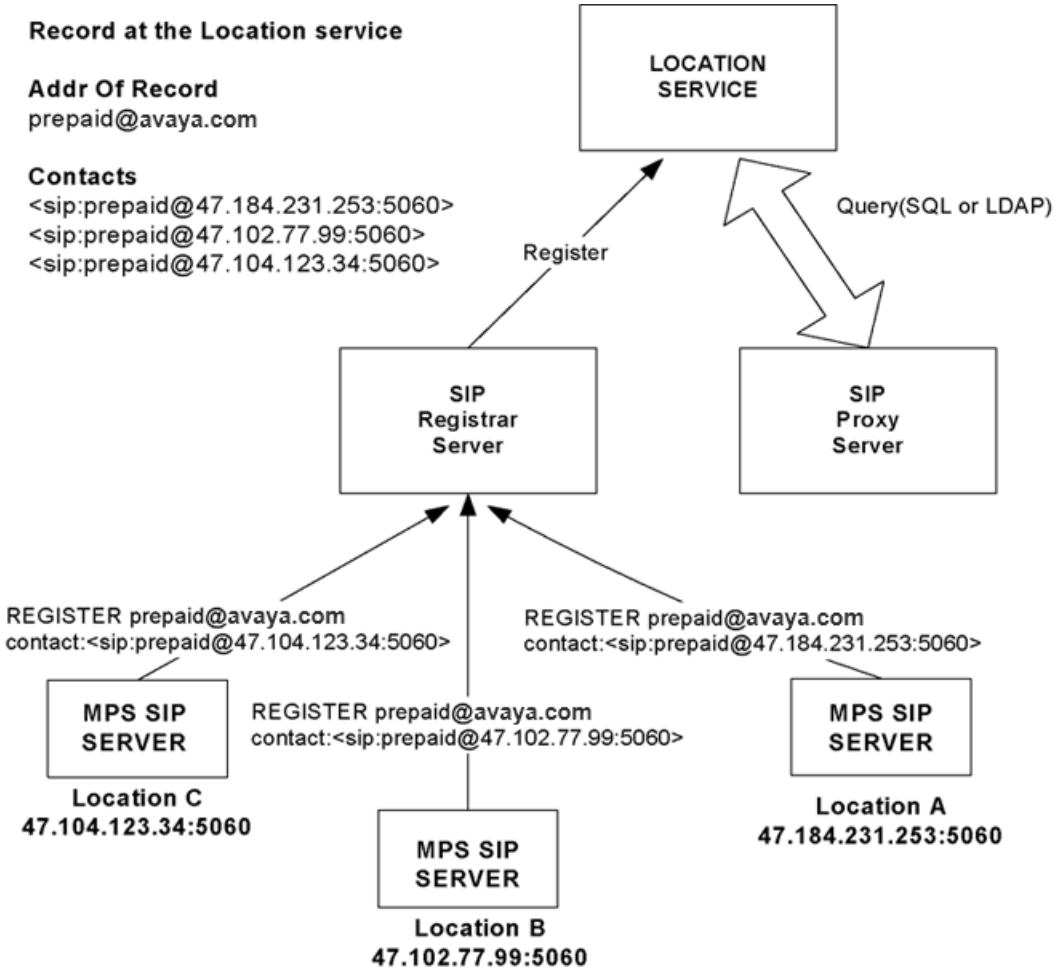
Multiple applications can provide a single service, and each application can provide multiple services. The application group configuration should provide the details of the application pools as shown in the following figure.



Registration Flow Diagram

Multiple Contacts

Singular registration with multiple contacts is not supported in the MPS SIP environment. However, multiple MPS SIP servers providing the same service can register with different contacts. This results in updating the proxy/registrars with multiple bindings for a single address-of-record, as is illustrated in the following figure.



Registration for Multiple Contacts

Using different contact addresses, MPS SIP servers in different locations register for prepaid@avaya.com. This results in updating the record in the location service with more than one contact. The selection of the contacts for invite messages is performed by the SIP proxy server.

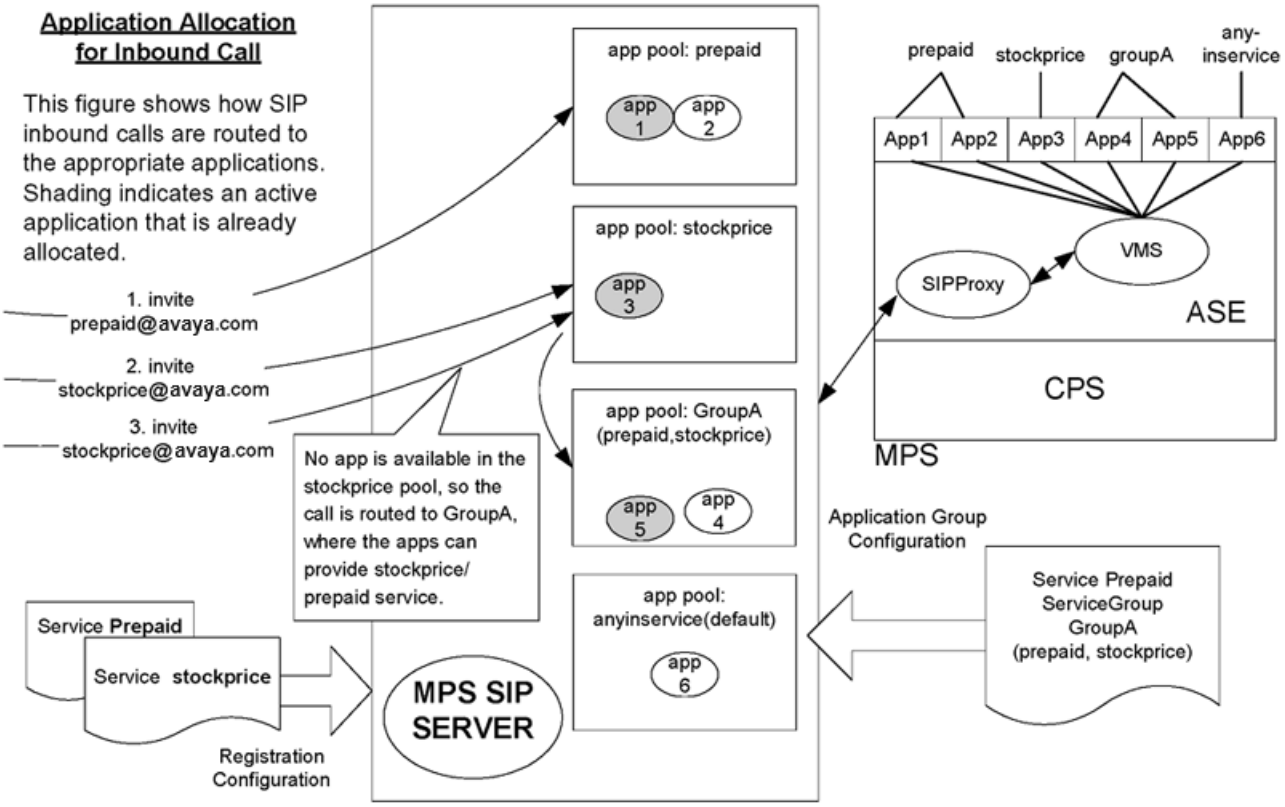
Application Pools and Allocation

An application can be registered to a generic inbound pool or a specific pool.

- generic (any) inbound pool: any inbound request that doesn't match the SIP URL specific pool allocates one application from the default pool.
- SIP URL-specific inbound pool: Applications can be logically grouped by their service functions. The group name is identified by the SIP URL specified in the Register command. If it does not exist, the pool manager creates the pool when the first application is registered.
  - When initially created, each application service pool can optionally send a SIP Register message to an external Proxy/Registrar server.
  - A timer is configured to maintain the application's registration when no application is available in the pool.
- Issue Get: the application specifies the pool name in the ISSUE GET Mode field (for example, ISSUE GET with mode prepaid places the application in the prepaid pool).

When an incoming call specifies To:prepaid@avaya.com, an application gets allocated from the prepaid pool. If no application is available in that pool, an application is allocated from the GroupA pool (which is capable of providing both prepaid and stockprice services). If no application is available in GroupA, the call is routed to the anyinservice pool.

If no application is available in any pools, the call is rejected. Incoming calls with To:anyuser@avaya.com are directed to the default pool, which is anyinservice.



Authentication Support

Authentication Background

SIP provides a stateless, challenge-based mechanism for authentication that is based on the authentication in HTTP. Any time a proxy server or UA receives a request, it may challenge the initiator of the request to provide assurance of its identity. Once the originator has been identified, the recipient of the request should ascertain whether or not the user is authorized to make the request.

The framework for SIP authentication closely parallels that of HTTP. In SIP, a UAS uses the 401 (Unauthorized) response to challenge the identity of a UAC. Additionally, registrars, and redirect servers can make use of 401 (Unauthorized) responses for authentication, but proxies must not, and instead may use the 407 (Proxy Authentication Required) response. The requirements for inclusion of the Proxy-Authenticate, Proxy-Authorization, WWW-Authenticate, and Authorization in the various messages are identical to those described in RFC 2617 HTTP Authentication.

MPS SIP Server Authentication Support

The Digest scheme is based on a simple challenge-response paradigm. The Digest scheme challenges using a nonce value. Nonce value is a server-specified data string, which should be uniquely generated each time a 401 response is made. A valid response contains a checksum (by default, the MD5 checksum) of the username, the password, the given nonce value, the HTTP method, and the requested URI. This way, the password is never sent in the clear.

The user name and the password are prearranged in the Proxy/Registrar.

Note:

The current version of the MPS SIP server handles the 401/407 responses only for the Register request. It is capable of only responding to the Digest challenge request.

Registration Configuration

Registration configuration is set up in the [REGISTRATION_CONTROL] section (see [REGISTRATION_CONTROL]) of the sip.conf file (see sip.conf on page 58).

Registration Configuration

The required information for application registration is shown in the following table.

Field	Description
ServiceName	Service provided by the application. Example: Prepaid, Stock price
Domain name	Domain of the location service for which the registration is intended. Example: avaya.com
Contact	Address binding. The MPS SIP server supports only the SIP URI scheme for contact. Example: Contact: "sip:pre paid@47.185.16.71:5060;transport=udp"
Contact-expires	The expires parameter of the contact field indicates how long the UA requests the binding to be valid. The value is a number indicating seconds. If this parameter is not provided, the value of the Expires header field is used instead.

Field	Description
	Example: Contact: <sip: prepaid@47.185.16.71: 5060;transport=udp>; expires=3600
Expires	Registration expiration value for contacts that do not have the expires parameter. The value of this field is an decimal integral number indicating seconds between 0 and (2^{31})-1. If neither mechanism for expressing an expiration time is present in a REGISTER, a default of one hour is used. Example: Expires: 3600
Registrar	List of registrars/proxies to which the registration request should be sent. The list of registrars for the given domain is obtained from the NETWORK_HOSTS = { references in the sip.conf file.

Application Group Configuration

An application provides a unique service or a group of services. These are configured in the [REGISTRATION_CONTROL] section of the sip.conf file.

The configuration includes the details shown in the following table.

Field	Description
ServiceGroupName	Service group name used to identify the group. Example: StockGroup.
ServiceName	List of services under the group definition. This should only include services that appear in the registration configuration. Example: Stockprice, Stocknews.

Line Configuration

The MPS lines can be preconfigured for their intended services. The line configuration command lists the line range and the service or service group it is going to provide. Line configuration is done in the [REGISTRATION_CONTROL] section of the sip.conf file. (See REG_LINE_CONFIG)

The configuration includes the details shown in the following table.

Field	Description
Mps HostName	Host name of the MPS. Example: tcmps01.
VPS/MPS number	The VPS/MPS ID. Example: 1, 2

Field	Description
Line Range	The range of lines. Example: 1-10, 30-40, 50
ServiceGroup	The name of services. Example: Prepaid, GroupA

Applications on preconfigured lines need not register to become part of the pool. They become members of the pool when the system starts.

Note:
Pre-configuration of lines is valid only for TMS integrated mode.

Command, Control, and Status

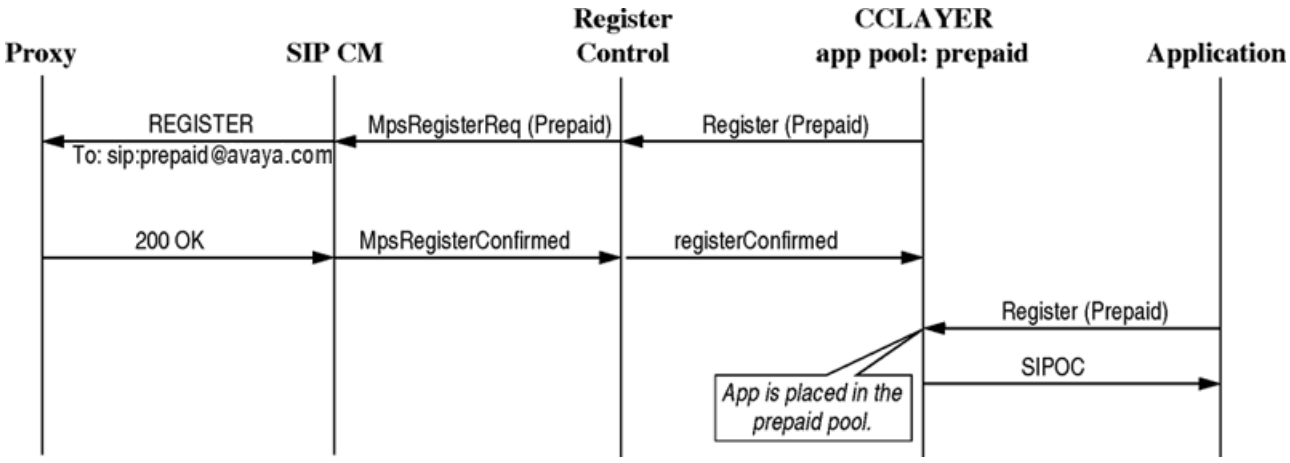
The command and control interface provides the commands to dynamically manipulate registrations. Registrations for the preconfigured services can be cancelled.

Command	Description
register deregister <service>	Creates pool and issues REGISTER.
deregister <service>	Informs all the active applications, removes the ready applications, and issues DEREGISTER.
passwd	Creates, changes, or deletes register passwords.
registrations	Lists all the current registrations showing their states.

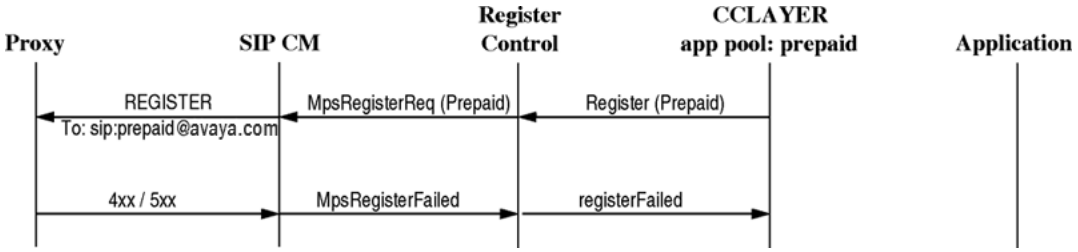
Note:
See DEBUG:root/mps/register on page 103 for more information about these commands.

Message Sequence Diagrams

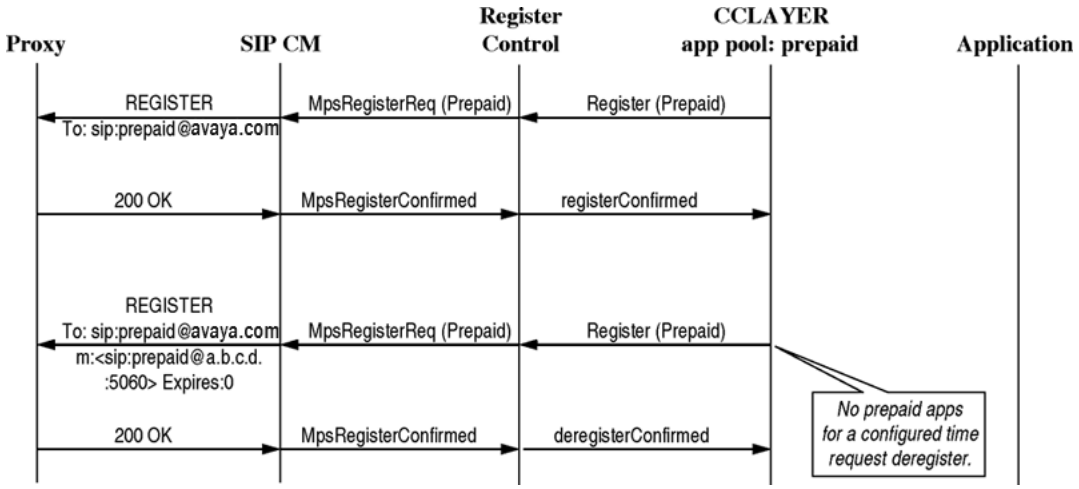
Registration Success



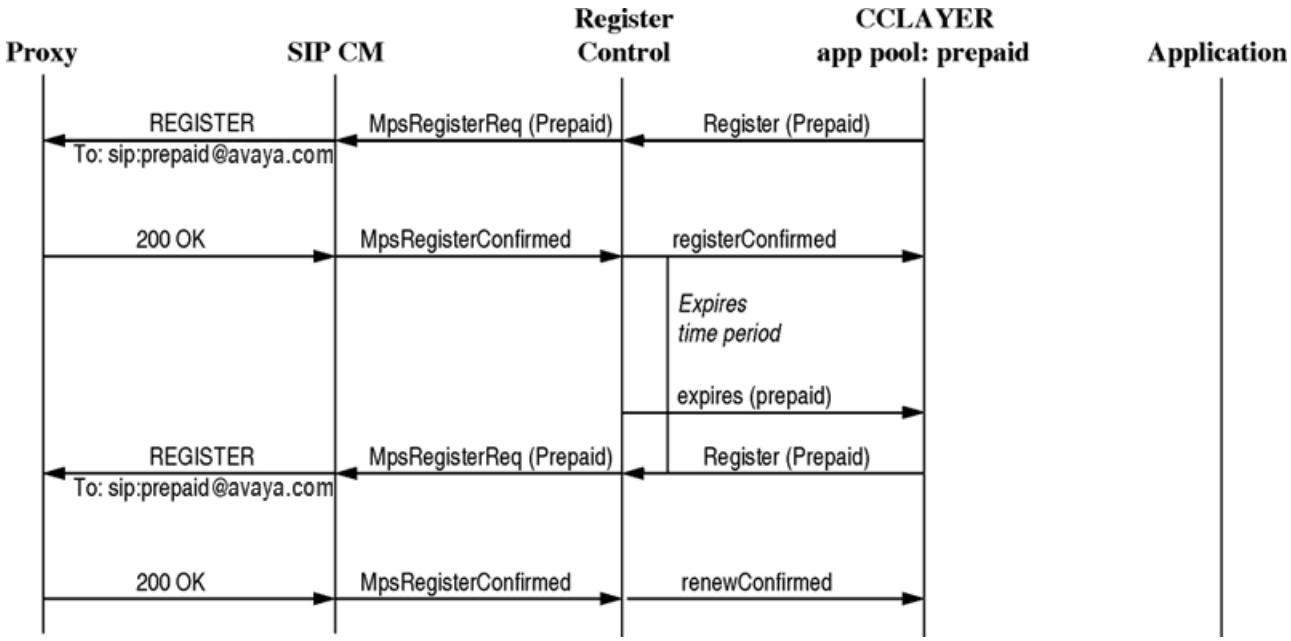
Registration Failure



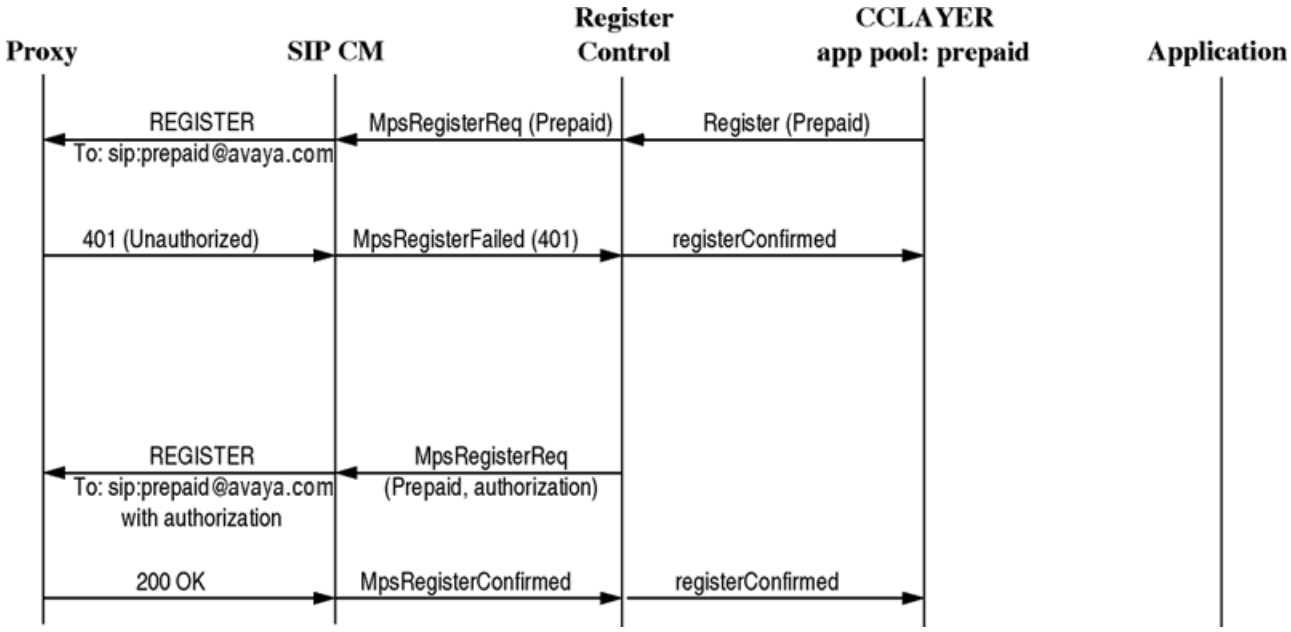
Deregistration (Cancel Registration)



