Avaya Aura®
Session Manager PPM Interface Specification for release 8.1.x
October 2019

Compas ID 152065
Version 2.10

Notices 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 Avaya shall not be 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. 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 End User.
Link disclaimer Avaya is not responsible for the contents or reliability of any linked Web sites referenced within this site or documentation(s) 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 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 to Avaya customers and other parties through the Avaya Support Web site: http://www.avaya.com/support Please note that if you acquired the product from an authorized reseller, the warranty is provided to you by said 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, AND 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 the Documentation(s) and Product(s) provided by Avaya. All content in this documentation(s) and the product(s) 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. 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(s) 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(s) 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.
Downloading documents For the most current versions of documentation, see the Avaya Support. Web site: http://www.avaya.com/support
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:// www.avaya.com/support**

Document Change History

Document Issue	Document Date	Compas MR	Change Description
0.1	May 2011	152065	First release of the interface specification for 6.2. Added ServiceType to getHomeCapabilitiesResponse.
0.2	May 2011	152065	Added Call Center PubType changes.
0.3	June 2011	152065	Modified the Introduction and added the getInitialEndpointConfiguration request/response, setOneTouchDialList request/response, a description of the Emergency Number information element and a brief note about the 3rd party Enhanced Call Forward button.
0.4	August 2011	152065	Added LBDS (Location Based Device Settings) information
0.5	October 2011	152065	Added information about supporting REMO users, multiple domains per user in PPM requests, Data Synchronization section updates, Speed Dial clarification, Device Settings data clarifications, and supported MAC address formats.
0.6	November 2011	152065	Added information on Multi Location Dial Plan (MLDP), Bridged call alerting (BCA), and Per Button Ring Control (PBRC).
0.7	November 2011	152065	Increased maximum size of device data values from 8,000 to 64,000 bytes.
0.8	November 2011	152065	Added description of primary number to contact handling.
0.9	January 2012	152065	Correct how the setOneTouchDialList operation is documented.
1.0	April 2012	152065	Corrected updateContact example in section 3.1.4. Added subsection 3.5.1 documenting PPMOperationalFault format and 3.5.2 explaining PPM's 503 response with Retry-Header. Added sections 7 & 8. Miscellaneous clarifications.

Document Issue	Document Date	Compas MR	Change Description
1.1	October 2012	152065	Added some clarifying information on the PPM URL, PPM Error Conditions, HTTP 503 generation, the PPM requests that should be made by AST Endpoints after receiving the Reload data SIP NOTIFYs, and the Endpoint Management Agent. Some unused PPM Fault codes were removed and various typos were corrected.
1.2	November 2012	152065	Miscellaneous changes in support of 6.2 and 6.2 FP1 including Endpoint Management; Mute On Shared Control (getAllEndpointConfiguration); general contact information; and changes to limits for getContactList, device limits, and emergency numbers
1.3	March 2013	152065	Added clarifying information regarding the <Name> and <Alias> elements in addContactList; added <EndpointDisplayName> element to getContactList (FP1); clarified the use of the term “short form” throughout the document; added clarifying information about the SpeedDial parameter in addContact; corrected description of <PpmServer> in getHomeServer; improved description of <DataValue> in get/setDeviceData; updated getAllEndpointConfiguration/setVolumeSettings to include the <Identity> element; updated description of external contact handle generation in section 3.7.
1.4	August 2013	152065	Added information about the new IM field in the ContactData section of getContactList responses, updateContact requests, and addContact requests.
1.5	September 2013	152065 (MR 81987)	Add changes for the FP3 Localization (Name1/Name2) feature and make some other necessary document updates.
1.6	April 2014	152065	Fixed the Table of Contents.
1.7	May 2014	152065	Added Presence Services to getHomeCapabilities response.

Document Issue	Document Date	Compas MR	Change Description
1.8	June 2014	152065	Add Call Journaling operations and enhancements to searchUser.
1.9	August 2014	152065	Add SM 6.2 Feature Pack 4 Team button enhancements, the Limit Number of Concurrent Calls button, Transfer-To-Voicemail enhancements, changes related to the SIP Phone Template feature, and some other miscellaneous changes.
2.0	October 2014	152065	Correct the description of the IsBuddy flag for contacts.
2.1	May 2015	85097	Add SM 7.0 end-to-end call security indicator and Service Observing button enhancements.
2.2	June 2015	86357	Add SM 7.0 gHCResponse changes regarding Listen Port, FQDN and FB Policy.
2.3	January / March 2016	87193, 87434	Add 7.0.1 buttons: Hunt Group Busy, Add/Remove Skill, VOA Repeat
2.4	May 2016		Add add/update/delete Contact request/response changes done for AADS
2.5	January 2017	87975, 88175, 88917	IPv6 changes for getHomeCapabilities; Complex Password changes for getAllEndpointConfiguration
2.6	February 2017	89028	IPv6 changes for getAllEndpointConfiguration and getInitialEndpointConfiguration
2.6.1	March 2017	89141, 89145	Minor corrections to §3.2.2.9 and §6; updated Copyright
2.6.2	May 2017	89453	Device settings are now current location based instead of home location.
2.6.3	June 2017	89500	Explained impact of SIP Entity Listen Port Endpoints column on TransportData; explained that PPM lets a contact have just a first or just a last name.
2.7	July 2018	86737, 87556, 88156, 88681, 90591, 90656, 90890, 91062	Clarify which handles gAEC returns; clarify what Dial Plan and Emergency Numbers are sent to REMO users; correct meaning of + symbol in DialPlanData; listed all fields mandatory for a new Contact; clarify Address & label_2 in addContact; clarify Category & PhoneNumber matching in Contact

Document Issue	Document Date	Compas MR	Change Description
			requests; add list of current DeviceModel strings; clarify DeviceIdentity contents; button labels can go past 20 to 255 characters; added 8.0 features: MCA bridges, CS1k endpoints, Extended Hostname Validation
2.8	September 2018	91202	Added more clarity on what fields are valid for device data requests.
2.9	December 2018	78229, 82008, 91201	Fixed aut-msg-wt button description; described crss-alert and no-hld-conf buttons; added call-appr buttons and CS1k sets to RingerOnOff description; added info on how to configure Public Contact; added section 3.3.3 to explain Device Model mappings.
2.10	October 2019		Extended ASCII character checking and string length checking were added/updated to/in many PPM request parameters in SM 8.1.0. A SIP Endpoint type to Device Model conversion table was added for 8.1.0. Terminal Group ID changes were added for 8.1.1. Device Data parameter change information was added for 8.1.1. Information about Multi-tenancy support was added.

Table Of Contents

1 INTRODUCTION

The Personal Profile Manager (PPM) is a web service that runs as part of the Avaya Aura® Session Manager (SM) and the System Manager (SMGR). In order to enable the Avaya SIP Telephony (AST) features, the endpoint must register to the SM, subscribe, and request configuration information from PPM.

Clients are configured with one or two SM IP-addresses through a config file or some other method. The client attempts to register with the first SM. If that SM is not the authoritative SM, that SM will redirect the phone to appropriate SM(s). Once the client successfully registers to all the SMs in its Proxy list, it will subscribe to several Event Packages including the feature-status and the avaya-ccs-profile event packages. If the feature-status event package subscription is successful (which implies the Avaya SIP phone has entered Advanced SIP Telephony (AST) mode), the phone will send a number of requests to PPM so it can receive its proxy list, button list, dialplan information, contact list, etc.

This document describes the SOAP message set, request parameters, and response parameters required to retrieve the configuration data and user data for a SIP endpoint. PPM processes SOAP messages over HTTP/HTTPS with digest authentication.

The PPM URL is the same for all requests:
scheme://*fqdn_or_ip_address*/axis/services/PPM. The port used by PPM clients is 80 when the *scheme* is http and 443 when it's https. This URL has been used since the inception of PPM and should never change. The parameters sent in each PPM request and those parameters returned in the corresponding PPM response may vary over time as the PPM interface is enhanced, but the intent is to keep the PPM interface backwards compatible, so it can support older Avaya SIP Endpoints as well as new ones.

Starting with 7.1, PPM correctly includes the SOAPAction HTTP header whenever it sends a response to a request. For example, for a getDeviceData request with the HTTP header *SOAPAction: getDeviceData\r\n*, the getDeviceDataResponse includes the same HTTP header (though with quotation marks): *SOAPAction: "getDeviceData"\r\n*

The data that PPM retrieves is stored in the local database on the SM. It is assumed that the reader is familiar with the SMGR, SM and CM configuration. The PPM messages are meant to service many different clients, so PPM clients should ignore any of the data they are not interested in or do not understand.

The reader of this document should assume that:

- only those PPM request parameter fields that are specifically mentioned below as being ASCII strings, require ASCII characters. And by ASCII characters, we mean the extended ASCII character set.
- all fields that aren't tagged as ASCII strings, are UTF-8 strings, and can accept ASCII characters because they're part of the UTF-8/Unicode character set.

2 SM 6.2 SUPPORTED PPM REQUESTS AND CLIENTS FOR SESSION MANAGER

Method	SM	Dev – Connect	Notes
addContact	5.2	SM 5.2	
deleteContact	5.2	SM 5.2	
getAllEndpointConfiguration	5.2	SM 5.2	In 6.1, Fields map support is added to the request. This request should follow the getHomeCapabilities request. In 6.2 a new "Identity" parameter was added.
getInitialEndpointConfiguration	6.2	SM6.2	This request is made by the SIP AST phone prior to displaying its login screen.
getContactList	5.2	SM 5.2	
getDeviceData	5.2	SM 6.0	In 5.2, this contains default data in the response
getHomeCapabilities	5.2	SM 5.2	This request follows the getHomeServer request.
getHomeServer	5.2	SM 5.2	Post phone registration and subscription, this is the first PPM request the SIP AST phone should make. This PPM request is now considered deprecated.
getPermissionType	n/a	n/a	Default data in the response
setDeviceData	5.2	SM 6.0	In 5.2, this contains default data in the response
setOneTouchDialList	6.2	SM6.2	Default data in response until 6.2.
setVolumeSettings	5.2	SM 6.0	A separate getVolumeSettings method is not needed as it is included in the getAllEndpointConfiguration response. In 6.2 a new "Identity" parameter was added.
updateContact	6.0	SM 6.0	
searchContact	6.1	SM 6.1	Supported for backwards compatibility with SIP 2.6 phones. Use searchUser for new applications.
searchContactCount	6.1	SM 6.1	Supported for backwards compatibility with SIP 2.6 phones.
searchUser	6.1	SM 6.1	Replaces searchContact.
getCallHistory	6.3.0.8	SM 6.3	Retrieve call logs from SM's centralized call journal DB.
deleteCallHistory	6.3.0.8	SM 6.3	Delete call logs from SM's centralized call journal DB.

3 PPM OPERATIONS

3.1 CONTACT MESSAGES

Administration of Enterprise and non-Enterprise contacts will be described in the sections below.

PPM will modify all string fields as noted below:

- Spaces will be removed from the beginning and end of each string
- HTML tags will be removed
- The following special characters will be removed
  - **‘** (single quote)
  - **;** (semi-colon)
  - \ (back-slash)
  - / (forward-slash)
  - \\ (double back-slash)
  - > (greater-than)
  - < (less-than)
  - **&** (ampersand)

3.1.1 addContact

The addContact PPM method allows the client to add a user to their contact list.

addContact(Handle, Contact);

Request Parameter	Type	Length / Value	Description
Handle	String	SIP URI Max 256 characters	The handle of the user in the form of “user@domain”. Also see the note in Section 3.8 Handles.
Contact	ContactData		Contact data

The greyed-out group-related fields are not supported.

ContactData	Type	Length / Value	Description
Address	String	Max 256 characters	Primary handle of a contact. If an avext parameter is added to the address, it will be included as part of the string’s length.
Name	String (optional*)	Max 256 characters	Localized Full Name of the contact

			(*) See text below.
FirstName	String (optional*)	Max 256 characters	Localized First Name of the contact (*) See text below.
LastName	String (optional*)	Max 256 characters	Localized Last Name of the contact (*) See text below.
PreferredLanguage	ASCII String (optional)	Max 32 characters	Preferred Language of the contact Note: ASCII checking not enforced.
EndpointDisplayName	ASCII String (optional)	Max 256 characters	ASCII Full Name of the contact Note: ASCII checking not enforced.
FirstNameAscii	ASCII String (optional)	Max 256 characters	ASCII First Name of the contact Note: ASCII checking not enforced.
LastNameAscii	ASCII String (optional)	Max 256 characters	ASCII Last Name of the contact Note: ASCII checking not enforced.
Alias	String (optional)	Max 256 characters	Alias of the contact
Group	String	Max 32 characters	Not available – always set to ‘top’.
ParentGroup	String	Max 32 characters	Not available – always set to nil
Email	String (optional)	Max 256 characters	Email address for contact
IM	String (optional)	Max 256 characters	XMPP address for contact
Notes	String (optional)	Max 1024 characters	Used internally by phone
IsBuddy	Boolean		Used to indicate whether the user

			wants to view the contact's presence
Manager	String (optional)	max 256 characters	Contact's manager from directory
PictureUrl	String (optional)	max 1024 characters	URL to contact's picture
Favourite	Boolean (optional)		indicates to client whether this contact is a favourite
InstantConnection	Boolean (optional)		indicates that client can auto answer calls from this contact
ContactPhones	ArrayOfContactPhoneData	Max 6 entries /private numbers.	Phone and Speed dial information.
EndpointDataList	ArrayOfEndpointData (optional)	Max 3500 bytes as the total combined lengths of all the name / value pairs. Max 20 name/value pairs.	Contact data stored by the endpoints which does not fit into the existing ContactData fields.
ServiceDataList	ArrayOfNameValuePair (optional)	Max 10 entries.	Contains service specific data in the form of name value / pairs associated with the contact
PostalAddressDataList	ArrayOfPostalAddresData (optional)	Max 3 entries.	Contact's postal addresses data e.g work, home

SM supports 2 types of contacts: An enterprise contact, also called internal, and a private contact, also called external. When an add contact request is received, PPM will search the administered handles for one that matches the ContactData.Address. If found, then the contact is an enterprise user. If not found, PPM will attempt to resolve it to a CM extension. If that is successful, then the contact is an enterprise user. If the ContactData.Address is not an administered handle, PPM will search the

ContactPhoneData entries for any that have a ContactPhoneData.Type of work. Using the ContactPhoneData.PhoneNumber, it will attempt to resolve it to an administered handle or a CM extension. If that is successful, then this contact is an enterprise user. Otherwise, the contact is private.

To add a contact through PPM, one or more of the Localized Name fields must be present in the request: the Localized Name field, or the Localized FirstName field, or the Localized LastName field. In addition to name information, the address field must be present, and for each ContactPhoneData entry in the body, the phone number and type fields are mandatory.

For enterprise contacts, the Email field should only be used to store a private email handle for that contact. If the contact's enterprise email handle is passed in that field, PPM will ignore it since it's already administered for the contact.

The IM field is used to assign a private XMPP address to the contact. This private XMPP address will be returned in the IM field in subsequent getContactList responses regardless of whether or not the contact has different XMPP addresses administered in the case of an enterprise contact.

See section 9.2, TYPES OF CONTACTS, for additional information regarding enterprise (internal) and private (external) contacts. Additional details regarding "Name" and "Alias" can be found in section 9.3.2.1, Names.

To obtain presence information on an enterprise contact, the ContactData.IsBuddy field must be set to true. This field is completely controlled by the endpoint and PPM simply stores the value for the contact.

ContactPhoneData	Type	Length / Value	Description
PhoneNumber	String	max 256 characters	Phone number or SIP URI
Label_1	String (optional)	max 255 characters	Used internally by phone
Label_2	String (optional)	max 255 characters	Used internally by phone
Category	String (optional)	max 256 characters	Contains the same value as the Phone Number if this is the Primary Number, otherwise this field will be blank
Type	ASCII String	max 32 characters	Type of contact
SpeedDialEnable	Boolean		Enable this as a speed dial entry or not

For enterprise contacts, the ContactPhoneData should only contain private numbers to be stored with the contact. PPM does look up each ContactPhoneData.PhoneNumber to check if it exists as an administered handle and if it does, PPM won't add it as a private number. The only exception is if the user wants to put the administered handle on speeddial, and/or associate a Label_1/2. Please note that only one ContactPhoneData entry per contact can be assigned to the user's speed dial list at any one time. This rule is enforced by the SMGR CIM (Common Information Model) which defines the schema for persistent data storage of contact information.

A maximum of 6 private numbers can be stored for contacts. Since enterprise contacts can have ContactPhoneData records for administered handles (i.e. speeddial, label_1/2), it's possible to have more than 6 ContactPhoneData records in an addContact request. But if more than 6 private numbers are passed in those records, PPM will return a SOAP fault.

ContactPhoneData.Category contains information on the primary number of a contact. If this ContactPhoneData is the primary number, the Category should be set to the ContactPhoneData.PhoneNumber. If the primary number is the same as the ContactData.Address field, there should be a ContactPhoneData object with the Address as the PhoneNumber and Category as well as the other required data. Otherwise, the Category field should be blank.

However, even though the Category field in PPM contact responses always matches the Category field from the corresponding PPM request, the PhoneNumber field may not match the Category field, as it may be postfixed with @domain. ASM-60361 has details.

EndpointData	Type	Length / Value	Description
Name	String		The name of the data.
Value	String (optional)		The value of the data. This field is required for addContact, but is optional for updateContact, where the entry is deleted if this field is not provided.

This EndpointData information element allows the endpoint to store additional data values for a specific contact. The endpoints are free to store whatever data they want to associate with the contact, with the restriction that the data must be a String.

During an updateContact operation, existing data will be overwritten by an EndpointData, or added if it doesn't exist. If the Value has the SOAP "nil=true" attribute, then any existing data with that name, is left intact. If the Value element is omitted, then any existing data with that name is deleted.

NameValuePair	Type	Length / Value	Description

Name	String	max 128 characters	The name of the data.
Value	String (optional)	max 256 characters	The value of the data.

This information element allows the servicedatalist to store additional data as name / value pairs for a specific contact. The endpoints are free to store whatever data they want to associate with the contact in a servicedatalist, with the restriction that the data must be a String.

PostalAddressData	Type	Length / Value	Description
AddressName	String	max 128 characters	The name of the address.
StreetAddress	String (optional)	max 128 characters	Contact's street address
Company	String (optional)	max 128 characters	Contact's company
Location	String (optional)	max 128 characters	Contact's location
City	String (optional)	max 128 characters	Contact's city
State	String (optional)	max 128 characters	Contact's state
Country	String (optional)	max 128 characters	Contact's country
PostalCode	String (optional)	max 128 characters	Contact's postal/zip code
Department	String (optional)	max 128 characters	Contact's department
WorkRoomNo	String (optional)	max 128 characters	Work space identifier for a contact
ServiceDataList	ArrayOfNameValuePair (optional)	Max 10 entries.	Contains service specific data in form of name value pairs associated with the addressData.

addContactResponse(Result, ListOfHandles, Handles, VideoCapable);

Response Parameter	Type	Description
Result	String	PPM_SUCCESS or fault message
ListOfHandles	HandleListInfo (optional)	Container for a list of associated handles when the contact is an enterprise user.

		The avext parameter will not be added to any handles in this list. This field should be considered deprecated, having been replaced by Handles. However it will remain in the response for backwards compatibility.
Handles	HandleDataList (optional)	List of all of the user's handles when the contact is an enterprise user with their type and subtype. An avext parameter will be added to each one as necessary to include the shortform (CM extension). See section 3.8, Handles, for more information about avext.
VideoCapable	Boolean (optional)	The user has a video-capable phone. This field is only present for enterprise contacts.
ContactId	String (optional)	The unique id of the contact is returned if the enable clustering flag is turned on, the SM is not a BSM.This id can be passed in the updateContact operation.

HandleListInfo	Type	Description
NoOfElements	Integer	Number of elements in the array
HandleList	ArrayOfHandles	Array of strings containing the handles associated with this contact

HandleDataList	Type	Description
NoOfElements	Integer	Number of records in the array
HandleList	ArrayOfHandleData	Array of HandleData record types

HandleData	Type	Description
Handle	String	The handle
HandleType	HandleTypeEnum (optional)	The type of handle
HandleSubtype	HandleSubtypeEnum (optional)	The subtype of handle

HandleTypeEnum	Type	Description
sip	String	SIP
smtp	String	SMTP
xmpp	String	XMPP

ibm	String	IBM

HandleSubtypeEnum	Type	Description
e164	String	Avaya E.164
username	String	Avaya SIP
msrtc	String	Microsoft OCS SIP
googletalk	String	GoogleTalk
jabber	String	Jabber
ibmsametime	String	IBM Sametime
lotusnotes	String	Lotus Notes
msexchange	String	Microsoft Exchange

The addContactResponse has 2 lists of handles: ListOfHandles and Handles. The former was added in SM 5.2 to provide the phone with the list of handles administered for the contact if the contact was for an enterprise user. Session manager supports various types of handles such as sip, xmpp and smtp, and subtypes within those types. The ListOfHandles does not provide the type and subtype information and will contain all handles. It will not contain the avext parameter. So in SM 6.1, the Handles element was added to provide that information for each administered handle including the primary handle. If the shortform/CMExtension is not one of the handles in this list, then the parameter ';avext=CMExtension' is appended to each handle in the list that is in the same communication profile set. Handles in the ContactPhoneData will not have the avext parameter appended. See section 3.8, Handles, for more information about avext.

