Notice
While reasonable efforts were made to ensure that the information in this
document was complete and accurate at the time of printing, Avaya Inc. can
assume no liability for any errors. Changes and corrections to the information
in this document may be incorporated in future releases.

For full support information, please see the complete document,
Avaya Support Notices for Software Documentation, document number
03-600758.
To locate this document on our Web site, simply go to
http://www.avaya.com/support and search for the document number in
the search box.

Documentation disclaimer
Avaya Inc. is not responsible for any modifications, additions, or deletions to
the original published version of this documentation unless such modifications,
additions, or deletions were performed by Avaya. Customer and/or End User
agree to indemnify and hold harmless Avaya, Avaya’s agents, servants and
employees against all claims, lawsuits, demands and judgments arising out of,
or in connection with, subsequent modifications, additions or deletions to this
documentation to the extent made by the Customer or End User.

Link disclaimer
Avaya Inc. is not responsible for the contents or reliability of any linked Web
sites referenced elsewhere within this documentation, and Avaya does not
necessarily endorse the products, services, or information described or offered
within them. We cannot guarantee that these links will work all of the time and
we have no control over the availability of the linked pages.

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

Copyright
Except where expressly stated otherwise, the Product is protected by copyright
and other laws respecting proprietary rights. Unauthorized reproduction,
transfer, and or use can be a criminal, as well as a civil, offense under the
applicable law.

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://www.avaya.com/support.
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete a User (deleteUser)</td>
<td>28</td>
</tr>
<tr>
<td>List Attribute Name (listAttributeNames)</td>
<td>28</td>
</tr>
<tr>
<td>Search for User (simpleSearch)</td>
<td>28</td>
</tr>
<tr>
<td>Verifying User Existence (extantUser)</td>
<td>29</td>
</tr>
<tr>
<td>User Service Requests for Service Control</td>
<td>29</td>
</tr>
<tr>
<td>Listing of Distributors (listDistributors)</td>
<td>29</td>
</tr>
<tr>
<td>Synchronization (synchronize)</td>
<td>29</td>
</tr>
<tr>
<td>Reset Failure Count (resetFailureCount)</td>
<td>30</td>
</tr>
<tr>
<td>Initialization (initialize)</td>
<td>30</td>
</tr>
<tr>
<td>Using the System Management Service</td>
<td>30</td>
</tr>
<tr>
<td>System Management Service Requests</td>
<td>31</td>
</tr>
<tr>
<td>Submit Request (submitRequest)</td>
<td>31</td>
</tr>
<tr>
<td>Release Request (release)</td>
<td>32</td>
</tr>
<tr>
<td>System Management Service Models and Operations</td>
<td>33</td>
</tr>
<tr>
<td>Exceptions</td>
<td>35</td>
</tr>
<tr>
<td>Cleanup</td>
<td>37</td>
</tr>
<tr>
<td>Security considerations</td>
<td>37</td>
</tr>
<tr>
<td>Authentication</td>
<td>37</td>
</tr>
<tr>
<td>Generating Certificates</td>
<td>38</td>
</tr>
<tr>
<td>Installing a self-signed certificate on a Web Service client</td>
<td>38</td>
</tr>
<tr>
<td>Chapter 4: Debugging</td>
<td>41</td>
</tr>
<tr>
<td>Improving performance</td>
<td>41</td>
</tr>
<tr>
<td>Getting support</td>
<td>42</td>
</tr>
<tr>
<td>Index</td>
<td>43</td>
</tr>
</tbody>
</table>
Chapter 1: AE Services Web Services

This chapter provides an overview of the web services that AE Services supports. These services are provided through three separate Web Service interfaces. They allow easy integration with enterprise applications and aggregation of services to implement compound operations.

This chapter describes the services provided by the following Web Service interfaces:

- Telephony Web Service
  Exposes core telephone functionality and enables access to basic 3rd party call control features on Communication Manager.

- User Service
  Implements a directory of communication application users. Users will be identified and authenticated by accessing this directory, and services will refer to it to get user related information.

- System Management Service
  Enables Web Service access to managed objects on Communication Manager.

Intended Audience

This document is written for applications developers. A developer must:

- have experience with telecommunications and basic Web Services programming
- be familiar with Web Service Description Language (WSDL)
- be familiar with XML Schema Definition (XSD)
- understand telephony concepts

You do not need to understand CSTA concepts or Avaya Communication Manager features and concepts, but they both might be helpful.

For those new to Avaya Communication Manager, you may wish to take a course from Avaya University (http://www.avaya.com/learning) to learn more about Communication Manager and its features. It is recommended that you start with the Avaya Communication Manager Overview course (course ID AVA00383WEN).
The Telephony Web Service provides a high level interface to a small subset of call control services. Call control services allow applications to control a device’s participation in calls on a switch. The Service hides the complicated concepts associated with traditional CSTA based call control such as connections, call identifiers and call states. Applications are not required to maintain any call state or information other than a session ID to manage their session on the server.

This service provides no events. For instance, it allows a user to originate an outbound call, but does not provide any subsequent events about the call progression. If monitoring is required, the user of the Web Service application must perform it.

This service presents a session-based model to clients. Sessions can be explicitly or implicitly created. Explicit sessions occur when the attach request is sent with credentials. Sessions are implicitly created when a new request is sent to the service and valid credentials are provided. The Telephony Web Service authenticates the given credentials with the User Service.

To learn how to use the Telephony Web Service call control services, see Using the Telephony Web Service on page 22.