Registration Renewal



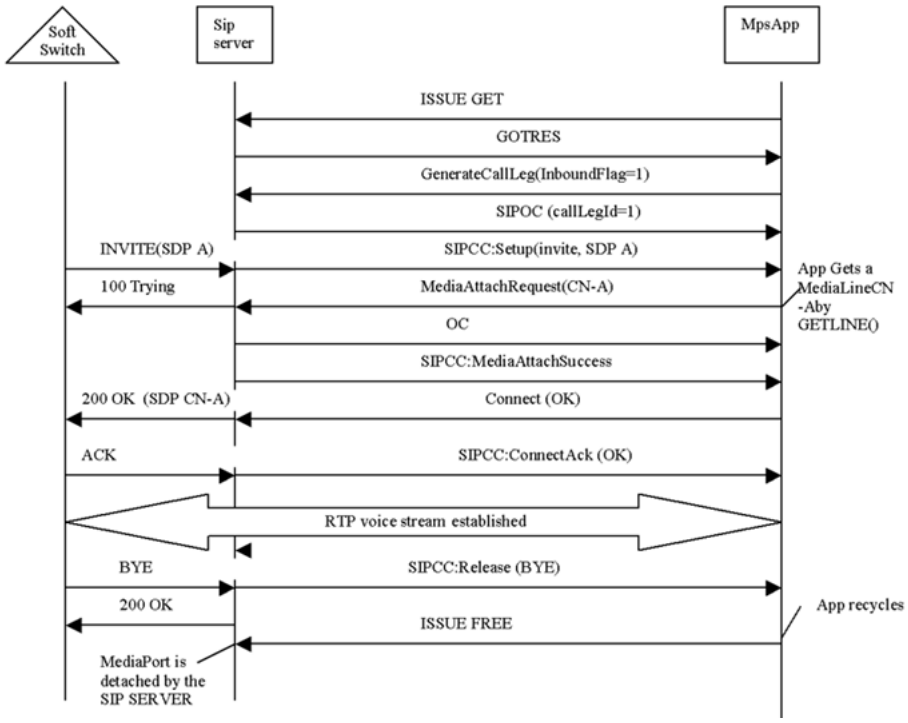
Registration Retry for 4xx Responses



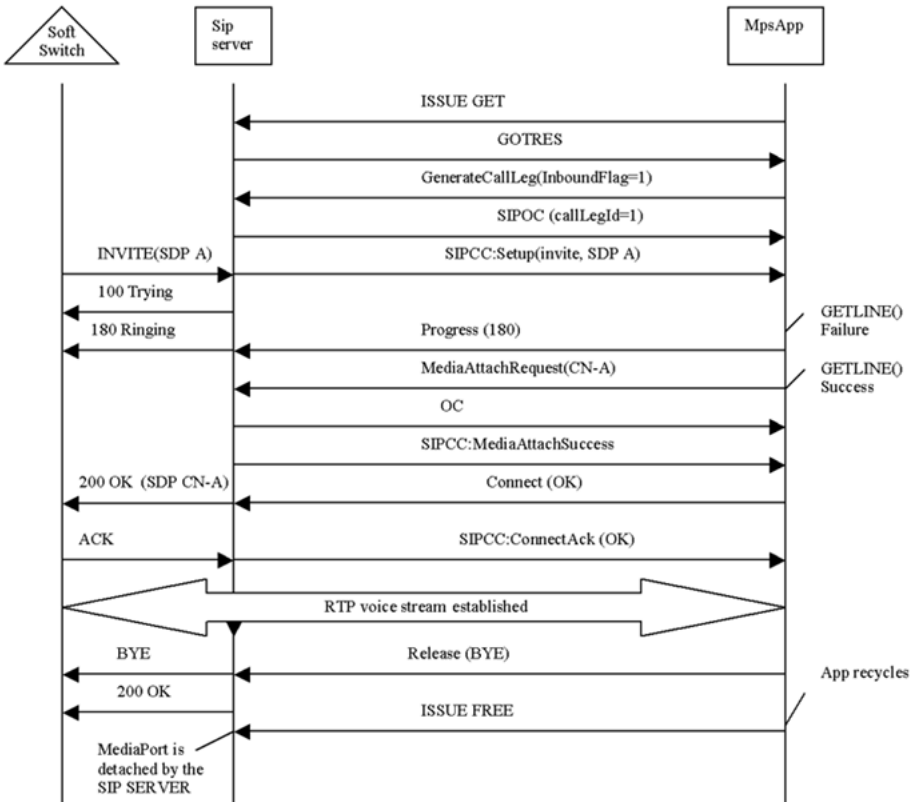
Application Call Flows

Call Flow Diagrams

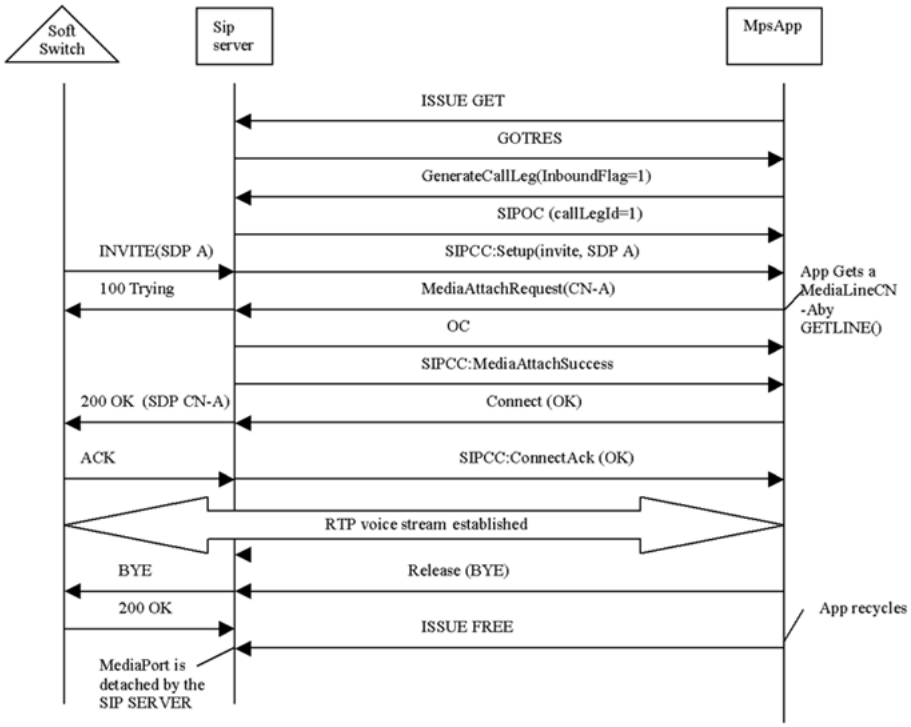
Incoming Call Flow (Single Leg)



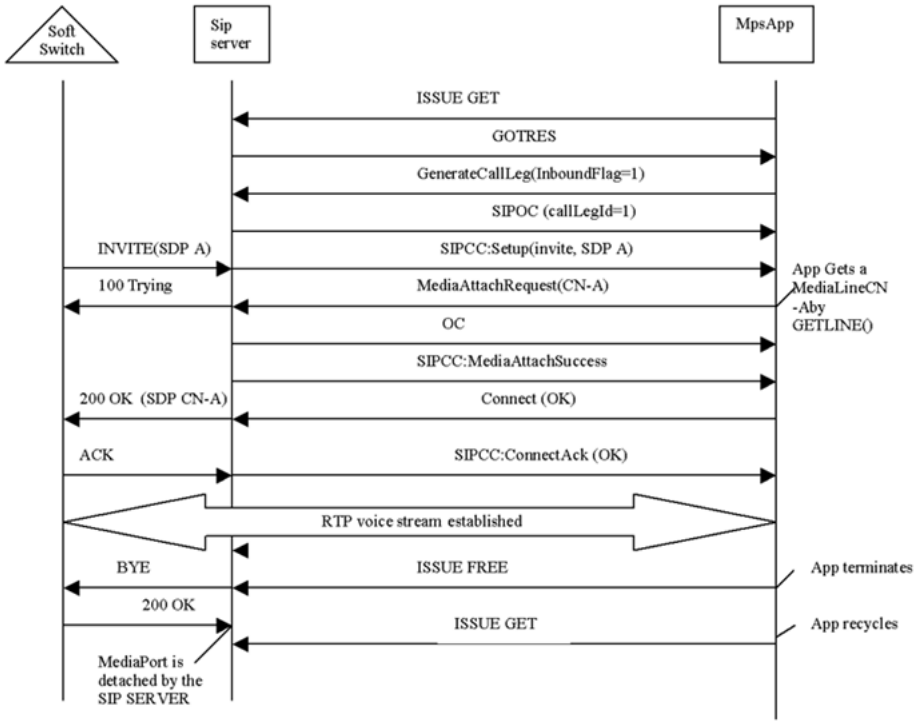
Waiting for CN



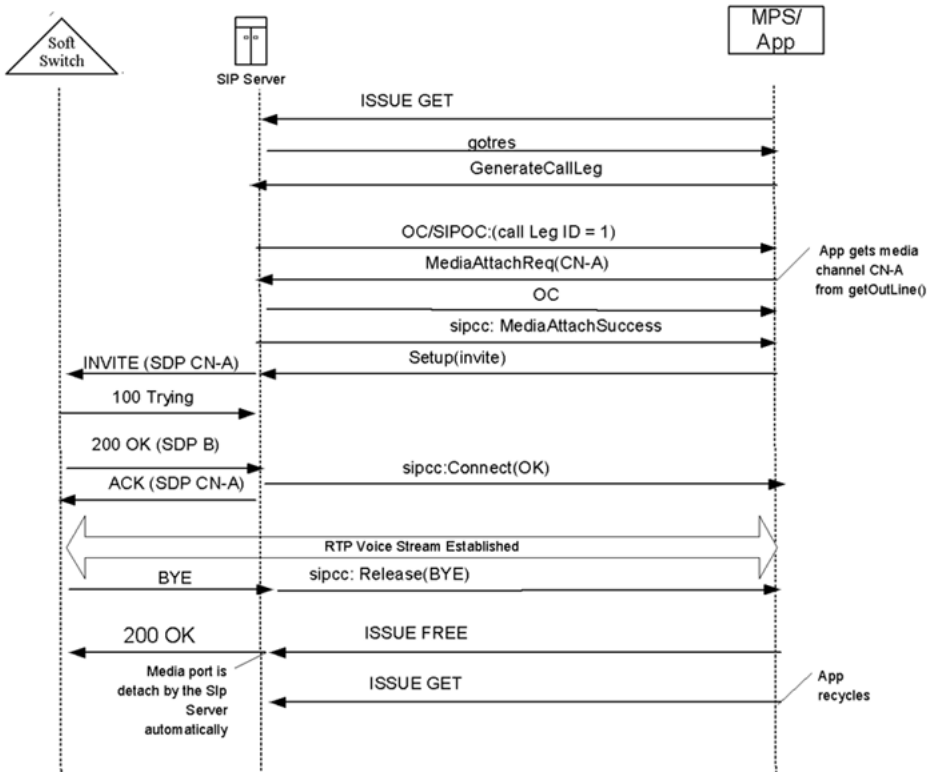
Application Releases



Application Terminates

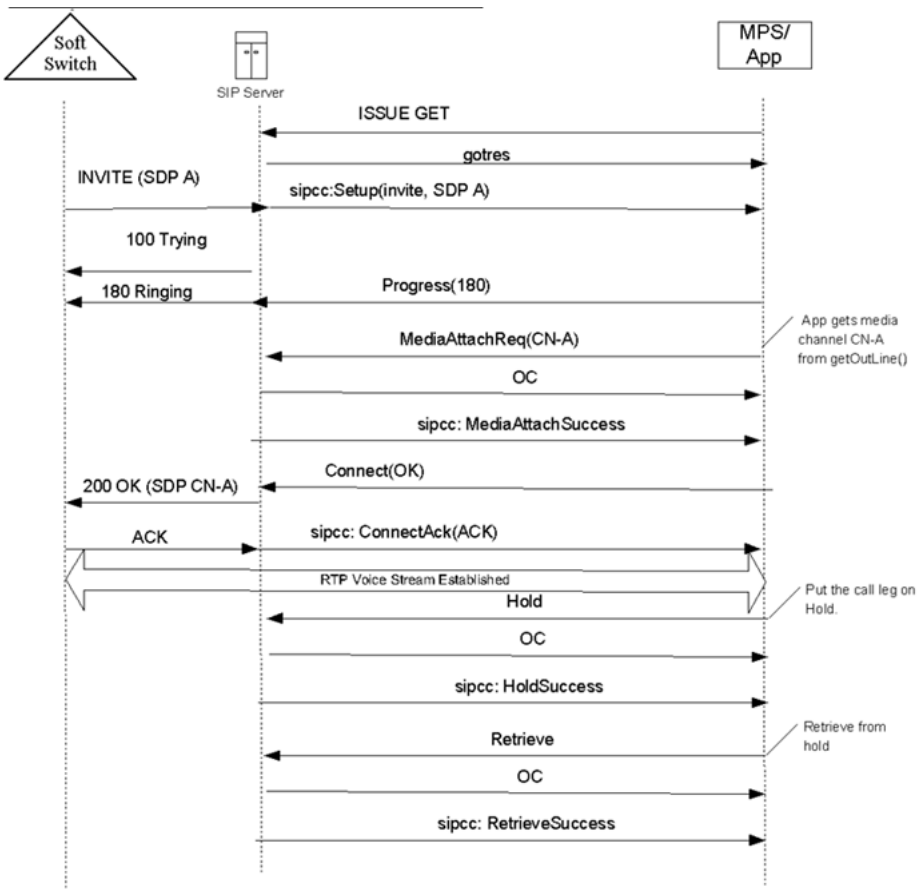


Outbound Call



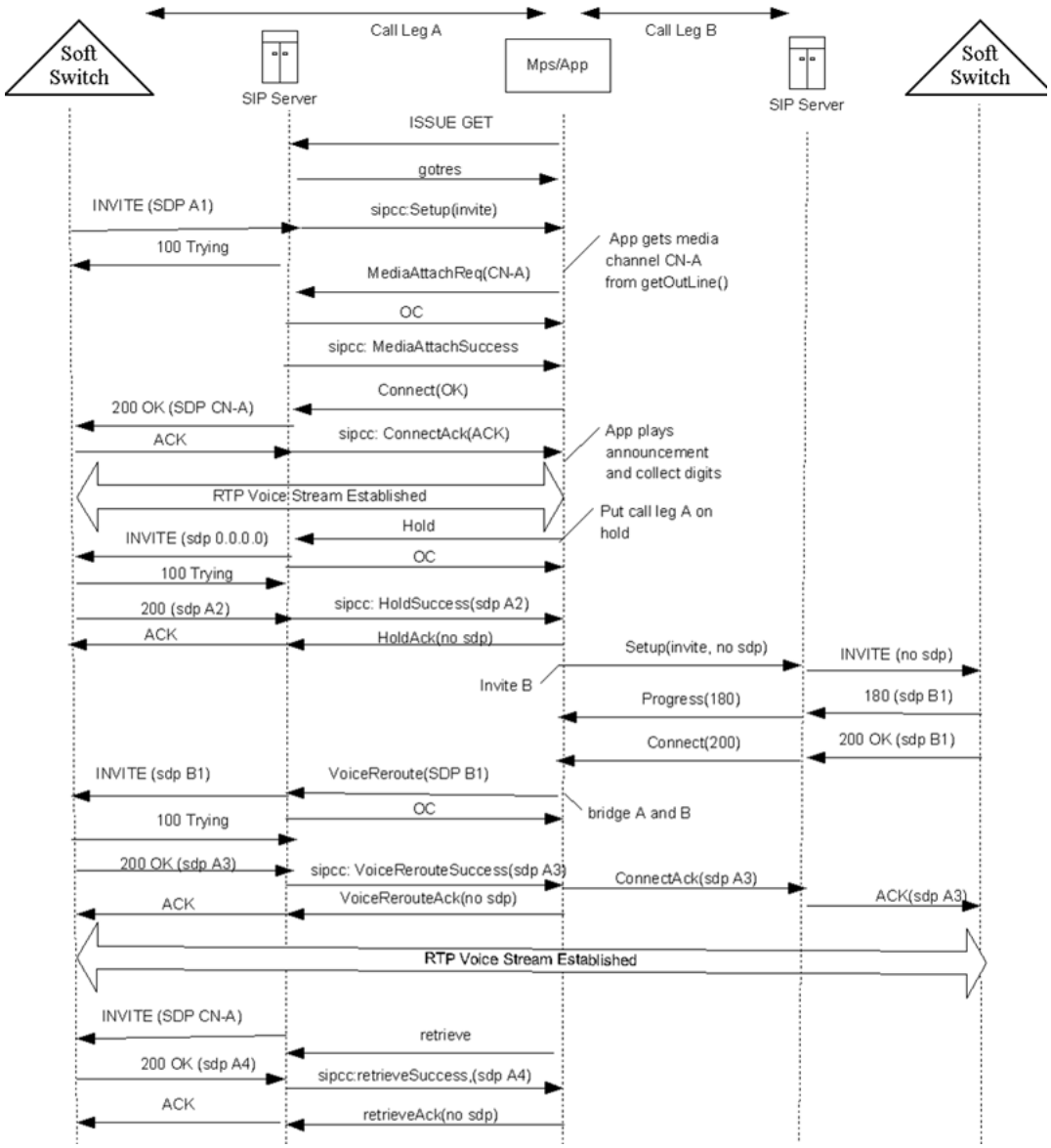
Hold and Retrieve

This is done by sending Hold with an empty message body.