Example: addContact request of a private contact. Note that the Address field has been generated by the phone. In this case, a phone number of type 'mobile' was provided by the user. The endpoint is adding supplemental information to this contact through the EndpointData element. Here it's adding a ring tone and favorite index to store with this contact. This information is only understood by the endpoints.

```
<ns1:addContact xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
  <Contact>
    <Address>ep__1273760598@avaya.com</Address>
<Name>みさき，えみ</Name>
    <FirstName>えみ</FirstName>
    <LastName>みさき</LastName>
    <PreferredLanguage>ja_JP</PreferredLanguage>
    <EndpointDisplayName>Misaki, Emi</EndpointDisplayName>
    <FirstNameAscii>Emi</FirstNameAscii>
    <LastNameAscii>Misaki</LastNameAscii>
    <Group>top</Group>
    <ParentGroup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
    <Email />
    <IM />
    <Notes />
    <IsBuddy>false</IsBuddy>
```

```
   <ContactPhones xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:ContactPhoneData[1]" xsi:type="soapenc:Array">
     <item>
      <PhoneNumber>9137235304</PhoneNumber>
      <Label_2 xsi:nil="true" />
      <Category>9137235304</Category>
      <Type>mobile</Type>
      <SpeedDialEnable>false</SpeedDialEnable>
     </item>
   </ContactPhones>
   <EndpointDataList  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:EndpointData [2]" xsi:type="soapenc:Array">
     <item>
       <Name>RingTone</Name>
       <Value>4</Value>
     </item>
     <item>
        <Name>Favorite</Name>
        <Value>1</Value>
     </item>
   </EndpointDataList>
  </Contact>
</ns1:addContact>
```

The ContactPhoneData elements contain private information to associate with this contact. Also note that we don't store the handle 70001@avaya.com nor any other administered handle in the ContactPhoneData. However, if the PPM client wants to put a handle on the speeddial list, then it would be added to the private information with SpeedDialEnable set to true. Please note that only one ContactPhoneData entry per contact can be assigned to the user's speed dial list at any one time.

```
<ns1:addContact xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
  <Contact>
   <Address>70001@avaya.com</Address>
   <Name>Doe, John</Name>
   <FirstName>John</FirstName>
   <LastName>Doe</LastName>
   <PreferredLanguage>en_US</PreferredLanguage>
   <Alias>D, Johnny</Alias>
   <Group>top</Group>
   <ParentGroup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
   <Email />
   <IM />
   <Notes />
   <IsBuddy>true</IsBuddy>
    <ContactPhones soapenc:arrayType="ns1:ContactPhoneData[2]"
xsi:type="soapenc:Array">
       <item>
         <PhoneNumber>70001@avaya.com</PhoneNumber>
         <Label_1 />
         <Label_2 />
```

```
        <Category>70001@avaya.com</Category>
        <Type>handle</Type>
        <SpeedDialEnable>true</SpeedDialEnable>
      </item>
      <item>
        <PhoneNumber>303 450 1234</PhoneNumber>
        <Label_1 />
        <Label_2 />
        <Category />
        <Type>home</Type>
        <SpeedDialEnable>false</SpeedDialEnable>
      </item>
      <item>
        <PhoneNumber>303 810 1234</PhoneNumber>
        <Label_1 />
        <Label_2 />
        <Category />
        <Type>mobile</Type>
        <SpeedDialEnable>false</SpeedDialEnable>
      </item>
    </ContactPhones>
    <EndpointDataList  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:EndpointData [2]" xsi:type="soapenc:Array">
     <item>
       <Name>RingTone</Name>
       <Value>4</Value>
     </item>
     <item>
        <Name>Favorite</Name>
        <Value>1</Value>
     </item>
   </EndpointDataList>
  </Contact>
</ns1:addContact></ns:addContact>
```

In 7.0.1, PPM is supporting new elements in the addContact request. Now “ContactData” element can have “Manager”, “PictureUrl”, “Favourite” and “InstantConnection” fields. Also “ContactData” can contain “PostalAddressDataList” and “ServiceDataList”. Apart from “ContactData”, the “ServiceDataList” can be added in the “PostalAddressData”. Please refer section 3.1.1 to find more details about these newly added fields. Please note that these new fields added in 7.0.1 are not returned in PPM’s getContactList SOAP response.

In Multi-Tenant situations, PPM will only allow a user to add another Enterprise user as a contact, if they reside in the same tenant partition as the user who is making the addContact request.

```
<ns1:addContact xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
```

```
  <Contact>
    <Address>70001@avaya.com</Address>
    <Name>Doe, John</Name>
    <FirstName>John</FirstName>
    <LastName>Doe</LastName>
    <PreferredLanguage>en_US</PreferredLanguage>
    <Alias>D, Johnny</Alias>
    <Group>top</Group>
    <ParentGroup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
    <Email />
    <IM />
    <Notes />
    <IsBuddy>true</IsBuddy>
    <Manager>Jones, Smith</Manager>
    <PictureUrl>http://135.27.162.111/ppm/files/john.jpg</PictureUrl>
    <Favourite>true</ Favourite >
    < InstantConnection/>
    <ContactPhones soapenc:arrayType="ns1:ContactPhoneData[2]"
xsi:type="soapenc:Array">
        <item>
          <PhoneNumber>70001@avaya.com</PhoneNumber>
          <Label_1 />
          <Label_2 />
          <Category>70001@avaya.com</Category>
          <Type>handle</Type>
          <SpeedDialEnable>true</SpeedDialEnable>
        </item>
        <item>
          <PhoneNumber>303 450 1234</PhoneNumber>
          <Label_1 />
          <Label_2 />
          <Category />
          <Type>home</Type>
          <SpeedDialEnable>false</SpeedDialEnable>
        </item>
        <item>
          <PhoneNumber>303 810 1234</PhoneNumber>
          <Label_1 />
          <Label_2 />
          <Category />
          <Type>mobile</Type>
          <SpeedDialEnable>false</SpeedDialEnable>
        </item>
    </ContactPhones>
    <EndpointDataList  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:EndpointData [2]" xsi:type="soapenc:Array">
      <item>
        <Name>RingTone</Name>
        <Value>4</Value>
      </item>
      <item>
         <Name>Favorite</Name>
         <Value>1</Value>
      </item>
```

```
   </EndpointDataList>
    <PostalAddressDataList xsi:type="ns:ArrayOfPostalAddressData"
soapenc:arrayType="ns:PostalAddressData[]">
          <item>
            <AddressName>Work_address</AddressName>
            <Company>Avaya</Company>
            <StreetAddress>4655 Great America Pkwy</StreetAddress>
            <Location/>
            <City>Santa Clara</City>
            <State>California</State>
            <Country>USA</Country>
            <PostalCode>95054</PostalCode>
            <Department>ECS</Department>
            <WorkRoomNo xsi:nil="true"/>
            <ServiceDataList xsi:type="ns:ArrayOfNameValuePair"
soapenc:arrayType="ns:NameValuePair[]">
              <item>
                <Name>USContact</Name>
                <Value>1866 GOAVAYA</Value>
              </item>
              <item>
                <Name>OtherLocations</Name>
                <Value>+1-908-953-6000</Value>
              </item>
            </ServiceDataList>
          </item>
        </PostalAddressDataList>
       <ServiceDataList xsi:type="ns:ArrayOfNameValuePair"
soapenc:arrayType="ns:NameValuePair[]">
          <item>
            <Name>RequestDetails</Name>
            <Value/>
          </item>
        </ServiceDataList>
  </Contact>
</ns1:addContact></ns:addContact>
```

PPM Response:

```
<ns1:addContactResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <PPMResponse>PPM_Success</PPMResponse>
  <ListOfHandles>
    <NoOfElements>1</NoOfElements>
    <HandleList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="xsd:string[1]"
xsi:type="soapenc:Array">
      <item>70001@avaya.com</item>
      <item>johndoe@avaya.com</item>
      <item>johndoe@pres.avaya.com</item>
```

```
   </HandleList >
  </ListOfHandles>
  <Handles>
      <NoOfElements>3</NoOfElements>
      <HandleList soapenc:arrayType="xsd:HandleData[3]"/>
        <item>
          <Handle>70001@avaya.com</Handle>
          <HandleType>sip</HandleType>
          <HandleSubtype>username</HandleSubtype>
        </item>
        <item>
          <Handle>johndoe@avaya.com</Handle>
          <HandleType>smtp</HandleType>
          <HandleSubtype>msexchange</HandleSubtype>
        </item>
        <item>
          <Handle>johndoe@pres.avaya.com</Handle>
          <HandleType>xmpp</HandleType>
          <HandleSubtype>jabber</HandleSubtype>
        </item>
      </HandleList>
   </Handles>
   <VideoCapable>false</VideoCapable>
 <ContactId>e9f0bbd0-c8fc-11e5-a303-5198bf5c0e11</ContactId>
</ns1:addContactResponse>
```

Error Conditions/Faults:
The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
“DatabaseError”	PPM cannot access the database for reading or writing.
“Duplicate Contact”	This contact already exists.
“NotIMUser”	Requesting user does not exist in the Enterprise database
“ContactAddressNull”	Address field is empty. This is the handle that is supposed to be generated by the phone if not provided by the user.
“InvalidContact”	Data provided failed a validation step.
“RequiredDataMissing”	Required fields are missing.
“InvalidValue”	Field provided is not ASCII.
“DatabaseNotAccessible”	There was a problem with the database
“InternalError”	Unexpected exception in PPM processing
"ContactDoesNotExist"	Name field is empty or not present.

3.1.2 deleteContact

The deleteContact request allows the client to delete a user from their contact list. On successful completion of the request, this method will return an acknowledgment stating that the contact has been deleted from the contact list.

deleteContact(Handle, Address)

Request Parameter	Type	Length / Value	Description
Handle	String	SIP URI Max 256 characters	The handle of the user in the form of "user@domain". Also see the note in Section 3.8 Handles.
Address	String	SIP URI max 256 characters	The contact's handle. If an avext parameter is added to the address, it will be included as part of the string's length.

deleteContactResponse(Result)

Response Parameter	Type	Description
Result	String	PPM_SUCCESS or fault message

Example deleteContact Request:

```
<ns1:deleteContact xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
  <Address>ep__1273760598@avaya.com</Address>
</ns1:deleteContact>
```

PPM Response:

```
<ns1:deleteContactResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <PPMResponse>PPM_Success</PPMResponse>
</ns1:deleteContactResponse>
```

Error Conditions/Faults:

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
"DatabaseError"	PPM cannot access the Database for reading or writing.
"NotIMUser"	Requesting user does not exist in the Enterprise database
"InvalidContact"	Data provided failed a validation step.
"DatabaseNotAccessible"	There was a problem with the database
"InternalError"	Unexpected exception in PPM processing

3.1.3 getContactList

This request retrieves the contacts associated with the requesting user.

See section 9.2, TYPES OF CONTACTS, for additional information regarding enterprise (internal) and private (external) contacts.

getContactList(Handle)

Request Parameter	Type	Value	Description
Handle	String	SIP URI Max 256 characters	The handle of the user in the form of "user@domain". Also see the note in Section 3.8 Handles.

getContactList Response(ContactList)

Response Parameter	Type	Description
ContactList	ContactListInfo	Container for contact information

ContactListInfo	Type	Description
NoOfElements	Integer	Number of elements in the array
ContactInfo	ArrayOfContactData	Array of contacts. There is a soft limit of 250 contacts. See section 9.1, Contact limits, for more details.

ArrayOfContactData	Type	Description
ContactDataList	ContactData	Array of individual contact information

ContactData	Type	Description
Address	String	Contact's URI – if the user has a CM extension that differs from all of the user's handle, its appended to the handle using the avext parameter. E.g. 3035551234@avaya.com;avext=5551234
Name	String (optional)	Localized Full Name of the contact.
FirstName	String (optional)	Localized First Name of the contact.
LastName	String (optional)	Localized Last Name of the contact.
PreferredLanguage	String (optional)	Preferred Language of the contact.

EndpointDisplayName	String (optional)	ASCII Full Name of the contact. Added in 6.2 FP1. May be used by the endpoint as the display name if UTF8 isn't supported by the endpoint.
FirstNameAscii	String (optional)	ASCII First Name of the contact.
LastNameAscii	String (optional)	ASCII Last Name of the contact.
Alias	String(optional)	Alias of the contact
Group	String	Not available – always set to 'top'.
ParentGroup	String	Not available – always set to nil
Email	String (optional)	Email contact
IM	String (optional)	XMPP address
Notes	String (optional)	
IsBuddy	Boolean (optional)	Used to indicate whether the user wants to view the contact's presence
ContactId	String (optional)	The unique id of the contact is returned if the enable clustering flag is turned on, the SM is not a BSM, and the value is available.
ContactPhones	ArrayOfContactPhoneData	Phone and Speed dial information – maximum of 6 items allowed for private contacts. Enterprise contacts can have up to 6 private items and will also have all administered SIP handles.
Handles	HandleDataList (optional)	List of all administered handles when the contact is an enterprise user. The avext parameter with the short-form CM extension is appended to any handle which requires it.
VideoCapable	Boolean (optional)	The user has a video-capable phone. This field is only present for enterprise contacts.
EndpointDataList	ArrayOfEndpointData (optional)	Contact data stored by the endpoints which does not fit into the existing ContactData fields.

The ContactData.Address field contains the primary handle for the contact. The primary handle can be any of a user's SIP handles from the primary communication profile set, prioritized in the following order based on the HandleDataList.Subtype:

1. username
2. e164
3. msrtc
4. otherSIP

If there are no SIP handles in the primary communication profile set, then the same algorithm is applied to the other communication profile sets.

If the user doesn't have a SIP handle, PPM generates a pseudo-handle of the form ep__ppm__(ui or ci)__(userid|contactid). If a CM extension exists and the primary handle is not the shortform/CMextension, then it will appended to the primary handle using the ';avext=CMextension' parameter.

If an enterprise contact has both a private and an enterprise email handle, the ContactData Email field will contain the private email handle and the enterprise email handle will be returned in the Handles element. If there isn't a private email handle for the contact, the Email field will contain the enterprise email handle. If there are multiple email handles administered for the user, priority is given to handles with subtype of "msexchange". If there are multiple email addresses of any of the above types, the email address with the lowest contact id in the database will be used so a consistent email address ordering is always maintained. Note that the enterprise email handles are always returned under the Handles element, regardless of whether they're returned in the ContactData Email element or not.

The IM field will be populated by an XMPP address that is administered for the contact. If the contact has multiple XMPP addresses, then the address will be chosen based on the priority listed below. In the event that multiple XMPP addresses of a single type exist for the contact, then the last address that is processed by PPM in the highest priority category will be chosen for the IM field.

1. Private XMPP address
2. Jabber
3. Googletalk
4. Other

HandleDataList as well as all other changes made to ContactData information are described in the addContact operation section (section 3.1.1).

Additional details regarding "Name", "Alias", "EndpointDisplayName" can be found in section 9.3.2.1, Names.

ContactPhoneData	Type	Description
PhoneNumber	String	Phone number or SIP URI
Label_1	String (optional)	Label
Label_2	String (optional)	Label
Category	String	Primary number information – see the description in 3.1.1
Type	String	Type of contact
SpeedDialEnable	Boolean	Enable this as a speed dial entry or not.

Private contact information means that the phone user has supplied supplemental information for an enterprise contact such as a home or cell phone number, a private email address, a name alias, or a set of notes. For enterprise contacts, PPM will populate the ContactPhoneData elements with private contact information and SIP handles. The order of the ContactPhoneData records will be random, except that the records containing private contact information will precede those containing administered SIP handles.

Example getContactList request:

```
<ns1:getContactList xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
</ns1:getContactList>
```

PPM response:

```
<ns1:getContactListResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ContactList>
    <NoOfElements>2</NoOfElements>
    <ContactInfo xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:ContactData[3]" xsi:type="soapenc:Array">
      <item>
        <Address>70005@avaya.com</Address>
        <Name>x70005, ppmscrush2</Name>
        <FirstName>ppmscrush2</FirstName>
        <LastName>x70005</LastName>
        <PreferredLanguage>en_US</PreferredLanguage>
        <Alias />
        <Group />
        <ParentGroup />
        <Email>ksmithjohnson@avaya.com</Email>
        <IM />
        <Notes />
        <IsBuddy>true</IsBuddy>
         <ContactId>e9f0bbd0-c8fc-11e5-a303-5198bf5c0e11</ContactId>
        <ContactPhones soapenc:arrayType="ns1:ContactPhoneData[2]"
xsi:type="soapenc:Array">
          <item>
            <PhoneNumber>303 665 1234</PhoneNumber>
            <Label_1/>
            <Label_2 />
            <Category>3036651234</Category>
            <Type>home</Type>
            <SpeedDialEnable>false</SpeedDialEnable>
          </item>
          <item>
            <PhoneNumber>303 538 1234</PhoneNumber>
            <Label_1 />
            <Label_2 />
            <Category />
            <Type>work</Type>
            <SpeedDialEnable>false</SpeedDialEnable>
```

```
        </item>
      </ContactPhones>
      <Handles>
        <NoOfElements>3</NoOfElements>
        <HandleList soapenc:arrayType="xsd:HandleData[3]"/>
          <item>
            <Handle>70005@avaya.com</Handle>
            <HandleType>sip</HandleType>
            <HandleSubtype>username</HandleSubtype>
          </item>
          <item>
            <Handle>ksmithjohnson@avaya.com</Handle>
            <HandleType>smtp</HandleType>
            <HandleSubtype>msexchange</HandleSubtype>
          </item>
          <item>
            <Handle>ksmithjohnson@pres.avaya.com</Handle>
            <HandleType>xmpp</HandleType>
            <HandleSubtype>jabber</HandleSubtype>
           </item>
         </HandleList>
       </Handles>
       <VideoCapable>true</VideoCapable>
       <EndpointDataList  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:EndpointData[2]" xsi:type="soapenc:Array">
         <item>
           <Name>RingTone</Name>
           <Value>4</Value>
         </item>
         <item>
            <Name>Favorite</Name>
            <Value>1</Value>
         </item>
       </EndpointDataList>
    </item>
    <item>
      <Address>extSip@extsip.com</Address>
      <Name>test, external</Name>
      <FirstName>external</FirstName>
      <LastName>test</LastName>
      <PreferredLanguage>en_US</PreferredLanguage>
      <Alias />
      <Group />
      <ParentGroup />
      <Email>extSmtp@extsmtp.com</Email>
     <IM>extXMPP@pres.extxmpp.com</IM>
      <Notes />
      <IsBuddy>false</IsBuddy>
      <ContactId>9f2186c0-c8f7-11e5-a303-5198bf5c0e11</ContactId>
      <ContactPhones soapenc:arrayType="ns1:ContactPhoneData[4]"
xsi:type="soapenc:Array">
        <item>
          <PhoneNumber>O=CompanyName,C=US</PhoneNumber>
          <Label_1 />
          <Label_2 />
```

```
          <Category />
          <Type>work</Type>
          <SpeedDialEnable>false</SpeedDialEnable>
        </item>
        <item>
          <PhoneNumber>sip:extMsrtc@extmst.com</PhoneNumber>
          <Label_1 />
          <Label_2 />
          <Category />
          <Type>work</Type>
          <SpeedDialEnable>false</SpeedDialEnable>
        </item>
        <item>
          <PhoneNumber>303-555-1234</PhoneNumber>
          <Label_1 />
          <Label_2 />
          <Category>3035551234</Category>
         <Type>work</Type>
          <SpeedDialEnable>false</SpeedDialEnable>
        </item>
      </ContactPhones>
      <EndpointDataList  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:EndpointData [1]" xsi:type="soapenc:Array">
        <item>
          <Name>Favorite</Name>
          <Value>2</Value>
        </item>
      </EndpointDataList>
    </item>
  </ContactInfo>
 </ContactList>
</ns1:getContactListResponse>>
```

In 7.0.1, PPM is supporting new elements “Manager”, “PictureUrl”, “Favourite”, “InstantConnection”, “PostalAddressDataList” and “ServiceDataList” in the addContact and updateContact request. However, these fields are not returned in the getContactList SOAP response. These fields are returned in a Java API.

Error Conditions/Faults:
The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
“DatabaseNotAccessible”	There was a problem with the database
“NotIMUser”	Requesting user does not exist in the Enterprise database
“ContactNoProfile”	An internal contact does not have a communication profile
“InvalidContact”	The contact is incorrectly configured for PPM

“ContactAddressNull”	The list of addresses for the contact could not be retrieved
“InternalError”	Unexpected exception in PPM processing

For more information on this response, see section 9, General Information Regarding Contacts at the end of this document.