This service supports these requests:

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make Call</td>
<td>Originates an outbound call from one party to another</td>
</tr>
<tr>
<td>Single Step Conference Call</td>
<td>Conferences in a new party to the existing active call</td>
</tr>
<tr>
<td>Single Step Transfer Call</td>
<td>Transfers the remote party on the current active call to the specified destination</td>
</tr>
<tr>
<td>Disconnect Active Call</td>
<td>Disconnects the specified extension from the active call</td>
</tr>
<tr>
<td>Answer Alerting Call</td>
<td>Answers an alerting call at the specified destination</td>
</tr>
<tr>
<td>Attach</td>
<td>Allows an application to authenticate the user and explicitly establish a session before invoking a telephony request</td>
</tr>
<tr>
<td>Release</td>
<td>Explicitly ends the session</td>
</tr>
</tbody>
</table>
User Service

The User Service is a shared service component enabling the management of user profile data in a common user store. The service provides the ability for web service applications to authenticate users against an LDAP data store of authorized users and allows you to move/add/change user profiles.

The User Service interface provides two sets of operations. The first set consists of operations to manage user data profiles. This set of operations allows an application using the User Service to list, search, add, modify and delete users in the common user store.

The second set of operations support administrative control of the service itself. These operations include the ability to (re)initialize the service with configuration settings and operations to manage any configured distributors.

The User Service provides software distributors (software adapters) for clients that wish to receive routed user events and/or information. Distributors are customized for each target product space. They can also be developed to integrate enterprise application spaces. Please contact support for more information on creating distributors.

A web page interface to the User Service is provided from the OAM web pages.

To learn how to use the User Service, see Using the User Service on page 26.

This service supports these requests for User Management:

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List Users</td>
<td>Lists the users in the common user store.</td>
</tr>
<tr>
<td>Get a User’s Information</td>
<td>Provides the user profile for a user in the common user store.</td>
</tr>
<tr>
<td>Create a User</td>
<td>Adds a user to the common user store.</td>
</tr>
<tr>
<td>Modify a User</td>
<td>Edits the user profile for a user in the common user store.</td>
</tr>
<tr>
<td>Authenticate a User</td>
<td>Authentificates a user by username or password.</td>
</tr>
<tr>
<td>Delete a User</td>
<td>Deletes the user from the common user store.</td>
</tr>
<tr>
<td>List Attribute Name</td>
<td>Lists what attributes are supported by the User Service.</td>
</tr>
</tbody>
</table>

Table 2: User Service Requests for User Management
Table 2: User Service Requests for User Management (continued)

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify User Existence</td>
<td>Determines if a user handle is in use.</td>
</tr>
<tr>
<td>Search for User</td>
<td>Searches for a particular user by ID, Last name, or common name.</td>
</tr>
</tbody>
</table>

Table 3: User Service Requests for Service Control

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Listing of Distributors</td>
<td>Retrieves information about the running distributors of the service.</td>
</tr>
<tr>
<td>Synchronization</td>
<td>Calls for a synchronization between the User Service and a specified list of application level user spaces.</td>
</tr>
<tr>
<td>Reset Failure Count</td>
<td>Allows the failure count to be reset.</td>
</tr>
<tr>
<td>Initialization</td>
<td>Allows an administrator to reinitialize.</td>
</tr>
</tbody>
</table>

This service supports these requests for Service Control:
System Management Service

The System Management Service exposes management features of Communication Manager. This service enables its clients to display, list, add, change and remove specific managed objects on Communication Manager. This service provides programmatic access to a subset of the administration objects available via the CM SAT screens. This service can be used to configure AE Services in Communication Manager. This service also provides the ability to find out the status of resources on Communication Manager.

To learn how to use the System Management Service, see Using the System Management Service on page 30.

This service supports these System Management Requests:

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit Request</td>
<td>Submits a service request. This method is used to invoke any of the available commands for any of the supported models.</td>
</tr>
<tr>
<td>Release Request</td>
<td>Used to explicitly end a session. Will release the resources associated with a session and terminate the session</td>
</tr>
</tbody>
</table>
Chapter 2: Getting started

This section describes what you need to do and what you need to know before you begin programming to this API, including:

- Setting up the development environment on page 11
- Request and response framework on page 14
- Learning from sample code on page 16

Setting up the development environment

Before running an application you will need to have a connector server machine set up as well as having the Application Enablement Services software installed. See the appropriate Installation guide for the offer you purchased (bundled server or software only) for instructions on setting up a connector server machine and installing AE Services.

Obtaining Web Service API SDK’s

The Web Service API SDK’s are included in the IP Communication SDK CD. They can also be found on the web at the Avaya Developer’s Connection web site. www.devconnectprogram.com.

The Web Service API SDK files are:

- Telephony Web Service: telsvc-sdk-<version number>.zip
- User Service: usersrvc-sdk-<version number>.zip
- System Management Service: smssvc-sdk<version number>.zip

Expand the SDK ZIP files using any application or tool that recognizes the ZIP file format.

- Telephony Web Service - All of the SDK files are placed into a directory named telsvcsdk.
- User Service - All of the SDK files are placed into a directory named usersrvc.
- System Management Service - All of the SDK files are placed into a directory named smssvc.
Getting started

The location of the sample applications provided with the SDK is:

- telsvcsdk/example
- usersrvc/example
- smssvc/example

Web Service RPM’s

The following Web Service RPM’s should have been installed on the AES server during the installation of AE Services. Verify that they have been installed and are configured correctly.

To verify the installation of the RPM’s, use the swversion command:

From the command line:

- /opt/mvap/bin/swversion -a

From the OAM web pages:

- log onto the OAM pages
- go to CTI OAM Home-->Help-->About AE Services

This will give you a list of all MVAP RPM’s that have been installed. This list should include:

- User Service RPM
- User Service LDAP module RPM
- Telephony Web Service RPM:
- System Management Service RPM
- Call Control RPM for Telephony Web Service

3rd party requirements

The following 3rd party RPM’s should have been installed on the AES server during the installation of AE Services. Verify that the following 3rd party requirements are met and have been configured correctly.