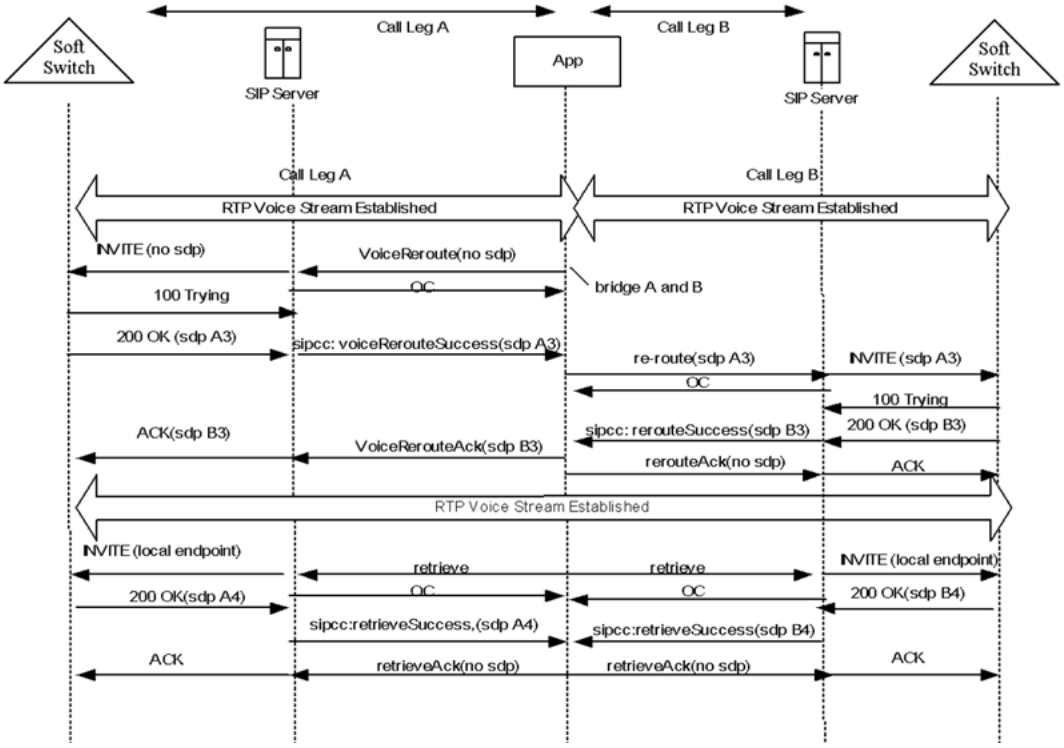


Prepaid Call

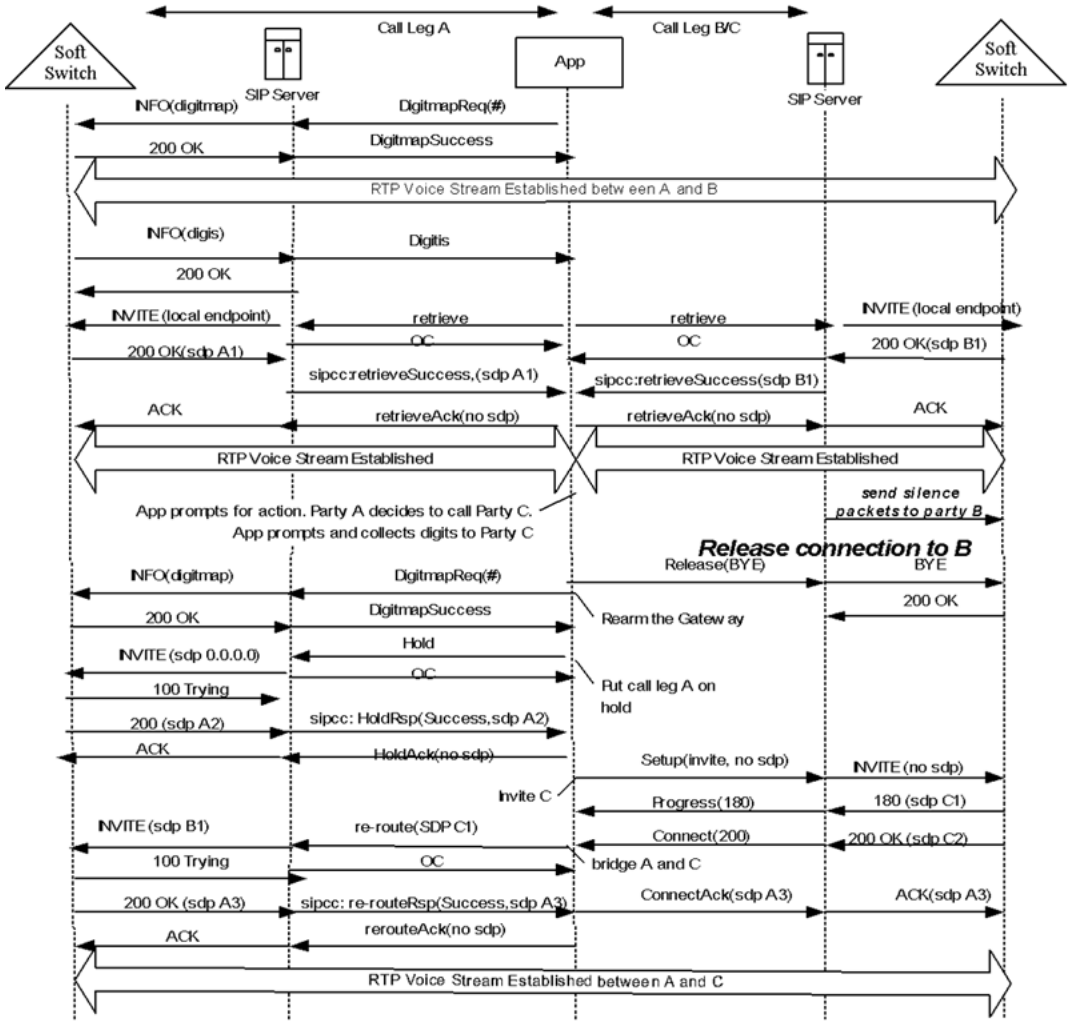
Bridging the voice on the network is done by sending the SDP of the B-leg to the A-leg through Re-Route. To bring the voice path back to the MPS, issue Retrieve on the call legs with Local TPM endpoint. (This is not shown in the following diagram.)


Call Leg A
Call Leg B
Soft Switch
SIP Server
Mps/App
SIP Server
Soft Switch
ISSUE GET
gotres
INVITE (SDP A1)
sipcc:Setup(invite)
100 Trying
App gets media channel CN-A from getOutLine()
MediaAttachReq(CN-A)
OC
sipcc: MediaAttachSuccess
Connect(OK)
200 OK (SDP CN-A)
ACK
sipcc: ConnectAck(ACK)
App plays announcement and collect digits
RTP Voice Stream Established
Put call leg A on hold
Hold
INVITE (sdp 0.0.0.0)
OC
100 Trying
200 (sdp A2)
sipcc: HoldSuccess(sdp A2)
ACK
HoldAck(no sdp)
Setup(invite, no sdp)
INVITE (no sdp)
Invite B
Progress(180)
180 (sdp B1)
Connect(200)
200 OK (sdp B1)
INVITE (sdp B1)
VoiceReroute(SDP B1)
OC
bridge A and B
100 Trying
200 OK (sdp A3)
sipcc: VoiceRerouteSuccess(sdp A3)
ConnectAck(sdp A3)
ACK
VoiceRerouteAck(no sdp)
ACK(sdp A3)
RTP Voice Stream Established
INVITE (SDP CN-A)
retrieve
200 OK (sdp A4)
sipcc:retrieveSuccess,(sdp A4)
ACK
retrieveAck(no sdp)

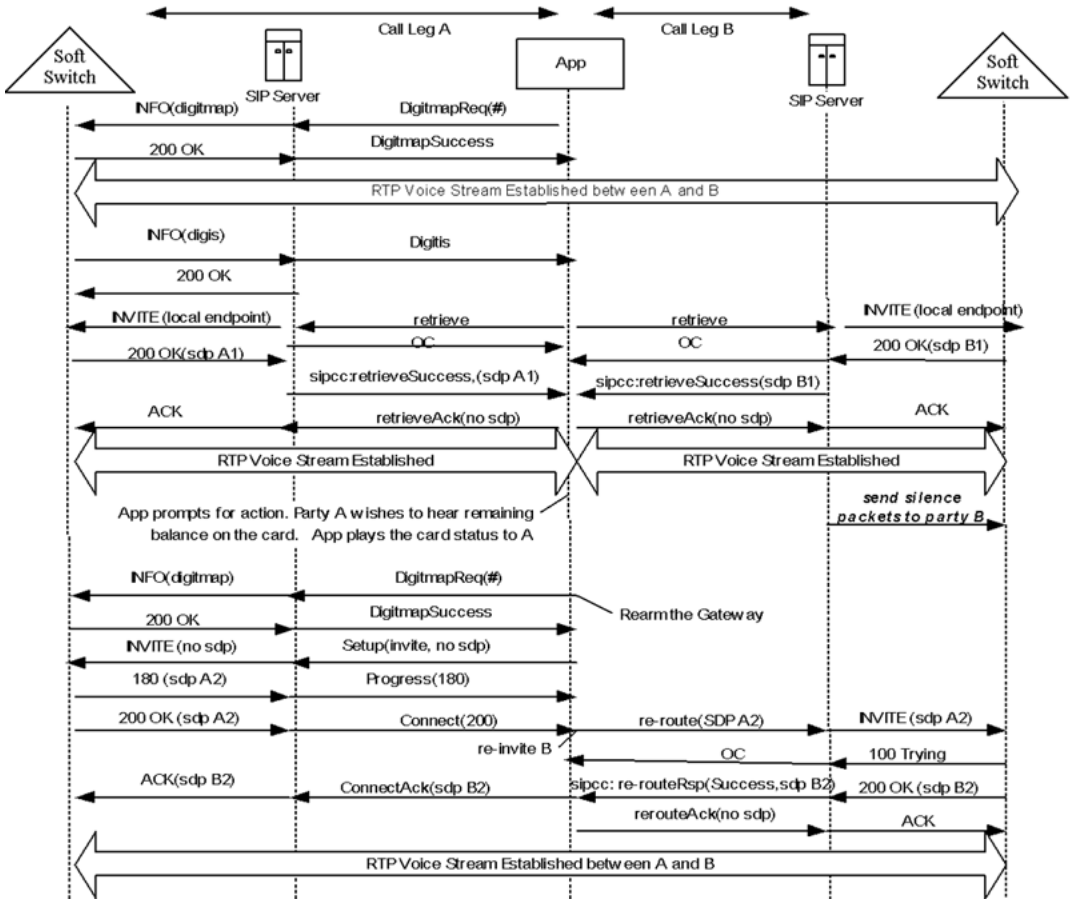
Voice Bridge



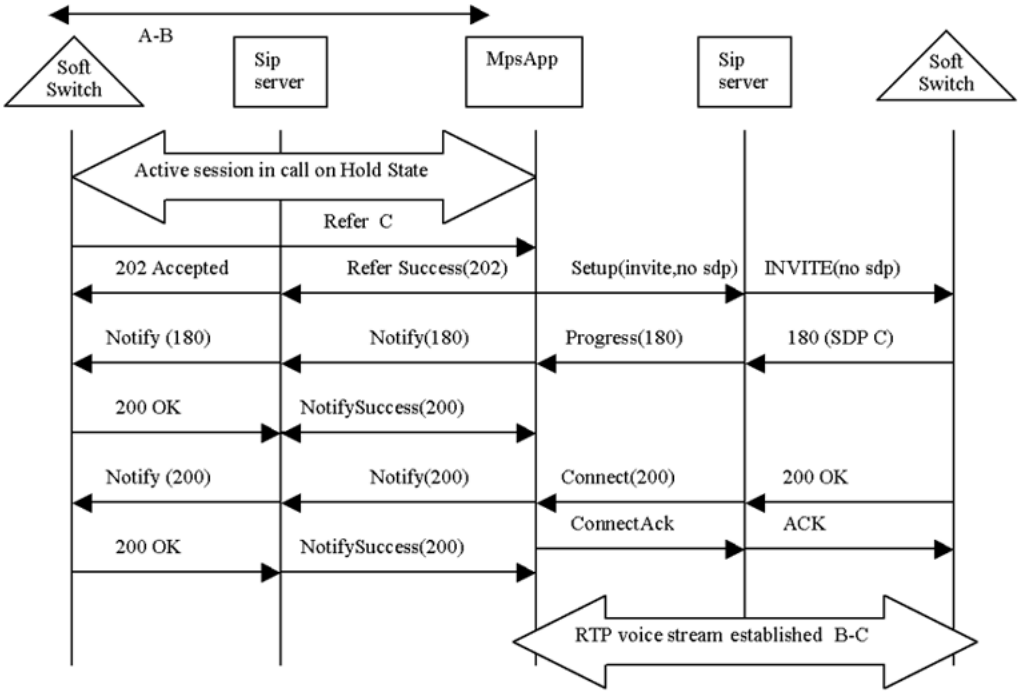
Re-origination



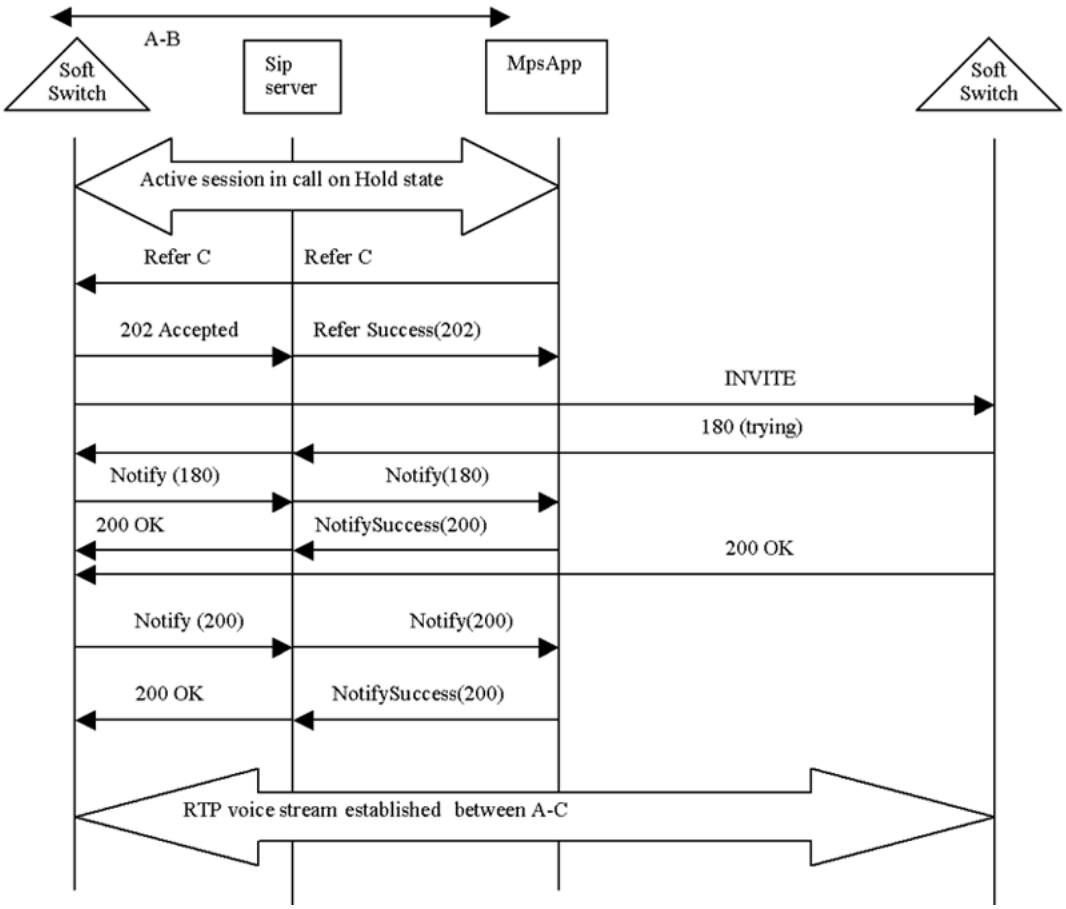
Mid-Call Card Status



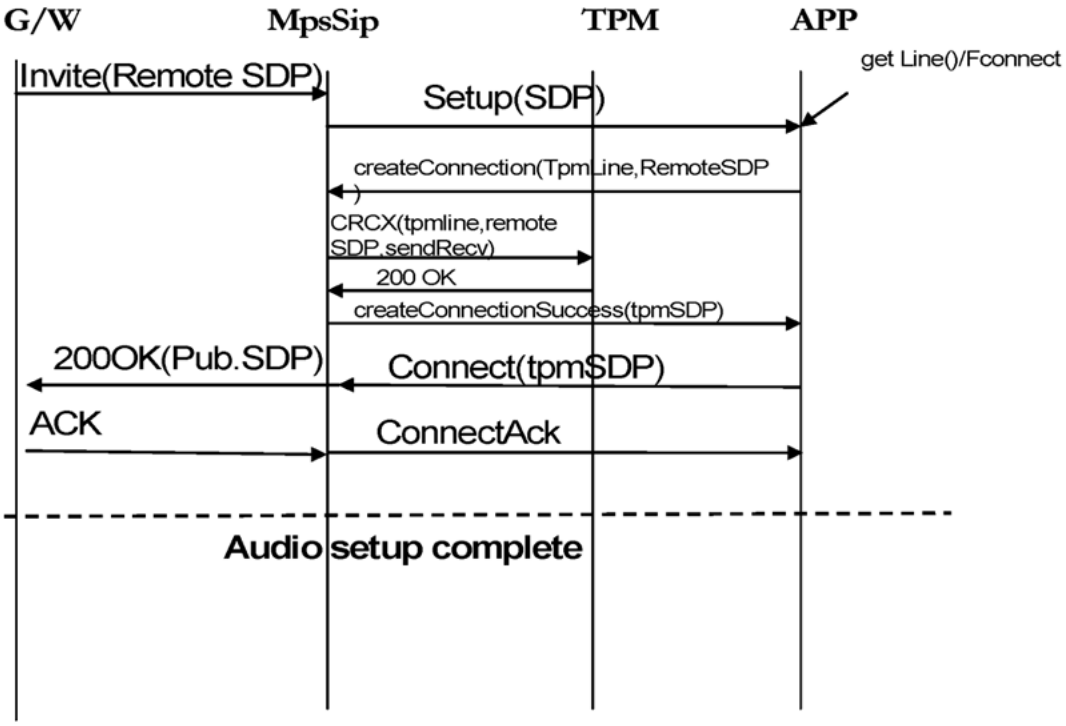
Application Receives Refer



Application Sends Refer



Inbound Call on ASE Mode using Create Connection Request

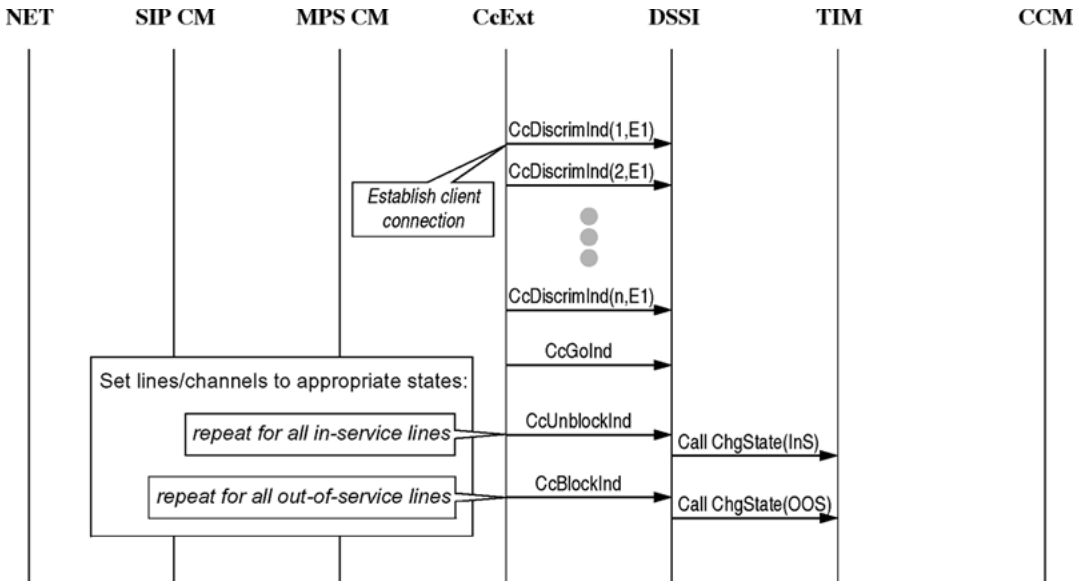


TMS Integrated Mode Message Flows

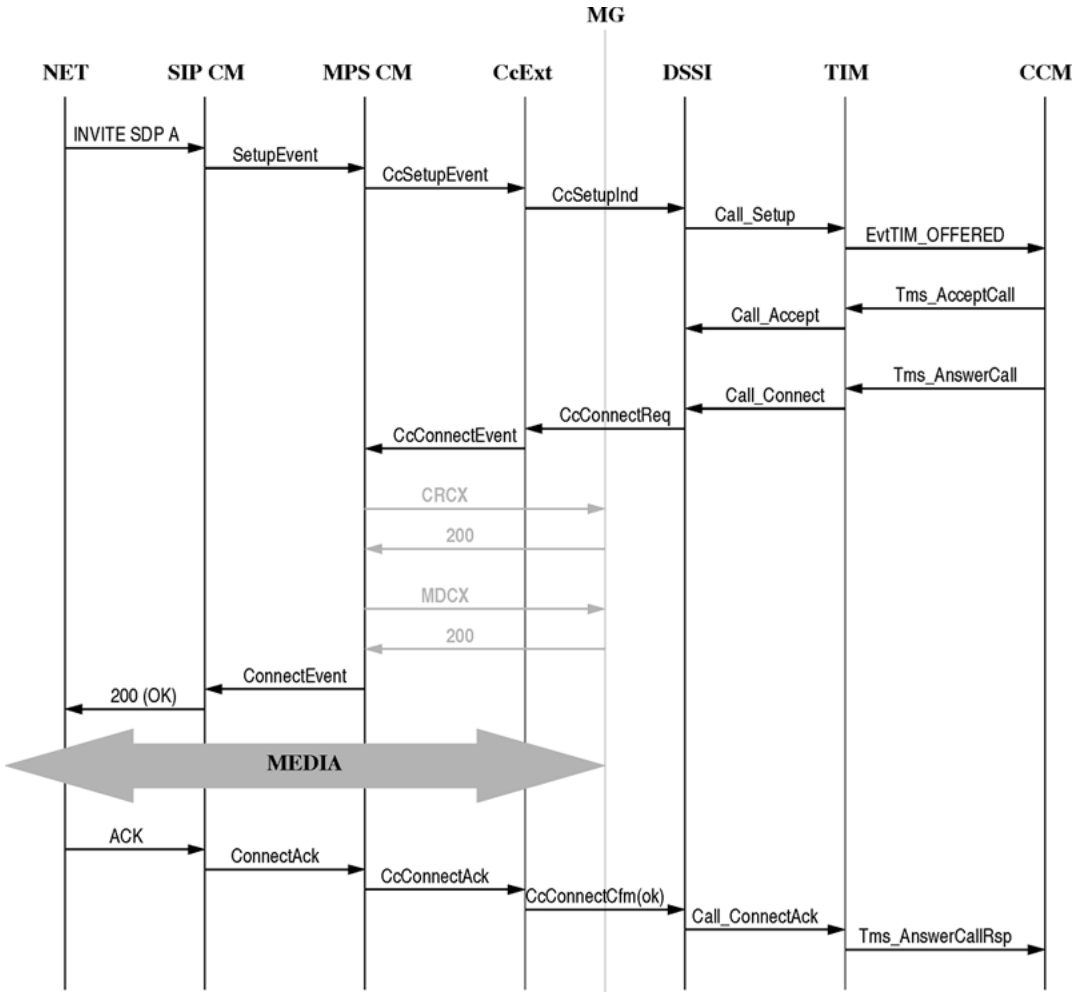
The following chapter details message flows in different call scenarios.