3.1.4 updateContact

The update contact method is responsible for managing any data that is associated with an existing contact. It is similar to the add contact method, but it updates a current contact rather than adding a new one.

updateContact(Handle, OldAddress, OldGroupName, Contact)

The greyed-out group-related fields in updateContact are not supported.

Request Parameter	Type	Length / Value	Description
Handle	String	SIP URI max 256 characters	The handle of the user in the form of “user@domain. Also see the note in Section 3.8 Handles.
OldAddress	String	max 256 characters	Contact’s current URI. Handle of contact that is being changed, but that Handle must already exist in the system. ”. If an avext parameter is added to the address, it will be included as part of the string’s length.
OldGroupName	String	max 32 characters	Not available - always set to ‘top’
Contact	ContactData		Contact data

ContactData	Type	Length / Value	Description
Address	String	max 256 characters	Primary handle of a contact. If an avext parameter is added to the address, it will be included as part of the string’s length.
Name	String (optional)	max 256 characters	Localized Full Name of the contact
FirstName	String (optional)	max 256 characters	Localized First Name of the contact

LastName	String (optional)	max 256 characters	Localized Last Name of the contact
PreferredLanguage	ASCII String (optional)	max 32 characters	Preferred Language of the contact
EndpointDisplayName	ASCII String (optional)	max 256 characters	ASCII Full Name of the contact
FirstNameAscii	ASCII String (optional)	max 256 characters	ASCII First Name of the contact
LastNameAscii	ASCII String (optional)	max 256 characters	ASCII Last Name of the contact
Alias	String (optional)	max 256 characters	Alias of the contact
Group	String	max 32 characters	Not available – always set to 'top'.
ParentGroup	String	max 32 characters	Not available – always set to nil
Email	String (optional)	max 256 characters	Email address of contact
IM	String (optional)	max 256 characters	XMPP address of contact
Notes	String (optional)	max 1024 characters	
IsBuddy	Boolean		Used to indicate whether the user wants to view the contact's presence
Manager	String (optional)	max 256 characters	Contact's manager from directory
PictureUrl	String (optional)	max 1024 characters	URL to contact's picture
Favourite	Boolean (optional)		Indicates to client whether this contact is a favourite
InstantConnection	Boolean (optional)		Indicates that client can auto answer calls from this contact
ContactId	String (optional)		The unique id of the contact which is returned in the addContactResponse or getContactListResponse. If the id is not passed in the request, then PPM will find the id based on other details from the request.

ContactPhones	ArrayOfContactPhoneData	Max of 6 entries.	Phone and Speed dial information.
EndpointDataList	ArrayOfEndpointData (optional)	The total combined lengths of all the strings for names and values cannot exceed 3,500 bytes. Max of 20 entries.	Contact data stored by the endpoints which does not fit into the existing ContactData fields.
ServiceDataList	ArrayOfNameValuePair (optional)	Max of 10 entries.	Contains service specific data in form of name/value pairs associated with the contact
PostalAddressDataList	ArrayOfPostalAddresData (optional)	Max of 3 entries.	Contact's postal addresses data e.g work, home

For usage of the ContactData type, please see the description under the addContact operation section (section 3.1.1) as it applies to updateContact, too.

updateContactResponse(Result, ListOfHandles, Handles, VideoCapable);

Response Parameter	Type	Description
Result	String	PPM_SUCCESS or fault message
ListOfHandles	HandleListInfo (optional)	Container for a list of associated handles. This field is only present for enterprise contacts and will not contain the avext parameter.
Handles	HandleDataList (optional)	List of all administered handles for that contact. This field is only present for enterprise contacts and may contain the avext parameter.

VideoCapable	Boolean (optional)	The user has a video-capable phone. This field is only present for enterprise contacts.
ContactId	String (optional)	The unique id of the contact is returned if the enable clustering flag is turned on, the SM is not a BSM, and the value is available.

The ListOfHandles and Handles are described in the addContact operation section (section 3.1.1).

Example updateContact Request:

```
<ns1:updateContact xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
  <OldAddress>70005@avaya.com</OldAddress>
  <OldGroupName>top</OldGroupName>
  <Contact>
    <Address>70005@avaya.com</Address>
    <Name>Doe, John</Name>
    <FirstName>Jonathan</FirstName>
    <Group>top</Group>
    <ParentGroup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
    <Email />
   <IM />
    <Notes />
    <IsBuddy>true</IsBuddy>
    <ContactPhones soapenc:arrayType="ns1:ContactPhoneData[1]"
xsi:type="soapenc:Array">
       <item>
         <PhoneNumber>70005@avaya.com</PhoneNumber>
         <Label_1>1</Label_1>
         <Label_2/>
         <Category>70005@avaya.com</Category>
         <Type>home</Type>
         <SpeedDialEnable>true</SpeedDialEnable>
       </item>
    </ContactPhones>
    <EndpointDataList  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:EndpointData [2]" xsi:type="soapenc:Array">
      <item>
        <Name>RingTone</Name>
      </item>
      <item>
         <Name>Favorite</Name>
         <Value>2</Value>
      </item>
      <item>
         <Name>TwitterAccount</Name>
         <Value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true">
      </item>
   </EndpointDataList>
```

```
  </Contact>
  </ns1:updateContact>
```

In the request, the phone is adding the contact's handle to the speeddial list with a Label_1 set to 1. Passing administered handles in the ContactPhoneData is unnecessary except to change the label or the speeddial for the handle. Please note that only one ContactPhoneData entry per contact can be assigned to the user's speed dial list at any one time.

The RingTone for this contact is deleted because there is no associated Value. The data value for Favorite is added or changed to 2, depending on whether it already exists or not. The TwitterAccount data is left intact if it exists. The same behavior can be achieved by not specifying an EndpointData item for TwitterAccount.

In 7.0.1, PPM is supporting new elements in the updateContact request. Now the "ContactData" element can have "Manager", "PictureUrl", "Favourite" and "InstantConnection" fields. Also "ContactData" can contain "PostalAddressDataList" and "ServiceDataList". Apart from "ContactData", the "ServiceDataList" can be added in the "PostalAddressData". Please refer section 3.1.1 to find more details about these newly added fields.

```
<ns1:updateContact xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
  <OldAddress>70005@avaya.com</OldAddress>
  <OldGroupName>top</OldGroupName>
  <Contact>
    <Address>70005@avaya.com</Address>
    <Name>Doe, John</Name>
    <FirstName>Jonathan</FirstName>
    <Group>top</Group>    <ParentGroup xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:nil="true" />
    <Email />
   <IM />
    <Notes />
    <IsBuddy>true</IsBuddy>
    <ContactId>e9f0bbd0-c8fc-11e5-a303-5198bf5c0e11</ContactId>
    <Manager>Jones, Smith</Manager>
    <PictureUrl>http://135.27.162.111/ppm/files/john.jpg</PictureUrl>
    <Favourite>true</ Favourite >
    < InstantConnection/>
    <ContactPhones soapenc:arrayType="ns1:ContactPhoneData[1]"
xsi:type="soapenc:Array">
        <item>
          <PhoneNumber>70005@avaya.com</PhoneNumber>
          <Label_1>1</Label_1>
          <Label_2/>
          <Category>70005@avaya.com</Category>
          <Type>home</Type>
          <SpeedDialEnable>true</SpeedDialEnable>
        </item>
     </ContactPhones>
     <EndpointDataList  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:EndpointData [2]" xsi:type="soapenc:Array">
      <item>
        <Name>RingTone</Name>
      </item>
      <item>
         <Name>Favorite</Name>
         <Value>2</Value>
      </item>
      <item>
         <Name>TwitterAccount</Name>
         <Value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true">
      </item>
   </EndpointDataList>
     <PostalAddressDataList xsi:type="ns:ArrayOfPostalAddressData"
soapenc:arrayType="ns:PostalAddressData[]">
            <item>
              <AddressName>Work_address</AddressName>
              <Company>Avaya</Company>
              <StreetAddress>4655 Great America Pkwy</StreetAddress>
              <Location/>
              <City>Santa Clara</City>
              <State>California</State>
              <Country>USA</Country>
              <PostalCode>95054</PostalCode>
              <Department>ECS</Department>
              <WorkRoomNo xsi:nil="true"/>
              <ServiceDataList xsi:type="ns:ArrayOfNameValuePair"
soapenc:arrayType="ns:NameValuePair[]">
                <item>
                  <Name>USContact</Name>
                  <Value>1866 GOAVAYA</Value>
                </item>
                <item>
                  <Name>OtherLocations</Name>
                  <Value>+1-908-953-6000</Value>
                </item>
              </ServiceDataList>
            </item>
          </PostalAddressDataList>
         <ServiceDataList xsi:type="ns:ArrayOfNameValuePair"
soapenc:arrayType="ns:NameValuePair[]">
            <item>
              <Name>RequestDetails</Name>
              <Value/>
            </item>
          </ServiceDataList>
  </Contact>
  </ns1:updateContact>
```

PPM Response:

```
<ns1:updateContactResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
  <PPMResponse>PPM_Success</PPMResponse>
  <ListOfHandles>
    <NoOfElements>1</NoOfElements>
    <HandleList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="xsd:string[1]"
xsi:type="soapenc:Array">
      <item>70005@avaya.com</item>
      <item>johndoe@avaya.com</item>
      <item>johndoe@pres.avaya.com</item>
    </HandleList>
  </ListOfHandles>
  <Handles>
       <NoOfElements>3</NoOfElements>
       <HandleList soapenc:arrayType="xsd:HandleData[3]"/>
         <item>
           <Handle>70005@avaya.com</Handle>
           <HandleType>sip</HandleType>
           <HandleSubtype>username</HandleSubtype>
         </item>
         <item>
           <Handle>johndoe@avaya.com</Handle>
           <HandleType>smtp</HandleType>
           <HandleSubtype>msexchange</HandleSubtype>
         </item>
         <item>
           <Handle>johndoe@pres.avaya.com</Handle>
           <HandleType>xmpp</HandleType>
           <HandleSubtype>jabber</HandleSubtype>
         </item>
       </HandleList>
   </Handles>
   <VideoCapable>false</VideoCapable>
<ContactId>e9f0bbd0-c8fc-11e5-a303-5198bf5c0e11</ContactId>
</ns1:updateContactResponse>
```

Error Conditions/Faults:

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
"Invalid Contact"	If the contact to update is invalid, i.e. the Address field contained a contact that wasn't in the list
"NotIMUser"	Requesting user does not exist in the Enterprise database
"ContactDoesNotExist"	If the contact that needs to be updated can't be found
"ContactAddressNull"	Address field is empty
"RequiredDataMissing"	Required fields are missing
"DatabaseNotAccessible"	There was a problem with the database
"InternalError"	Unexpected exception in PPM processing

3.1.5 searchContact

The SearchContact operation has been replaced by the SearchUser operation which provides more searching capabilities than the older SearchContact. SearchContact is still supported in order to provide backwards compatibility with the older SIP phones. New applications should use SearchUser instead.

This operation allows a phone to search the Session Manager database for administered users. This search is based on search criteria sent in the request. The response will contain the user information for each user that matches the search criteria. This information includes the first and last name, handles and address info. SearchContact provides pagination capability where a start index and limit can be specified for retrieving the list of matched users. The data returned should allow the user doing the search to add any users from the search results to their list of contacts.

searchContact(SearchQuery, start, limit)

Request Parameter	Type	Length / Value	Description
SearchQuery	SearchCriteria		The criteria to use for the search.
Start	Integer	0 to 1,000,000	Return the user beginning at this start index.
Limit	Integer	1 to 250	Maximum number of users to return.

The greyed-out address-related fields in SearchCriteria are not supported.

SearchCriteria	Type	Length / Value	Description
FirstName	String (optional)	Max 512 characters	A pattern matching the user's first name in Localized characters.
LastName	String (optional)	Max 512 characters	A pattern matching the user's last name in Localized characters.
FirstNameAscii	ASCII String (optional)	Max 512 characters	A pattern matching the user's first name in ASCII characters. Note: ASCII checking not enforced.
LastNameAscii	ASCII String (optional)	Max 512 characters	A pattern matching the user's last name in ASCII characters. Note: ASCII checking not enforced.
Handle	String (optional)	Max 512 characters	A pattern matching a user's handle. If `@domain` is not specified, the domain of the user running this request is used. `handle@*` will match *handle* against any domain.
Address1	String (optional)	Max 512 characters	A pattern matching the administered street address of the user.

Address2	String (optional).	Max 512 characters	A pattern matching the administered postal address of the user.
City	String (optional)	Max 512 characters	A pattern matching the administered locality name of the user.
State	String (optional)	Max 512 characters	A pattern matching the administered state or province of the user.
Zip	String (optional)	Max 512 characters	A pattern matching the administered postal code of the user.
OfficeLocation	String (optional)	Max 512 characters	A pattern matching the administered room of the user.
Country	String (optional)	Max 512 characters	A pattern matching the administered country of the user.

Any number of the SearchCriteria fields can be specified and PPM will "and" these fields together. So if LastName is set to "Smith" and Handle is set to "303538*", PPM will search for all users with last name "Smith" and who have a handle beginning with "303538". The search is done with alphabetic case ignored.

The pattern is a sequence of characters which is anchored at the beginning and end of the pattern. It may contain the "*" operator which expands out any sequence of characters, including no characters. The expression "John*" matches both "John" and "Johnson". If the asterisk is to be taken literally, it must be escaped with another "*". So "John**" will only match the name "John*".

Regarding the anchoring of the pattern, "53*8" will match "538 5678" but will not match "303 538 5678". Use the "*" to not anchor the beginning or end of a pattern.

searchContactResponse(SearchContactsResult);

Response Parameter	Type	Value	Description
SearchContactsResult	SearchResultListInfo	Complex	Results of the user search.

SearchResultListInfo	Type	Value	Description
NoOfElements	Integer	0 – "limit"	Number of records in the array
SearchResultsInfo	ArrayOfSearchResult	Maximum of "limit" elements.	List of SearchResult data types. The sort order for the records will be by last name. For equal last names, the first name is used. If both first and last names are the same, the internal database id of the records is used.

The greyed-out address-related fields in the SearchResult type are not supported.

SearchResult	Type	Description
SearchIndex	Integer	Index in the search result
PrimaryHandle	String	The "primary" handle of the user. Note: this will include the avext parameter if necessary. See section 3.8 for which handle constitutes the primary handle.
FirstName	String	First name in Localized characters.
LastName	String	Last name in Localized characters.
FirstNameAscii	ASCII String	First name in ASCII characters
LastNameAscii	ASCII String	Last name in ASCII characters
Address1	String	Address
Address2	String	Postal Address. Note: This isn't currently settable in the System Manager UI.
City	String	Locality name
State	String	State or province
Zip	String	Postal code
OfficeLocation	String	Room number
Country	String	Country Name
AlternateAddresses	AlternateAddressListInfo	List of all user's handles including their PrimaryHandle.

AlternateAddressListInfo	Type	Description
noOfElements	Integer	Number of addresses in alternateAddressList.
alternateAddressList	Array	Array of String elements.

See searchUser below, in particular SearchUserResponse, for important information regarding public contacts in the searchContactResponse. See section 9.2.1, Public Contacts for general information regarding public contacts.

Example searchContact Request:

```
<ns1:searchContact xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <SearchQuery>
    <LastName>*john*</LastName>
  </SearchQuery>
  <Start>0</Start>
  <Limit>50</Limit>
</ns1:searchContact>
```

PPM Response:

```
<ns1:searchContactResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SearchContactsResult>
   <NoOfElements>2</NoOfElements>
   <SearchResultsInfo soapenc:arrayType="ns1:SearchResult[2]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    <item>
     <SearchIndex>0</SearchIndex>
     <PrimaryHandle>70005@avaya.com</PrimaryHandle>
     <FirstName>Steve</FirstName>
     <LastName>Johnson</LastName>
     <FirstNameAscii>Steve</FirstNameAscii>
     <LastNameAscii>Johnson</LastNameAscii>
     <AlternateAddresses>
      <NoOfElements>3</NoOfElements>
      <alternateAddressList soapenc:arrayType="xsd:HandleData[3]"/>
        <item>70005@avaya.com</item>
        <item>stevejohnson@avaya.com</item>
       <item>stevejohnson@pres.avaya.com</item>
     </alternateAddressList>
     </AlternateAddresses>
   </item>
   <item>
     <SearchIndex>1</SearchIndex>
     <PrimaryHandle>70002@avaya.com</PrimaryHandle>
     <FirstName>Kelly</FirstName>
     <LastName>Smith-Johnson</LastName>
     <FirstNameAscii>Kelly</FirstNameAscii>
     <LastNameAscii>Smith-Johnson</LastNameAscii>
     <AlternateAddresses>
      <NoOfElements>3</NoOfElements>
      <alternateAddressList soapenc:arrayType="xsd:HandleData[3]"/>
        <item>70002@avaya.com</item>
        <item>ksmithjohnson@avaya.com</item>
        <item>ksmithjohnson@pres.avaya.com</item>
       </alternateAddressList>
     </AlternateAddresses>
    </item>
   </SearchResultsInfo>
  </SearchContactsResult>
</ns1:searchContactResponse>
```

Error Conditions/Faults:

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
DatabaseError	PPM cannot access the database for reading.

3.1.6 searchContactCount

This operation is provided for backwards compatibility with phones which may use this operation. This operation is identical to searchContact except it only returns the number of users found who match the given search criteria, instead of the user records themselves. This operation is useful if the endpoint wants to make sure that it can properly display the returned data as a large number of users in the results may force the endpoint to paginate the results. If the number of users returned is excessive, the phone may want to reject the search and display an error message to the user requesting a more selective search criterion.

searchContactCount(SearchQuery)

Request Parameter	Type	Length / Value	Description
SearchQuery	SearchCriteria		The criteria to use for the search. See Section 3.1.5 SearchContact.

Response Parameter	Type	Description
searchContactCountResult	Integer	The number of matching entries

Example SearchContactCount Request:

```
<ns1:SearchContactCountRequest
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <SearchQuery>
    <LastNameAscii>*john*</LastNameAscii>
  </SearchQuery>
</ns1:SearchContactCountRequest >
```

PPM Response:

```
<ns1:SearchContactCountResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <searchContactCountResult>2</searchContactCountResult>
</ns1:SearchContactCountResponse>
```

Error Conditions/Faults:

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
DatabaseError	PPM cannot access the database for reading.

3.1.7 searchUser

The SearchUser request allows a phone to search the Session Manager database for administered users. This search is based on search criteria sent in the request. The response will contain the user information for each user that matches the search criteria. This information includes the first and last name, handles and address info. SearchUser provides pagination capability where a start index and limit can be specified for retrieving the list of matched users. The data returned should allow the user doing the search to add any users from the search results to their contact list, or communicate with the user via IM. It should also allow the user to press a button and dial a user returned in the search results.

searchUser(Handle, SearchQuery, Start, Limit)

Request Parameter	Type	Length / Value	Description
Handle	String	SIP URI max 256 characters	The handle of the user in the form of "user@domain". Also see the note in Section 3.8 Handles.
SearchQuery	SearchCriteria		The criteria to use for the search.
Start	Integer	0 to 1,000,000	Return the user beginning at this start index.
Limit	Integer	1 to 250	Maximum number of users to return.

The greyed-out address-related fields in SearchCriteria are not supported.

SearchCriteria	Type	Length / Value	Description
FirstName	String (optional)	max 512 characters	A regular expression matching the user's first name in Localized characters.
LastName	String (optional)	max 512 characters	A regular expression matching the user's last name in Localized characters.
FirstNameAscii	ASCII String (optional)	max 512 characters	A regular expression matching the user's first name in ASCII characters. Note: ASCII checking not enforced.
LastNameAscii	ASCII String (optional)	max 512 characters	A regular expression matching the user's last name in ASCII characters. Note: ASCII checking not enforced.

LoginName	String (optional)	max 512 characters	A regular expression matching the user's login name. LoginName cannot be used to search for public contacts (see section 9.2.1-Public Contacts).
Handle	String (optional)	max 512 characters	A regular expression matching a user's handle. If `@domain` is not specified, then any domain is used for the search. To narrow the search against a specific domain, append it to the pattern. `handle@avaya.com` will match *handle* for only users under the "avaya" domain.
HandleType	ASCII String (optional)	max 512 characters	A regular expression matching the type of the handle that the user may have. Note: ASCII checking not enforced.
HandleSubtype	ASCII String (optional)	max 512 characters	A regular expression matching the subtype of the handle that the user may have. HandleSubtype cannot be used to search for public contacts (see section 9.2.1-Public Contacts). Note: ASCII checking not enforced.
VideoCapable	Boolean (optional)		The user has a video-capable phone. VideoCapable cannot be used to search for public contacts (see section 9.2.1-Public Contacts).
Address1	String (optional)	max 512 characters	A regular expression matching the administered street address of the user.
Address2	String (optional).	max 512 characters	A regular expression matching the administered postal address of the user.
City	String (optional)	max 512 characters	A regular expression matching the administered locality name of the user.
State	String (optional)	max 512 characters	A regular expression matching the administered state or province of the user.
Zip	String (optional)	max 512 characters	A regular expression matching the administered postal code of the user.
Building	String (optional)	max 512 characters	A regular expression matching the administered building of the user.
OfficeLocation	String (optional)	max 512 characters	A regular expression matching the administered room of the user.
Country	String (optional)	max 512 characters	A regular expression matching the administered country of the user.