- Required by User Service
  - Tomcat version 5.5.9
  - Axis 1.2
  - OpenLDAP 2.2.13-4
- Required by Telephony Web Service
  - Tomcat version 5.5.9
  - Axis 1.2
Setting up the development environment

- Required by System Management Service
  - Apache HTTP Server, version 2.0.52-22
  - PHP 4.3.9-3.9
  - PEAR SOAP 4.3.9-3.1, HTTP_Request and NET_URL
  - OpenSSL 0.9.7a-43.8

Note:
Both Tomcat and Axis require the latest J2SE JDK installed (currently 1.5.0_02).

Note:
See the appropriate Installation Guide for the offer you have purchased (bundled server or software only) for details on installing and customizing Axis and Tomcat.

Obtaining the WSDL document(s) and generating the stubs

The WSDL's for the Web Services are available for convenience in the SDK. To guarantee that you have a WSDL that is up to date, they are also retrievable from the AE Services server. The following URL's must be used to retrieve the WSDL from the server.

- User Service
  - https://ServerName:8443/axis/services/UserService?wsdl (secure)
  - http://ServerName:8080/axis/services/UserService?wsdl (non-secure)

- Telephony Web Service
  - https://ServerName:8443/axis/services/TelephonyService?wsdl (secure)
  - http://ServerName:8080/axis/services/TelephonyService?wsdl (non-secure)

- System Management Service

Note:
For System Management Service ports 443 and 80 are defaults. Those customers who customized their installation should refer to the apache configuration files for their installation.

⚠️ Important:
The secure port is always recommended for all of the Web Services
The page retrieved should be saved as a WSDL file on the application development machine. These files can be compiled into client application stub code with a language specific SOAP toolkit’s WSDL compile/code generator. This document will describe stub code generation for the AXIS client side Java development scenario. Other client side technologies would feature a similar approach in general, but will differ in the detailed instructions. See your preferred toolkit’s documentation for specifics.

For the Axis client side Java binding, the instructions for generating client stub code are as follows:

1. Install the Apache Axis Java library files from: http://ws.apache.org/axis/.
2. Use the Axis wsdl2Java program to generate source code for the target WSDL file.
3. Use the Java compiler to compile the generated stub source code. Do this by navigating to the bottom of the package directory structure (i.e. /com/avaya/ws/stub/UserService) and issuing the following command: javac *.java.
4. Ensure that these classes are available on your development class path.

Note:
The Axis client side jars are also included in the SDK along with the generated stubs for the Telephony and User Services.

---

**Request and response framework**

Your application will need to be able to:

- make service requests
- parse service responses
- catch exceptions

The SOAP and HTTP headers in the requests and responses carry user credentials and session information. This section describes what requests, responses and faults are, and how to use them.

---

**Service requests**

A client application sends service requests to the connector server. Examples of a request are “originate an outbound call” and “disconnect an active call” when using the Telephony Web Service, or “add a user to the user store”, and “list the users in the user store” when using the User Service.
To make a request, the application calls a method on the client side service stub that was generated by their SOAP platform. The client side SOAP framework then sends a SOAP request to the service. Each request is processed by the service. Once the request is completed, the service will either send a positive response or a SOAP fault.

Service responses

Each service request is acknowledged with a service response (positive acknowledgement). The results of the request are reported in the service response.

In some cases, response to a request may result in an error. These errors are indicated through SOAP faults (under Axis, through Axis faults).

Service responses will be returned in the form of objects by the service request method on the client side stubs generated by your SOAP framework.

SOAP (or Axis) Faults

Each service request may generate an exception, referred to as a SOAP or Axis fault. An exception indicates that the service request has failed for some reason.

Each exception will include a description of what happened and the appropriate response to a received exception will vary according to what the fault was. It will be up to the application as to how it chooses to handle such messages.

Details on exceptions that can be received, and possible causes and solutions can be found in section Exceptions on page 35.

We recommend that the application log all possible exceptions since this will be an important source of information for debugging the application. Look at the server side logs for more information about what happened. The logs will contain helpful information that you may need to provide to DevConnect.

The server side logs are located at:

- Telephony Web Service and User Service
  /opt/mvap/logs/tomcat
- System Management Service
  /var/log/httpd
Learning from sample code

Another key learning tool is the set of sample code files that are provided with this API.

- **User Service**
  
  A test client, with source code, is included as part of the SDK. The test client demonstrates usage of the generated stubs, and basic interaction with the User Service. The source code for the client application is thoroughly documented to explain the steps involved with connecting to the target service, and interacting with the service for basic user profile oriented operations.

  The sample code is located in the following directory:

  usersrvc/examples/src/cusclientdemo

  The client stubs have been compiled in the file `cusstubs.jar` and placed into the following directory:

  usersrvc/example/lib

  To run the program, navigate to the `usersrvc/examples/src/cusclientdemo` folder and issue the following command on the command line:

  `runclient`

- **Telephony Web Service**

  A sample is provided which demonstrates basic call control flow.

  The sample code is located in the following directory:

  telsvc-sdk/example/src/sampleapps/telSvcClient

  To run the Telephony Web Service sample application navigate to the `telsvc-sdk/example/bin` folder and execute the following command:

  `ant.sh runTelSvcClient`

- **System Service**

  A test client, with source code, is included as part of the SDK. The test client demonstrates usage of the generated stubs, and basic interaction with System Management Service. It executes a single service request and displays the result, releases the session and exits.

  The sample code is located in the following directory:

  smssvc/example/src

  The client stubs have been compiled and placed into the following directory:

  smssvc/example/bin
To run the program, navigate to the `usersrvc/example/src` folder and issue the following command on the command line:

```
run -root http(s)://<smshost> -login mylogin@cmserveraddr -pw mypassword
```

This will execute the default command for the test application using the selected Communication Manager and login. You can experiment with the command-line options to try different commands.