Initial Client Connectivity

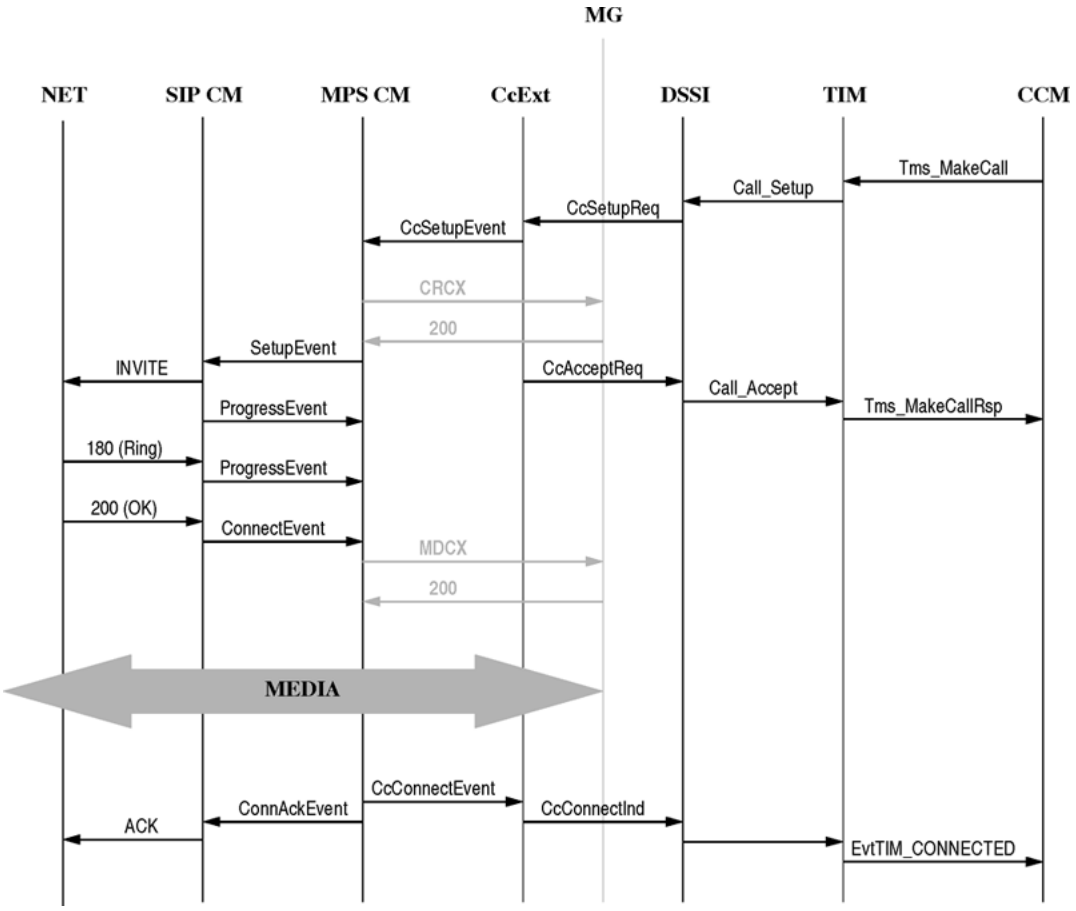
When the client establishes a connection, discrim messages are sent for each span, indicating the span number and type (E1). Systems having a TPM-200 with 60 media ports or a TPM-800 with 240 media ports are segmented by 30 channels (similar to E1 type) with 2 spans in TPM-200 and 8 spans for TPM-800.


NET
SIP CM
MPS CM
CcExt
DSSI
TIM
CCM
CcDiscrimInd(1,E1)
CcDiscrimInd(2,E1)
Establish client connection
CcDiscrimInd(n,E1)
CcGoInd
Set lines/channels to appropriate states:
repeat for all in-service lines
CcUnblockInd
Call ChgState(InS)
repeat for all out-of-service lines
CcBlockInd
Call ChgState(OOS)

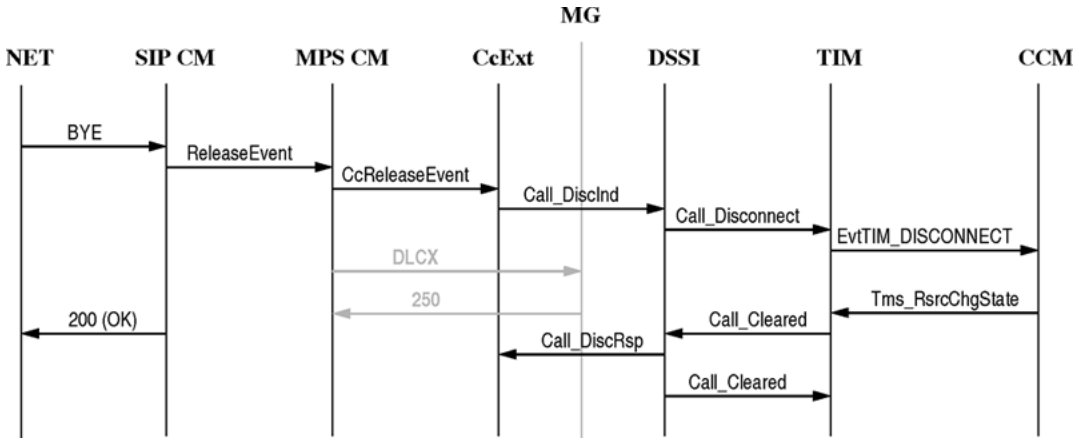
Inbound Call Setup



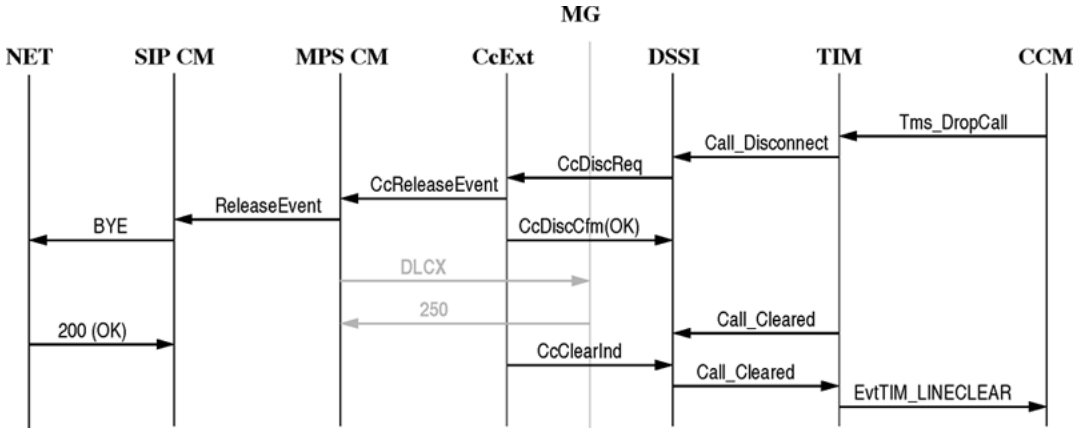
Outbound Call Setup



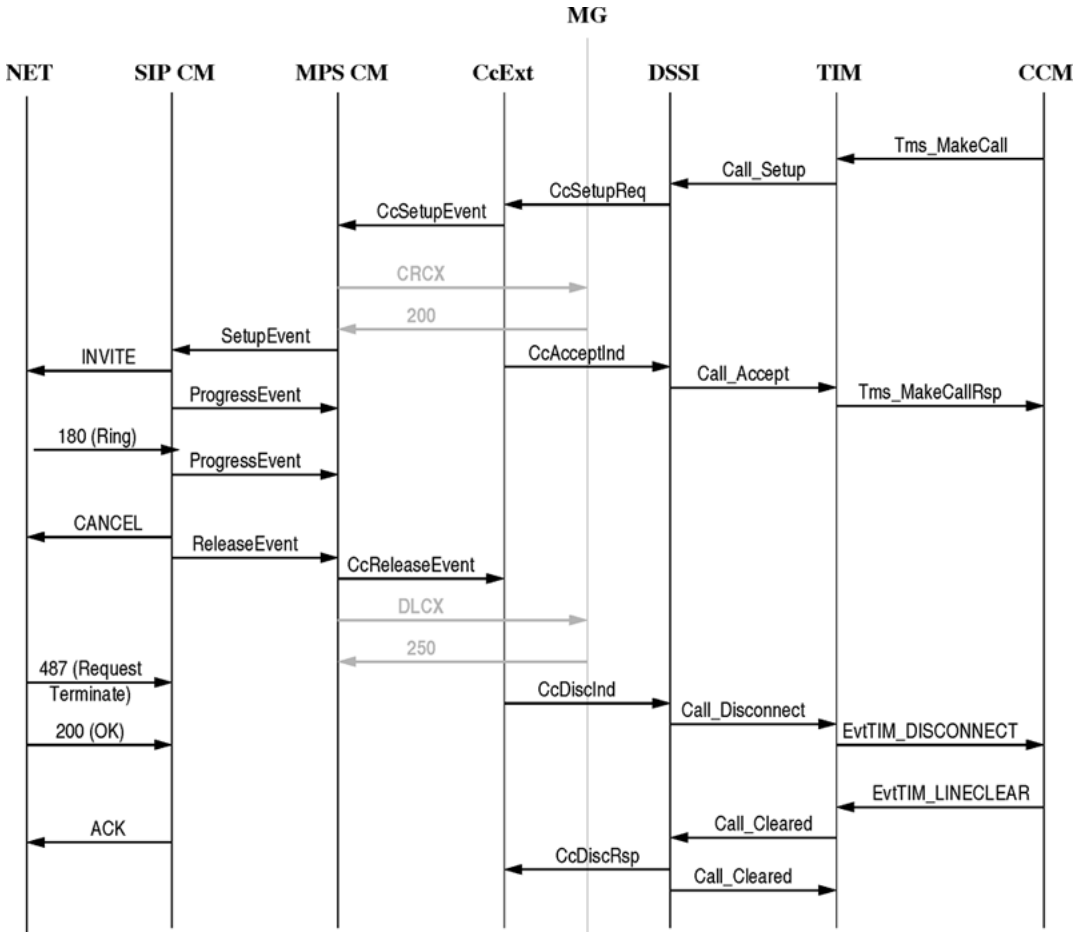
Network Disconnect



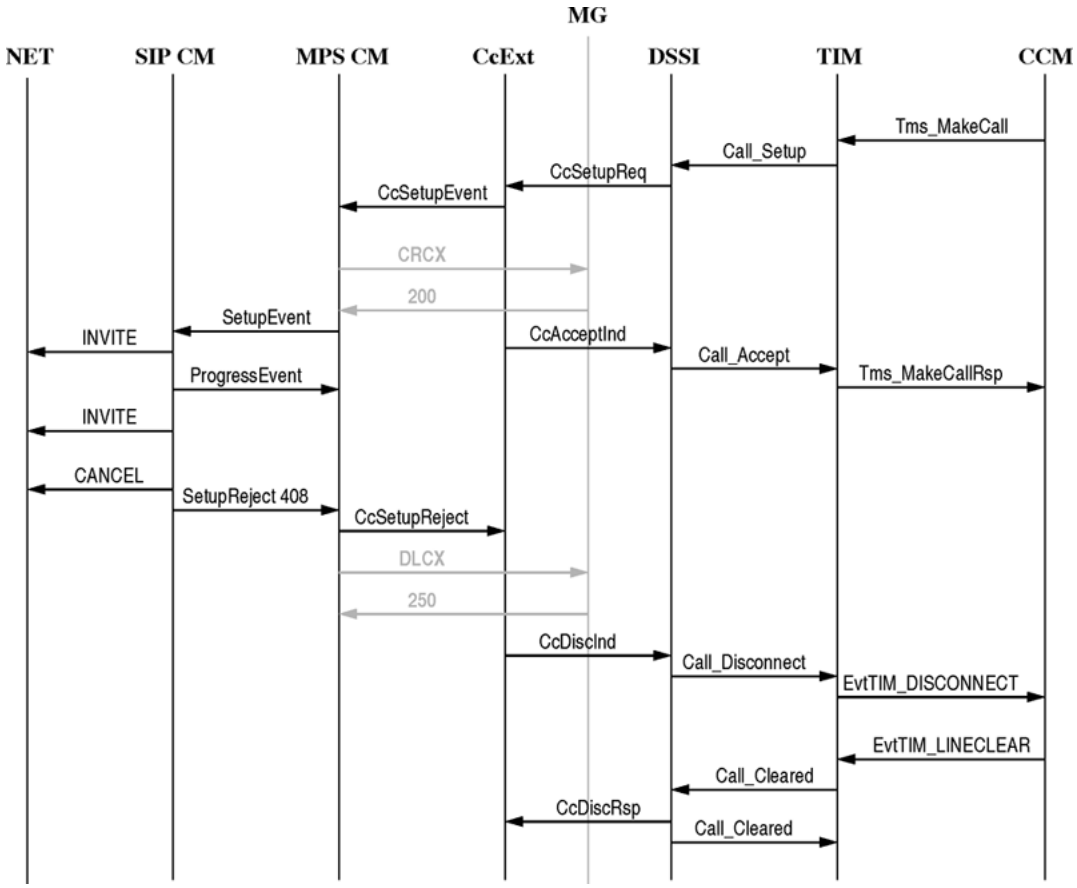
Application Disconnect



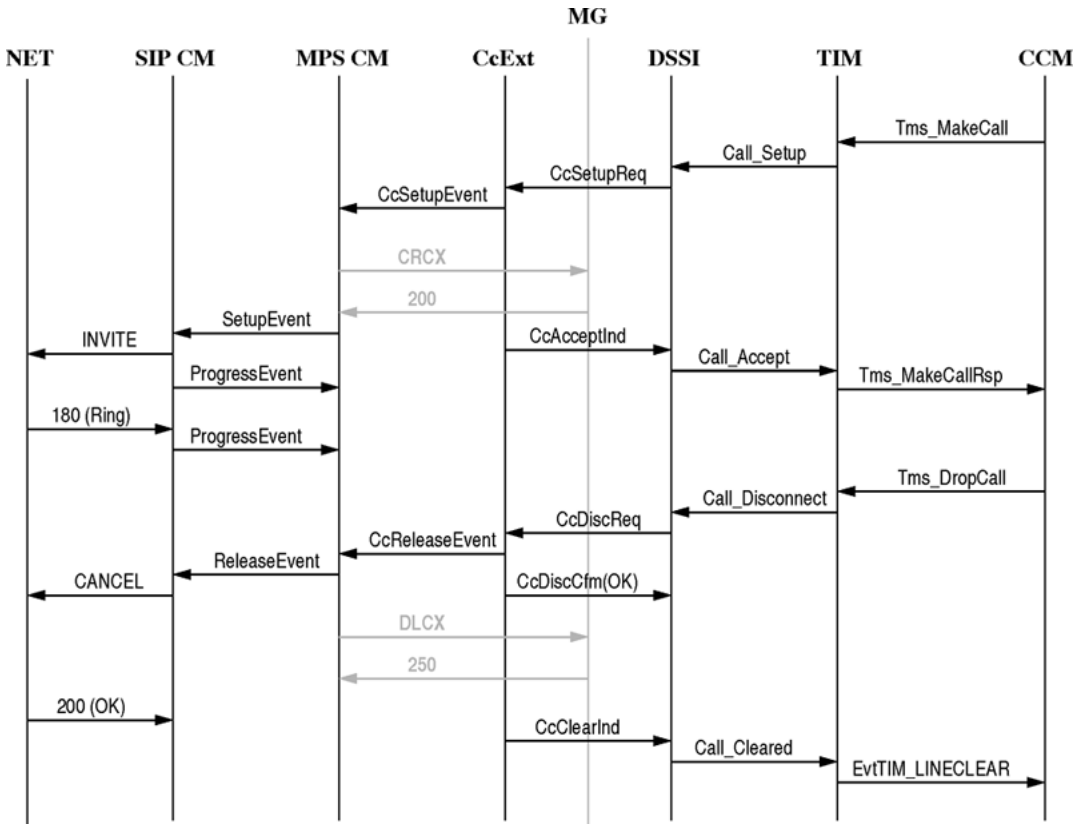
Outgoing Call - No Answer



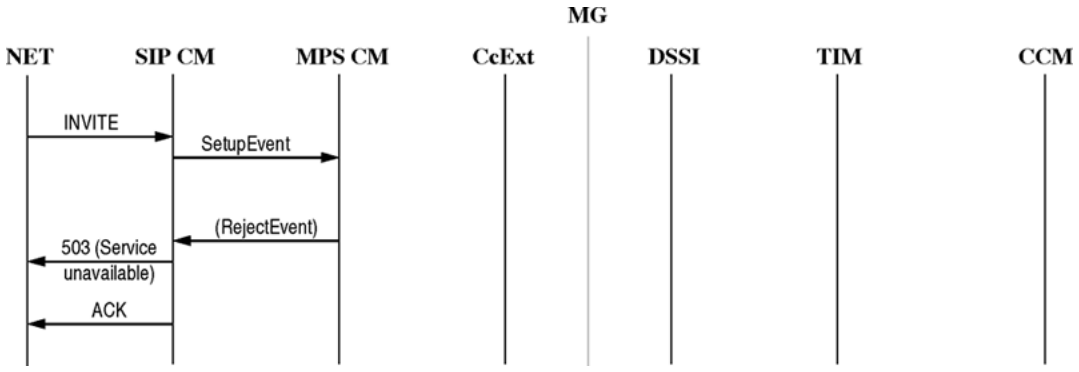
Outgoing Call - No Response, Setup Reject



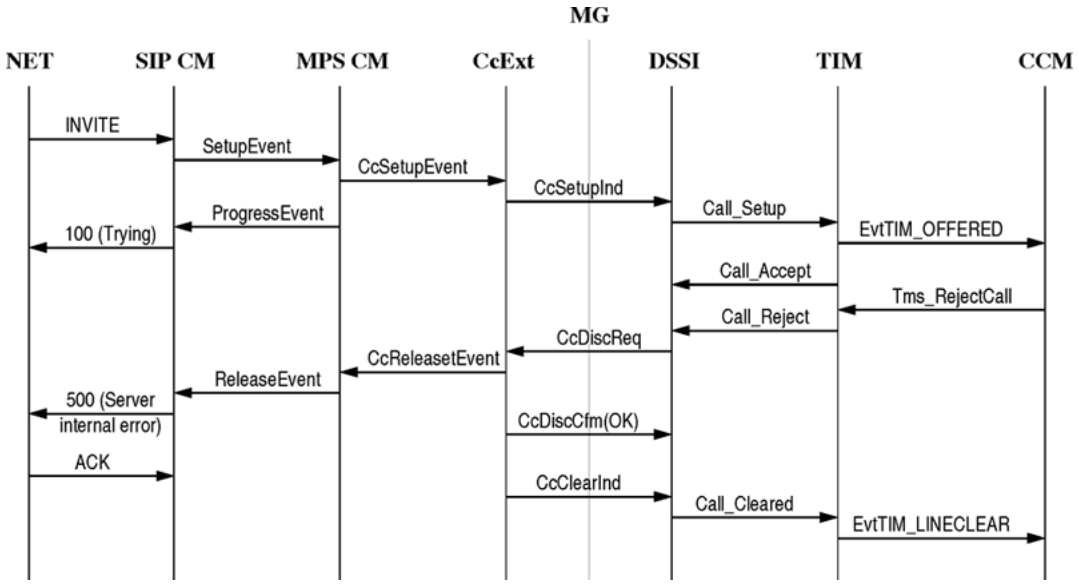
Outgoing Call - Application Drops Call before Far End Answers