Any number of the SearchCriteria fields can be specified and PPM will "and" these fields together. So, if LastName is set to "Smith" and VideoCapable is set to "true", PPM will search for all users whose last name contains "Smith" who have video-capable phones. The search is done with alphabetic case ignored.

Per 100326-175, true regular expressions (RE) are supported. The following characters have special meaning and must be escaped with a backslash (\) if they're not to be interpreted as RE characters: `*+?.^$(){}[]`. As an example, to search for handles beginning with a + character (e.g. e.164 handles), one must use `\+[0-9]+`.

The RE patterns are NOT anchored. The behavior is similar to the Unix *egrep* utility for searching for lines matching a specified RE. As an example, if LastName is "John", users with the last name of John, Johnson and Smith-Johnson will match. As is standard with regular expressions, the caret character (^) can be used to anchor the beginning of the pattern. The dollar character ($) anchors the end of the pattern.

Depending on their values and how user names are administered in the SMGR, there may be special handling with the FirstName, LastName, FirstNameAscii and LastNameAscii fields. First, if both the Unicode and ascii field is set (e.g. FirstName and FirstNameAscii), then there is no special handling. PPM will only return results where the Unicode value matches an administered user's Unicode administered name and the ascii value matches the administered users's ascii administred name. But if only the ascii version of the field(s) is specified, then PPM will match users having an ascii name field or Unicode name field that matches. That is, PPM extends the search to the administered Unicode names. If only the non-Ascii field(s) is specified but the value is ascii, then PPM will return users whose administered Unicode name matches or whose ascii name matches. That is, the search is extended to the administered ascii names.

searchUserResponse(SearchUserResult):

Response Parameter	Type	Value	Description
SearchUserResult	SearchResultListInfo	Complex	Results of the user search.

SearchResultListInfo	Type	Value	Description
NoOfElements	Integer	0 – "limit"	Number of records in the array
MoreResults	Boolean (optional)		If the full set of results cannot be returned due to the Limit parameter, then this is "true".
SearchResultsInfo	ArrayOfSearchResult	Maximum of "limit" elements.	List of SearchResult data types. The sort order for the records will be by last name. For equal last names, the first name is used. If both first and last names are the same, the internal database id of the records is used.

The greyed-out address-related fields in the SearchResult type are not supported.

SearchResult	Type	Description
SearchIndex	Integer	Index in the search result
PrimaryHandle	String	The "primary" handle of the user. See section 3.8 for which handle constitutes the primary handle. This field exists for backwards compatibility and should really be picked up from the <handles> list which also contains type and subtype for each handle. Note: this will include the avext parameter if necessary.
FirstName	String	First name in Localized characters.
LastName	String	Last name in Localized characters.
PreferredLanguage	ASCII String	Preferred Language of the contact.
FirstNameAscii	ASCII String	First name in ASCII characters
LastNameAscii	ASCII String	Last name in ASCII characters
LoginName	String	Login name
VideoCapable	Boolean (optional)	User has a video-capable phone
Address1	String	Address
Address2	String	Postal Address. Note: This isn't currently settable in the System Manager UI.
City	String	Locality name
State	String	State or province
Zip	String	Postal code
Building	String	Building number
OfficeLocation	String	Room number
Country	String	Country Name
Handles	HandleDataList	List of all user's handles including their PrimaryHandle. Note: this will include the avext parameter if necessary.

HandleDataList is described in the addContact operation section (section 3.1.1 addContact).

There is a limitation when the SearchResult references a public contact (see section 9.2.1-Public Contacts). Specifically, a public contact can only be directly dialed from a SearchResult if the <PrimaryHandle> contains an E.164 SIP handle administered via the System Manager. This handle does not have to represent an internal contact, it only needs to be defined as a SIP handle in E.164 format. This SearchResult can be added as a

contact. If it is, a subsequent getContactList will contain all the dialable numbers or handles defined for the public contact.

Here are the steps to configure an E.164 number for a public contact:

- In SMGR go to: User Management → Public Contacts
- For each Public Contact, do the following:
  - Select that contact and press Edit
  - Go to the Contact Address section and press New
  - Select a Type of "SIP" (the Category doesn't matter) and in the address field, enter a phone number dialable by the SM in the following format: sip:+CountryCodeAndSubscriberNumber@Domain

In Multi-Tenant situations, PPM will only return users that meet the searchCriteria specified in the request, i.e. if they reside in the same tenant partition as the user who is making the searchUser request.

Example searchUser Request:

```
<ns1:searchUser xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
  <SearchQuery>
    <LastNameAscii>john</LastNameAscii>
  </SearchQuery>
  <Start>0</Start>
  <Limit>50</Limit>
</ns1:searchUser>
```

PPM Response:

```
<ns1:searchUserResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SearchUserResult>
    <NoOfElements>2</NoOfElements>
    <MoreResults>false</MoreResults>
    <SearchResultsInfo soapenc:arrayType="ns1:SearchResult[2]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
     <item>
      <SearchIndex>0</SearchIndex>
      <PrimaryHandle>70005@avaya.com</PrimaryHandle>
      <FirstName>Steve</FirstName>
      <LastName>Johnson</LastName>
      <PreferredLanguage>en_US</PreferredLanguage>
      <FirstNameAscii>Steve</FirstNameAscii>
      <LastNameAscii>Johnson</LastNameAscii>
      <LoginName>sjohnson@avaya.com</LoginName>
      <VideoCapable>false</VideoCapable>
      <Handles>
       <NoOfElements>3</NoOfElements>
       <HandleList soapenc:arrayType="xsd:HandleData[3]"/>
        <item>
```

```
          <Handle>70005@avaya.com</Handle>
          <HandleType>sip</HandleType>
          <HandleSubtype>username</HandleSubtype>
        </item>
        <item>
          <Handle>stevejohnson@avaya.com</Handle>
          <HandleType>smtp</HandleType>
          <HandleSubtype>msexchange</HandleSubtype>
        </item>
        <item>
          <Handle>stevejohnson@pres.avaya.com</Handle>
          <HandleType>xmpp</HandleType>
          <HandleSubtype>jabber</HandleSubtype>
        </item>
    </HandleList>
    </Handles>
  </item>
  <item>
    <SearchIndex>1</SearchIndex>
    <PrimaryHandle>70002@avaya.com</PrimaryHandle>
    <FirstName>Kelly</FirstName>
    <LastName>Smith-Johnson</LastName>
    <PreferredLanguage>en_CA</PreferredLanguage>
    <FirstNameAscii>Kelly</FirstNameAscii>
    <LastNameAscii>Smith-Johnson</LastNameAscii>
    <LoginName>ksmithjohnson@avaya.com</LoginName>
    <VideoCapable>true</VideoCapable>
    <Handles>
      <NoOfElements>3</NoOfElements>
      <HandleList soapenc:arrayType="xsd:HandleData[3]"/>
        <item>
          <Handle>70002@avaya.com</Handle>
          <HandleType>sip</HandleType>
          <HandleSubtype>msexchange</HandleSubtype>
        </item>
        <item>
          <Handle>ksmithjohnson@avaya.com</Handle>
          <HandleType>smtp</HandleType>
          <HandleSubtype>msexchange</HandleSubtype>
        </item>
        <item>
          <Handle> ksmithjohnson@pres.avaya.com</Handle>
          <HandleType>xmpp</HandleType>
          <HandleSubtype>jabber</HandleSubtype>
        </item>
      </HandleList>
    </Handles>
    </item>
  </SearchResultsInfo>
 </SearchUserResult>
</ns1:searchUserResponse>
```

Error Conditions/Faults:

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
DatabaseError	PPM cannot access the database for reading.
RegExpError	An invalid regular expression was specified.
NotIMUser	Requesting user does not exist in the Enterprise database
InvalidValue	String length exceeded maximum

3.2 CONFIGURATION INFORMATION

3.2.1 getInitialEndpointConfiguration

This method retrieves the endpoint's configuration information required prior to user login. This differs from getAllEndpointConfiguration which returns configuration information after user login. Since getInitialEndpointConfiguration is unauthenticated, only non-sensitive data can be retrieved. In the future, this request will be used to return the data in the phone settings file; thus reducing the need for that file.

The objects returned are described in alphabetical order but are not returned that way in the SOAP message.

getInitialEndpointConfiguration(Fields)

Request Parameter	Type	Length / Value	Description
Fields	ArrayOfEndpointConfigurationFields (optional)	Array of an enumeration	List of configuration fields that are to be returned in response. If not specified, all supported fields are returned.
Scope	String (optional)	0 through 999	This value corresponds to the Group parameter administered on SIP phones. It is passed by the phone to PPM and is used to filter the data returned in the IpAddressFamilySettings.

EndpointConfigurationFields is the same enumeration used with getAllEndpointConfiguration but with only a subset supported. The list of supported fields is:

EndpointConfigurationFields	Data Returned In Response
ListOfEmergencyNumbers	Emergency numbers that are applicable to the phone based on its ip address/location. For Avaya Remote Office Solution (REMO) users, only the global ListOfEmergencyNumbers will be returned.
IpAddressFamilySettings	The IpAddressFamilySettings contains information about Ip address type supported by signaling mode and media mode.

Note that the endpoint must request each of the information elements that it wants using these fields. Unsupported configuration fields passed in the Fields element are ignored. If the Fields element doesn't exist in the request, then all supported information elements are returned. If the Fields element exists but the list is empty, no information elements are returned as one would expect.

getInitialEndpointConfigurationResponse(AllEndpointConfigInfo)

Response Parameter	Type	Description
ConfigInfo	AllEndpointConfigInfo	Container for objects listed below.

AllEndpointConfigInfo is also used by getAllEndpointConfigurationResponse. However, only a subset of elements is supported for the getInitialEndpointConfiguration response. The supported ones are shown below:

AllEndpointConfigInfo	**Corresponding Data**
ListOfEmergencyNumbers	All emergency numbers for the phone as described below.
IpAddressFamilySettings	Ip address type (IPv4 / IPv6 / Dual stack) configuration information as described below

3.2.1.1 ListOfEmergencyNumbers

This contains the list of emergency numbers as configured through SMGR.

For non-REMO users, these numbers are based on the SM location associated with the IP address of the phone. If no SM location matches the IP address of the phone, the numbers come from the phone's assigned home location. In either case the global ListOfEmergencyNumbers is returned.

For REMO users, only the global ListOfEmergencyNumbers will be returned.

Response Parameter	Type	Description
ListOfEmergencyNumbers	EmergencyNumberListInfo	Container for Emergency Number information

EmergencyNumberListInfo	Type	Description
NoOfElements	Integer	Number of elements in array
EmergencyNumberList	Array of EmergencyNumberData	A list of EmergencyNumberData. There is a maximum of 10. Starting with SM 6.2 Feature Pack 1 this limit is increased to 100.

EmergencyNumberData	Type	Value	Description
Type	String	Max of 255 characters	This is the type of emergency such as Police, Fire and Ambulance.
Number	String	Max of 32 characters	This is the dialable number.

3.2.1.2 Data Description

The first EmergencyNumberData element will be the primary number and should be used by the endpoints emergency number soft button.

3.2.1.3 IpAddressFamilySettings

The IpAddressFamilySettings contains information about Ip address type supported by signaling mode and media mode. These values can be configured from Device Settings Groups or Location Settings on SMGR.

Response Parameter	Type	Description
IpAddressFamilySettings	IpAddressFamilySettings	Container for IpAddressFamilySettings

IpAddressFamilySettings	Type	Value	Description
SignalingMode	String	4 or 6	4 indicates IPv4 6 indicates IPv6
MediaMode	String	4 or 6 or 46 or 64	4 indicates IPv4 6 indicates IPv6 46 indicates Dual stack with IPv4 preferred 64 indicates Dual stack with IPv6 preferred

getInitialEndpointConfiguration(Fields)

Example getInitialEndpointConfiguration request without specifying Fields:

```
<ns1:getInitialEndpointConfiguration
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
</ns1:getInitialEndpointConfiguration>
```

Example getInitialEndpointConfiguration request using the Fields:

```
<ns1:getInitialEndpointConfiguration
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Fields xsi:type="ns:ArrayOfEndpointConfigurationFields"
soapenc:arrayType="ns:EndpointConfigurationFields[1]">
    <item>ListOfEmergencyNumbers</item>
    <item>IpAddressFamilySettings</item>
  </Fields>
```

```
   <Scope>0</Scope>
</ns1:getInitialEndpointConfiguration>
```

PPM response:

```
<ns1:getInitialEndpointConfigurationResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ConfigInfo>
    <ListOfEmergencyNumbers>
      <NoOfElements>3</NoOfElements>
      < EmergencyNumberList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:EmergencyNumberData[3]" xsi:type="soapenc:Array">
        <item>
          <Type>Police</Type>
          <Number>911</Number>
        </item>
        <item>
          <Type>Ambulance</Type>
          <Number>911</Number>
        </item>
        <item>
          <Type>Fire</Type>
          <Number>911</Number>
        </item>
      </ EmergencyNumberList>
    </ListOfEmergencyNumbers>
<IpAddressFamilySettings>
            <SignalingMode>4</SignalingMode>
            <MediaMode>6</MediaMode>
  </IpAddressFamilySettings>
  </ConfigInfo>
</ns1:getInitialEndpointConfigurationResponse>
```

E**rror Conditions/Faults:**

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
“DataNotAvailable ”	PPM is unable to obtain expected data.
“InternalError”	Unexpected exception in PPM processing
“DatabaseNotAccessible”	There was a problem with the database

3.2.2 getAllEndpointConfiguration

This method retrieves the endpoint's required configuration information. The objects returned are described in alphabetical order but are not returned that way in the SOAP message.

getAllEndpointConfiguration(Handle, Identity, Fields, Scope)

Request Parameter	Type	Length / Value	Description
Handle	String	SIP URI max 256 characters	The handle of the user in the form of "user@domain". Also see the note in Section 3.8 Handles.
Identity	DeviceIdentity (optional)	Identity	The identity of the specific device to retrieve information for. If not supplied the method returns volume setting information applicable to all devices owned by the user.
Fields	ArrayOfEndpointConfigurationFields (optional)	Array of an enumeration	List of configuration fields that are to be returned in response. If not specified, all fields are returned.
Scope	String (optional)	0 through 999	This value corresponds to the Group parameter administered on SIP phones. It is passed by the phone to PPM and is used to filter the data returned in the ListOfMaintenanceData and VolumeSettings.

DeviceIdentity	Type	Length / Value	Description
DeviceHandle	String (optional)	nn:nn:nn:nn:nn:nn Max 128 characters	MAC address of the device. "nn" values are 6 pairs of hexadecimal digits separated by hyphens or colons (as shown here).

EndpointConfigurationFields is an enumeration with the following values:

EndpointConfigurationFields	Data Retrieved in Response
AutoAnswer	Auto Answer Type. Described below.
DialPlanData	Dial plan elements. Described below.
IpAddressFamilySettings	IpAddressFamily settings data. Described below.
LinePreferenceInfo	Line Preference. Described below.
ListOfButtonAssignments	Button Assignments. Described below.
ListOfCmSystemParameters	CM System Parameter values. New for SM 6.2 Feature Pack 4. Described below.
ListOfConfigPacketVersions	Version information for each packet.
ListOfContacts	Not supported, use getContactList
ListOfDeviceData	Not supported, use getDeviceData
ListOfDevices	Not supported, use getDeviceData
ListOfEmergencyNumbers	Emergency Number elements, see 3.2.1.1
ListOfIdentities	Identities defined for the endpoint. Described below.
ListOfMaintenanceData	Maintenance data used by the endpoint. Described below.
ListOfNumberFormatRules	Number Format Rules. Described below.
ListOfOneTouchDialData	One Touch Dials. Described below.
ListOfPermissions	Not supported.
ListOfRegistrations	Not supported
ListOfRingerOnOffData	Ringer settings. Described below.
ListOfSpeedDialData	Not supported.
ListOfTimers	Timers required by the endpoint. Described below.
ListOfWatchers	Who's watching the user's presence. Not supported.
MWExt	Message Waiting Extension. Described below.
MuteOnRemoteOffHook	Specifies the Endpoint behavior when a 96xx hard phone and a 1XCommunicator (soft phone) are linked together in Shared Control mode. New for 6.2 FP1. Described below.
SoftMenuKeyList	Information on the user's soft menu keys. Described below.
TerminalGroupId	The user's IP Phone Group Id configured on the SMGR.
VideoInfo	Softphone and video assignments. Described below.
VMNumber	Voice mail number. Described below.
VMONInfo	VMON data items. Described below.
VolumeSettings	Volume settings data. Described below.

Note that the endpoint must request each of the XML packets that it wants using these fields. This is a change from SM 6.0 and earlier releases where these fields were ignored, and all XML packets were returned. If the Fields element doesn't exist in the request, then all XML packets are returned. If the Fields element exists but the list is empty, no

XML packets are returned as one would expect. The one exception is the AutoAnswerType. It will always be returned in the getAllEndpointConfigurationResponse.

getAllEndpointConfigurationResponse(AllEndpointConfigInfo)

Response Parameter	Type	Description
ConfigInfo	AllEndpointConfigInfo	Container for objects listed below.

AllEndpointConfigInfo	**Corresponding Data**
AutoAnswer	Auto Answer Type. Described below.
DialPlanData	Dial plan elements. Described below.
IpAddressFamilySettings	IpAddressFamily settings data. Described below.
LinePreferenceInfo	Line Preference. Described below.
ListOfButtonAssignments	Button Assignments. Described below.
ListOfCmSystemParameters	CM System Parameter values. New for SM 6.2 Feature Pack 4. Described below.
ListOfConfigPacketVersions	Default data in response. Described below.
ListOfContacts	Not supported, use getContactList
ListOfDeviceData	Not supported, use getDeviceData
ListOfDevices	Not supported, use getDeviceData
ListOfEmergencyNumbers	Emergency Number elements, see 3.2.1.1
ListOfIdentities	Identities defined for the endpoint. Described below.
ListOfMaintenanceData	Maintenance data used by the endpoint. Described below.
ListOfNumberFormatRules	Number Format Rules. Described below.
ListOfOneTouchDialData	One Touch Dials. Described below.
ListOfPermissions	Not supported.
ListOfRegistrations	Not supported
ListOfRingerOnOffData	Ringer settings. Described below.
ListOfSpeedDialData	Contacts on the speed dial list. Described below. Currently not used.
ListOfTimers	Timers required by the endpoint. Described below.
ListOfWatchers	Who's watching the user's presence. Not supported.
MWExt	Message Waiting Extension. Described below.

AllEndpointConfigInfo	Corresponding Data
MuteOnRemoteOffHook	Specifies the Endpoint behavior when a 96xx hard phone and a 1XCommunicator (soft phone) are linked together in Shared Control mode. New for 6.2 FP1. Described below.
SoftMenuKeyList	Information on the user's soft menu keys. Described below.
TerminalGroupId	The user's IP Phone Group Id configured on the SMGR.
VideoInfo	Softphone and video assignments. Described below.
VMNumber	Voice mail number. Described below.
VMONInfo	VMON data items. Described below.
VolumeSettings	Volume settings data. Described below.

3.2.2.1 AutoAnswer

Object to retrieve the value of the AutoAnswer field in CM.

Auto Answer Types:

all	Enter all to allow all calls (ACD and non-ACD) terminated to an idle station to be cut through immediately. Does not allow automatic hands-free answer for intercom calls.
acd	Enter acd to allow only ACD split /skill calls and direct agent calls to auto answer. If this field is set to acd, Non-ACD calls terminated to a station ring audibly.
none	Enter none to cause all calls terminated to this station to receive an audible ringing treatment.
icom	Enter icom to allow a phone user to answer an intercom call from the same intercom group without pressing the intercom button.

Response Parameter	Type	Description
AutoAnswer	AutoAnsType	See Below

AutoAnsType	Type	Value	Description
AutoAnsType	Enum	{none, icom, acd, all}	The Auto Answer setting for this station on CM.

3.2.2.2 ListOfConfigDataPacketVersionsInfo

Default data is returned in this object.

Response Parameter	Type	Description
ListOfConfigDataPacketVersions	ConfigDataPacketVersionsInfo	Container for data packet versions

ConfigDataPacketVersionsInfo	Type	Description
NoOfElements	Integer	Number of elements in array
ConfigDataPacketVersionsList	Array of ConfigDataPacketVersions	See ConfigDataPacketVersions below

ConfigDataPacketVersions	Type	Description
PacketName	String	This message is currently providing predefined data. No changes can be made to this field.
PacketVersion	String	This message is currently providing predefined data. No changes can be made to this field.