The source code is fully commented and also serves as a brief tutorial for coding System Management Service client applications in Java using Axis. The source file `SMSTest.java` is located in the following directory:

```
smssvc/sampleapps/sms
```

**Note:**

While the System Management Service does not require a JDK to be installed, the sample code does require the latest J2SE JDK installed (currently 1.5.0_02).
Getting started
Chapter 3: Writing a client application

This chapter describes how to write an application using the Web Services.

Your application may have these different parts:

- **Setup**
  - The HTTP header
  - The SOAP header
  - Setting the URL of the service to be accessed

- **Using the Telephony Web Service**
  - Telephony Web Service Requests

- **Using the User Service**
  - User Service Requests for User Management
  - User Service Requests for Service Control

- **Using the System Management Service**
  - System Management Service Requests
  - System Management Service Models and Operations

- **Exceptions**

- **Cleanup**

- **Security considerations**

- **Authentication**

- **Generating Certificates**

Each part is described in the sections below.

---

**Setup**

This section describes the different setup steps that must be taken by every application.
The HTTP header

During processing of the initial request, the application will need to provide credentials in the HTTP header. When credentials are provided in the header, a new session will automatically be created if needed.

Credentials are passed in the HTTP authorization header using basic authentication semantics. The basic authentication username consists of `username@switch:port`. The basic authentication password is the password for the specified user. The sample applications demonstrate how to do this for an Axis client side framework.

⚠️ Important:
The `username/password` are passed in clear-text when using HTTP (non-secure port) which is why we recommend using HTTPS (secure port).

The `switch` and `port` are optional:

- **port:**
  
The `port` portion of the `username` only applies to System Management Service. You will not specify a "port" for the Telephony or User Service. If the port is not specified, the default port from the ini file will be used.

- **switch:**
  
The `switch` portion of the `username` only applies to the Telephony Web Service and the System Management Service.

  If the `switch` portion is not specified:

  - System Management Service - the default `switch` from the ini file will be used. The `switch` name should be the actual Communication Manager machine (DNS) name or ip address.
  
  - Telephony Web Service - the first `switch` on the list of switches administered through the OAM web pages is tried. The `switch` name should be the `Connection Name` administered in Application Enablement Services.

The SOAP header

SessionID information will be passed between the Web Service Client and the AE Server in the SOAP header of the requests and responses. The SessionID is included in the SOAP header of the response and the application will need to copy this SessionID from this response and include it in the SOAP header of the next request to maintain the session. The sample applications demonstrate how to do this for an Axis client side framework.
The attributes of the SessionID are:

1. the SessionID is a 32-character identifier (128 bit hex number).
2. the namespace is "http://xml.avaya.com/ws/session"
3. the name of the element is "sessionID"

**Note:**
SessionID's are used for the Telephony Web Service and the System Management Service. They are not used for the User Service.

A session shall be created in the following 2 cases:

- Valid credentials are provided in the HTTP header and no SessionID is provided in the SOAP header. If no SessionID is included in a request, then the service will use the credentials in the HTTP header to implicitly create the session and return a new SessionID in the SOAP header of the response.
- Valid credentials are provided in the HTTP header and the SessionID specified in the SOAP header does not exist (either it has expired or is invalid).

**Note:**
It is possible for the SessionID to change from one request to the next (though not likely). For example, if the server is restarted between the client requests, the old session will be invalidated and a new session (with a new SessionID) created for the subsequent client request.

---

**Setting the URL of the service to be accessed**

You must set the URL of the service to be accessed:

- **User Service**
  - https://ServerName:8443/axis/services/UserService (secure)
  - http://ServerName:8080/axis/services/UserService (non-secure)
- **Telephony Web Service**
  - https://ServerName:8443/axis/services/TelephonyService (secure)
  - http://ServerName:8080/axis/services/TelephonyService (non-secure)
- **System Management Service**
  - https://ServerName:443/sms/SystemManagementService.php (secure)
  - http://ServerName:80/sms/SystemManagementService.php (non-secure)
Using the Telephony Web Service

The Telephony Web Service provides a high level interface to a small subset of call control services. Call control services allow applications to control a device’s participation in calls on a switch. The service hides the complicated concepts associated with traditional CSTA based call control such as connections, call identifiers and call states.

The following sections provide details about the service requests provided by the Telephony Web Service as well as common exceptions your application may receive in response to these requests.

This service provides no events. For instance, it allows a user to originate an outbound call, but does not provide any subsequent events about the call progression. If monitoring is required, the user of the Web Service application must perform it.

Telephony Web Service Requests

Each request will contain the username, password and optional sessionID (all SOAP headers sent subsequent to establishing the session will contain the sessionID).

Attach (attach)

Allows an application to authenticate the user and establish a session before invoking a telephony request.

The Attach request takes no parameters

A successful response indicates that the session has been successfully authenticated and initiated. The response contains no data. It simply acts as a positive acknowledgement to the request.

The Telephony Web Service returns an exception to indicate the Make Call request was unsuccessful.

Fault values for this request include.

- InvalidSessionException - The session has expired or was invalid
- InvalidCredentialsException - The supplied credentials are invalid
- SwitchNotReachableException - The switch cannot be reached

Make Call (makeCall)

Originates an outbound call from one party to another.
The Make Call request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>originatingExtension</td>
<td>String</td>
<td>The originating extension.</td>
</tr>
<tr>
<td>destinationNumber</td>
<td>String</td>
<td>Number to dial to make the call. For external calls this must contain the necessary ARS code.</td>
</tr>
</tbody>
</table>

A successful response indicates that a call has been successfully initiated from the originating extension and is now alerting at the destination. It does not signify that the destination has answered the call. The response contains no data. It simply acts as a positive acknowledgement to the request.

The Telephony Web Service returns an exception to indicate the Make Call request was unsuccessful.