Incoming Call - No Application Available

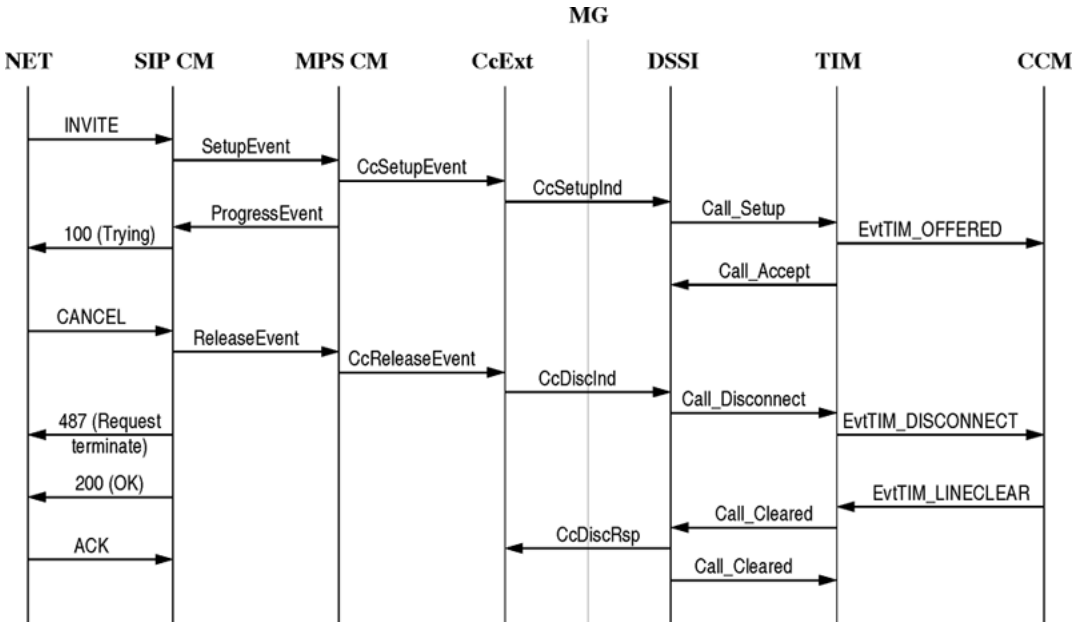


Incoming Call - Rejected by TIM

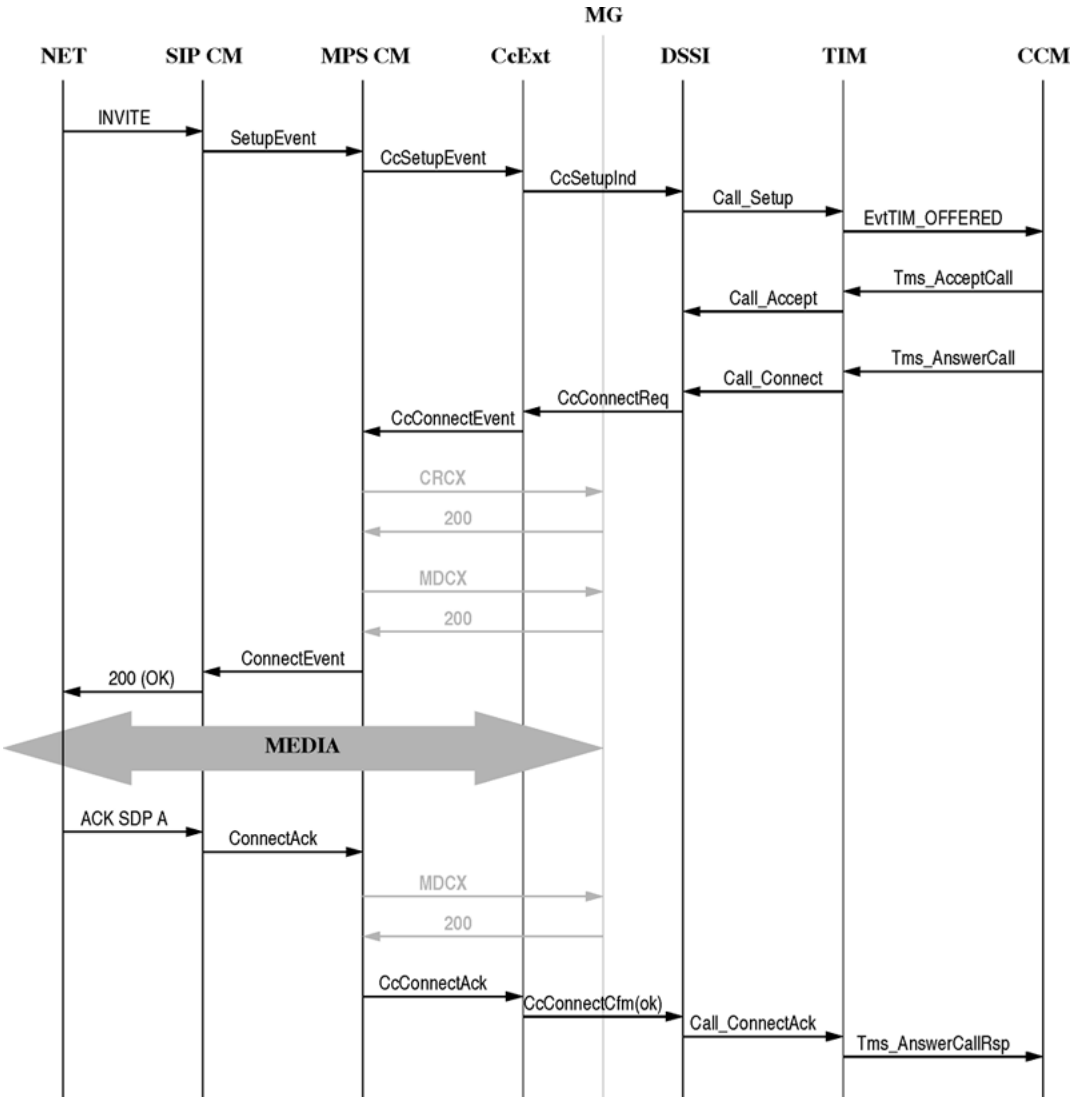


Incoming Call - Not Answered

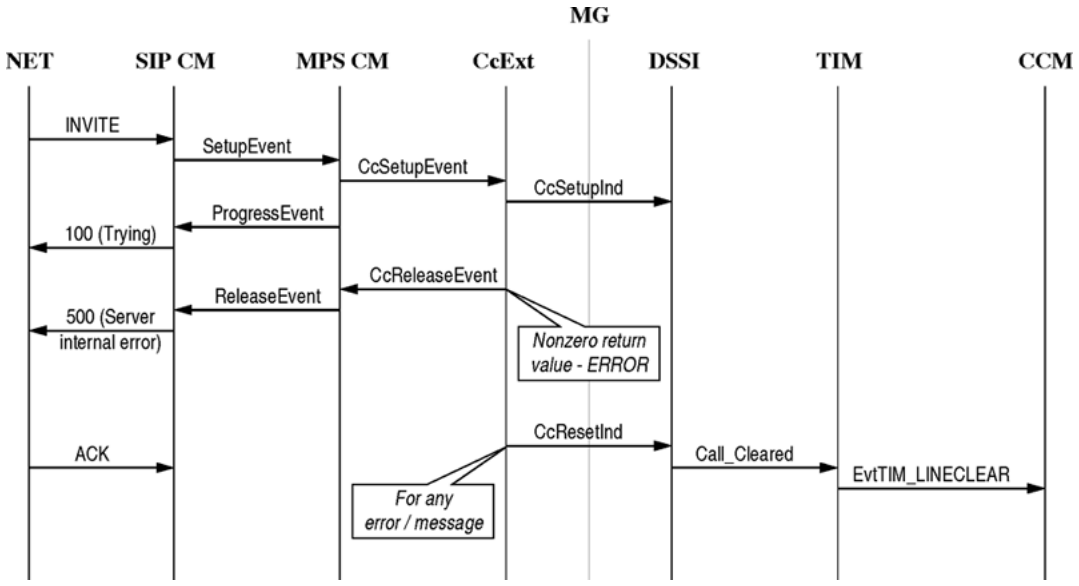
In this scenario, it is required that the far end send the CANCEL message when the MPS does not answer the incoming call.


MG
NET
SIP CM
MPS CM
CcExt
DSSI
TIM
CCM
INVITE
SetupEvent
CcSetupEvent
CcSetupInd
Call_Setup
EvtTIM_OFFERED
ProgressEvent
100 (Trying)
Call_Accept
CANCEL
ReleaseEvent
CcReleaseEvent
CcDiscInd
Call_Disconnect
EvtTIM_DISCONNECT
487 (Request terminate)
200 (OK)
EvtTIM_LINECLEAR
Call_Cleared
CcDiscRsp
ACK
Call_Cleared

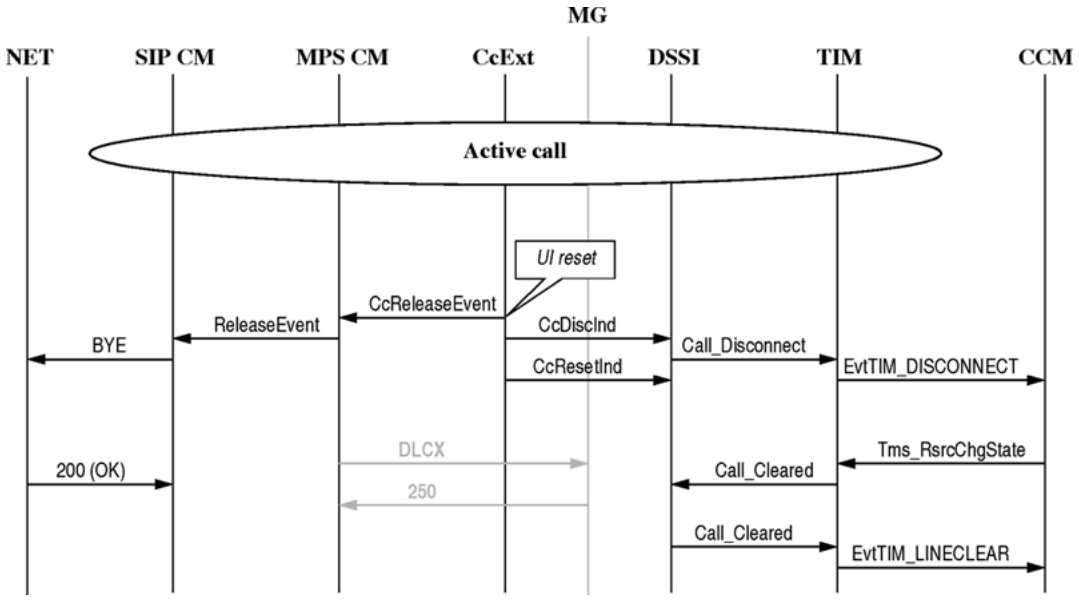
Incoming Call - Slow Start



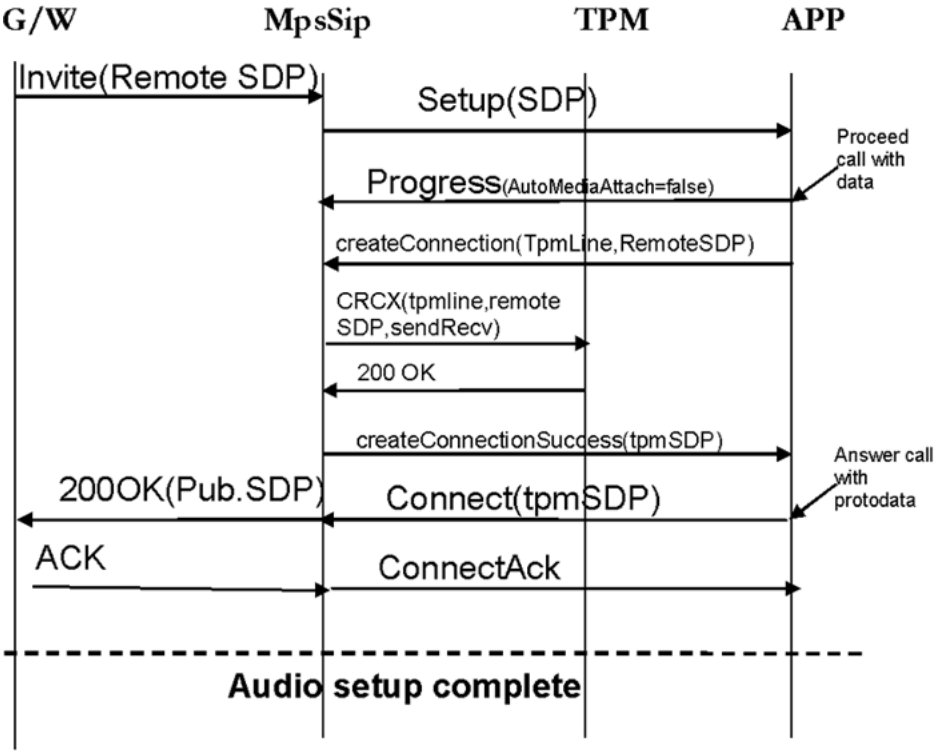
Error Sending CcExt Message to DSSI



UI Resets



Inbound Call on TMS Mode using Create Connection Request



Configuration Information Worksheets

CCSS SIP

CCSS server name	Public IP address or hostname	Port number (default: 5060)

SIP Network Servers

SIP network Server name	Public IP address or hostname	Port number (Default:5060)	SIP domain name	Supports registrar (Y/N)

SIP URL Registration Information

User name	SIP domain	Password

TPM VoIP Gateway Module

TPM name	Public IP address or hostname	Ethernet address	TPM type (TPM200 or TPM800)	Default gateway/ router IP address or hostname

TMS/TPM

Chassis	Back plane slot# (BPS)	TMS IP address or hostname	TMS secondary IP address or hostname (optional)	Ethernet address	Corresponding TPM name

MPS Information

MPS IP address or hostname	MPS IP secondary address or hostname (optional)	MPS VOS group number	Line range

Alarms

Alarms

Alarms are viewed in the alarm window. Alarm logs are placed in $CCSSHOME/log/sip/Log in Unix and %common/log/alarm.ccss.N.log in Windows systems.

Note:

The alarms field '%s' in the alarm message displayed, represents a character string. The alarms field '%d' in the alarm message displayed, represents a decimal value. The alarms field '%x' in the alarm message displayed, represents a hexadecimal value.

For example, if the alarm message is (%d,%d,%d) T5 expired, circuit down, a possible display would be (0,44,3). This corresponds to point code # always 0, trunk group number 44, channel 3.

Configuration

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
Type: Configuration			
INFO	0	Configuration file loaded + filename	Sip server is loading the configuration information from sip.conf and sipdefault.properties.
INFO	0	loadProps file + propFileName	Sip server is loading the configuration information from sip.conf and sipdefault.properties.
INFO	0	[NETWORK] section configured with no CLIENT_ACCESS entry	[NETWORK] section configured with no CLIENT_ACCESS entry.
MAJOR	0	File not found + filename	The configuration file is not found. Check if sip.conf and aipdefault.properties are

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
			presented in the correct directory.
MAJOR	0	Unable to set initial self context. Unknown host: hostname	Sip server host name is not valid. Check sip.conf SYSTEM section, verify the SIP SERVER HOST NAME is valid.
MAJOR	0	Configuration error, no SIP_TIMER_PROFILE entry defined for: profilename.	The sip timer profile with name profilename is not defined. Add the definition to SIP_TIMER_PROFILE.
MAJOR	0	KEY: parsing error	The number or format of parameters is incorrect for this KEY. Check the number or format of parameters for this KEY.
MAJOR	0	parsing error CallControlSection : CC_LINEMAP: HOST ID: No CC_HOSTS entry found for this HOST ID.	The host id in CC_LINEMAP is not defined in a CC_HOSTS entry. Define the host id in a CC_HOSTS entry.
MAJOR	0	parsing error MediaGatewaySection: MG_LIST: HOST NAME: invalid host name.	The hostname in this MG_LIST entry is not a valid hostname. Supply a valid hostname.
MAJOR	0	parsing error NetworkSection: NETWORK_HOSTS: PRIMARY HOSTNAME : invalid Host name	The hostname in this NETWORK_HOSTS entry is not a valid hostname. Supply a valid hostname.
MAJOR	0	parsing error SystemSection :SYSTEM:RE DUND ENABLE: invalid value, has to be "true" or "false".	The value for REDUND ENABLE is not valid. Put "true" if the system is redundant enabled, "false" otherwise.
MAJOR	0	configuration error, no CONTACT_PROFILE entry defined for cpName.	In REGISTRATION_CONTROL section, there is no CONTACT_PROFILE entry defined with name:cpName. The contact profile name must be a defined profile.
MAJOR	0	configuration error REGISTRATION_CONTROL: REG_LINE_CONFIG, no	The hostId used in REG_LINE_CONFIG entry is

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
		CC_HOSTS entry defined with HOST ID: hostId.	not correct. Supply a valid hostId.
MAJOR	0	configuration error CALL_CONTROL:CC_HOSTS: no HOST_PROFILE entry defined for pName.	In CALL_CONTROL section, there is no HOST_PROFILE entry defined with name:pName. The host profile name must be a defined profile.
MAJOR	0	configuration error, MEDIA_GATEWAY:MG_LIST: no MG_PROFILE entry defined for pName.	In MEDIA_GATEWAY section, there is no MG_PROFILE entry defined with name:pName. The mg profile name must be a defined profile.
MAJOR	0	configuration error MEDIA_GATEWAY:MG_LIST: no TMS_HOST_PROFILE entry defined for tmsname	The tmsname specified in this MG_LIST entry is not defined in TMS_HOST_PROFILE. The tms host profile name must be a defined profile.
MAJOR	0	configuration error, type of IP transport for CLIENT_ACCESS has to be UDP.	The transport type configured for CLIENT_ACCESS is not correct. The type of transport for CLIENT_ACCESS has to be UDP.
MAJOR	0	configuration error in [NETWORK] section: there has to be either CLIENT_ACCESS entry or NETWORK_HOSTS entry or both.	CLIENT_ACCESS and NETWORK_HOSTS both are empty. CLIENT_ACCESS and NETWORK_HOSTS can not be both empty.
MAJOR	0	[NETWORK] section configured with no NETWORK_HOSTS entry	[NETWORK] section configured with no NETWORK_HOSTS entry. Remove or comment the [REGISTRATION_CONTROL] section and then restart.
MAJOR	0	The proxy SS2 of avaya.com is not reachable	The Keep Alive option is enabled and the proxy is considered to be unreachable. Check network availability.
Type: System Family			

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
INFO	e	SystemName :systemName More than one active file found in Path	More than one active file found in path.
INFO	1	OM Agent mailbox full	Message noting that the Om Agent mailbox is full.
INFO	1	OM Agent mailbox full OMAgent trying to post to OMRecSysTask.	OM Agent mailbox is full. OMAgent trying to post to OMRecSysTask.
INFO	1	OM Agent mailbox full MpsOMAgent trying to post to OMRecSysTask	OM Agent mailbox is full. MpsOMAgent trying to post to OMRecSysTask.
INFO	1	OM Collector mailbox full OMCollector trying to post to File Writer Task	OM Collector mailbox is full. OMCollector trying to post to File Writer Task.
INFO	3	OM Collector mailbox inactive OMCollector::methodName	OM Collector mailbox is inactive.
INFO	3	OM Collector mailbox inactive OMCollector trying to post to File Writer Task	OM Collector mailbox is inactive. OMCollector trying to post to File Writer Task.
INFO	3	OM Agent mailbox inactive MpsOMAgent trying to post to OMRecSysTask	OM Agent mailbox is inactive. MpsOMAgent trying to post to OMRecSysTask
INFO	3	OM Agent mailbox inactive OMAgent trying to post to OMRecSysTask	OM Agent mailbox is inactive. OMAgent trying to post to OMRecSysTask
INFO	3	OM Agent mailbox inactive	Message noting that the OM Agent mailbox is inactive.
INFO	4	OM File Retention Period: retentionPeriod	OM File Retention Period.
INFO	5	OM Office Transfer Period: value	OM Office Transfer Period.
INFO	6	OMCollector:handleMENoResponse received - reason:	OMCollector:handleMENo Response received.
INFO	6	OM MEResponse error in OMAgent::sendOMGroup QueryToEM	There is error in OM MEResponse in OMAgent.Send OM Group Query to EM.
INFO	6	OM MEResponse error in OMAgent::sendOMGroup NamesQueryToEM	There is error in OM MEResponse in

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
			OMAgent.Send OM Group names to EM.
INFO	6	OM MEResponse error in OMAgent::sendOMGroupAll QueryToEM	There is error in OM MEResponse in OMAgent.Send all OM Group query to EM.
INFO	6	ME: name cannot send OM Group: GroupName data to FPM: pmName	ME: cannot send OM GroupName data to FPM.
INFO	6	OM MEResponse error in MpsOMAgent::sendOMGroup QueryToEM	SendOMGroupQueryToEM.
INFO	6	OM MEResponse error in MpsOMAgent::sendOMGroup NamesQueryToEM	SendOMGroupNamesQuery ToEM.
INFO	6	OM MEResponse error in MpsOMAgent::sendOMGroup AllQueryToEM	SendOMGroupAllQueryToEM.
INFO	7	InvalidOMGroupName Exception in handleMEEvent of OMAgent: omGroupNameinvalid – omGroupNameStr	InvalidOMGroupName.
INFO	7	OMException in handleMEEvent of OMAgent: omGroupName does not exists - omGroupName	OmGroupName does not exist.
INFO	7	SystemName : sysName Active File Extension Updated: ActExtension	Active File Extension updated.
INFO	7	SystemName : sysName Closed File Extension Updated: ClosedExtension	Closed File Extension updated.
INFO	7	InvalidOMGroupNameExcepti on in handleMEEvent of MpsOMAgent: omGroupName invalid	Invalid OMGroupName.
INFO	7	OMException in handleMEEvent of MpsOMAgent: omGroupName does not exists	MpsOMAgent does not exist.