3.2.2.3 DialPlanData

An endpoint may obtain a simplified Dial Plan with this operation. The Dial Plan information will contain all global data administered on CM as well as any location-specific data relevant to the endpoint's CM location.

PPM determines the user's CM location based on data from the CM station form, or from the endpoint's ip-address if it is administered on the CM ip-network-map form. If it is determined the user has a specific location other than the global location administered, then both the global and location-specific entries are sent as part of the DialPlanData. If the user is not administered with a CM location, or one cannot be determined from their ip-address, then only the global dial plan entries are included in the DialPlanData.

For REMO users, the DialPlanData follows the same logic (it includes both location-specific and global location entries, or just the global entries if the user has no CM location), but the location-based entries are derived from the user's home location on SM.

Note that enbloc entries on the CM dialplan page, denoted with call-type of "enb-ext", are not included in the dialplan sent to the endpoint.

The Dial Plan will be represented as a list of terms, where each term is a nearly generic regular expression form.

The rules for each term are:

- x = any digit
- Z = play dialtone

- + = 0 or more instances of the preceding character, implied inter-digit timer

(The PPM use of the + symbol differs from standard regular expressions, where + means 1 or more instances of the preceding characters.)

Some sample terms for a dialplan:

- 538xxxx = a 7-digit number starting with 538
- 9Z1xxxxxxxxxx = ARS access to a long-distance number in North America (dialtone after the 9)
- 9Z011xxxxx+ = ARS access to an international number of unknown length, but at least 7 digits (including 011)

Response Parameter	Type	Description
DialPlanData	DialPlanListInfo	Container for all dial plan information. For REMO users, only global emergency numbers configured for the Session Managers will be mixed into the dialplan. For other users, the global dial plan data as well as any dial plan information specific to an endpoint's CM location will be included with this parameter. In addition, local and global emergency numbers configured for the Session Managers will be mixed into the dialplan.

DialPlanListInfo	Type	Value	Description
DialPlanDomain	String	Max 255 characters	The domain of the dial plan. Currently this is same as the enterprise SIP domain.
NoOfElements	Integer		Number of elements in the array
DialPlan	Array of Strings	Expression representing rules described above	Each string in the array represents a single term of the CM dial plan. Configurable limit – maximum of 300 terms returned.

3.2.2.4 ListOfButtonAssignments

Contains the list of buttons assigned to this endpoint through CM.

Response Parameter	Type	Description
ListOfButtonAssignments	ButtonListInfo	Container for Button Assignments information

ButtonListInfo	Type	Description
NoOfElements	Integer	Number of elements in array
ButtonAssignment	Array of ButtonData	Button assignments (below)

ButtonData	Type	Value	Description
Location	Short	1 – 96	This value specifies the number (location) of the button. CM set types can support as many as 96 buttons.
ButtonType	String	Max 10 characters	This specifies the button type and consists of letters, numbers and dashes.
Label	String	Max 255 characters	A string that can be displayed for the button. If no label has been defined, PPM will provide a default label from known data (for example, from the extension for a bridged appearance). This comes from a button-label mapping configuration file. For information about the team button and the label, see 3.2.2.4.1 Team Button Label.
LineID	Short	1-96	This gives the LineID value, or no value if a line ID is not relevant for the button type. - For a normal (numeric) bridged appearance button, this is the Button number subfield found on CM. - If a normal (numeric) bridged appearance button points to a CS1k endpoint, PPM maps the Button number range 0-95 in the database to the range 1-96. - For an MCA bridged appearance button (8.0 and higher), the CM Button number subfield is the letter “a”, but PPM sends instead the numeric Location value as the LineID. - In most instances, the value is 0, but it is the button number for call-apprs.
Address	String	Max 256 characters	If the button type is brdg-appr, the bridge extension goes here. If the button type is autodial, the destination digits go here. If auto-msg-wt, the destination extension goes here. If the button is a team button, the extension being monitored is given here. For some FNUs (e.g. call-fwd), the source

			extension, if different than the user's, is specified here.
FNUType	String	A pre-defined string unique to each supported feature	See "FNU Description" below. Only appears for buttons whose pushes result in INVITES with FNUs sent from the endpoint to CM.
PUBType	String	A pre-defined string unique to each supported feature	See "PUB Description" below. Only appears for buttons whose pushes result in a PUBLISH message sent from the endpoint to CM.
App	Boolean		If "true", the endpoint must request a line appearance (+line) on the Contact header in an INVITE with this FNU.
Media	Boolean		If "true", the endpoint must include an SDP offer in an INVITE with this FNU.
PickupByGoing OffHook	Boolean		Only present for the Team button. It is a field retrieved from the user's Class of Restriction information on the CM.
SilentIfActive	Boolean		Only present for the Team button. It is a field retrieved from the user's Class of Restriction information on the CM.
SACCFOverride	String	Can have the values: 'a'(ask), 'n' (no), or 'y' (yes).	Only present for the Team button. It is data retrieved from the user's Station form on the CM. (Added in SM 6.2 Feature Pack 4)
FNUInfo	Array of FNUData	Varies by FNU	See "FNU Description" below. Only appears for FNU Button Types. Contains the data needed by the endpoint to construct the INVITE message to CM for the specified button push.
PUBPayload	Container for PublishPayload data for this button	Varies by Button	See "PUBPayload Description" below. Only appears for PUB Button Types. Contains the data needed by the endpoint to construct the PUBLISH message to CM for the specified button push.
ListenOnly	Boolean		Only present for Service Observe button. The "listen-only" qualification set to "y" will not allow the user to toggle to the listen/talk mode for any of the SO available options activated.
Coaching	Boolean		Only present for Service Observe button. The coach qualification set to "y" will allow the observer to enable SO Coaching mode where he/she can talk only to the agent while the

			caller is connected and the agent can hear and talk to the caller and the caller does not hear the observer.

FNUData	Type	Description
Parameter	String	A parameter to the FNU (as presented in FNUType). If the parameter requires completion by the client, it is the parameter string ending with "=" (e.g., "avaya-cm-action="). If the parameter is filled in by PPM and is complete (e.g., "avaya-cm-group=3"), the client would use the parameter as is.
Required	Boolean	If "true", the client must include this parameter. If "false", the client may include the parameter if it is able to complete the parameter or provide the feature processing.
MustComplete	Enum	If "none", the parameter is already complete. If "offon", the endpoint must complete the parameter with either "off" or "on". If "address", the endpoint must complete the parameter with an address (e.g., telephone number or extension). If "number", the endpoint must complete the parameter with a number. If "dialcode", the endpoint must complete the parameter with a dial code. If "pickupORspeeddial", the endpoint must complete the parameter with either "pickup" or "speeddial". If "call_id", the endpoint must complete the parameter with a call-id.

PublishPayload	Type	Description
SIPEvent	String	A string defining the subscription event type to be used in the PUBLISH message sent from the endpoint to CM. Currently, only "avaya-cm-cc-info" is supported.
CCInfoEvent	String	Contains the schema, namespace, version and state information for the specified event package.
clientPublish	Container for ClientPub	Container that specifies the type of entity (native/physical station or agent) and the specific button type for processing. See "clientPublish" below.

ClientPub	Type	Description

agentEntity or nativeEntity	String	Each clientPublish object will contain either an agentEntity or nativeEntity item. When "agentEntity" is present it indicates the button push associated with the agent extension that is logged into the physical station. When the "nativeEntity" item is present it indicates the button push is associated with the endpoint's physical station extension. In both cases, PPM sends the value "$Required" which indicates the endpoint must fill in the extension when sending the PUBLISH message to CM.
requestedAfterCall, requestedAutoIn, requestedAux, requestedCWC, requestedManIn, requestedOverride, requestedQStats, requestedStrokeCount, requestedSupAssist, requestedVuStats or requestedSO.	Data container for 0 or more parameters. (see below)	This item is one of a list of items that indicate the actual button push that CM is to process. Of these, requestedAfterCall,, requestedAutoIn, and requestedManIn, have no additional parameters. The others do have parameters which are listed below.

AuxParams	Type	Description (For requestedAux)
reasonCode	Int	If CM has a reason code administered in the RC field for this button, then PPM will send this value as stored in the database. Otherwise PPM sends "$Required" and the endpoint fills in a value.

StrokeCountAndWorkCodeParams	Type	Description (For requestedCWC and requestedStrokeCount)
code	String	PPM sends "$Required" indicating the endpoint must provide this data as input by the user.

LogoutOvrParams	Type	Description (For requestedOverride)
overrideState	String	PPM will set this value to "$Required", indicating the endpoint must provide a value of "true" or "false" when creating the PUBLISH message for logout override to CM. The endpoint sends "true" the first time the agent pushes the button, indicating the agent wishes to remain logged in

		past the administered forced logout time. The button then becomes a toggle.

QStatsParams	Type	Description (For requestedQStats)
objectSkill	int	PPM sends the group number administered on the CM station form for this button..
buttonNumber	int	PPM stores the value from the "Location" field of the button table into this field. It is here because CM needs this data in the PUBLISH message that will be created by the endpoint.

AssistParams	Type	Description (For requestedSupAssist)
skill	String	This is the hunt (skill) group number configured with the "assist" button on the CM station form. If no button is administered, PPM will send "$Required", indicating the endpoint must supply a skill number.
dialogID	String	PPM sends "$Required", indicating the endpoint must provide this information.

VuStatsParam	Type	Description (For requestedVuStats)
formatNumber	int	PPM supplies the value administered on the station for the Fmt parameter for this button. It will always be present.
objectValue	String	This value corresponds to the "ID" parameter administered on the CM station form for the vu-display button. If it is not administered on the station form, PPM will send the string "$Optional" for this field.
buttonNumber	int	PPM stores the value from the "Location" field of the button table into this field. It is here because CM needs this data in the PUBLISH message that will be created by the endpoint.

SOParam	Type	Description (For requestedSO)
soType	String	Type is any of the following Service Observing types: "Basic", "No-Talk", "Next-Call", or "By-Location".
soEntity	String	This value is the extension of the observed entity.
location	int	This is the location number associated with the observed VDN. The valid range is 0 – 2000. The value 0 means that the <soType> is not "By-Location".

mode	String	This value indicates whether the observer will join the call or should be in “listenonly” or “listen-talk” mode.

3.2.2.4.1Team Button Label

The Label field for the Team button can be the monitored endpoint’s extension, station name or station native name. It can also be formatted to prepend the data with a “T:”. The value is dependent upon three different settings in the CM.

The field, “Extension only label for Team button on 96xx H.323 terminals?” on the System-Parameters Features form controls the prepending of the “T:” to the label. If this is set to “n”, the data will be formatted to start with “T:”. If “y”, it will not have the “T:”.

The value that controls whether the extension or the name is used in the Label, is located on the Class of Restriction form. The field is called “Team Btn Display Name”. A setting of “n” will send the extension of the monitored station and “y” will send the name associated with the monitored station. The field that determines whether the station name or the Unicode name will be in the Label field, is on the Station form and is the “Display Language”. If the value is set to “Unicode” and a native name has been administered for the monitored station, the native name will be sent in the Label field. If the “Team Btn Display Name” is set to “y” but there is no station name or native name administered, the extension of the monitored station is sent.

Here are some examples:

Extension Only field	**Team Btn Display Name field**	**Display Language field**	**Station name administered?**	**Native name administered?**	**Label**
Yes	No	Unicode	Y	Y	53487
Yes	Yes	English	Y	Y	Muller, John
Yes	Yes	Unicode	Y	Y	Müller, Johann
No	Yes	Unicode	Y	N	T:Muller, John
No	Yes	Unicode	N	N	T:53487

3.2.2.5 FNU Description

The following table shows the list of features accessible via FNU in Communication Manager. Many of the FNUs require additional parameters. A “true” value in the column labeled “Required” indicates that the endpoint must send the indicated parameter. A “false” value in the column labeled “Required” indicates that the FNU parameter

provided by PPM in FNUInfo is fully qualified and may not require any further addition from the endpoint. The “MustComplete” value indicates the param type that is to be used to send in the FNU. The FNUInfo parameter names cannot exceed 31 characters in length due to restrictions in the Avaya SIP stack.

Feature Name	FNUType	App	Media	FNUInfo		
				Parameter	Required	MustComplete
Agent Login	avaya-cm-fnu=agnt-login	false	false	avaya-cm-agent-id={extension} avaya-cm-agent-password={password}	true false	extension password, if needed
Auto Callback	avaya-cm-fnu=auto-callback	false	false	avaya-cm-action={on|off}	true	offon
Auto Intercom	avaya-cm-fnu=auto-intercom	true	true	avaya-cm-group={group} avaya-cm-dial-code={code}	true true	none none
Busy Indicator	avaya-cm-fnu=busy-indicator	true	true	avaya-cm-destination={address}	true	address
Call forwarding all	avaya-cm-fnu=call-forwarding-all	true	true	avaya-cm-destination={address} avaya-cm-action={on|off}	false true	address offon
Call forwarding busy/no answer	avaya-cm-fnu=call-forwarding-busy-no-answer	true	true	avaya-cm-destination={address} avaya-cm-action={on|off}	false true	address offon
Call park	avaya-cm-fnu=call-park	false	false			
Call unpark (answer back)	avaya-cm-fnu=call-unpark	true	true	avaya-cm-extension={address}	false	address
Call pick-up Group	avaya-cm-fnu=call-pickup	true	true			
Call pickup Directed	avaya-cm-fnu=call-pickup-directed	true	true	avaya-cm-extension={address}	false	address
Call pickup Extended group	avaya-cm-fnu=call-pickup-extended	true	true	avaya-cm-pickup-number={num}	false	number
Per call calling party number block	avaya-cm-fnu=calling-party-block	true	true	avaya-cm-destination={address}	false	address
Per call calling party number unblock	avaya-cm-fnu=calling-party-unblock	true	true	avaya-cm-destination={address}	false	address
Crisis Alert (8.1)	avaya-cm-fnu=crisis-alert	true	true	avaya-cm-monitor-option	true	noneORlisten
Dial Intercom	avaya-cm-fnu=dial-intercom	true	true	avaya-cm-group={group} avaya-cm-dial-code={code}	true false	none dialcode
Drop	avaya-cm-fnu=drop	false	false			
Exclusion	avaya-cm-fnu=exclusion	false	false	avaya-cm-action={on|off}	true	offon
Extended Call	avaya-cm-fnu=extend-call	false	false			

Enhanced Call Forwarding	avaya-cm-fnu=enhanced-call-forwarding * see note 1 about this FNU	true	true	avaya-cm-cfall-external-dest={address} avaya-cm-cfall-external-action={on|off} avaya-cm-cfall-internal-dest={address} avaya-cm-cfall-internal-action={on|off} avaya-cm-cfbusy-external-dest={address} avaya-cm-cfbusy-external-action={on|off} avaya-cm-cfbusy-internal-dest={address} avaya-cm-cfbusy-internal-action={on|off} avaya-cm-cfnr-external-dest={address} avaya-cm-cfnr-external-action={on|off} avaya-cm-cfnr-internal-dest={address} avaya-cm-cfnr-internal-action={on|off}	false true false true false true false true false true false true	address offon address offon address offon address offon address offon address offon
Hunt group busy position	avaya-cm-fnu= hunt-group-busy-position	true	false	avaya-cm-group={group} avaya-cm-action={on|off}	true true	none offon
Off-PBX call	avaya-cm-fnu=off-pbx	false	false	avaya-cm-action={on|off}	True	Offon
Off hook	avaya-cm-fnu=off-hook	true	false			
Last number dialed	avaya-cm-fnu=last-number-dialed	true	true			
Limit Number of Concurrent Calls	avaya-cm-fnu=limit-call	false	false	avaya-cm-action={on|off}	true	offon
Malicious Call Trace	avaya-cm-fnu=mct	false	false			
No Hold Conference (8.1)	avaya-cm-fnu=no-hold-conf	false	false	avaya-cm-destination	true	address
One Touch Recording	avaya-cm-fnu=one-touch-recording	false	false	avaya-cm-action={on|off} avaya-cm-extension={address}	true true	offon address
Priority call	avaya-cm-fnu=priority-call	true	true	avaya-cm-destination={address}	false	address
Send all calls	avaya-cm-fnu=sac	false	false	avaya-cm-action={on|off}	true	offon

Team	avaya-cm-fnu=team	true	true	avaya-cm-action={pickup|speeddial} avaya-cm-destination={address} Note : this option is only needed when the action is set to speeddial avaya-call-id={call_id} Note : this option is only needed when the action is set to pickup avaya-cm-redirection-override={on|off} Note : this option is only used when the avaya-cm-action is set to speeddial.	true false false false	pickupORspeeddial address call_id offon
ASAI UUI	avaya-cm-fnu=uui-info	false	false			
Whisper page	avaya-cm-fnu=whisper-page	true	true	avaya-cm-extension={address}	false	address

Note 1: This FNU consists of pairs of information {destination, action}. When PPM sends this information to the endpoint, it will send all 12 FNU items. If a third-party extension is administered to this button, the 12 FNU items will contain the enhanced call forward settings of the third party extension.

3.2.2.6 PUB Description

The following table shows the list of buttons that, when pushed, result in a PUBLISH message to CM. The getAllEndpointConfigurationResponse message for these button items will include a "PUBType" and a PUBPayload element. Data from the PUBPayload element is used by the endpoint to construct a PUBLISH message to CM. Examples of the XML sent by PPM in the button item for each PUB button are included below.

Button Label	PUBType	PUBPayload
After Call Work	after-call	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> < CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" version="1" state="partial"/> <clientPublish> <agentEntity>$Required</agentEntity> <requestedAfterCall /> </clientPublish> </PUBPayload>
Supervisor Assist	assist	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> < CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" version="1" state="partial"/> <clientPublish> <agentEntity>$REQUIRED</agentEntity> <requestedSupAssist> <skill>$REQUIRED</skill> ** filled by PPM if admin'd on CM

Button Label	PUBType	PUBPayload
		<dialogID>>$REQUIRED</dialogID> ** filled by the endpoint if there is an active call when the button is pressed </requestedSupAssist> </clientPublish> </PUBPayload>
Auto In	auto-in	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> < CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" version="1" state="partial"/> <clientPublish> <agentEntity>$Required</agentEntity> <requestedAutoIn /> </clientPublish> </PUBPayload>
Auxiliary	aux-work	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> < CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" version="1" state="partial"/> <clientPublish> <agentEntity>$REQUIRED</agentEntity> <requestedAux> <reasonCode>$REQUIRED</reasonCode> ** filled by PPM if admin'd on CM </requestedAux> </clientPublish> </PUBPayload>
Logout Override	logout-ovr	<PUBPayload> <SIPEvent="avaya-cm-cc-info" /SIPEvent> <CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" state="partial" version="1" /> <clientPublish> <agentEntity>$REQUIRED</agentEntity> <requestedOverride> <overrideState>$REQUIRED</overrideState> filled by endpoint based on feature state </requestedOverride> </clientPublish> </PUBPayload>
Manual In	manual-in	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> < CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" version="1" state="partial"/> <clientPublish> <agentEntity>$REQUIRED</agentEntity> <requestedManIn /> </clientPublish> </PUBPayload>
Queue Status	q-calls	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> < CCInfoEvent

Button Label	PUBType	PUBPayload
		xmlns="http://www.avaya.com/schemas/ccinfo.xsd" version="1" state="partial"/> <clientPublish> <nativeEntity>$REQUIRED</nativeEntity> <requestedQStats> <objectSkill>7</objectSkill> <buttonNumber>12</buttonNumber> </requestedQStats> </clientPublish> </PUBPayload
Stroke Count	stroke-cnt	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> < CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" version="1" state="partial"/> <clientPublish> <agentEntity>$REQUIRED</agentEntity> <requestedStrokeCount> <code>3</code> </requestedStrokeCount> </clientPublish> </PUBPayload>
Call Work Code	work-code	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> < CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" version="1" state="partial"/> <clientPublish> <agentEntity>$REQUIRED</agentEntity> <requestedCWC> <code>$REQUIRED</code> ** filled by endpoint via user prompt </requestedCWC> </clientPublish> </PUBPayload>
Vu Stats	vu-display	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> <CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" version="1" state="partial"/> <clientPublish> <nativeEntity>$REQUIRED</nativeEntity> <requestedVuStats> <formatNumber>2</formatNumber> <objectValue>$OPTIONAL</objectValue> *** filled by PPM if admin'd on CM <buttonNumber>8</buttonNumber> ** filled by PPM </requestedVuStats> </clientPublish> </PUBPayload>
Service Observe	sip-sobsrv	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> <CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" state="partial" version="1" />

Button Label	PUBType	PUBPayload
		<clientPublish> <nativeEntity>$Required</nativeEntity> <requestedSO> <soType>$Required</soType> <soEntity>$Required</soEntity> <location>$Required</location> <mode>$Required</mode> </requestedSO> </clientPublish> </PUBPayload>
Add/Remove Skill	add-rem-sk	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> <CCInfoEvent xmlns=http://www.avaya.com/schemas/ccinfo.xsd version=”1” state=”partial”> <clientPublish> <nativeEntity>$Required</nativeEntity> <requestedSkillChg> <agentID>$Required</agentID> <action>$Required</action> <skill>$Required</skill> <skillLevel>$Required</skillLevel> </requestedSkillChg > </clientPublish> </CCInfoEvent> </PUBPayload>
VOA Repeat	voa-repeat	<PUBPayload> <SIPEvent>avaya-cm-cc-info</SIPEvent> < CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" version="1" state="partial"/> <clientPublish> <nativeEntity>$Required</nativeEntity> <requestedVOA> </requestedVOA> </clientPublish> </PUBPayload>

Note: Although PPM supports the Add/Remove Skill and VOA Repeat buttons in 7.0.1, the Avaya Aura offer does not fully support these two buttons yet.