Fault values for this request include:
- `InvalidPartyException` - Invalid extension, destination or destination busy
- `ActiveCallExistsException` - The extension already has a call active
- `InvalidSessionException` - The session has expired or was invalid
- `InvalidCredentialsException` - The supplied credentials are invalid
- `SwitchNotReachableException` - The switch cannot be reached

**Single Step Conference Call (singleStepConferenceCall)**

Conferences in a new party to the existing active call. This request will initiate a call to the specified destination and conference it into the currently active call.

The Single Step Conference Call request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>originatingExtension</td>
<td>String</td>
<td>The originating extension with an active call that the destination party will be conferenced to.</td>
</tr>
<tr>
<td>destinationNumber</td>
<td>String</td>
<td>Number of the party to conference in to the call. For external calls this must contain the necessary ARS code.</td>
</tr>
</tbody>
</table>

A successful response indicates that a call has been successfully initiated from the originating extension and is now alerting at the destination. It does not indicate that the destination has answered the call. The response contains no data. It simply acts as a positive acknowledgement to the request.
Writing a client application

The Telephony Web Service returns an exception to indicate the Single Step Conference Call request was unsuccessful.

Fault values for this request include.

- InvalidPartyException - Invalid extension, destination or destination busy
- NoActiveCallException - The originating extension does not have a call active
- InvalidCredentialsException - The supplied credentials are invalid
- SwitchNotReachableException - The switch cannot be reached
- InvalidSEssionException - The session has expired or was invalid.

**Single Step Transfer Call (singleStepTransferCall)**

Transfers the remote party on the current active call to the specified destination.

The Single Step Transfer Call request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>originatingExtension</td>
<td>String</td>
<td>The originating extension with an active call which will be transferred</td>
</tr>
<tr>
<td>destinationNumber</td>
<td>String</td>
<td>Number to transfer the call to. For external calls this must contain the necessary ARS code.</td>
</tr>
</tbody>
</table>

A successful response indicates that a call originated from the specified extension is alerting at the destination number and the remote parties on the original active call have been transferred to the alerting destination. It does not indicate that the destination has answered the call. The response contains no data. It simply acts as a positive acknowledgement to the request.

The Telephony Web Service returns an exception to indicate the Single Step Transfer Call Request was unsuccessful.

Fault values for this request include.

- InvalidPartyException - Invalid extension, destination or destination busy
- NoActiveCallException - There is no active call to be transferred on the originating extension.
- InvalidCredentialsException - The supplied credentials are invalid
- SwitchNotReachableException - The switch cannot be reached
- InvalidSEssionException - The session has expired or was invalid.

**Disconnect Active Call (disconnectActiveCall)**

Disconnects the specified extension from the active call.
The Disconnect Active Call request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>extension</td>
<td>String</td>
<td>Extension with an active call.</td>
</tr>
</tbody>
</table>

A successful response indicates that an active call at the specified extension was successfully disconnected. The response contains no data. It simply acts as a positive acknowledgement to the request.

The Telephony Web Service returns an exception to indicate the Disconnect request was unsuccessful.

Fault values for this request include:
- InvalidPartyException - Invalid extension, destination or destination busy
- NoActiveCallException - The extension does not have a call active
- InvalidCredentialsException - The supplied credentials are invalid
- SwitchNotReachableException - The switch cannot be reached
- InvalidSEssionException - The session has expired or was invalid.

**Answer Alerting Call (answerAlertingCall)**

Answers an alerting call at the specified destination.

The Answer Alerting Call request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>extension</td>
<td>String</td>
<td>Extension with an alerting call</td>
</tr>
</tbody>
</table>

A successful response indicates that the alerting call at the specified destination has been answered. The response contains no data. It simply acts as a positive acknowledgement to the request.

The Telephony Web Service returns an exception to indicate the Answer Alerting Call request was unsuccessful.

Fault values for this request include:
- InvalidPartyException - Invalid extension, destination or destination busy
- NoAlertingCallException - The extension specified does not have an alerting call
- InvalidCredentialsException - The supplied credentials are invalid
- SwitchNotReachableException - The switch cannot be reached
- InvalidSEssionException - The session has expired or was invalid.
- ActiveCallExistsException - The extension already has a call active
Release (release)

Ends the current session, irrespective of whether it was created explicitly (using attach) or implicitly.

The Release request takes no parameters.

The Telephony Web Service returns a Release response to indicate the request was successful. The response contains no data. It simply acts as a positive acknowledgement to the request. If the session has already expired prior to the release request being sent, a positive response is still returned.

Using the User Service

The User Service is a shared service component enabling the management of user profile data in a common user store. The service provides the ability for web service applications to authenticate users against an LDAP data store of authorized users and allows you to move/add/change user profiles.

The User Service interface provides two set of operations.

- operations to manage user data profiles
- operations support administrative control of the service itself

User Service Requests for User Management

The User Service supports specific operations to allow for management of the users in the user store.

List Users (listUsers)

Lists the users in the common user store. The operation supporting the listing of users will let the caller scroll through the entire set of users (if desired) via a cursor type method. The result of a "list users" call is a batch of user handles (that can be used to retrieve a user profile) and information indicating whether or not there are more users to be listed in a subsequent call.

The number of users returned in the listUsers operation is a configurable parameter in the User Service property file.
The List Users request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>afterHandle</td>
<td>String</td>
<td>The handle of the last user returned in a previous call to ListUsers, or blank if this is the first call to ListUsers.</td>
</tr>
</tbody>
</table>

**Get a Users Information (getUser)**

Provides the user profile for a user in the common user store which consists of the common attributes of the user. These attributes are:

- uid - the unique identification handle for the user in the identity management realm
- ch - Common Name, as in John Smith
- sn - surname, as in Smith

The Get a Users Information request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>handle</td>
<td>String</td>
<td>The handle of the user you wish to retrieve.</td>
</tr>
</tbody>
</table>

**Create a User (createUser)**

Adds a user to the common user store.

The Create a User request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>User</td>
<td>A User object representing the user to be created.</td>
</tr>
</tbody>
</table>

**Modify a User (modifyUser)**

Edits the user profile for a user in the common user store.