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
INFO	7	OMGroup Query command failure. Reason:	OMGroup Query command failure.
INFO	7	OMGroup names Query command failure. Reason:	OMGroup names Query command failure.
INFO	7	OMGroup QueryAll command failure. Reason:	OMGroup QueryAll command failure.
INFO	28d	OMGroup QueryAll command failure	Periodic Call Audit failed.
INFO	258	<SIP/MPS/ADT> CPS [%d-sec-avg]:"+ %d	Periodic report of average calls per second.
INFO	320	<SIP/SYS/ADM> "current state" ◊ "new state"	System state transition information.
INFO	320	Uninitialized	Indicates that the state is Uninitialized.
INFO	320	Uninitialized→Init Requested	Indicates State change from Uninitialized to Init Requested.
INFO	320	Init Requested→Bootstrapping	Indicates state change from Init Requested to Bootstrapping.
INFO	320	Bootstrapping→Initializing Static	Indicates state change from Bootstrapping to Initializing Static.
INFO	320	Initializing Static→Initializing Dynamic	Indicates state change from Initializing Static to Initializing Dynamic.
INFO	320	Initializing Dynamic→Cold Standby	Indicates state change from Initializing Dynamic to Cold Standby.
INFO	320	Cold Standby→Activation Requested	Indicates state change from Cold Standby to Activation Requested.
INFO	320	Activation Requested→Activating	Indicates state change from Activation Requested to Activating.
INFO	320	Activating→Active	Indicates state change from Activating to Active.
INFO	320	Deactivation Requested→Deactivating	Indicates state change from Deactivation Requested to Deactivating.

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
INFO	320	Deactivation Requested	Deactivation requested.
INFO	320	Deactivating→Shutting Down	Indicates state change from Deactivating to Shutting Down.
INFO	320	Shutting Down→Shutdown	Indicates state change from Shutting Down to Shutdown.
INFO	321	Delaying 100 milliseconds before system shutdown	Notification of system shutdown.
INFO	322	Fault System is shutting down	The Fault System is shutting down.
INFO	323	FPM not provisioned	No FPM is available on the CM.
INFO	324	<SIP/SYS/ADM> Activating Cold	Activating system from cold standby.
INFO	324	Activating Warm	Activating system from warm standby.
INFO	325	Aborting synchronization from peer	Aborting synchronization from peer.
INFO	325	Terminating synchronization from peer	Terminating synchronization from peer.
INFO	326	Checkpoint sending started	Checkpoint sending started.
INFO	326	Checkpoint sending terminated	Checkpoint sending terminated.
INFO	326	Checkpoint sending aborted	Checkpoint sending aborted.
INFO	326	Checkpoint receiving started	Checkpoint receiving started.
INFO	326	Checkpoint receiving terminated	Checkpoint receiving terminated.
INFO	326	Checkpoint receiving aborted	Checkpoint receiving aborted.
INFO	384	Peer restarted before peer failure was detected	Peer restarted before peer failure was detected.
MINOR	2BC	<SIP/SYS/ABN> A desired configuration is required before starting the system.	Internal initialization error. Contact Avaya personnel.
CRITICAL	ca	Critical call processing overload condition. The call attempts are throttled at "CRITICAL_OVERLOAD_ THROTTLE_RATE" percent	Critical call processing overload condition. The call attempts are throttled at "CRITICAL_OVERLOAD_ THROTTLE_RATE" percent.

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
CRITICAL	c9	The call attempt rate has exceeded the licensed limit of bhcaLicensedRate “busy-hour-call-attempts". Incoming calls will be throttled until the attempt rate drops below the licensed limit.	The call attempt rate has exceeded the licensed limit of bhcaLicensedRate “busy-hour-call-attempts". Incoming calls will be throttled until the attempt rate drops below the licensed limit.
CRITICAL	d2	Recording System: Cannot create base file path	Recording System: Cannot create base file path.
CRITICAL	d3	Recording System: Cannot open a new path	Recording System: Cannot open a new path
CRITICAL	2bd	Peer instance has been activating for at least "dur" milliseconds and is thus presumed failed. Dispatching assassin to kill peer instance.	Peer instance has been activating for at least "dur" milliseconds and is thus presumed failed. Dispatching assassin to kill peer instance.
CRITICAL	2bd	Peer instance has been deactivating for at least "dur” milliseconds and is thus presumed failed. Dispatching assassin to kill peer instance.	Peer instance has been deactivating for at least "dur” milliseconds and is thus presumed failed. Dispatching assassin to kill peer instance.
CRITICAL	6c	Sip Message from the Application can not be parsed successfully. Could not be delivered to the MPSCM.	Sip Message from the Application can not be parsed successfully. Could not be delivered to the MPSCM.
CRITICAL	65	Peer instance presumed failed.	Peer instance presumed failed.
CRITICAL	66	Receiving status from peer for which I am not provisioned.	Receiving status from peer for which I am not provisioned.
CRITICAL	68	Network isolation detected; shutdown imminent.	Network isolation detected; shutdown imminent.
CRITICAL	69	Peer instance is network isolated.	Peer instance is network isolated.
Type: MPS Call Layer Family			
INFO	a	call redirected through CLIENT:	Call is redirected through the CLIENT.
INFO	a	redirect: MULTIPLE_CHOICES, call to got redirected to:	The address in the request resolved to several choices, each with its own specific location, and the user (or UA) can select a preferred

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
			communication end point and redirect its request to that location.
INFO	a	redirect: MOVED_PERMANENTLY call to got redirected to:	The user can no longer be found at the address in the Request–URI, and the requesting client should retry at the address given by the Contact header field. Call to is redirected.
INFO	a	redirect: MOVED_TEMPORARILY call to got redirected to	The requesting client should retry the request at the new address given by the Contact header field. Call to is redirected.
INFO	a	redirect: USE_PROXY call to got redirected to:	The requested resource must be accessed through the proxy given by the Contact field. Call to is redirected.
INFO	a	call redirected through : ssname	Call is redirected through the subsystem.
INFO	258	<SIP/MPS/ADT> CPS [%d-sec-avg]:"+ %d	Periodic report of average calls per second.
INFO	258	<SIP/MPS/ADT> CPS [%d-sec-avg]:"+ %d [%d-sec-avg]:"+ %d	Periodic report of average calls per second plus a second average of a longer sample period.
MINOR	64	<SIP/MPS/COM> Application release the call at state:%s GCID:%s key:%s Call Started at: %s Duration:%s msec	Call Terminated before answered. Check application logic.
MINOR	64	<SIP/MPS/COM> Network release the call at state:%s GCID:%s key:%s Call Started at: %s Duration:%s msec	Call Terminated before answered. Check application logic and configuration.
MINOR	64	<SIP/MPS/COM> Outbound call timeout at state:%s GCID: %s key:%s Cause:%d Call Started at: %s Duration:%s msec	Outbound call timeout 408 request timeout received.
MINOR	64	<SIP/MPS/COM> Outbound call timeout at state:%s GCID: %s key:%s Cause:%d Call	Outbound call rejected.

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
		Started at: %s Duration:%s msec	
MAJOR	65	<SIP/MPS/COM> Inbound call attempt on application %s while previous call still active at state: %s	Internal Error. Contact Avaya Personnel.
MAJOR	71	<SIP/MPS/COM> Media Modify failed for SDP in ACK .	Media Modify failed for SDP in ACK.
MAJOR	C8	<SIP/MPS/RSC> No Inbound Application Available	No available applications to handle inbound call.
MINOR	C9	<SIP/MPS/COM> Caller hangup queued call at state: %s GCID:%s key:%s Call Started at: %s Duration:%s msec	A queued call terminated by caller.
MINOR	C9	<SIP/MPS/COM> Abort queued inbound request GCID:%s key:%s Call Started at: %s Duration:%s msec	A queued call is terminated guarded by queue timer.
Type: Registration Family			
INFO	2	<SIP/MPS/DEB> Registration Success	Indicates registration success.
INFO	2	<SIP/MPS/DEB> Deregistration Success	Indicates deregistration success.
INFO	3	NATH : Deregister timer is:	Indicates the deregister timer.
INFO	3	Registrar MOVED_PERMANENTLY to redirectedHost. Registration for resent through: nSsname	Indicates that the Registrar is MOVED_PERMANENTLY to redirectedHost and Registration is resent through the subsystem.
INFO	3	Registrar MOVED_TEMPORARILY to redirectedHost. Registration for resent through : nSsname	Indicates that the Registrar is MOVED_TEMPORARILY to redirectedHost. Registration is resent through the subsystem.
INFO	3	<SIP/MPS/DEB> Registration Failed	Indicates registration failure.
INFO	3	<SIP/MPS/DEB> Deregistration Failed	Indicates deregistration failure.

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
INFO	3	Registration for got redirected, resent through :nSsname	Registration is redirected, resent through subsystem.
INFO	3	Registration Failed. Server Error	Indicates registration failure due to server error.
INFO	3	Registration Failed due to Internal Error. Will retry after 10 Secs	Indicates registration failure due to internal error.
INFO	3	Register Failed. Password not found for user	Indicates registration failure due to unavailability of password for user.
INFO	3	Register Failed for 423 Received.	Indicates registration failure because the expiration time of the resource refreshed by the request is too short.
INFO	4	<SIP/MPS/DEB> Registration time expired	Indicates registration expired and will be renewed.
INFO	4	DeRegistration Guard time expired	Indicates DeRegistration Guard time has expired.
INFO	4	Register Retry Timer Expired	Indicates Register Retry Timer has expired.
MAJOR	10	<SIP/MPS/COM> The Proxy %s(name) of %s (domain) Not Reachable	Indicates the proxy is not reachable and no response for 2 consecutive "OPTIONS".
MAJOR	11	<SIP/MPS/COM> The Proxy %s(name) of %s (domain) Is Alive	Indicates the proxy became reachable.
Type: CC Family (ASE Mode)			
INFO	1	<SIP/CC/DBG> CC-ASE Proxy mode started	During the ccss-start, this messages indicates the Call Control module is started without any error.
INFO	6a	Proxy Resource connection Paused :VasHostName	Indicates that the Proxy Resource Connection is paused.
INFO	66	<SIP/CC/COM> Connected with Proxy Resource	Indicates successful connection with SIP Proxy resource daemon.
INFO	68	Proxy Resource Connection became STANDBY :VasHostName	Indicates that the Proxy Resource Connection has become standby.

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
INFO	69	Proxy Resource Connection became Active :VasHostName	Indicates that the Proxy Resource Connection has become active.
MAJOR	2BC	<SIP/CC/COM> unsatisfied link error in loading the proxy sip client library PLIBJNI Load failed	JNI Library load failed during start-up. Check the /opt/ccss/lib if libsipclient.so found. If not Contact Avaya personnel.
MAJOR	65	<SIP/CC/COM> Proxy Resource Connection Disconnected	Indicates the connection status with the SIP Proxy resource daemon is disconnected. MPS SIP server tries to connect automatically again. If fails, Check the connection status between MPS SIP Server and SIP Proxy resource daemon. Check SIP Proxy deamon log message for more info.
MAJOR	67	<SIP/CC/COM> Connecting with proxy resource %s-cchostname	Indicates the connection status with the SIP Proxy resource deamon. During startup, do nothing. Otherwise, check the link between the MPS SIP Server and the MPS SIP Proxy daemon.
MAJOR	64	<SIP/CC/COM> AVS Format Error AVS Decoding Error or AVS Encoding Error	Indicates that there is error with the AVString format message to/from the application. Turn on the Trace details and check the values and AVString.
MAJOR	71	<SIP/CC/COM> Message to Proxy Failed	Indicates the message could not be delivered to the application. Check the MPS SIP Server-SIP Proxy resource daemon connection status.
MAJOR	72	<SIP/CC/COM> Sip Message from the Application can not be parsed successfully. Could not be delivered to the MPSCM.	Indicates the sip message from the application has some error. Turn the trace detail on to review the values of the sip message from the application and correct the application.
MAJOR	0	<SIP/CC/ABN> Abnormal Category Alarms Invalid	Indicates some internal error. Contact Avaya personnel.

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
		VASID Invalid State Transition Duplicate Issue Get Invalid PrimitiveID Duplicate Transaction ID Invalid State Operation	
INFO	6e	<SIP/CC/COM> Primary MPS Node XXX XXX XXX XXX failed. Failover initiated for Secondary MPS Node XXX XXX XXX XXX to become Active	Indicates that failover occurred on Primary MPS Node failure and secondary/standby MPS Node is active now.
INFO	6f	<SIP/CC/COM> Primary MPS Node XXX XXX XXX XXX is active now	Indicates that failback occurred on Primary MPS Node coming up into service
INFO	70	<SIP/CC/COM> nlb defaultls is enabled	Indicates that NO load balancing set.
INFO	70	<SIP/CC/COM> nlb roundrobinls is enabled	Indicates that round-robin Network Load Balancing (NLB) set.
INFO	70	<SIP/CC/COM> nlb relativels is enabled	Indicates that NLB relative-ration algorithm is set.
INFO	70	<SIP/CC/COM> nlb weightagels is enabled	Indicates that load-sharing using weightage ratio set,
Type: CC Family (MIXED Mode)			
INFO	1	<SIP/CC/COM> CC-MIXED Mode started	During the ccss-start, this message indicates the Call Control module is started in MIXED mode without any error.
Type: MG Family			
INFO	0	Configuration file loaded + filename.	Sip server is loading the configuration information from sip.conf and sipdefault.properties.
INFO	0	[NETWORK] section configured with no CLIENT_ACCESS entry.	[NETWORK] section configured with no CLIENT_ACCESS entry.
MAJOR	0	File not found + filename	The configuration file is not found. Check if sip.conf and aipdefault.properties are

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
			presented in the correct directory.
MAJOR	0	Unable to set initial self context. Unknown host: hostname	Sip server host name is not valid. Check sip.conf SYSTEM section, verify the SIP SERVER HOST NAME is valid.
MAJOR	0	Configuration error, no SIP_TIMER_PROFILE entry defined for : profilename.	The sip timer profile with name profilename is not defined. Add the definition to SIP_TIMER_PROFILE.
MAJOR	0	KEY: parsing error	The number or format of parameters is incorrect for this KEY. Check the number or format of parameters for this KEY.
MAJOR	0	parsing error CallControlSection : CC_LINEMAP: HOST ID: No CC_HOSTS entry found for this HOST ID.	The host id in CC_LINEMAP is not defined in a CC_HOSTS entry. The host id in CC_LINEMAP has to be defined in a CC_HOSTS entry.
MAJOR	0	parsing error MediaGatewaySection : MG_LIST: HOST NAME: invalid host name.	The hostname in this MG_LIST entry is not a valid hostname. The hostname in MG_LIST entry must be a valid hostname.
MAJOR	0	parsing error NetworkSection: NETWORK_HOSTS: PRIMARY HOSTNAME : invalid Host name	The hostname in this NETWORK_HOSTS entry is not a valid hostname. Supply a valid host name.
MAJOR	0	parsing error SystemSection :SYSTEM:RE DUND ENABLE: invalid value, has to be "true" or "false".	The value for REDUND ENABLE is not valid. Put "true" if the system is redundant enabled, "false" otherwise.
MAJOR	0	configuration error, no CONTACT_PROFILE entry defined for cpName.	In REGISTRATION_CONTROL section, there is no CONTACT_PROFILE entry defined with name:cpName. The contact profile name must be a defined profile.

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
MAJOR	0	configuration error REGISTRATION_CONTROL: REG_LINE_CONFIG, no CC_HOSTS entry defined with HOST ID: hostId.	The hostId used in REG_LINE_CONFIG entry is not correct. The hostId must be presented in one of CC_HOSTS entry.
MAJOR	0	configuration error CALL_CONTROL:CC_HOSTS: no HOST_PROFILE entry defined for pName.	In CALL_CONTROL section, there is no HOST_PROFILE entry defined with name:pName. The host profile name must be a defined profile.
MAJOR	0	configuration error, MEDIA_GATEWAY:MG_LIST: no MG_PROFILE entry defined for pName.	In MEDIA_GATEWAY section, there is no MG_PROFILE entry defined with name:pName. The mg profile name must be a defined profile.
MAJOR	0	configuration error MEDIA_GATEWAY:MG_LIST: no TMS_HOST_PROFILE entry defined for tmsname	The tmsname specified in this MG_LIST entry is not defined in TMS_HOST_PROFILE. The tms host profile name must be a defined profile.
MAJOR	0	configuration error, type of IP transport for CLIENT_ACCESS has to be UDP.	The transport type configured for CLIENT_ACCESS is not correct. The type of transport for CLIENT_ACCESS has to be UDP.
MAJOR	0	configuration error in [NETWORK] section: there has to be either CLIENT_ACCESS entry or NETWORK_HOSTS entry or both.	CLIENT_ACCESS and NETWORK_HOSTS both are empty. CLIENT_ACCESS and NETWORK_HOSTS can not be both empty.
MAJOR	0	[NETWORK] section configured with no NETWORK_HOSTS entry Pease remove or comment the [REGISTRATION_CONTROL] section and then restart.	[NETWORK] section configured with no NETWORK_HOSTS entry. Remove or comment the [REGISTRATION_CONTROL] section and then restart.
Type: MG			
INFO	258	Gateway: gatewayID, audit starting.	Start audit on gateway gatewayID.

Alarm Level	Alarm Code	Message Displayed	Description/Resolution Resolution Required
INFO	259	Gateway: gatewayID, audit complete.	The audit on gateway gatewayID is completed.
INFO	e6	Endpoint Group : groupNum INSV	The state of endpoint group groupNum is INSV.
INFO	f0	Gateway: gatewayID,INSV	The state of gateway gatewayID is INSV.
INFO	2bf	Protocol Error: Unknown or Invalid RequestId : requestId received in GWNotifyEvent from gateway: gatewayID.	Unknown or Invalid RequestId : requestId received in GWNotifyEvent from gateway: gatewayID.
INFO	2bf	Protocol Error: null requestId received in GWNotifyEvent from gateway: gatewayID.	Null requestId received in GWNotifyEvent from gateway: gatewayID.
INFO	2bf	Protocol Error: Unknown endpointId: endpointId received in GWDeleteRequestEvent from gateway: gatewayID.	Unknown endpointId: endpointId received in GWDeleteRequestEvent from gateway: gatewayID.
INFO	2bf	Protocol Error: Null endpointId received in GWDeleteRequestEvent from gateway:gatewayID.	Null endpointId received in GWDeleteRequestEvent from gateway: gatewayID.
MINOR	e7	Endpoint Group : groupNum OOSV	The state of endpoint group groupNum is OOSV.
MINOR	f1	Gateway: gatewayID,OOSV	The state of gateway gatewayID is OOSV.
CRITICAL	f2	Media Gateway is out-of-service:	Media Gateway is out-of-service.

TMS Integrated Mode Diagnostics

Alarms

There are four categories of alarms: Warning, Minor, Major, and Critical. These are listed in order of increasing severity.

Alarms specific to TMS integrated mode are numbered starting at 51.

Alarm events are one of following: Abnormal, Administrative, Communication, Resource Availability, Threshold, and Uncategorized.

The following table lists all of the alarms for TMS integrated mode that can be generated from the Call Control layer.