3.2.2.7 Description of Other Buttons

The following table shows the buttons that are neither FNU nor PUB buttons. Examples of the XML sent by PPM in the button item for each button are included.

Button Type	Internal Name	XML
Autodial	autodial	<item> <Location>25</Location> <ButtonType>autodial</ButtonType> <Label>70005</Label> <LineID>0</LineID> <Address>70005</Address> <FNUType /> <App>false</App> <Media>false</Media>

Button Type	Internal Name	XML
		<FNUInfo soapenc:arrayType="ns1:FNUData[0]" xsi:type="soapenc:Array" /> </item>
Automatic Message Waiting	aut-msg-wt	<item> <Location>5</Location> <ButtonType>aut-msg-wt</ButtonType> <Label /> <LineID>0</LineID> <Address>3030999</Address> <FNUType /> <App>false</App> <Media>false</Media> <FNUInfo soapenc:arrayType="ns1:FNUData[0]" xsi:type="soapenc:Array" /> </item>
Bridged Appearance	brdg-appr	<item> <Location>10</Location> <ButtonType>brdg-appr</ButtonType> <Label>70005</Label> <LineID>1</LineID> <Address>70005</Address> <FNUType /> <App>false</App> <Media>false</Media> <FNUInfo soapenc:arrayType="ns1:FNUData[0]" xsi:type="soapenc:Array" /> </item>
Call appearance	call-appr	<item> <Location>1</Location> <ButtonType>call-appr</ButtonType> <Label>70004</Label> <LineID>1</LineID> <Address /> <FNUType /> <App>false</App> <Media>false</Media> <FNUInfo soapenc:arrayType="ns1:FNUData[0]" xsi:type="soapenc:Array" /> </item>
Conference*	conference	same as Call Appearance with Label=”conference”
Transfer*	transfer	same as Call Appearance with Label=”transfer”

* Conference and Transfer buttons are only supported on CS1k-* endpoints. In SM 8.1.1, if a user is using both CS1k and non-CS1k Endpoints, PPM will replace the button labels for Conference and Transfer buttons of the non-CS1k Endpoints with the user’s extension. CS1k Endpoints shall use the Identity parameter in the getAllEndpointConfiguration request to identity themselves as such.

3.2.2.8 ListOfCmSystemParameters

This object/XML element is used to contain the list of all the CM System Parameters which PPM sends to the SIP Endpoints. For SM Feature Pack 4, it only contains two parameters. One of these two parameters, MuteOnRemoteOffHook was an existing parameter sent to SIP Endpoints, so it appears on its own in another place in the getAllEndpointConfigurationResponse for backwards compatibility reasons.

Response Parameter	Type	Description
ListOfCmSystemParameters	CmSystemParameterInfo	Container for all CM system parameters.

CmSystemParameterInfo	Type	Value	Description
NoOfElements	Int	0 ... n	Number of name/value pairs in the array.
CmSystemParameters	ArrayOfCmSystemParameter	SOAP Array	Array of name/value pairs.

CmSystemParameter	Type	Value	Description
Name	String	Max 255 characters	String length varies based on parameter name.
Value	String	Max 255 characters	String length varies based on parameter value.

3.2.2.9 ListOfIdentities

This object returns a list of the user's handles and extensions. Handles include SIP, email, and presence. Extensions are CM extensions.

Response Parameter	Type	Description
ListOfIdentities	IdentityListInfo	Container for the identity list

IdentityListInfo	Type	Value	Description
NoOfElements	Int	0 … n	Number of identities in the array
IdentityList	ArrayOfIdentity	SOAP Array	Array of Identity objects

Identity	Type	Value	Description
Address	String (optional)	SIP Address Max 512 characters	Public address, handle, etc
Type	String (optional)	Max 64 characters	The type of identity

The various Identity Types are shown in the following table.

Class	Type
CM extension	shortform
SIP	username e164

	msrtc otherSip
SMTP	msexchange otherSmtp
XMPP	googletalk jabber otherXmpp
IBM	ibmsametime lotusnotes

If a SIP handle matches a CM extension, only one Identity record is used and the Type is set to "shortform". That is, precedence is given to the CM extension over the SIP handle.

PPM does not return all the configured handles. PPM only returns the handles from the communication profile set that includes the handle the user entered when logging in (plus the CM extension, if it differs from all those handles).

For more information on this object, see the example at the end of the document (section 3.8 on page 131).

3.2.2.10 ListOfMaintenanceData

This object provides a remote mechanism to enable various settings on the phones for maintenance purposes. In addition to the eight possible items returned with the Maintenance Data in 6.1, 6.2 PPM supports device settings on a per location basis. This includes existing timer and Quality of Service (QoS) parameters, as well as newly added Centralized VLAN and QoS settings. Two sub pages exist on the SMGR Elements-> Session Manager-> Device and Location Configuration-> Device Settings Group page: Terminal Groups and Location Groups. These subpages, when administered, contain all maintenance, timer and device specific data for endpoints administered in the specified terminal or location group. New fields are added that allow the values for VLAN, DIFF/TOS parameters and 802.1 P/Q parameters to be entered. The new DIFF/TOS parameters, PPM will now send values for are CALL_CONTROL_PHB, AUDIO_PHB, and VIDEO_PHB. The new 802.1 P/Q parameters are CALL_CONTROL_802_PRIORITY, AUDIO_802_PRIORITY, and VIDEO_802_PRIORITY.

When the endpoint passes the “scope” parameter (which is the “Group” parameter from the phone’s Admin screen) in the getAllEndpointConfiguration request and if this Group is configured in the System Manager, PPM will use this value as the Terminal Group number and return the maintenance data values from the associated Device Settings Groups Terminal Groups page. If there was no “scope” parameter in the getAllEndpointConfiguration request or there is no data configured for the Group received by PPM, PPM will try to determine the phone’s SM location via its IP address. If PPM can map the phone’s IP address to an SM location, then PPM sends the phone all the maintenance data values configured on the Device Settings Groups Location Groups page. If neither a Group was received nor could the phone’s location

be determined by the client's IP Address, PPM will try and use the user's configured Home Location on the SMGR to determine their Device Settings. If none of these ways to determine the user's Device Settings succeed, then PPM will return the maintenance data values configured on the Device Settings Groups Default Group page.

For REMO users, only the Terminal Group, the user's configured Home Location on the SMGR or the Default Device Settings will be used. The IP address-based Device Settings cannot be used because the client's IP address (actually the IP address of SBC) cannot be used to determine the location of the client. Also, the SNMPADD field is not transformed for REMO users.

The 7.1 PPM adds three new Station Access Code items: Salt, Hash Algorithm, and Hash. These are derived from the Station Access Code field under the Maintenance heading of the Device Settings Group, and are always included in the getAllEndpoint-Configuration response.

The SMGR administration location and the default values are listed below:

MaintenanceData item	Default value	SMGR administration location
SNMPADD		Device Settings → IP Address For SNMP Queries
SNMPSTRING		Device Settings → SNMP Community
PROCPSWD		Device Settings → Station Admin Password
STATION_ACCESS_CODE_SALT		Set automatically by SMGR when Station Access Code is set
STATION_ACCESS_CODE_HASH_ALGORITHM		Set automatically by SMGR to "SHA-512" when Station Access Code is set
STATION_ACCESS_CODE_HASH		Device Settings → Station Access Code (password is encrypted)
QKLOGINSTAT	0	Device Settings → Quick Login Status
RECOVERYREGISTERWAIT	60	Device Settings → Reactive Monitoring Interval (secs)
FAST_RESPONSE_TIMEOUT	2	Device Settings → Timer B (sec)
FAILBACK	auto	Session Manager Administration → Failbacks Policy
SIGREGPROXYPOLICY	simultaneous	Not administrable
VLAN_NUMBER	0	Device Settings Groups → (Default Group OR Terminal Groups OR Location Groups)-> VLAN Parameters
CALL_CONTROL_PHB	46	Device Settings Groups → (Default Group OR Terminal Groups OR Location Groups) ->DIFFSERV/QOS Parameters
AUDIO_PHB	46	Device Settings Groups → (Default Group OR Terminal Groups OR Location Groups) -> DIFFSERV/QOS Parameters

MaintenanceData item	Default value	SMGR administration location
VIDEO_PHB	26	Device Settings Groups → (Default Group OR Terminal Groups OR Location Groups) -> DIFFSERV/QOS Parameters
CALL_CONTROL_802_PRIORITY	6	Device Settings Groups → (Default Group OR Terminal Groups OR Location Groups) ->802.1 P/Q Parameters
AUDIO_802_PRIORITY	6	Device Settings Groups → (Default Group OR Terminal Groups OR Location Groups)->802.1 P/Q Parameters
VIDEO_802_PRIORITY	5	Device Settings Groups → (Default Group OR Terminal Groups OR Location Groups) ->802.1 P/Q Parameters

The three STATION_ACCESS_CODE parameters are always included. If the administrator leaves the Station Access Code field empty, these values are sent to the endpoint as the empty string. When they are the empty string, this tells the endpoint to remove the Station Access Code. This handles the case where the administrator enters a Station Access Code, and then changes their mind and removes it. The endpoint then reverts back to using the PROCPSWD.

It should be noted that any of the above Device Settings parameters passed to the phone by PPM in the ListOfMaintenanceData will override the same parameters received by the phone from its settings file.

Response Parameter	Type	Description
ListOfMaintenanceData	MaintenanceDataInfo	Container for the maintenance data

MaintenanceDataInfo	Type	Description
NoOfElements	Integer	Number of elements in the array
MaintenanceDataList	ArrayOfMaintenanceData	Array of MaintenanceData

MaintenanceData	Type	Description
MDScope	String (optional)	Maintenance Data Scope (e.g. groups)
MDName	String	Maintenance Data Name
MDValue	String	Maintenance Data Value

3.2.2.11 ListOfNumberFormatRules

The number formatting table is a feature available in Communication Manager for configuring how numbers are displayed under certain conditions. The table specifies the extension length, the inter-location format and the intra-location format.

The number formatting table is located in CM on the dial plan parameters form for the system wide configuration.

Response Parameter	Type	Description
ListOfNumberFormatRules	NumberFormatListInfo	Container for number format rules

NumberFormatListInfo	Type	Description
NoOfElements	Integer	Number of elements in the array
NumberFormatList	ArrayOfNumberFormatRule	NumberFormatRule

NumberFormatRule	Type	Description
ExtLength	Int (optional)	Length of the extension
InterLocation	String (optional)	Format of the inter-domain number, where x represents a number and all other characters are for formatting
IntraLocation	String (optional)	Format of the intra-domain number, where x represents a number and all other characters are for formatting

3.2.2.12 ListOfOneTouchDialData

The One Touch Dial List is a set of buttons and associated addresses that will be called when the button is pressed. Note that the button location is determined from the administration of an autodial button in the station form on Communication Manager.

Response Parameter	Type	Description
ListOfOneTouchDialData	OneTouchDialListInfo	Container for the one touch dial list

OneTouchDialListInfo	Type	Description
NoOfElements	Integer	Number of elements in the array
OneTouchDialList	Array of OneTouchButton	Array of oneTouchButton elements

OneTouchButton	Type	Value	Description
ButtonLocation	Short	1-96	the location of button
Address	String		number (or general SIP URI) assigned to the button for One Touch Dial
Label	String (optional)		This is a label that can be displayed for the button. If no label has been set (via setOneTouchDialList), PPM will return a label formed from the Address.
ReadOnly	Boolean		This indicates if the one touch button is read-only or not. Read-only buttons are maintained in Communication Manager and the user may not change the address of the button.

3.2.2.13 ListOfRingerOnOffData

An endpoint can retrieve its Ringer On/Off settings with this object. This value is configurable for call-appr buttons, brdg-appr buttons, the team button and the call-pkup button. For the latter two buttons, the OnOffSet value is set using the "R:" or "Rg:" field of the button assignment on the CM station form. (The Rg field only appears for call-pkup buttons if the *Enhanced Call Pickup Alerting* system parameter is enabled.)

For call-appr appearance buttons, the OnOffSet value depends on the settings of the "Per Button Ring Control?" (PBRC) field on the CM station form.

- If the station's PBRC field is "y", then the OnOff value is based on what is administered on the call-appr button's Rg parameter.
- If the station's PBRC field is "n", then no ListOfRingerOnOffData is returned for any of the call-appr buttons.

For brdg-appr appearance buttons, the OnOffSet value depends on the settings of the "Per Button Ring Control?" (PBRC) and "Bridged Call Alerting?" (BCA) fields on the CM station form.

- If the station's PBRC field is "y", then the OnOff value is based on what is administered on the brdg-app button's R parameter.
- If the station's PBRC field is "n", the BCA field value determines the setting.
    - If the station's BCA field is "y", the OnOffSet value will be set to "on".
    - If the station's BCA field is "n", the OnOffSet value will be set to "off".

For endpoints of type CS1k-* (e.g., CS1k-IP), the PBRC field is hidden on the CM station form, because it is always enabled. This means for call-appr and brdg-appr buttons on a CS1k-* endpoint, the OnOff value is always taken from the button's Rg or R field.

Response Parameter	Type	Description
ListOfRingerOnOffData	RingerOnOffListInfo	Container for ringer on/off data

RingerOnOffListInfo	Type	Description
NoOfElements	Integer	Number of elements in the array
RingerOnOffList	Array of RingerOnOffData	Array of ringerOnOffData

RingerOnOffData	Type	Value	Description
OnOffSet	Enum	{on, off, abbreviated, delayed, icom, continuous, if-busy-silent, if-busy-single, single}	Configurable for call-appr buttons, brdg-appr buttons, the team button and call-pkup button. The text at the beginning of this section explains how the CM station form controls the setting of this field.
ButtonNumber	Short	1-96	the location of the button for this ringer on/off setting
Address	String	SIP URI	The Address of the Button
PatternSet	Enum	{half-ring, intercom-ring}	Only applies to the call-pkup button. It is the ringPattern of the button and is assigned in the "P:" field of the button assignment on the CM station form.
AutoAbbrDelayedTransInterval	Short	1-16	New to SM 6.2 Feature Pack 1. Specifies the number of ring cycles to elapse before an Automatic Abbreviated/Delayed transition is triggered for a call. For example, if the number of ring cycles is 2, and the Ring Type for the primary (or bridged call appearance) is Delayed Ringing, an incoming call

RingerOnOffData	Type	Value	Description
			would silently alert the station for 2 ring cycles and then audibly alert. Similarly, if the number of ring cycles is 2, and the Ring Type for the primary (or bridged call appearance) is Abbreviated Ringing, an incoming call would audibly alert the station for 2 ring cycles and then silently alert the station.

3.2.2.14 ListOfTimers

This provides a method for the endpoint to receive required timers. Currently the phone will receive the following timers.

Timer Name	Description	Default Value
registration	Registration timer (configurable)	3600 seconds
line-reservation	Line-reservation timer (configurable)	30 seconds
subscription	Subscription timer (configurable)	86400 seconds

Response Parameter	Type	Description
ListOfTimers	TimerListInfo	Container for list of timer information

TimerListInfo	Type	Description
NoOfElements	Int	Number of elements in the array
TimerList	ArrayOfTimer	SOAP Array of Timers

Timer	Type	Description
TimerName	String	Name of the timer
TimerValue	String	Value of the timer
Precision	String	{seconds, milliseconds}

3.2.2.15 LinePreferenceInfo

This object provides the endpoint with the Idle Line preferences as defined on CM for its call-appearances and bridged-appearances. These fields are used to specify that the selected line for incoming calls is always an idle line. If you enter y, then as the user goes off-hook, the phone will connect to an idle call appearance instead of the ringing call.

Response Parameter	Type	Value	Description
LinePreferenceInfo	LinePrefInfo		Container of line preference information

LinePrefInfo	Type	Value	Description
callAppPreference	Enum (LinePref)	{y, n}	the Idle Appearance Preference defined for this station in CM
bridgeAppPreference	Enum (LinePref)	{y, n}	the Bridged Appearance Preference defined for this station in CM

3.2.2.16 MessageWaitingExt

This object provides the endpoint with the Message Waiting Extension as defined on CM.

Response Parameter	Type	Description
MWExt	String (optional)	The message waiting extension assigned to the station on CM (Note that the Message Waiting Extension assigned on CM is generally used to light a message waiting lamp on the telephone)

3.2.2.17 MuteOnRemoteOffHook

This setting was added as part of SM 6.2 Feature Pack 1.

When Avaya one-X Communicator is operating in Shared Control Mode, the same service state is maintained on the softphone and the hard phone. Any actions performed at the one-X Communicator are reflected on the desk phone and vice versa. CM establishes the media path to the desk phone. Using this mode, the user leverages on the rich user experience provided by the one-XC client for the call control (signaling) and uses the traditional endpoints for media.

Typically, the desk phone resides inside the enterprise premises. However, it is possible for a user to be away from their desk and initiate a call on their soft phone that, because of the shared control, causes the desk phone to go off hook. This puts the desk phone in

speaker mode and the conversation can potentially be heard by other parties that are near the desk phone. In this scenario it is also possible for the user to listen in on conversations occurring near their desk phone.

Response Parameter	Type	Value	Description
MuteOnRemoteOffHook	String	{y, n}	If set to "y", then the speakerphone of the 96xx hard phone shall be muted when a call from the 1XCommunicator is made.

3.2.2.18 SoftMenuKeyList

This object is not being used.

Response Parameter	Type	Description
SoftMenuKeyList	ButtonListInfo	Container for SoftMenuKeys information

ButtonListInfo	Type	Description
NoOfElements	Integer	Number of elements in the array
ButtonList	Array of ButtonData	See ButtonData above

3.2.2.19 TerminalGroupId

This object provides the endpoint with the IP Phone Group Id that is configured on the CM/SMGR. If the IP Phone Group Id is not configured on the CM/SMGR, then PPM will return the Scope value it received in the getAllEndopointConfiguration request. If neither the IP Phone Group Id value is configured nor the Scope value was received by PPM in the gAEC Request, then PPM will return a value of 0.

Response Parameter	Type	Description
TerminalGroupId	Integer	0 through 999

3.2.2.20 VideoInfo

This object provides the endpoint with the IP Softphone Enabled flag and IP Video Enabled flag from CM.

Response Parameter	Type	Description
VideoInfo	VideoInformation	Container for features associated with a softphone

VideoInformation	Type	Value	Description
IPSoftphoneEnable	Boolean (optional)	Y/N	Enables features available with a softphone and is defined in CM.
IPVideoEnable	Boolean (optional)	Y/N	Enables video and is defined in CM.

3.2.2.21 VolumeSettings

This object provides the endpoint with the endpoint's volume settings. In 6.2 PPM supports default volume settings on a per location basis. Two sub pages exist on the Elements-> Session Manager -> Device and Location Configuration -> Device Settings Group Page: Terminal Groups and Location Groups. Volume settings for the getAllEndpointConfigurationResponse message are taken from the administered values on these pages.

When the endpoint passes the "scope" parameter (which is the "Group" parameter from the phone's Admin screen) in the getAllEndpointConfiguration request and if that Group is configured in the System Manager, PPM will use this value as the Terminal Group number and return the Volume Settings data values from the associated Device Settings Groups Terminal Groups page. If there was no "scope" parameter in the getAllEndpointConfiguration request or there is no data configured for the Group received by PPM, PPM will try and determine the phone's SM location via its IP address. If PPM can map the phone's IP address to an SM location, then PPM sends the phone all the Volume data values configured on the Device Settings Groups Location Groups page. If neither a Group was received nor could the phone's location be determined, PPM will return the Volume Settings data values configured on the Device Settings Groups Default Group page.

Volume settings set on the endpoint will take precedence over any of the settings described in the previous 2 paragraphs.

ResponseParameter	Type	Description
VolumeSettings	VolumeSet	Container for volume settings

VolumeSet	Type	Value	Description
RingerVolume	Short	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	Relative volume on a scale of 1 – 10
SpeakerVolume	Short	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	Relative volume on a scale of 1 - 10
ReceiverVolume	Short	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	Relative volume on a scale of 1 - 10

RingerCadence	Short	{1, 2, 3 ,4 ,5 ,6 ,7 , 8, 9, 10}	One from a choice of ringer cadences

3.2.2.22 VMONInformation

This object provides the endpoint with the RTCP Monitor IP Address, Port Number, and Report Period (in seconds). For REMO users, the RtcpServer value is not transformed. They are treated in the same way as Enterprise users.