The Modify a User request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>User</td>
<td>A User object representing the user to be modified.</td>
</tr>
</tbody>
</table>
Writing a client application

Authenticate a User (authenticateUser)

Authenticates a user by user name or password. The User Service will indicate whether or not the provided user name and password match what is stored in the LDAP repository.

The Authenticate a User request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>String</td>
<td>The username of the user to be authenticated.</td>
</tr>
<tr>
<td>password</td>
<td>String</td>
<td>The password of the user to be authenticated.</td>
</tr>
</tbody>
</table>

Delete a User (deleteUser)

Deletes the user from the common user store

The Delete a User request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>handle</td>
<td>String</td>
<td>The user you wish to delete.</td>
</tr>
</tbody>
</table>

List Attribute Name (listAttributeNames)

Lists what attributes are supported by the User Service

The List Attribute Name request takes no parameters:

Search for User (simpleSearch)

Searches for a particular user by ID, Last name, or common name. The results are returned in the same data structure as the ListUsers() operation. This function can search on any of three attributes:

- uid - unique id
- sn - last name
- cn - common name

The Search for User request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchAttribute</td>
<td>String</td>
<td>Attribute to search on.</td>
</tr>
<tr>
<td>searchValue</td>
<td>String</td>
<td>Value of the attribute to search on</td>
</tr>
</tbody>
</table>
Verifying User Existence (extantUser)

Determines if a user handle is in use.

The extantUser() request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>handle</td>
<td>String</td>
<td>User to verify.</td>
</tr>
</tbody>
</table>

User Service Requests for Service Control

The User Service will support specific operations to allow the service’s synchronization capability and current configuration to be controlled by a MultiVantage Application Platform administrator.

Listing of Distributors (listDistributors)

Retrieves information about the running distributors of the service. Distributor information includes:

- the name of the Distributor
- the name of the Distributor’s implementation class
- the current status of a synchronization (whether one is in progress or not and if not whether the last one was successful, failed or partial failed)
- the results of the last synchronization effort performed by the Distributor
- number of failures encountered by this Distributor
- size of internal queues holding events to be published to the data store associated with this Distributor

The Listing of Distributors request takes no parameters.

Synchronization (synchronize)

Calls for a synchronization between the User Service and a specified list of application level user spaces. The synchronization logic is encapsulated within the Distributor for the specific user space. The work of synchronization is conducted asynchronously and does not block normal operation of the User Service. The results of a synchronization are summarized in the DistributorInfo data structure returned from the ListDistributors() method. Additional detail would be found in the logs for the effected Distributors.
Writing a client application

The Synchronization request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>distributors</td>
<td>String[]</td>
<td>A list of distributor names.</td>
</tr>
</tbody>
</table>

**Reset Failure Count (resetFailureCount)**

Allows the failure count associated with a Distributor to be reset.

The Reset Failure Count request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>distributorName</td>
<td>String</td>
<td>Name of the distributor to be reset.</td>
</tr>
</tbody>
</table>

**Initialization (initialize)**

Allows an administrator to reinitialize. The User Service will auto-initialize when it is first instantiated by a client call. Subsequent requests to the `initialize()` operation will result in a re-initialization. The service allows the administrator to change the running configuration of the service by first changing file based properties of the service and then calling the `initialize()` operation.

The Initialization request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>String</td>
<td>File containing the properties of the service.</td>
</tr>
</tbody>
</table>

**Using the System Management Service**

The System Management Service exposes management features of Communication Manager. This service enables its clients to display, list, add, change and remove specific managed objects on Communication Manager. This service provides programmatic access to a subset of the administration objects available via the CM SAT screens. This service can be used to configure AE Services in Communication Manager. This service also provides the ability to find out the status of resources on Communication Manager.
System Management Service Requests

Submit Request (submitRequest)

This method is used to invoke any of the available commands for any of the supported models. The web service will check that the specified model name is valid, and if so, instantiates the specified OSSI Model-derived class. It then invokes on that class to perform the requested operations.

Submit Request takes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>String</td>
<td>The model name. For example, “Station” is the model name required to access station data via any of the commands: list, display, add, change or remove.</td>
</tr>
<tr>
<td>operation</td>
<td>String</td>
<td>This specifies the command to perform. Not all operations are supported by all models. The online model reference information available at https://&lt;smshost&gt;/sms/getModelClasses.php will indicate which operations are supported for each model.</td>
</tr>
<tr>
<td>objectname</td>
<td>String</td>
<td>Reserved for possible future use. This should be left blank.</td>
</tr>
<tr>
<td>qualifier</td>
<td>String</td>
<td>This is the &quot;key&quot; for the operation. For a command such as &quot;change station 12345&quot;, the qualifier is 12345. The qualifier may also be used to specify additional parameters for some commands. Users who are familiar with SAT commands will recognize this as the third part of a SAT command. Documenting the full set of possible qualifier values for all available commands is beyond the scope of SMS. This information may be obtained from other available sources (such as the CM SAT help screens, for example).</td>
</tr>
</tbody>
</table>
Writing a client application

The Submit Request Response includes the following parameters:

<table>
<thead>
<tr>
<th>Name of Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fields</td>
<td>String</td>
<td>For a read command, this is an optional pipe-delimited list of fieldnames to read. For read operations, the value &quot;*&quot; may be used to read all the defined fields in the model. For write commands (commands that set the value of data attributes), this is a list of &quot;fieldname=value&quot; pairs, specified in the form: <code>fieldname=value[fieldname=value...]</code></td>
</tr>
<tr>
<td>result_code</td>
<td>int</td>
<td>0 for success, 1 for failure. If a connection to OSSI fails, a SOAP fault is sent back. If a connection is established, but the OSSI request fails (for instance an invalid qualifier was specified), then a submitRequest response is returned and this field will contain a 1.</td>
</tr>
<tr>
<td>result_data</td>
<td>String</td>
<td>For display or list commands (commands that get the value of data attributes), this is a list of fields and values, specified in the form: <code>fieldname=value[fieldname]value...</code> For certain array-based fields, and for commands that return a list of objects, array-indexes are used, i.e., the result data is in the form: `field1[0]=v0</td>
</tr>
<tr>
<td>message_text</td>
<td>String</td>
<td>If result_code is 1, this field contains the protocol failure message received from OSSI.</td>
</tr>
</tbody>
</table>

Release Request (release)

This request will release the resources associated with a session and terminate the session. This operation must be called when a client is finished with the SMS session.