Alarm Type /Level	Alarm Code	Message Displayed	Description / Resolution
Type: CC Family (TMS mode)			
MAJOR	2ef	Invalid TMS Application state	TMS App object is not in valid state while receiving or starting a new call. Contact Avaya personnel.
MAJOR	2f0	CcCall unavailable for the App:#	There is no Call Control object assigned to the TMS App. Contact Avaya personnel.
WARNING	2f1	INSV line going to INSV, App#	Line which is already in service is again going to in service. Contact Avaya personnel.
WARNING	2f2	OOSV line going to OOSV, App#	Line which is already out of service is again going to out of service. Contact Avaya personnel.
MINOR	2f3	Invalid Message 'MessageName' from DSSI:#	Unknown request or response received from the JNI thread. Contact Avaya personnel.
MINOR	2f4	Unsupported SIP Event, 'EventName' forTmsApp:#	Event received from the MPS CM not supported. Contact Avaya personnel.
MINOR	2f5	Invalid Protocol ID received from the Application, Protocol ID#	Protocol ID does not match SIP=5 in the protocol generic header in VarData. Contact Avaya personnel.
MINOR	2f6	Invalid protocol variant ID received from Application, Variant ID#.	Protocol variant ID is set to default 0 for SIP protocol. Contact Avaya personnel.
MINOR	2f7	Call Control Message received from DSSI on wrong DCCID, ID#	Each connection to TMS has an ID. Call Control (cc) Message received on ID that is different than the known value. Contact Avaya personnel.

Alarm Type /Level	Alarm Code	Message Displayed	Description / Resolution
MINOR	2f8	Client Connection Message received on wrong DCCID, ID#	Each connection to TMS has an ID. Call Control (cli) Message received on ID that is different than the known value. Contact Avaya personnel.
MINOR	2f9	Call Control Message received from DSSI on wrong Span#, Line#	Call Control (cc) Message received on unknown span or channel. Contact Avaya personnel.
MINOR	2fa	Duplicate TmsAppMgr instance	TMS Application Manager already created; trying to create again. Contact Avaya personnel.
MAJOR	2fb	Line out of range in the sip.conf: Span ID#	The Call Control line map in the sip.conf file has a line out of range. Valid line ranges are 1-30. Contact Avaya personnel.
MAJOR	2fc	Line configured for unknown Host Id:#	Lines for host are not configured in the Call Control host table in sip.conf. Contact Avaya personnel.
MAJOR	2fd	Pre Configured lines on wrong Host Name: #	Registration line configuration that specifies line pooling has been configured for an unknown host in sip.conf. Contact Avaya personnel.
MAJOR	2fe	Pre Configured lines on wrong Span: #	Registration line configuration that specifies line pooling has been configured for an unknown span in sip.conf. Contact Avaya personnel.
MAJOR	2ff	Application data not sent to SIP server for setup, App#	ANI and DNIS not received. Contact Avaya personnel.
MAJOR	64	AVS decoding error, App:#	SIP server was not able to decode the folder structure received as an AVS string from the application. Turn on the Trace details and check the values and AVString.
MAJOR	97	Connection established to host: 'IP Address' port:#	Perm state changed to PERM_CONNECTED.

Alarm Type /Level	Alarm Code	Message Displayed	Description / Resolution
MAJOR	98	Discrim Indication Fail, DCCID: #, Span#	Unable to transmit discrim message on the span. Contact Avaya personnel.
MAJOR	99	GO Indication FAIL for DCCID: #	Unable to send cliGoInd. This signals the client to begin signaling on the connection. Contact Avaya personnel.
MAJOR	9a	Failure to send CcExt-Message to DSSI on Span:# Line:#	Unable to communicate CcExt message to App, in line#, from the Call Control layer. Contact Avaya personnel.
MAJOR	9b	Disconnected from the host : IP Address	PERM_DISCONNECT on the existing connection between the Call Control layer and DSSI task. Check connection between sip server and tms.
MINOR	9c	Cannot send message, due to link drop, DCCID:#	Link between the DSSI and SIP server is lost while attempting to send SIP events to DSSI. Check connection between sip server and tms.
MINOR	9d	Line state changed to INSV, App:#	CcUnblockReq/Ind received. Line state changes to the In Service.
MINOR	9e	Line state changed to OOSV, App:#	CcBlockReq received. Line state changed to Out of Service.
MAJOR	9f	Proxy Name received null, App:#	Application did not provide the proxy name to forward the request. Check application and sip.conf file to make sure the proxy name in application already defined in config file.
MAJOR	a0	AVS Encoding error, App:#	SIP server was not able to encode data into AVS format, before sending to DSSI. Check application.
MAJOR	a1	Trying to communicate on TMS connection which is down, hostID#	TMS is disconnected. Message sent from network to TMS not received. Check connection between sip server and tms.

Alarm Type /Level	Alarm Code	Message Displayed	Description / Resolution
MAJOR	a1	Trying to communicate on Line which is disconnected in SIP server, key:	Network is disconnected. Message sent from TMS to the network not received. This is similar to the previous alarm, but in the reverse direction. Check connection between sip server and tms.
CRITICAL	fb	CcExt JNI Mailbox Queue full	Queue is full. TMS App objects and the CcExtJNIMailbox task cannot communicate to make native calls. Contact Avaya personnel.
CRITICAL	fc	CcExt JNI Mailbox Queue Inactive	Queue is inactive. TMS App objects and the CcExtJNIMailbox task cannot communicate to make native calls. Contact Avaya personnel.
CRITICAL	fd	TmsApp Queue full	Queue is full. The native thread and TMS App cannot communicate. Contact Avaya personnel.
CRITICAL	fe	TmsApp Queue Inactive	Queue is inactive. The native thread and TMS App cannot communicate. Contact Avaya personnel.
CRITICAL	ff	Unable to Load CcExt Native Library	SIP server was not able to load the CcExt Native Library. Contact Avaya personnel.
MAJOR	100	Unable to allocate media line, CN#	SIP server was unable to create media connection on the endpoint. MGCP created connection fails. Contact Avaya personnel.
MAJOR	101	Unable to modify existing media connection, CN#.	SIP server was not able to perform MGCP media modify on the connection. Contact Avaya personnel.
MINOR	102	Unable to delete existing media connection, CN#	The application tied with this CN enters the block state, since it was not able to delete existing media connection. Contact Avaya personnel.

Alarm Type /Level	Alarm Code	Message Displayed	Description / Resolution
MAJOR	3b7	Invalid From field from application.	In the outgoing call, the application did not provide a valid From or ANI field. Check application
MAJOR	3b8	Invalid To field from the application.	In the outgoing call, the application did not provide a valid To or DNIS field. Check application
MAJOR	3b9	Invalid proxy name from the application.	In the outgoing call, the application did not provide a valid configured proxy name or CLIENT for client access. Check application
MINOR	3ba	Null Pointer Exception.	Pointer to object is null. Contact Avaya personnel.

Alarm Logs

Logs are classified as Alert, Info, and Swer. All logs specific to TMS integrated mode start with log number 21.

Logged events can be any one of the following types: Abnormal, Administrative, Audit, Communication, Debug, Resource Availability, Threshold, Uncategorized.

Log Level	Code	Log Type/Name	Message Displayed	Description
ALERT	21	ADMINISTRATIVE ccCcExtLibLoaded	Loaded CcExt Native Library	Loading native code using static class method System.loadLibrary.
ALERT	21	COMMUNICATION ccLineResetFromCC	Line Reset from Call Control, Line:#	Line reset from the Call Control (UI) by calling CcResetInd.
ALERT	22	COMMUNICATION ccLineResetFromApp	Line Reset from Application, Line:#	Line reset received from application.
ALERT	23	ABNORMAL ccAppNoAnswer	Application in line # did not answer before network release	Verify that the application is assigned to the line.

Advanced Configuration

Advanced Configuration

This appendix provides manufacturing procedures for configuring TPM cards. This information is intended for use only by the Avaya manufacturing department.

TPM 1100 variants

The TPM card itself does not have a PEC code. It is always encased in a CTPM-1000 or CTPM-2000 assembly, which does have a PEC code.

The following table lists the TMP1100 variants and their information.

Variant	Image file	Initialization	Carrier/Type	PEC code
240 Channels G.711	ramTPM1100.cmp	tp1100.ini	CTPM 1000/non-Rohs	NTV60313
60 Channels G.711	ramTPM1100.cmp	tp1100-60.ini	CTPM 1000/non-Rohs	NTV60315
240 Channels G.711/AMR	ramTPM1100AMR.cmp	tp1100AMR.ini	CTPM 1000/non-Rohs	NTV60314
240 Channels G.711	ramTPM1100.cmp	tp1100.ini	CTPM 1000/Rohs	NTV60313E5
60 Channels G.711	ramTPM1100.cmp	tp1100-60.ini	CTPM 1000/Rohs	NTV60315E5
240 Channels G.711/AMR	ramTPM1100AMR.cmp	tp1100AMR.ini	CTPM 1000/Rohs	NTV60314E5
240 Channels G.711	ramTPM1100.cmp	tp1100.ini	CTPM 2000/Rohs	NTV60333E5
60 Channels G.711	ramTPM1100.cmp	tp1100-60.ini	CTPM 2000/Rohs	NTV60335E5

240 Channels G.711/AMR	ramTPM1100AMR.cmp	tp1100AMR.ini	CTPM 2000/Rohs	NTV60334E5

Configuration of TPM Cards will require editing the following files:

```
/etc/bootptab
/etc/hosts
/etc/ethers
```

Configuration of BOOTP daemon:

The file /etc/bootptab is used by the bootp daemon (bootpd) to reply to bootp requests made by the TPM cards in a system containing VOIP spans. The configuration of the bootptab entries are automated with the use of ccssinstall script. Discussion follows on how manually inserted the TPM entries:

The TPM entries are configured with the following specifications:

#TPM200:\ :sm=255.255.255.0:gw=47.185.16.1:td=/tftpboot:
bf=ramTPM200.cmp;tp200.ini:ht=ethernet:ha=08004e2fbc00:ip=47.185.16.242

sm – specifies the subnetmask

gw – specifies default gateway (both primary and secondary gateway).

td – (target directory) specifies the directory on the boot server to obtain images and initialization files.

Note:
For Windows systems, td (target directory) must be replaced by hd (home directory).

bf – This bootptab entry needs to specify the TPM image name (ramTPM200.cmp TPM 200 card and ramTPM800A.cmp for TPM800 card) and initialization file (tp200.ini or tp800.ini).

ht – The hardware address type (ht) is set to Ethernet.

ha – The TPM hardware address (ha).

ip – The IP address for the TPM card.

Hosts file configuration:

Enter the IP addresses used in the bootptab file for the TPM and into the /etc/hosts file.

```
/etc/hosts
127.0.0.1 localhost
#
#
# The External Network
#
192.168.204.201 ny4-ap1 ap1 loghost www.nodomain.nocom
```

```
#
#
#Application processor to TMS LAN
#
192.168.101.200 ap-to-tms
#
#tms resource cards, private LAN
# IP Addresses associated with chassis nbr 1
#
192.168.114.1 tms1
192.168.114.2 tms2
192.168.114.3 tms3
192.168.114.4 tms4
192.168.114.7 nic1
192.168.114.8 nic2
```

```
47.185.16.242 tpm200
```

```
192.168.114.22 dcc
#
#
#
192.84.100.1 ppp-DialIn
192.84.100.2 ppp-DialOut
```

Ethernet address configuration

Enter each Ethernet hardware address for every TPM200 or TPM800 cards connected to the system into the /etc/ethers file.

```
/etc/ethers:
```

08:00:4e:2f:bc:00 tpm200 08:00:20:c6:23:7e tcmps01

Windows system

Configuring using ccssinstall.plx script:

This section describes how the hardware and IP addresses for the TPM cards are configured using the ccssinstall script located at \Program Files\Avaya\SelfService\bin. Running this script modifies the \etc\hosts and \etc\bootptab files and also the tpxxxx.ini file located at \tftpboot where xxxx is the TPM type-200, 800, 1100.

The following is displayed when ccssinstall.plx is executed.

1 file(s) copied.

1 file(s) copied.

========== CCSSINSTALL V1.0 (c) 2006 Avaya

CCSS SIP Network Address : [10.7.0.1] Subnet Mask : [255.255.0.0]255.0.0.0 Primary Gateway Address : [47.185.0.1] Secondary Gateway Address (Optional) : [47.185.0.1] Bootp Configuration: 1. TPM200 2. TPM800 3. TPM1100 4. TPM1100-AMR 5. TPM1100-60 Select (1-5): 3 IP of TPM1100 Card : [10.7.1.3] Insert TPM1100 card's

Chassis Number : 17 Insert TPM1100 card's Chassis Number : 2 Insert TPM1100 card's Backplane Slot : 2 Hostname of TPM1100 Card : [TPM1100-2-2] Ethernet Address : [00:80:01:80:nn:nn]11:22:33:44:55:66 1 file(s) copied. tp1100-2-2.ini is the initialized file name for the TPM1100 card placed in ChassisNo:2 and BPS:2 Configuring TPM1100 Parameters in the tp1100-2-2.ini file. Please select a PCM Law (1.aLaw 2.uLaw) [2] : Please enter the call agent IP address : Please enter the call agent IP address : 1.2.3.4.5.6 Please enter the call agent IP address : 0.0.0.0 Please enter the gateway name : san Please select the RFC2833 payload type [101] : Do you want silence compression (0.No 1.Yes) [0] : tp1100-2-2.ini file has been updated For modifying other parameters use the TPM web page or edit the tp1100-2-2.ini file -- File C:\WINDOWS\system32\drivers\etc\hosts updated -- -- File C:\WINDOWS \system32\drivers\etc\bootptab updated -- Do you want to install another card (y/n)? : [n]n EXITING.

Note:

The ccssinstall.plx script updates the bootptab and hosts file. Based on your input for the Chassis Number/Backplane Slot and the type of TPM1100, the script creates the respective .ini file at C:\tftpboot. For example, if you select option 5 in the Bootp Configuration menu with the Chassis and BackPlane Slot (BPS) as 1 and 2 respectively, then tp1100-60-1-2.ini is created at C:\tftpboot.

Ensure that the corresponding default tpxxxx.ini files (tp1100.ini and tp1100-AMR.ini.tp1100-60.ini) are present at C:\tftpboot before executing the script.

Solaris System

1. To configuring using ccssinstall script, use the following.

This section describes how the Hardware and IP addresses, for the TPM cards can be configured using the ccssinstall script located at $CCSSHOME/bin.

To run the ccssinstall script: 1. Login as superuser. peri@ccss {1} su Password: ***** # 2. Execute ccssinstall to configure the Hostname, Ethernet Address and BOOTPTAB, and respond to all prompts. The following is displayed when ccssinstall executes.

```
==========================================
CCSSINSTALL V1.2 Mon Dec 10 (c) 2005 Avaya
```

CCSS SIP/SS7 Network Address :[10.7.0.1] Subnet Mask : [255.255.0.0] Primary Gateway Address : [47.185.11.1]47.185.121.1 Secondary Gateway Address (Optional) : [47.185.121.1] Bootp Configuration: 1. TPM200 2. TPM800 3. TPM1100 4. DCC 5. COM 6. TPM1100-AMR 7. TPM1100-60 Select (1-7): 3 IP of TPM1100 Card : [10.7.1.3]10.7.1.3 Insert TPM1100 card's Chassis Number : 1 Insert TPM1100 card's Backplane Slot : 4 Hostname of TPM1100 Card : [TPM1100-1-4] Ethernet Address : [00:80:01:80: nn:nn]11:22:33:44:8c:9d Save changes (y/n)? : [n] y -- File /etc/hosts updated -- -- File /etc/ethers updated -- -- File /etc/bootptab updated -- TPM Gateway Configuration: Please select a PCM Law (1.aLaw 2.uLaw) [2] : 1 Please enter the call agent IP address : 0.0.0.0 Please enter the gateway name : gatenort Please Enter the RFC2833 PayloadType [101] : 111 Do you want silence compression (0.No 1.Yes) [0] : tp1100-1-4.ini file has been initialized/updated. For modifying other parameters use the TPM web

page or edit the tp1100-1-4.ini file. Install another card (y/n)? : [n] y Bootp Configuration: 1. TPM200 2. TPM800 3. TPM1100 4. DCC 5. COM 6. TPM1100-AMR 7. TPM1100-60 Select (1-7): 7 IP of TPM1100-60 Card : [10.7.1.3]10.3.11.1 Insert TPM1100-60 card's Chassis Number : 1 Insert TPM1100-60 card's Backplane Slot : 2 Hostname of TPM1100-60 Card : [TPM1100-60-1-2] Ethernet Address : [00:80:01:80:nn: nn]11:d5:22:33:43:22 Save changes (y/n)? : [n] n Install another card (y/n)? : [n] n EXITING

2. To configure TPM1100 parameters using tpm_config.pl, use the following.

The tpm_config.pl script is used to configure the various TPM1100 Gateway Cards (TPM1100/TPM1100-AMR/TPM1100-60). Using this script, TPM parameters like PCM law, PayloadType, and silent compression are configured. Execute the tpm_config.pl script only in the case of TPM1100 configuration.

Ensure that the GW Card initialization file, tpxxxx.ini, that you intend to modify is located at /tftpboot before executing tpm_config.pl.

The following is displayed when tpm_config.pl executes.

perl tpm_config.pl
TPM Gateway Configuration: Please enter TPM1100/TPM1100-AMR/TPM1100-60 GW Card initialization file. e.g. tpxxxx.ini to update GW parameters: tp1100-AMR-1-1.ini Please select a PCM Law (1.aLaw 2.uLaw) [2] : 1 Please enter the call agent IP address : 0.0.0.0 Please enter the gateway name :gateway_avaya Please Enter the RFC2833 PayloadType [101] : 111 Do you want silence compression (0.No 1.Yes) [0] : tp1100-AMR-1-1.ini file has been initialized/updated. For modifying other parameters use the TPM web page or edit the tp1100-AMR-1-1.ini file. Do you want to modify another .ini file(y/n)? [n]: y Please enter TPM1100/TPM1100-AMR/TPM1100-60 GW Card initialization file. e.g. tpxxxx.ini to update GW parameters: tp1100-3-4.ini Please select a PCM Law (1.aLaw 2.uLaw) [2] : Please enter the call agent IP address : 1.2.3.4 Please enter the gateway name :nort_gate Please Enter the RFC2833 PayloadType [101] : Do you want silence compression (0.No 1.Yes) [0] :0 tp1100-3-4.ini file has been initialized/updated. For modifying other parameters use the TPM web page or edit the tp1100-3-4.ini file. Do you want to modify another .ini file(y/n)? [n]: n EXITING.

SIP.CONF file configuration

You must modify the SIP.CONF file to support the TPM card.

The SIP.CONF file is located:

- Solaris: $CCSSHOME/etc/sip.conf
- Windows: %MPSHOME%/ ccssN/etc/sip.conf (N is the ccss component number)

Modify the MG_LIST section with an entry for the TPM card.

```
#################################################################
#
```

```
# The configuration of the MG_LIST, the media gateway configured for sip
server
# NAME: name of this MG_LIST entry.
# DOMAIN:  domain name of the media gateway.
# HOST Name : Audiocode's card's IP Host Name, it can be IP address string
#             in "%d.%d.%d.%d" format.
# MG PORT : gateway transport port number use "2427" for standard,
#           or the number configured for this card.
# MG TYPE :  MG_Profile type for this entry, it has to be defined in the
MG_PROFILE.
# CHASSIS ID: chassis number.
# SLOT ID: slot number.
# DEFAULT CODEC:  default payload Codec "G711"|"G723"|"G726"|"G729".
# TMS_HOST NAME: Profile name of the tms host which controls this TPM
card, it must be one
#                of the PROFILE NAME entry of TMS_HOST_PROFILE in
CALL_CONTROL section.
#
################################################################
MG_LIST = {
! ---------- --------------- --------------------- ----- ------ -------
---- ------- ------------
! NAME       DOMAIN          HOST Name               MG    MG    CHASSIS
SLOT DEFAULT TMS_HOST
!                                                    PORT  TYPE
ID      ID    CODEC  PROFILE NAME
! ---------- --------------- --------------------- ----- ------ -------
---- ------- ------------
AudioCodes com            47.185.24.123          2427  tpm1100     3
1     G711   tms1
}
```

The parameters are defined as follows:

- NAME: (AudioCodes) name of MG_LIST entry
- DOMAIN: (com) domain name of media gateway
- HOST Name: (47.185.24.123) IP address of TPM card
- MG Port (2427): always 2427
- MG Type (tpm1100): always tpm1100 for the TPM 1100
- Chassis ID (3): TMS chassis number (must agree with TMS.CFG)
- Slot ID (1): TMS slot number (must agree with TMS.CFG)
- Default Codec (G711): default payload codec
- TMS_HOST profile name (tms1): must agree with TMS_HOST_PROFILE setting, also in SIP.CONF.

sipdefault.properties file configuration

The sipdefault.properties file is located:

- Solaris: $CCSSHOME/etc/sipdefault properties
- Windows: %MPSHOME%/ ccssN/etc/sipdefault.properties (N is the ccss component number)

sipdefault.properties file contains internal configuration parameters that set global properties for debugging and task-oriented operations on a system wide basis.