ResponseParameter	Type	Description
VMONInfo	VMONInformation	Container for VMON settings

VMONInformation	Type	Value	Description
RtcpServer	String	IP address	The IP address of the RTCP Monitor
VmonPort	int	Port	The port of the RTCP Monitor
ReportPeriod	int	5-30 Seconds	How frequently the end point should be sending the receiver report

3.2.2.23 VMNumber

This object provides the endpoint with the Voice Mail Number as defined on CM.

Response Parameter	Type	Description
VMNumber	String (optional)	The voice mail extension assigned to the station on CM.

3.2.2.24 IpAddressFamilySettings

The IpAddressFamilySettings contains information about Ip address type supported by signaling mode and media mode. These values can be configured from Device Settings Groups or Location Settings on SMGR.

Response Parameter	Type	Description
IpAddressFamilySettings	IpAddressFamilySettings	Container for IpAddressFamilySettings

IpAddressFamilySettings	Type	Value	Description
SignalingMode	String	*4 or 6*	*4 indicates IPv4* *6 indicates IPv6*
MediaMode	String	4 or 6 or 46 or 64	*4 indicates IPv4* *6 indicates IPv6* *46 indicates Dual with IPv4 preferred* *64 indicates Dual with IPv6 preferred*

getAllEndpointConfiguration(Handle, Fields)

Example getAllEndpointConfiguration request without specifying Fields:

```
<ns1:getAllEndpointConfiguration
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
  <Scope>123</Scope>
</ns1:getAllEndpointConfiguration>
```

PPM response:

```
<ns1:getAllEndpointConfigurationResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ConfigInfo>
    <VolumeSettings>
      <RingerVolume>5</RingerVolume>
      <ReceiverVolume>5</ReceiverVolume>
      <SpeakerVolume>5</SpeakerVolume>
      <RingerCadence>3</RingerCadence>
    </VolumeSettings>
    <ListOfRingerOnOffData>
      <NoOfElements>2</NoOfElements>
      <RingerOnOffDataList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:RingerOnOffData[3]" xsi:type="soapenc:Array">
       <item>
          <Address>76011</Address>
          <ButtonNumber>4</ButtonNumber>
          <OnOffSet>delayed</OnOffSet>
          <AutoAbbrDelayedTransInterval>2</AutoAbbrDelayedTransInterval >
        </item>
        <item>
          <Address />
          <ButtonNumber>9</ButtonNumber>
          <OnOffSet>single</OnOffSet>
          <Pattern>half-ring</Pattern>
        </item>
        <item>
          <Address>70005</Address>
          <ButtonNumber>10</ButtonNumber>
          <OnOffSet>on</OnOffSet>
        </item>
       </RingerOnOffDataList>
    </ListOfRingerOnOffData>
    <ListOfTimers>
      <NoOfElements>4</NoOfElements>
      <TimerList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="ns1:Timer[4]"
xsi:type="soapenc:Array">
        <item>
```

```
        <Precision>seconds</Precision>
        <TimerName>registration</TimerName>
        <TimerValue>3600</TimerValue>
       </item>
       <item>
        <Precision>seconds</Precision>
        <TimerName>line-reservation</TimerName>
        <TimerValue>30</TimerValue>
       </item>
       <item>
        <Precision>seconds</Precision>
        <TimerName>subscription</TimerName>
        <TimerValue>86400</TimerValue>
       </item>
     </TimerList>
   </ListOfTimers>
   <LinePreferenceInfo>
     <callAppPreference>n</callAppPreference>
     <bridgeAppPreference>n</bridgeAppPreference>
   </LinePreferenceInfo>
   <MWExt>70004</MWExt>
   <AutoAnswer>none</AutoAnswer>
   <ListOfOneTouchDialData>
     <NoOfElements>1</NoOfElements>
     <OneTouchDialList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:OneTouchButton[1]" xsi:type="soapenc:Array">
       <item>
        <ButtonLocation>25</ButtonLocation>
        <Address>70005</Address>
        <Label />
        <ReadOnly>true</ReadOnly>
       </item>
     </OneTouchDialList>
   </ListOfOneTouchDialData>
   <ListOfButtonAssignments>
     <NoOfElements>9</NoOfElements>
     <ButtonAssignment xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:ButtonData[8]" xsi:type="soapenc:Array">
       <item>
        <Location>1</Location>
        <ButtonType>call-appr</ButtonType>
        <Label>70004</Label>
        <LineID>1</LineID>
        <Address />
        <FNUType />
        <App>false</App>
        <Media>false</Media>
        <FNUInfo soapenc:arrayType="ns1:FNUData[0]" xsi:type="soapenc:Array" />
       </item>
       <item>
        <Location>2</Location>
        <ButtonType>call-appr</ButtonType>
        <Label>70004</Label>
        <LineID>2</LineID>
```

```
      <Address />
      <FNUType />
      <App>false</App>
      <Media>false</Media>
      <FNUInfo soapenc:arrayType="ns1:FNUData[0]" xsi:type="soapenc:Array" />
    </item>
    <item>
      <Location>3</Location>
      <ButtonType>call-appr</ButtonType>
      <Label>70004</Label>
      <LineID>3</LineID>
      <Address />
      <FNUType />
      <App>false</App>
      <Media>false</Media>
      <FNUInfo soapenc:arrayType="ns1:FNUData[0]" xsi:type="soapenc:Array" />
    </item>
    <item>
      <Location>4</Location>
      <ButtonType>FNU</ButtonType>
      <Label>76011</Label>
      <LineID>0</LineID>
      <Address>76011</Address>
      <FNUType>avaya-cm-fnu=team</FNUType>
      <App>true</App>
      <Media>true</Media>
      <PickupByGoingOffHook>true</PickupByGoingOffHook>
      <SilentIfActive>false</SilentIfActive>
      <SACCFOverride>ask</SACCFOverride>
      <FNUInfo soapenc:arrayType="ns1:FNUData[3]" xsi:type="soapenc:Array">
        <item>
          <Parameter>avaya-cm-action=</Parameter>
          <Required>true</Required>
          <MustComplete>pickupORspeeddial</MustComplete>
        </item>
        <item>
          <Parameter>avaya-cm-destination=</Parameter>
          <Required>false</Required>
          <MustComplete>address</MustComplete>
        </item>
        <item>
          <Parameter>avaya-call-id=</Parameter>
          <Required>false</Required>
          <MustComplete>call_id</MustComplete>
        </item>
        <item>
          <Parameter>avaya-cm-redirection-override=</Parameter>
          <Required>false</Required>
          <MustComplete>offon</MustComplete>
        </item>
      </FNUInfo>
    </item>
    <item>
      <Location>5</Location>
      <ButtonType>FNU</ButtonType>
      <Label>Busy Indicator</Label>
```

```
      <LineID>0</LineID>
      <Address />
      <FNUType>avaya-cm-fnu=busy-indicator</FNUType>
      <App>true</App>
      <Media>true</Media>
      <FNUInfo soapenc:arrayType="ns1:FNUData[1]" xsi:type="soapenc:Array">
        <item>
          <Parameter>avaya-cm-destination=71000</Parameter>
          <Required>true</Required>
          <MustComplete>address</MustComplete>
        </item>
      </FNUInfo>
    </item>
    <item>
      <Location>6</Location>
      <ButtonType>FNU</ButtonType>
      <Label>Hunt Busy</Label>
      <LineID>0</LineID>
      <Address />
      <FNUType>avaya-cm-fnu=hunt-group-busy-position</FNUType>
      <App>true</App>
      <Media>false</Media>
      <FNUInfo soapenc:arrayType="ns1:FNUData[2]" xsi:type="soapenc:Array">
        <item>
          <Parameter>avaya-cm-group=1234</Parameter>
          <Required>true</Required>
          <MustComplete>none</MustComplete>
        </item>
        <item>
          <Parameter>avaya-cm-action=</Parameter>
          <Required>true</Required>
          <MustComplete>offon</MustComplete>
        </item>
      </FNUInfo>
    </item>
    <item>
      <Location>9</Location>
      <ButtonType>FNU</ButtonType>
      <Label>Call Pickup</Label>
      <LineID>0</LineID>
      <Address />
      <FNUType>avaya-cm-fnu=call-pickup</FNUType>
      <App>true</App>
      <Media>true</Media>
      <FNUInfo soapenc:arrayType="ns1:FNUData[0]" xsi:type="soapenc:Array" />
    </item>
    <item>
      <Location>10</Location>
      <ButtonType>brdg-appr</ButtonType>
      <Label>70005</Label>
      <LineID>1</LineID>
      <Address>70005</Address>
      <FNUType />
      <App>false</App>
      <Media>false</Media>
      <FNUInfo soapenc:arrayType="ns1:FNUData[0]" xsi:type="soapenc:Array" />
```

```
</item>
<item>
  <Location>25</Location>
  <ButtonType>autodial</ButtonType>
  <Label>70005</Label>
  <LineID>0</LineID>
  <Address>70005</Address>
  <FNUType />
  <App>false</App>
  <Media>false</Media>
  <FNUInfo soapenc:arrayType="ns1:FNUData[0]" xsi:type="soapenc:Array" />
</item>
<item>
  <Location>11</Location>
  <ButtonType>PUB</ButtonType>
  <Label>Service Observe</Label>
  <PUBType>sip-sobsrv</PUBType>
  <ListenOnly>false</ListenOnly>
  <Coaching>false</Coaching>
 <PUBPayload>
   <SIPEvent>avaya-cm-cc-info</SIPEvent>
   <CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" state="partial" version="1" />
   <clientPublish>
     <nativeEntity>$Required</nativeEntity>
     <requestedSO>
       <soType>$Required</soType>
       <soEntity>$Required</soEntity>
       <location>$Required</location>
       <mode>$Required</mode>
     </requestedSO>
   </clientPublish>
 </PUBPayload>
</item>
<item>
  <Location>12</Location>
  <ButtonType>PUB</ButtonType>
  <Label>Add/Remove Skill</Label>
  <PUBType>add-rem-sk</PUBType>
  <PUBPayload>
    <SIPEvent>avaya-cm-cc-info</SIPEvent>
   <CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" state="partial" version="1" />
    <clientPublish>
      <nativeEntity>$Required</nativeEntity>
      <requestedSkillChg>
        <agentID>$Required</agentID>
        <action>$Required</action>
        <skill>$Required</skill>
        <skillLevel>$Required</skillLevel>
      </requestedSkillChg>
    </clientPublish>
  </PUBPayload>
</item>
<item>
  <Location>13</Location>
  <ButtonType>PUB</ButtonType>
  <Label>VOA Repeat</Label>
  <PUBType>voa-repeat</PUBType>
```

```
        <PUBPayload>
          <SIPEvent>avaya-cm-cc-info</SIPEvent>
          <CCInfoEvent xmlns="http://www.avaya.com/schemas/ccinfo.xsd" state="partial" version="1" />
          <clientPublish>
            <nativeEntity>$Required</nativeEntity>
            <requestedVOA />
          </clientPublish>
        </PUBPayload>
      </item>
    </ButtonAssignment>
  </ListOfButtonAssignments>
  <DialPlanData>
    <DialPlanDomain>avaya.com</DialPlanDomain>
    <NoOfElements>10</NoOfElements>
    <DialPlan xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="xsd:string[10]"
xsi:type="soapenc:Array">
      <item>#</item>
      <item>*xx</item>
      <item>1xx</item>
      <item>2xxxx</item>
      <item>3xxxx</item>
      <item>4xxxx</item>
      <item>5xxxx</item>
      <item>6xxxx</item>
      <item>7xxxx</item>
      <item>8xxxx</item>
    </DialPlan>
  </DialPlanData>
  <VMONInfo>
    <RtcpServer />
    <VmonPort>5005</VmonPort>
    <ReportPeriod>5</ReportPeriod>
  </VMONInfo>
  <VideoInfo>
    <IPSoftphoneEnable>false</IPSoftphoneEnable>
    <IPVideoEnable>false</IPVideoEnable>
  </VideoInfo>
  <ListOfMaintenanceData>
    <NoOfElements>15</NoOfElements>
    <MaintenanceDataList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:MaintenanceData[15]" xsi:type="soapenc:Array">
      <item>
        <MDName>STATION_ACCESS_CODE_SALT</MDName>
        <MDValue />
      </item>
      <item>
        <MDName> STATION_ACCESS_CODE_HASH_ALGORITHM</MDName>
        <MDValue />
      </item>
      <item>
        <MDName> STATION_ACCESS_CODE_HASH </MDName>
        <MDValue />
      </item>
      <item>
```

```
        <MDName>QKLOGINSTAT</MDName>
        <MDValue>0</MDValue>
       </item>
       <item>
        <MDName>RECOVERYREGISTERWAIT</MDName>
        <MDValue>60</MDValue>
       </item>
       <item>
        <MDName>FAILBACK_POLICY</MDName>
        <MDValue>auto</MDValue>
       </item>
       <item>
        <MDName>FAST_RESPONSE_TIMEOUT</MDName>
        <MDValue>2</MDValue>
       </item>
       <item>
        <MDName>SIPREGPROXYPOLICY</MDName>
        <MDValue>simultaneous</MDValue>
       </item>
       <item>
        <MDName>VLAN_NUMBER</MDName>
        <MDValue>0</MDValue>
       </item>
       <item>
        <MDName>CALL_CONTROL_PHB</MDName>
        <MDValue>46</MDValue>
       </item>
       <item>
        <MDName>AUDIO_PHB</MDName>
        <MDValue>46</MDValue>
       </item>
       <item>
        <MDName>VIDEO_PHB</MDName>
        <MDValue>26</MDValue>
       </item>
       <item>
        <MDName>CALL_CONTROL_802_PRIORITY</MDName>
        <MDValue>6</MDValue>
       </item>
       <item>
        <MDName>AUDIO_802_PRIORITY</MDName>
        <MDValue>6</MDValue>
       </item>
       <item>
        <MDName>VIDEO_802_PRIORITY</MDName>
        <MDValue>5</MDValue>
       </item>
     </MaintenanceDataList>
    </ListOfMaintenanceData>
    <ListOfNumberFormatRules>
     <NoOfElements>2</NoOfElements>
     <NumberFormatList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:NumberFormatRule[2]" xsi:type="soapenc:Array">
       <item>
        <ExtLength>6</ExtLength>
```

```
        <InterLocation>xx.xx.xx</InterLocation>
        <IntraLocation>xx.xx.xx</IntraLocation>
       </item>
       <item>
        <ExtLength>7</ExtLength>
        <InterLocation>xx-xxxxx</InterLocation>
        <IntraLocation>xx-xxxxx</IntraLocation>
       </item>
     </NumberFormatList>
   </ListOfNumberFormatRules>
   <ListOfIdentities>
     <NoOfElements>1</NoOfElements>
     <IdentityList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:Identity[5]" xsi:type="soapenc:Array">
       <item>
        <Address>70004</Address>
        <Type>shortform</Type>
       </item>
       <item>
        <Address>otherSip70004@yahoo.com</Address>
        <Type>otherSip</Type>
       </item>
       <item>
        <Address>xchg70004@xchg.com</Address>
        <Type>msexchange</Type>
       </item>
       <item>
        <Address>yahoo70004@yahoo.com</Address>
        <Type>otherSip</Type>
       </item>
       <item>
        <Address>sip:ocs70004@ocssip.com</Address>
        <Type>msrtc</Type>
       </item>
</IdentityList>
   </ListOfIdentities>
   <ListOfConfigDataPacketVersions>
     <NoOfElements>1</NoOfElements>
     <ConfigDataPacketVersionsList
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:ConfigDataPacketVersions[1]" xsi:type="soapenc:Array">
       <item>
        <PacketName>ListOfContacts</PacketName>
        <PacketVersion>2009-05-05 21:23:49.832659</PacketVersion>
       </item>
     </ConfigDataPacketVersionsList>
   </ListOfConfigDataPacketVersions>
   <VMNumber />
<ListOfEmergencyNumbers>
     <NoOfElements>1</NoOfElements>
     <EmergencyNumberList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:EmergencyNumberData[1]" xsi:type="soapenc:Array">
       <item>
```

```
        <Type>Police</Type>
        <Number>911</Number>
       </item>
     </ EmergencyNumberList>
   </ListOfEmergencyNumbers>
   <MuteOnRemoteOffHook>n</MuteOnRemoteOffHook>
   <ListOfCmSystemParameters>
     <NoOfElements>2</NoOfElements>
     <CmSystemParameters xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:CmSystemParameter[2]" xsi:type="soapenc:Array">
      <item>
       <Name>MuteOnRemoteOffHook</Name>
       <Value>n</Value>
      </item>
      <item>
       <Name>TransferUponHangup</Name>
       <Value>y</Value>
      </item>
     </CmSystemParameters>
  </ListOfCmSystemParameters>
  <IpAddressFamilySettings>
         <SignalingMode>4</SignalingMode>
         <MediaMode>6</MediaMode>
  </IpAddressFamilySettings>
  <TerminalGroupId>5</TerminalGroupId>
</ConfigInfo>
</ns1:getAllEndpointConfigurationResponse>
```

Example getAllEndpointConfiguration request using the Fields:

```
<ns1:getAllEndpointConfiguration
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
  <Fields xsi:type="ns:ArrayOfEndpointConfigurationFields"
soapenc:arrayType="ns:EndpointConfigurationFields[2]">
    <item>ListOfRingerOnOffData</item>
    <item>VolumeSettings</item>
    <item>TerminalGroupId</item>
  </Fields>
  <Scope>321</Scope>
</ns1:getAllEndpointConfiguration>
```

PPM response:

```
<ns1:getAllEndpointConfigurationResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ConfigInfo>
   <VolumeSettings>
    <RingerVolume>5</RingerVolume>
    <ReceiverVolume>5</ReceiverVolume>
    <SpeakerVolume>5</SpeakerVolume>
    <RingerCadence>3</RingerCadence>
   </VolumeSettings>
```

```
   <ListOfRingerOnOffData>
     <NoOfElements>2</NoOfElements>
     <RingerOnOffDataList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:RingerOnOffData[2]" xsi:type="soapenc:Array">
       <item>
         <Address>70005</Address>
         <ButtonNumber>10</ButtonNumber>
         <OnOffSet>On</OnOffSet>
       </item>
     </RingerOnOffDataList>
   </ListOfRingerOnOffData>
   <TerminalGroupId>5</TerminalGroupId>
  </ConfigInfo>
</ns1:getAllEndpointConfigurationResponse>
```

For more explanation on the ListOfIdentities element, see the explanation in section 3.7 Identities List.

E**rror Conditions/Faults:**
The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
“DataNotAvailable ”	PPM is unable to obtain expected data.
“ConfigurationMismatch”	The station has not been administered on CM.
“NotIMUser”	Requesting user does not exist in the Enterprise database
“InternalError”	Unexpected exception in PPM processing
“DatabaseNotAccessible”	There was a problem with the database

3.3 DEVICE DATA

The two Device Data requests (one for Identity data and one for Config/XML data) are designed to store data pertaining to a particular Avaya AST endpoint or model of Avaya AST endpoint.

An endpoint should only send a setDeviceData Config/XML request if its internally configured data differs from the data received in a prior getDeviceDataResponse.

Since SM 6.3 FP4 and the coupling of PPM and IPTCM with respect to Device Data / Profile Settings, the DeviceIdentity must be restricted to DeviceVendor + DeviceModel. The reason is as follows: Even though SIP endpoints can provide all five DeviceIdentity values just listed, IPTCM can only provide the DeviceVendor and DeviceModel values.

getDeviceData or setDeviceData PPM request expects ONLY one of the following as Identities:

Device Vendor + Device Model:

```
<Identity xsi:type="ns:DeviceIdentity">
        <DeviceVendor>Avaya</DeviceVendor>
        <DeviceModel>96xx</DeviceModel>
 </Identity>
```

DeviceHandle:

```
<Identity xsi:type="ns:DeviceIdentity">
        <DeviceHandle>02:03:04:05:06:ab</DeviceHandle>
 </Identity>
```

Blank Identity:

```
<Identity xsi:type="ns:DeviceIdentity">
</Identity>
```

Currently PPM gives preference to DeviceHandle if the user sends both DeviceHandle and DeviceVendor + DeviceModel. So, the endpoints should send only one of the above mentioned DeviceIdentity values. Identity can be left blank or not specified if the SIP Endpoint wants to retrieve/store data common to all of that user's devices, per the DataCategory and DataName specified in the request.

Currently there are three data categories: ‘public’, ‘Volume’ and ‘Config’. The data category ‘public’ has five data elements: ‘deviceVendor’, ‘deviceModel’, ‘contactURI’, ‘deviceVersion’, ‘deviceType’. The data category ‘Config’ has one data element ‘XML’. The data category ‘Volume’ has four data elements, and is covered under section 3.4.4 setVolumeSettings.