The SessionID to release shall be specified in the SOAP header with the request. A release response shall be sent back to the client. If the SessionID is already expired, a positive response shall still be sent back.
This operation takes no parameters. It returns the same result structure as the submitRequest method, however, in all cases, the **result_code** will be 0 (success), and **result_data** and **message_text** will be empty. (Any severe error conditions encountered during a release call will result in a SOAP fault being raised, as opposed to returning a result_code of 1.

---

**System Management Service Models and Operations**

It is beyond the scope of this usage document to provide detailed documentation of all the possible combinations for each of the available models and operations. Detailed help is currently available for each of the Communication Manager administration commands via the SAT.

Following is a listing of all the models currently supported by the System Management Service. Detailed help is currently accessible by pointing your browser to the installation host machine at http://<hostname>/sms/getModelClasses.php. The information provided for each supported model includes the set of allowable operations, and the available fields.

**Currently supported models:**

- AARAnalysis
- AESSvcsCTILink
- AESSvcsInterface
- AESSvcsLink
- ARSAnalysis
- Agent
- AliasStation
- BCMSAgentReport
- BCMSSkillReport
- BCMSTRunkReport
- BCMSVDNReport
- AuthorizationCode
- COR
- CTILink
- Configuration
- CoveragePath
- CustomerOptions
- DefaultRegion
- DialPlan
Writing a client application

- ExtensionStation
- FeatureAccessCodes
- HuntGroup
- IPAddressUsage
- IPCCodec
- IPHoteling
- IPInterface
- IPNetworkMap
- IPNetworkRegion
- IPNetworkRegionStatus
- IPEndPointNames
- IPRegistration
- IPServices
- ListExtensionType
- ListHistory
- ListIPInterface
- ListIPNetworkRegionMonitor
- ListNodeNames
- ListUsageExtension
- Locations
- OffPbxStationMapping
- SignalingGroup
- SiteData
- SoftwareVersion
- SpecialApplications
- Station
- SystemFeatures
- SystemParameters
- TrunkGroup
- VDN
- WitnessStation
## Exceptions

Exceptions that you may encounter when using the Telephony Web Service are listed below along with their potential solutions:

### Table 5: Telephony Web Service exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Potential cause and solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Session Exception</td>
<td>This means that the session has expired or is invalid. Ensure that the AE Services connector server is up and administered correctly. To recover from this exception, your application will need to send valid credentials in the HTTP header of the next request.</td>
</tr>
<tr>
<td>Active Call Exists Exception</td>
<td>This occurs when a Make Call request is received for an extension and an active call already exists at the specified extension. Send the request again when there are no active calls on the extension.</td>
</tr>
<tr>
<td>Invalid Credentials Exception</td>
<td>This means that the credentials you provided in the HTTP header were invalid. This occurs if either the username or password was invalid when establishing a new session. Ensure that the given user has been administered in the User Management OAM pages.</td>
</tr>
<tr>
<td>Invalid Party Exception</td>
<td>This means that one of the specified parties was invalid or the user did not have permission to access the extension. Ensure that the extension is configured correctly on Communication Manager and on the AE Services connector server. Ensure that the originating device has been administered in the SDB, and that the user is authorized to control the originating device.</td>
</tr>
</tbody>
</table>
Exceptions that you may encounter when using the User Service are listed below along with their potential solutions:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Potential cause and solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Active Call Exception</td>
<td>This occurs when a Single Step Conference Call, Single Step Transfer Call, or Disconnect Active Call request has been received for an extension, but there is no active call on that device. Send the request again when an active call exists on the extension.</td>
</tr>
<tr>
<td>No Alerting Call Exception</td>
<td>this occurs when an Answer Alerting Call request is received for an extension, but there is no alerting call on that extension. Send the request again when an alerting call exists on the extension.</td>
</tr>
<tr>
<td>Switch Not Reachable Exception</td>
<td>This occurs if either the TSAPI service is not accessible or if there is not an active Switch Connection to the specified switch. Verify that the Switch Connection to the given switch is operational, and that the given switch name matches the Switch Connection name exactly.</td>
</tr>
</tbody>
</table>

Exceptions that you may encounter when using the System Management Service actually come from the switch. It is out of the scope of this documentation to try to list all these. However the cause -- and where possible the resolution -- should be clearly indicated in the fault description text.
Cleanup

This section applies only to applications using the System Management Service and the Telephony Web Service.

It is important to free resources when they are no longer needed. The System Management Service and the Telephony Web Service have a "Release" request available. This request ends the session and frees all resources associated with that session. Applications should call this request when they are done using the service.

Security considerations

Your application development organization has the responsibility of providing the appropriate amount of security for your particular application and/or recommending appropriate security measures to your application customers for the deployment of your application. Therefore you should be aware of the security measures that Application Enablement Services already takes and what risks are known.

Security measures taken by AE Services include:

- A valid username and password are required to access all of the web services.
- All of the web services may be secured by using the secure SSL port (8443 for User and Telephony Web Service and 443 for System Management Service).
- Authorizations may be administered in the TSAPI Security Database to specify which devices may be controlled by which users of the Telephony Web Service.

The security risks include:

- SOAP messages transmitted between the client application and the connector server software are unencrypted if using the non-secure port (HTTP). To avoid this risk we recommend that you use the secure SSL port.