Sample sipdefault.properties file

The following is a sample sipdefault.properties file for a stand alone MPS SIP server system.

sipdefault.properties

- foundation.task.cpu.count=1
- foundation.debug.out=stdout
- foundation.debug.port=21000
- dbg=true
- dbg.file=true
- dbg.module.mg=true
- dbg.module.gen=true
- dbg.module.cc=true
- dbg.level=3
- mg.audit.enable=false
- mg.rfc2833.pt=101
- alarm.usejni=false
- mps.inbq.max.time=120
- mps.inbq.max.count=1024
- mps.inbq.update.interval=10
- mps.api.header.all=true
- mps.proxy.ping=true
- mps.proxy.ping.p1=10
- mps.proxy.ping.p2=25
- mps.ContAddr.max.count=20

- mps.registration.useragent.enable=true
- ip.tos.value=112
- history.enable=true
- codec.default.ptime=30
- mps.redundancy.enable
- mps.sip.nlb

mps.inbq.max.time

Maximum time in seconds for which an inbound call can remain queued in the system.

mps.inbq.max.count

Maximum number of calls that can get queued in the system.i.e the Inbound queue Size. This value is 128 bytes or 1024 bits.

mps.inbq.update.interval

Time interval after which the inbound call's queue gets updated/refreshed.

dbg

If this parameter is set to true then debug is enabled.

dbg.level

This parameter can take the values 1|2|3 depending on the level of debug needed.

dbg.console

If this parameter is set to true then debug will be directed to the users console

dbg.file

If this parameter is set to true then debug will be directed to file

dbg.module.mg

This parameter when set to true will print diagnostic text in the Media Gateway module on to a file/console.

dbg.module.gen

This parameter when set to true will print diagnostic text in the Media Gateway module on to a file/console.

dbg.module.cc

This parameter when set to true will print diagnostic text in the Call Control module on to a file/console.

dbg.module.cm

This parameter when set to true will print diagnostic text in the Call Management module onto a file/console.

mps.api.header.all

This parameter when set to true will get the API to deliver all SIP headers.

Parameters for pinging hosts

- 1.mps.proxy.ping
- 2.mps.proxy.ping.p1
- 3.mps.proxy.ping.p2

The above mentioned parameters enable pinging and ping intervals of all the hosts configured in the NETWORK_HOSTS section of the sip.conf file.

When mps.proxy.ping is set to true, it enables the sip server to send "OPTIONS" periodically every P1 seconds, to all configured sip proxy/registrar in NETWORK HOSTS within sip.conf. When there is no reply for two consecutive "OPTIONS" form a proxy/registrar, it is considered to be unreachable.

A MAJOR alarm occurs the first time an unreachable situation occurs. Pinging continues every P2 seconds to that proxy and no alarm is thrown for later failures once it is found unreachable. A MAJOR alarm is thrown when the proxy is available again. If there is any succesful SIP registration with that proxy, they will have to be reregistered.

mps.proxy.ping

If this parameter is set to true then the MPS SIP server will send SIP “OPTIONS” periodically to all configured NETWORK_HOSTS within sip.conf.

mps.proxy.ping.p1

Time interval in seconds after which the MPS SIP server will send “OPTIONS” to all configured sip proxy/registrar in NETWORK_HOSTS, if mps.proxy.pingis set to true. The proxy/registrar is considered unreachable after two consecutive “OPTIONS” are sent.

mps.proxy.ping.p2

Time interval after which pinging will continue to the unreachable proxy/registrar.

mps.sdptransparency.enable

If this option is set to true, the MPS SIP server shall accept the SIP re-invite with no (zero) SDP and respond with a MediaModified which translates to 200 OK in SIP.

sip.longformat.enable

If this parameter is set to true then the MPS SIP server will use the long format for Event header in SIP messages (e.g. SUBSCRIBE). If this parameter is set to false then the compact format will be used.

foundation.debug.out

This parameter conditionally prints diagnostic text to a number of destinations called console devices. Devices corresponding to the standard output and standard error streams are built into the debug framework; a device is also created for each user session when the user logs on. Users may also create devices that map to ordinary files. Devices must be started to receive console output. There are four possibilities for this property:

- foundation.debug.out=stdout
- foundation.debug.out=stderr
- foundation.debug.out=<Filename>

foundation.debug.port

Port number for telnet access to the DebugSessionServer. Default value is 21000. A debug session provides a context for user interaction with the debug framework.

mg.codec.selection

This parameter decides which audio codec to be selected between the various codec.list entries in the sip.conf. It takes the values first/all depending on the user. Default is all.

mg.audit.enable

Enable/disable audit of TPM end-points. When enabled the system will ping the TPM periodically and tries to get the end-points in service.

mg.rfc2833.pt

This parameter indicates the RFC2833 Payload type to be used for the Media Gateway. It's required only for TPM800 boards.

mps.ContAddr.max.count

This parameter is used for supporting higher number of contact address list. For example, if the Contact Address List needs to be supported for 20 contacts, then add the entry mps.ContAddr.max.count=20in sipdefault.properties

mps.registration.useragent.enable

This parameter is used for supporting user-agent header in Register request. By default this value is set to false. This would also ensure that the user-agent header is present in the authenticated Register messages in case of a challenge. If the user-agent header is to be supported in Register message, add the entrymps.registration.useragent.enable=true in sipdefault.properties.

ip.tos.value

This parameter is used for supporting ToS bytes in IPv4 header in Layer 2 of SIP/MGCP protocol. The default ToS value is 0. The IPv4 header will use the default ToS value if:

- No ip.tos.value is set
- ip.tos.value is below 0 or above 254

If the IPv4 header needs to have the ToS bytes set to 112, add the entry ip.tos.value=112in sipdefault.properties.

history.enable

This parameter is used for enabling/disabling history on CCSS SIP server. By default history is OFF. To explicitly turn history ON, add the entry history.enable=true in sipdefault.properties.

codec.default.ptime

This parameter is used for modifying ptime for all outbound calls from the server. By default its value is set to 20 ms. If the ptime needs to be updated to 30, add the entrycodec.default.ptime=30 in sipdefault.properties.

mps.redundancy.enable

Enables CCSS SIP ASE Mode configured system to integrate with MPS N+1 redundancy. By default mps.redundancy.enable is false. To enable MPS SIP redundancy feature in ASE mode, entry mps.redundancy.enable = true must be set in sipdefault.properties file.

mps.sip.nlb

To enable Network Load Balancing (NLB) feature on a SIP system on start-up, any of the following entry needs to be configured in “sipdefault.properties” file:

- mps.sip.nlb = roundrobinls must be set to enable round-robin load-sharing method.
- mps.sip.nlb = relativels must be set to enable relative ratio method.
- mps.sip.nlb = weightagels must be set to enable weightage based load-sharing.

Configuring .ini File Parameters

The following .ini file parameters can be modified. The .ini files (such as tp200.ini, tp800.ini, and tp1100.ini) are located in the directory /tftpboot.

.ini File Field Name	Description	Valid Range
CallAgentIP	IP Address is specified in dotted notation. This should match the IP Address of the SIP server which is configured to use this TPM board.	Legal IP address. For tpm200 and tpm800 boards, this value can be set to 0.0.0.0. For tpm1100, this value should match SIP server IP Address.
GatewayName	Defines the Identification Name of the media gateway.	For tpm200 and tpm800 boards, this field can be left blank. For a tpm1100 board, this value should match the entry in the

.ini File Field Name	Description	Valid Range
		form Name.Domain that is specified in the MG_LIST section of the sip.conf file. For example, if Name is tpm01 and Domain is us.avaya.com, the GatewayName should be tpm01.us.avaya.com
PCMLawSelect	Selects the type of PCM companding law in input/output TDM Bus. This value should match the system backplane law setting in the tms.cfg file.	1= A-law 3= Mu-law
EnableEchoCanceller	Enables or disables echo canceller. 0 = Disable; 1 = Enable Default value is 1.	0 or 1
MaxEchoCancellerLength	Defines the maximum board EC (Echo Canceller). 0 = EC length determined internally to reach maximum channel capacity. 4 = 32 milliseconds 11 = 64 milliseconds 22 = 128 milliseconds TPM (TrunkPack Module) LBR (Low Bit Rate) Using 64 or 128 msecs can reduce the channel capacity to 5/6 of the original capacity. TPM (TrunkPack Module) PCM only Using 128 msecs reduces the channel capacity to 5/6 of the original capacity. Host/Default value is 0.	See Description
MGCPCompatibility Profile	Controls MGCP function. Default value is 1.	See Description
EnableSilence Compression	Enables or disables Silence Suppression Mode. 0 = Disable = SILENCE_COMPRESION_ DISABLE 1 = Enable = SILENCE_COMPRESION_ ENABLE 2 = Enable without adaptation = SILENCE_COMPRESION_	0 to 2

.ini File Field Name	Description	Valid Range
	ENABLE_NOISE_ADPTATIO N_ DISABLE Default value is 0.	
RFC2833RxPayloadTyp e	Controls the RFC 2833 Rx Relay RTP Payload type. Default value is 96.	96 to 127
RFC2833TxPayloadTyp e	Controls the RFC 2833 Tx Relay RTP Payload type. Default value is 96.	96 to 127
DtmfTransportType	Defines the type of DTMF transport. 0 = Erase DTMF from voice transport not relayed to remote. 2 = DTMFs not erased are not relayed to remote. 3 = DTMFs are muted from the voice stream and relayed according to RFC 2833. Default value is 3.	0 to 3
FaxTransportMode	Sets the fax over IP transport method. 0 = Transparent 1 = Relay 2 = Bypass 3 = Transparent with events Default value is 1.	0 to 3
ExternalPCMHighway Profile	Sets the TPM module specific profile of the PCM Highway. Relevant only to the TPM-1100. Default value is 0.	0 to 20

.ini file configuration

You must make additional modifications to the .ini file. See target on page 237 for the correct .ini file.

The .ini file locations are:

- Solaris: /tftpboot/*.ini
- Windows: C:/tftpboot/*.ini

Modify the following entries:

- CallAgentIP: the IP address of the SIP server Example: CallAgentIP = 47.185.21.41
- GateWayName: must match the MG_LIST name in the SIP.CONF file Example: GateWayName = AudioCodes.com
- PCMLawSelect: must agree with the SYS_coding_law name in the MPS Application Processor's TMS.CFG file (1 for A-Law, 3 for U-Law) Example for U-Law: PCMLawSelect = 3

After you make these modifications, you must visit the TPMs OAM page using a web browser. Type the TPM card's IP address in the web browser's address field and use the default login credentials (unless an administrator has changed them): User: Admin Password: Admin

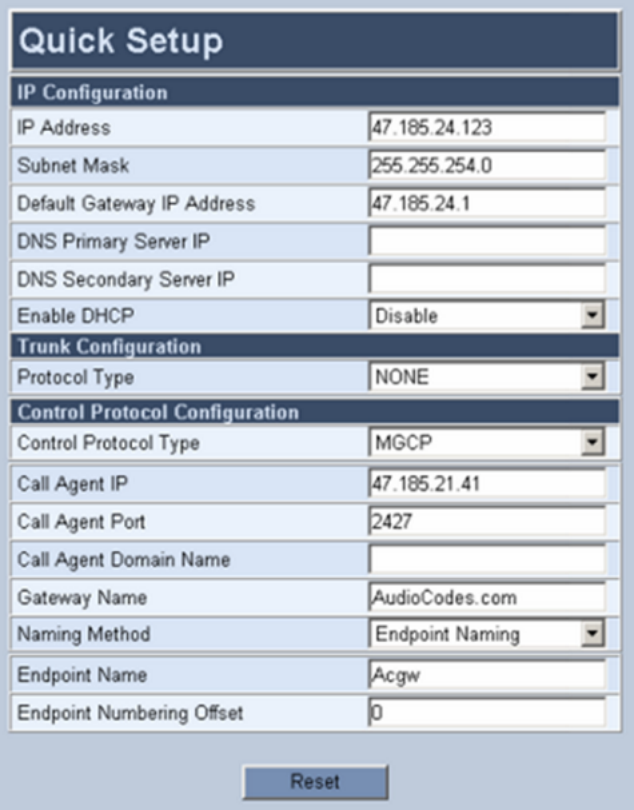


Figure 1: The following appears in the web browser after you log in.

The following information must agree with the entries in the *.ini file:

- Call Agent IP
- Gateway Name

The following information must agree with the entries in the bootptab file:

- IP address
- Subnet Mask
- Default Gateway IP Address

If the correct information does not appear, click the Reset button and wait a minute or two. The correct entries then appear.

Confirm TPM is in service

To confirm the TPM card is in service, perform the following steps.

1. On the SIP server, type sipui to open the command line interface to the SIP server.
2. Change to the mps/mg directory by typing cd /mps/mg.
3. Type ep list to see the status of each line on the TPM card.

The output resembles the following.

```
DEBUG:root/mps/mg/>ep list
endpoint info
GatewayId                    EndpointId                  State
HasConnection    CallID    ConnectionID

AudioCodes.com     ACgw0@AudioCodes.com    INSV      false
AudioCodes.com     ACgw1@AudioCodes.com    INSV      false
AudioCodes.com     ACgw2@AudioCodes.com    INSV      false
AudioCodes.com     ACgw3@AudioCodes.com    INSV      false

:
:
:
AudioCodes.com     ACgw236@AudioCodes.com    INSV      false
AudioCodes.com     ACgw237@AudioCodes.com    INSV      false
AudioCodes.com     ACgw238@AudioCodes.com    INSV      false
AudioCodes.com     ACgw239@AudioCodes.com    INSV      false
```

If each line shows INSV (in service), the TPM is connected to the SIP server and ready for a call. If one or more lines shows OOSV (out of service), the TPM is not fully in service and the configuration should be reviewed.

You may also go to the SIP user interface and locate the mps directory. Then enter tpmreset.

Field factory test documentation

For more information, see the field factory test documentation in the PERIfft package on the Application Processor or the SIP server. For Solaris, see /opt/vps/PERIfft/fftests/docs/

0136688-mps1000-tpm-fft.pdf. For Windows, see C:\Program Files\Avaya\SelfService\PERIfft \fftests\docs\0136692-tpmFFT-win2003.pdf.

Troubleshooting the TPM card

The following table covers some common issues with the TPM card and suggests resolutions.

Symptom	Probable cause	Resolution
The TPM card is not pingable or the TPM OAM web page does not appear. The DCC and TMS are pingable.	• 1. The TPM card may not be connected to the Ethernet. • 2. The TPM card was not booted by bootp on the SIP server. • 3. The Ethernet cable is unplugged or plugged into the wrong TPM port.	• 1. Verify the TPM is connected through a cable to the Ethernet. • 2. Verify that the entries for the TPM card and it's mother DCC card exist and have the correct information. If you suspect an incorrect Ethernet address, run a packet sniffer (for example, Ethereal) and look for BOOTP requests coming from the Ethernet address. A TPM Ethernet address always starts with 00:90:8f. Any incorrect information can cause the TPM card to fail to boot. • 3. Ensure the Ethernet cable is plugged into the ETHA or ENETA port on the TPM card.
The TPM card is not pingable. Neither are the DCC or the TMS.	• 1. The TMS may not be powered on. • 2. The TMS or DCC may not be connected to the Ethernet. • 3. The NIC card in the TMS chassis is not functional	• 1. Turn the TMS power supply on. • 2. Verify the TMS or DCC is connected through a cable to the Ethernet. • 3. Ensure the NIC card is connected and booted properly.

The TPM card is pingable, but the SIP server ep list shows the TPM lines as OOSV (out of service).	• 1. The INI file may not be configured with the correct information. • 2. The SIP.CONF file may not be configured properly • 3. The TPM may need to be reset. • 4. TFTP is not running on the SIP server. The tftp service in /etc/inetd.conf is disabled.	• 1. Verify that the INI file referenced in bootptab contains the correct CallAgentIP and GatewayName. • 2. Verify the MG_LIST section of the SIP.CONF file contains the correct information. Any incorrect information can cause the SIP server to not put the TPM in service. • 3. In the sipui tool, go to the mps directory and run the tpmreset command. • 4. Open /etc/inetd.conf in a text editor and uncomment the tftp line. Run the inetconv utility
The ep list command shows the TPM card is in service, but the SIP server still fails to take a SIP call.	• 1. The TMS.CFG file is not configured for the proper mode (Proxy or DSSI). • 2. The REG_LINE_CONFIG section of the SIP.CONF file is incorrect.	• 1. See Configuration on page 57 for details on TMS.CFG configuration. • 2. Check that you are dialing a valid SIP address based on the information in REG_LINE_CONFIG. See Configuration on page 57 for details on REG_LINE_CONFIG.

Enabling JVM debug options on window machine

For example, if the JVM debug options "-Xrs" needs to be specified.

1 Add the "-Xrs" option in file vos.cfg (path:C:\Program Files\Avaya\SelfService\ccss1\etc) e.g. "perisip -Xrs".

2. Restart Avaya MPS Service using windows services utility.

3. Open the srp.log located at C:\Program Files\Avaya\SelfService\common\log directory, look out for perisip entry and it's arguments passed.

4. Expected Result: debug print appears in srp.log e.g when arguments passed, -Xrs argv[2] = -Xrs joptv[11] = -Xrs

This assures that java process is started with enabled JVM options.

Note:

- Enables "-Xrs" JVM option by default.
- Enables the user to specify JVM debug options passing as arguments to perisip in vos.cfg file.
- Max heap size (-Xmx) defaulted to 256MB.
- Min heap size (-Xms) defaulted to 128MB.
- Min & Max heap size can be altered by passing through command line arguments in the range >128 MB and <256 MB.
- Maximum JVM options allowed in perisip.exe up to 32 entries.

Glossary

Index

Numerics

installation 41, 42
pkgadd 42
pkgrm 41
setenv CCSSSTARTSCRIPT 42
setenv CCSSSTOPSCRIPT 42
setup_protocol 42
SIPUI 94, 101, 103, 104
appstatus 101
calls 94
conn 104
createpool 101
deregister 103
ep 104
history 94
info 94
passwd 103
pools 101
register 103
registrations 103
reset 101
stats 94
summary 94
trace 94
system 41, 42, 89, 90, 92, 93
cp 89, 90
help 93
pwd 42
su 41, 42
tar 89, 92
operations 87
ccss_start 87
ccss_stop 87
configuration 60
CALL_CONTROL 60
ASE_HOST_PROFILE 60
CC_HOSTS 60
CC_LINEMAP 60
TMS_HOST_PROFILE 60
MEDIA_GATEWAY 60
MG_LIST 60
MG_PROFILE 60
MG_TIMER_PROFILE 60
NETWORK 60
CLIENT_ACCESS 60
NETWORK_HOSTS 60
SIP_TIMER_PROFILE 60
REGISTRATION_CONTROL 60
APPSVC_GROUP 60
CONTACT_PROFILE 60
REG_LINE_CONFIG 60
REG_TIMER_PROFILE 60
REGISTRATION_SERVICE 60
SYSTEM 60
SYSTEM 60
tms.cfg 85
LOAD 85
gen.cfg 84
sipproxy 84

A

ASE mode 32
ASE mode versus TMS integrated mode 32, 85, 109, 115
ASE mode vs. TMS integrated mode 32

B

backing up 89, 90, 92
cp 90
devices and options 89
methods 89
restore 90
tar 90
verify 90

C

CCSS 2000 ....24, 41, 42, 51, 53, 54, 87
Ethernet connections 24
software 41, 42, 51, 53, 54
installation 41, 42, 53
revisions and patches 51, 54
system 87
shutdown 87
startup 87
CCSSALARM 36
CCXML 32, 60
configuration files 58, 60, 84, 85
gen.cfg 84

SIP 58
syntax examples 58
sip.conf 58, 60
sipdssi_proto.cfg 85
sipproxy_proto.cfg 85
tms.cfg 85
copy 90
See also backing up 90
customer service 13

D

distributor 14
documentation 13

E

ethereal utility 106

J

Java Runtime Environment 30, 42

K

key=value pairs 58

M

MPS Manager 42, 54

P

packet voice connectivity 29
PERIvoip 29

R

reseller 14
restore - See backing up 90

S

SIP resource daemon (sipproxy) 84
SIP server 30, 32
snoop utility 106

T

tar 90
backup 90
restore 90
verify 90
TMS integrated mode 32
TPM-200/800 gateway module 29
TPM100 variants 237
training 14
troubleshooting 252
TPM card 252

V

verify, backup 90
Voice over IP (VoIP) 29