Avaya recommends that the application machine, connector server and Communication Manager machines reside on a private LAN.

Authentication

Users of the Telephony Web Service are by default authenticated with the User Service just as users of the TSAPI services are.

System Management Service users are authenticated just as Communication Manager users are.
Generating Certificates

The Avaya Tomcat RPM includes a default certificate and a default keystore of encryption keys for use in connecting to the AE Services server via SSL. However, if you installed a Tomcat RPM from a different source, you must create these yourself.

Installing a self-signed certificate on a Web Service client

Some Web service client frameworks require server authentication when using SSL. If using such a framework, you must install the self-signed certificate on the Web service client machine(s).

To install the self-signed certificate:

1. Run the following command on the server:

   ```
   keytool -export -alias key_alias -keystore keystore_name -storepass keystore_password -file certificate_filename
   ```

   The self-signed certificate is exported from the keystore and stored in a file called AEcert.crt.

   **Note:**
   When running `keytool -export`, you must use the values you specified for `alias`, `keystore`, and `storepass` when you created the original server keystore.
   For example, if you are using the default keystore that ships with the Avaya-Tomcat RPM, then the command becomes:

   ```
   keytool -export -alias avayakey -keystore $CATALINA_HOME/conf/defaultkeystore -storepass defaultpassword -file AEcert.crt
   ```

   However, if you have replaced the default server keystore with a new keystore, you must use the values specified for `alias`, `keystore`, and `password` when the new server keystore was created. For example, if you used `makecert.sh` to create a new keystore, then `alias = avayakey`, `keystore = $CATALINA_HOME/conf/keystore`, and `password = no_default` (you must remember what password was used).

2. Move the generated certificate file from the server to the Web Service client machine.
3. Install the AE server certificate on the Web Service client machine. For example, this is how to install the certificate when using a Java-based Web Service client framework, such as Axis.

```
keytool -import -alias certificate_alias -file certificate_filename
   -keystore keystore_name -storepass keystore_password -noprompt
```

The certificate is imported into the keystore (keystore_name) as a trusted certificate. This keystore should then be utilized by the client application when opening a secure connection with the AE Server.

**Note:**

The Telephony Web Service SDK has a shell script/batch file to import a certificate into a keystore, and provides an example of how to securely connect using Axis. You can use that provided script instead of running the `keytool -import` command.
Writing a client application
Chapter 4: Debugging

In debugging your application, you will rely on:

1. **Exceptions** that your application received and logs. See Exceptions on page 35.

2. **Server-side logs** found at:
   - Telephony Web Service and User Service
     
     /opt/mvap/logs/tomcat/
   - System Management Service
     
     /var/log/httpd/

   See the Management guide to learn more about the server's logs.

**Note:**
You can also download the logs from the OAM web pages.

The remainder of this chapter describes:

- **Improving performance** on page 41
- **Getting support** on page 42

---

Improving performance

Many factors can affect the performance of your system. The system has three main parts that may be affected:

- the AE server
- Communication Manager
- the network

An excessive load on any of these has the potential to slow down the overall system. Here are other factors to check that also may affect your system performance.

**On the AE server:**

- Ensure that your AE server machine meets the minimum requirements specified in the Installation guide.
- Avoid running any other applications on the AE server machine.
Debugging

- Check that the AE servers Linux operating system resources are tuned correctly for your application needs. The AE server software makes no assumptions concerning your application needs and therefore does not tune the kernel for you. The Installation guide provides some guidance in tuning Linux.

- Update the Linux kernel with the latest updates available.

On Communication Manager:

- Ensure that Communication Manager is properly configured for your network and business needs. Misconfigured switch elements can lead to performance hits.

On the network:

- Ensure that your network traffic is properly balanced. One way to do this professionally is to ask Avaya to perform a network assessment. There is also a VoIP Readiness Guide available from the Avaya Support Centre (http://www.avaya.com/support). For more information about improving the performance of your network, see the “Network Quality and Management” section of Administration for Network Connectivity for Avaya Communication Manager (555-233-504).

Getting support

Development support is only available through Avaya's Developer Connection Program (DevConnect) at this time. As an Innovator/Premier/Strategic level member of the DevConnect Program, technical support questions can be answered through the DevConnect Portal at www.devconnectprogram.com.

As a Registered member of the program, support is not available. If you require support as a Registered member, you can apply for a higher level of membership that offers support and testing opportunities through the DevConnect Portal. Membership at the Innovator/Premier/Strategic level is not open to all companies. Avaya determines membership status above the Registered level.
# Index

## A
- acknowledgements
  - positive .................................. 15
- application
  - developing ................................ 19
  - improving performance ..................... 41
- Avaya University .................................. 5

## C
- cleanup ........................................ 37
- Communication Manager
  - network configuration .......................... 42
  - performance .................................. 41
- connector server
  - performance .................................. 41

## D
- demonstration source code ....................... 16
- deployment
  - security ..................................... 37
  - development environment ....................... 11

## E
- exceptions
  - used for debugging .................................. 41

## F
- freeing up resources .................................. 37

## L
- Linux kernel
  - updates ....................................... 42
- Linux operating system
  - tuning ......................................... 42
- logs
  - server ........................................ 41

## N
- network
  - performance .................................. 41

## P
- performance ....................................... 41
- Physical Device Services ............................ 6, 7, 9
- physical elements .................................. 6, 7, 9
- positive acknowledgement ............................ 15

## R
- requests ......................................... 14
- responses ......................................... 15

## S
- sample code ....................................... 16
- security
  - considerations .................................. 37
- server-side logs .................................. 41
- service
  - requests ....................................... 14
  - responses ....................................... 15

## V
- VOIP
  - readiness guide .................................. 42

## W
- websites
  - Avaya University .................................. 5

## X
- XSD ................................................. 5