For a DataCategory of “Config" and a DataName of "XML", the value of the DataValue field in the past was opaque to PPM. Meaning that PPM had no understanding of the data contained in it, especially given that PPM did not attempt to validate those contents as this data is meaningful only to the client that populated it. However in SM 8.1.1, as a remedy to customer escalations, it was decided that PPM would break the Config XML Blob into its individual name/value pairs and store them that way. Additionally, this was

done so that common Device Data parameters could be updated (as part of the setDeviceData operation) and shared (as part of the getDeviceData operation) by the various different SIP Endpoint types that an end user has. The SMGR Administrator also wanted to be able to update and view these same common Device Data parameters (AwayTimer, AwayTimerValue, ButtonClicksEnabled, CallPickupIndication, CurrentLogo, DefaultAudioPath, EffectOfRedialButton, Favorites, HeadsetSignaling, PersonalLabels, PersonalRingToneWave, PhoneEditedDialing, PhoneScreenWidth, ShowPhoneScreenOnCall, ShowQuickTouchPanel, TimeFormat). It should be of further note that some of these common Device Data parameters (Favorites, PersonalLabels, and TimeFormat) require special handling before they can be returned to a SIP Endpoint or the SMGR Adminstrator GUI, or when they are updated by a SIP Endpoint or the SMGR Administrator GUI.

Please also see section 3.3.3 SIP Endpoint Type to Device Model Family Mapping, for additional information on PPM's Device Data operations.

3.3.1 getDeviceData

This PPM operation allows an endpoint to retrieve its Device Data. A user can retrieve the Config XML Device Data for any of its devices with an Identity of Device Vender plus Device Model, and for its single global device with no Identity, should that data exist.

getDeviceData(Handle, Identity, DataCategory, DataName)

Request Parameters	Type	Length / Value	Description
Handle	String	URI max 256 characters	The handle of the user in the form of "user@domain". Also see the note in Section 3.8 Handles.
Identity	DeviceIdentity (optional)		The device identity, supports storing information for an entire class of device – see below
DataCategory	String (optional)	Category name, max 128 characters	Name of a category of information
DataName	String (optional)	Parameter name, max 128 characters	The name of the generic data to retrieve, DataCategory cannot be null if DataName is populated

DeviceIdentity	Type	Length / Value	Description

DeviceHandle	String (optional)	max 128 characters	Type, e.g. a unique handle to assign to a device. This handle is usually the MAC address of the device. Note: The MAC address format must be 6 pairs of hexadecimal digits separated by either colons or hyphens. Example: 00-01-02-0a-0b-0c.
DeviceVendor	String (optional)	max 128 characters	Vendor of the phone, e.g. Avaya
DeviceModel	String (optional)	max 128 characters	Model of the device, e.g. 96xx, 96x1, CS1k-IP, J100, H1xx, A175
DeviceVersion	String (not supported)	max 128 characters	Version of the phone, e.g. R2.5
DeviceType	String (optional)	max 128 characters	Type, e.g. telephone

getDeviceDataResponse(DeviceDataListInfo)

Response Parameter	Type	Description
DeviceDataListInfo	ListofDeviceData	Container element for the array of device data

ListofDeviceData	Type	Description
NumOfElements	Integer	Number of elements in the array
DeviceIdentityList	ArrayOfDeviceIdentityInfo	Array of DeviceData data types

DeviceIdentityInfo	Type	Description
Identity	DeviceIdentity	Identity of the endpoint
NumOfElements	Integer	Number of elements in device data list
DeviceDataList	ArrayOfDeviceData	Array of device data types

DeviceData	Type	Description
DataCategory	String	Category of the data

DataName	String	Name of the data parameter
DataValue	String	Value of the data, max 64,000 characters

Example getDeviceData request:

```
<ns1:getDeviceData xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
  <Identity>
    <DeviceType>one-X Communicator</DeviceType>
    <DeviceVendor>Avaya</DeviceVendor>
    <DeviceModel>96x1</DeviceModel>
  </Identity>
  <DataCategory>Config</DataCategory>
  <DataName>XML</DataName>
</ns1:getDeviceData>
```

Example getDeviceData Response:

```
<ns1:getDeviceDataResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <DeviceDataListInfo>
    <NoOfElements>1</NoOfElements>
    <DeviceIdentityList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:DeviceIdentityInfo[1]" xsi:type="soapenc:Array">
      <item>
        <Identity>
          <DeviceType>one-X Communicator</DeviceType>
          <DeviceVendor>Avaya</DeviceVendor>
          <DeviceModel>96x1</DeviceModel>
        </Identity>
        <NoOfElements>1</NoOfElements>
        <DeviceDataList soapenc:arrayType="ns1:DeviceData[1]" xsi:type="soapenc:Array">
          <item>
            <DataCategory>Config</DataCategory>
            <DataName>XML</DataName>
            <DataValue>&lt;ConfigData xmlns="http://xml.avaya.com/endpointAPI"&gt;
&lt;version&gt;1273866296&lt;/version&gt;
&lt;parameter&gt;
&lt;name&gt;ShowPhoneScreenOnCall&lt;/name&gt;
&lt;alias/&gt;
&lt;value&gt;0&lt;/value&gt;
&lt;/parameter&gt;
&lt;parameter&gt;
&lt;name&gt;ShowPhoneScreenOnAlert&lt;/name&gt;
&lt;alias/&gt;
&lt;value&gt;1&lt;/value&gt;
&lt;/parameter&gt;
&lt;parameter&gt;
&lt;name&gt;DisplayCallTimers&lt;/name&gt;
&lt;alias/&gt;
&lt;value&gt;1&lt;/value&gt;
```

```
&lt;/parameter&gt;
&lt;parameter&gt;
&lt;name&gt;UseVisualAlerting&lt;/name&gt;
&lt;alias/&gt;
&lt;value&gt;1&lt;/value&gt;
&lt;/parameter&gt;
&lt;parameter&gt;
&lt;name&gt;CallPickupRingType&lt;/name&gt;
&lt;alias/&gt;
&lt;value&gt;1&lt;/value&gt;
&lt;/parameter&gt;
&lt;parameter&gt;
&lt;name&gt;CallPickupIndication&lt;/name&gt;
&lt;alias/&gt;
&lt;value&gt;3&lt;/value&gt;
&lt;/parameter&gt;
&lt;parameter&gt;
&lt;name&gt;ProvideEditedDialing&lt;/name&gt;
&lt;alias/&gt;
&lt;value&gt;2&lt;/value&gt;
&lt;/parameter&gt;
&lt;/ConfigData&gt;</DataValue>
        </item>
      </DeviceDataList>
    </item>
  </DeviceIdentityList>
 </DeviceDataListInfo>
</ns1:getDeviceDataResponse>
```


Error Conditions/Faults:

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
“NotIMUser”	Requesting user does not exist in the Enterprise database
“InvalidValue”	String length exceeded maximum
“DatabaseNotAccessible”	There was a problem with the database
“InternalError”	Unexpected exception in PPM processing.

3.3.2 setDeviceData

This PPM operation allows an endpoint to save its Device Data. A user can save Config XML Device Data for 5 devices with an Identity of Device Vender plus Device Model and/or 1 global device with no Identity. PPM also allows the user to save Identity Device Data for 5 devices with the device's Mac Address and/or save Device Data for 1 global device with no Identity. This Device Data is stored mostly agnosticly whether the data is Identity or Config XML Device Data. The reason that it is not stored "completely" agnosticly is because the sharing of common Device Data between all of a user's different SIP Endpoints which happened in SM 8.1.1, which caused PPM to need to be able to understand and manipulate a small set of these Device Data parameters. The first time an endpoint sets data for a device identity, it creates the identity in the system.

In SM 6.2 Feature Pack 4, a new capability was added on the System Manager so that the System Administrator could update the values of a limited set of Device Data parameters for one or more users at a time. In SM 8.1.1, additional functionality was added which made this limited set of Device Data parameters, common amongst all of a user's different SIP Endpoints.

setDeviceData(Handle, Identity, DeviceDataList)

Request Parameters	Type	Length / Value	Description
Handle	String	URI max 256 characters	The handle of the user in the form of "user@domain". Also see the note in Section 3.8 Handles.
Identity	DeviceIdentity (see getDeviceData operation)		The device identity, supports storing information for an entire class of device.
DeviceDataList	Array	ArrayOfDeviceData	List of DeviceData (see the getDeviceData operation)

DeviceData	Type	Length / Value	Description
DataCategory	String	max 128 characters	Category of the opaque data
DataName	String	max 128 characters	Name of the opaque data
DataValue	String	max 64,000 characters	The opaque data

setDeviceDataResponse(Result)

Response Parameter	Type	Description
Result	String	PPM_Success or fault message

Example setDeviceData request:

```
<ns1:setDeviceData xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
  <Identity>
    <DeviceHandle>00:ff:88:5a:e2:89</DeviceHandle>
  </Identity>
  <DeviceDataList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:DeviceData[5]" xsi:type="soapenc:Array">
    <item>
      <DataCategory>public</DataCategory>
      <DataName>deviceVendor</DataName>
      <DataValue>Avaya</DataValue>
    </item>
    <item>
      <DataCategory>public</DataCategory>
      <DataName>deviceType</DataName>
      <DataValue>one-X Communicator</DataValue>
    </item>
    <item>
      <DataCategory>public</DataCategory>
      <DataName>deviceModel</DataName>
      <DataValue>96x1</DataValue>
    </item>
    <item>
      <DataCategory>public</DataCategory>
      <DataName>deviceVersion</DataName>
      <DataValue>R6.0000-Beta-21376</DataValue>
    </item>
    <item>
      <DataCategory>public</DataCategory>
      <DataName>contactURI</DataName>
      <DataValue>70004@135.9.36.148</DataValue>
    </item>
  </DeviceDataList>
</ns1:setDeviceData>
```

Example setDeviceData Response:

```
<ns1:setDeviceDataResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <PPMResponse>PPM_Success</PPMResponse>
</ns1:setDeviceDataResponse>
```

Error Conditions/Faults:
The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
"InvalidValue"	String length exceeded maximum
"DeviceDataLimitExceeded"	Value exceeded maximum data items.
"InternalError"	Unexpected exception in PPM processing
"NotIMUser"	Requesting user does not exist in the Enterprise database
"DatabaseError"	PPM cannot access the database for reading or writing
"DatabaseNotAccessible"	There was a problem with the database

3.3.3 SIP Endpoint Type to Device Family Mapping

To keep the storage of Avaya SIP Endpoint Device Data to a minimum and to promote the sharing of Family-specific Device Data across SIP Endpoints within the same Device Family, SIP Endpoints, PPM, and the SMGR SM Element Manager convert each Endpoint's specific Device Model to a generic Device Family string. In the following table, the mapping of individual SIP Endpoint Models to generic Device Family strings is shown. It should be noted that some SIP Endpoint Models are not mapped to generic Device Family strings, meaning that every Endpoint model in that family has its own set of Family-specific Device Data in PPM. However, there is still sharing of common Device Data between Endpoints within that family, as well as outside of that family.

CM Templates supported	SMGR Release CM Template Support Began	Generic Device Model Mapping	Release in which Device Model Mapping began
9600SIP, 9620SIP, 9630SIP, 9640SIP, 9650SIP	6.0	96xx	N/A
9608SIP(CC), 9611SIP(CC), 9621SIP(CC), 9641SIP (CC)	6.0	96x1	6.3 FP4
J129, J169(CC), J179(CC)	8.0	J100	8.0.1
CS1k-IP, CS1k-1col, CS1k-2col, CS1k-39xx, CS1k-ana	8.0	No mapping done. The actual CS1k Endpoint types are used.	N/A

3.4 PROXY AND SERVER INFORMATION

3.4.1 getHomeCapabilities

The getHomeCapabilities request allows an endpoint to discover various capabilities about that particular server. This method is used for discovering the list of servers in the endpoint's proxy list and provides information to the endpoint on how they can use them.

getHomeCapabilites(Handle)

Request Parameter	Type	Length / Value	Description
Handle	String	URI max 256 characters	The handle of the user in the form of "user@domain". Also see the note in Section 3.8 Handles.

getHomeCapabilitesResponse(ServerCapabilites)

Response Parameter	Type	Description
ServerCapabilities	CapabilitiesList	Container element for the server capabilities

CapabilitiesList	Type	Description
ServicesList	ArrayOfServiceData	
NoOfServiceElements	int	Number of elements in ServicesData array
FNUList	ArrayOfFNUFeatures	Array of FNUFeature types
NoOfFNUElements	int	Number of elements in the FNUFeature array

ServiceData	Type	Description
ServiceName	String	The name of the service. It can be one of "endpoint-reflection", "ppm-features", "proxy-server", "presence-services" or "sm-global-features".
ServiceURI	String	URL of the service if available. For Enterprise users, the URL will include the Session Manager IP address or FQDN. For Presence Services, the URL will be an IP

ServiceData	Type	Description
		address or FQDN. For REMO users, the primary proxy server IP address will be transformed as per Section 6: REMO USER Support. The secondary and survivable server IP address will be transformed as well if they are configured.
ServiceFQDN	String	The FQDN for the SIP Proxy service if available. This field is populated starting in 8.0
ServiceDomain	String	The SIP domain as provisioned in SMGR admin. The endpoint should always send the register for user@domain, where domain is the SIP domain and not the hostname of the SIP server.
PPMServiceURI	String	The URL of the PPM Service if available. This information will be available only for "proxy-server" ServiceName. If the user is a REMO user, then the IP address in the URL will be transformed as per Section 6: REMO User Support.
PPMServiceFQDN	String	The FQDN for the PPM service if available. This information will be available only for "proxy server" ServiceName. This field is populated starting in 8.0.
ServiceURIV6	String	Same as ServiceURI, but an IPv6 address
PPMServiceURIV6	String	Same as PPMServiceURI, but an IPv6 address
ServiceType	String	SM Server Type, value can be: {CoreSM, BranchSM, SurvivabilityServer, PS, Other} Any SipEntity which is not CoreSM, BranchSM, SurvivabilityServer or PS would be of type Other, where CoreSM = Core Session Manager, BranchSM=Branch Session Manager PS=Presence Services
ServiceTransport	ArrayOfTransportData (optional)	Transport information, see the data type definition under getHomeServer for additional information
NoOfTransportElements	Int	Number of elements in ServiceTransport array

ServiceData	Type	Description
ServiceTransportV6	ArrayOfTransportData (optional)	Same as ServiceTransport, but for IPv6
NoOfTransportElementsV6	int	Number of elements in ServiceTransportV6 array
ServiceVersion	String	Versioning tag for the service. This information is only available for the server that responded to the original request. For other ServiceData entries returned in the response, this field is set to "unknown".
ServiceFeatures	ArrayOfFeatureData	Array of 0 or more feature data elements
NoOfFeatureElements	int	Number of elements in FeatureData array

FeatureData	Type	Description
FeatureName	String	The name of the feature supported
FeatureVersion	Integer (optional)	Versioning tag for the feature
FeatureValue	String (optional)	Value for a feature

FNUFeature	Type	Description
FNUType	String	See FNU Description below
App	Boolean	See FNU Description below
Media	Boolean	See FNU Description below
CMVersion	String (optional)	Current CM version
FNUInfo	Array of FNUData	See FNU Description below
NoOfFNUDataElements	int	Number of elements in FNUData array

3.4.1.1 FNU Description

The following table shows a list of features accessible via FNU in Communication Manager. These particular FNUs are sent with every getHomeCapabilities response. The phone can choose which items to display.

See the getAllEndpointConfiguration response for more information about FNUs and a complete list of available items. Note that the call-park and call-pickup-extended FNUs are also available as administrable features in the Communication Manager.

Feature Name	FNUType	App	Media	FNUInfo		
				Parameter	Required	MustComplete
Agent Logout	avaya-cm-fnu=agent-logout	false	false	avaya-cm-agent-id={extension} avaya-cm-agent-logout-reason-code={code}	true false	extension code
Off-hook	avaya-cm-fnu=off-hook	true	false			
Transfer to voice mail	avaya-cm-fnu=transfer-to-voicemail	false	false			

As of SM 6.2 Feature Pack 4, for the Transfer to Voicemail non-Button related FNU to be sent by PPM to a SIP Endpoint, that user must have had VM setup in at least one of two ways:

- On the System Manager: select **Home -> Users -> User Management -> Manage Users -> select a user -> Edit -> select the Communication Profile Tab ->** go to the **CM Endpoint Profile** part of the screen, and configure a Voice Mail number for the user.
- Coverage Path #1is configured on the Station form of the Communication Manager or on the related System Manager screen: select **Home -> Users -> User Management -> Manage Users -> select a user -> Edit -> select the Communication Profile Tab -> go to the CM Endpoint Profile part of the screen -> select the Endpoint Editor -> select the General Options** tab.

Note: Prior to SM 6.2 Feature Pack 4, the Transfer to Voicemail non-button related FNU was always sent to SIP Endpoints, regardless of whether that station/user had voicemail configured or not.

Example getHomeCapabilities request:

```
<ns1:getHomeCapabilities
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
</ns1:getHomeCapabilities>
```

Example PPM Response, showing one SM with IPv4 and IPv6, and one with just IPv4:

```
<ns1:getHomeCapabilitiesResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ServerCapabilities>
    <ServicesList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:ServiceData[3]" xsi:type="soapenc:Array">
      <item>
        <ServiceName>endpoint-reflection</ServiceName>
        <ServiceURI>1.2.3.4</ServiceURI>
        <NoOfTransportElements>0</NoOfTransportElements>
        <ServiceVersion>1</ServiceVersion>
```

```
    <ServiceFeatures soapenc:arrayType="ns1:FeatureData[0]" xsi:type="soapenc:Array" />
    <NoOfFeatureElements>0</NoOfFeatureElements>
  </item>
  <item>
    <ServiceName>sm-global-features</ServiceName>
    <ServiceVersion>1</ServiceVersion>
    <ServiceFeatures soapenc:arrayType="ns1:FeatureData[1]" xsi:type="soapenc:Array">
      <item>
        <FeatureName>e2eSecureIndication</FeatureName>
        <FeatureVersion>1</FeatureVersion>
        <FeatureValue>Enabled</FeatureValue>
      </item>
    </ServiceFeatures>
    <NoOfFeatureElements>1</NoOfFeatureElements>
  </item>
  <item>
    <ServiceName>ppm-features</ServiceName>
    <ServiceVersion>1</ServiceVersion>
    <ServiceFeatures soapenc:arrayType="ns1:FeatureData[3]" xsi:type="soapenc:Array">
      <item>
        <FeatureName>FS-DeviceData</FeatureName>
        <FeatureVersion>2</FeatureVersion>
        <FeatureValue>FS-Available</FeatureValue>
      </item>
      <item>
        <FeatureName>setVolumeSettings</FeatureName>
        <FeatureVersion>1</FeatureVersion>
        <FeatureValue>Method-Available</FeatureValue>
      </item>
      <item>
        <FeatureName>getAllEndpointConfiguration</FeatureName>
        <FeatureVersion>1</FeatureVersion>
        <FeatureValue>Method-Available</FeatureValue>
      </item>
      <item>
        <FeatureName>getCallHistory</FeatureName>
        <FeatureVersion>1</FeatureVersion>
        <FeatureValue>Method-Available</FeatureValue>
      </item>
      <item>
        <FeatureName>deleteCallHistory</FeatureName>
        <FeatureVersion>1</FeatureVersion>
        <FeatureValue>Method-Available</FeatureValue>
      </item>
    </ServiceFeatures>
    <NoOfFeatureElements>3</NoOfFeatureElements>
  </item>
  <item>
    <ServiceName>proxy-server</ServiceName>
    <ServiceFQDN>sm1.avaya.com</ServiceFQDN>
    <ServiceDomain>avaya.com</ServiceDomain>
    <PPMServiceFQDN>http://sm1.avaya.com/axis/services/PPM</PPMServiceFQDN>
    <ServiceType>CoreSM</ServiceType>
    <ServiceURI>135.9.183.69</ServiceURI>
    <PPMServiceURI>http://135.9.183.69/axis/services/PPM</PPMServiceURI>
    <ServiceTransport soapenc:arrayType="ns1:TransportData[2]" xsi:type="soapenc:Array">
```

```
        <item>
          <transportName>TLS</transportName>
          <transportPort>5061</transportPort>
        </item>
        <item>
          <transportName>TCP</transportName>
          <transportPort>5060</transportPort>
        </item>
      </ServiceTransport>
      <NoOfTransportElements>2</NoOfTransportElements>
      <ServiceURIV6>2a07:2a41:ad07:11f4::443</ServiceURIV6>

<PPMServiceURIV6>http://2a07:2a41:ad07:11f4::443/axis/services/PPM</PPMServiceURIV6
>
      <ServiceTransportV6 soapenc:arrayType="ns1:TransportData[2]"
xsi:type="soapenc:Array">
        <item>
          <transportName>TLS</transportName>
          <transportPort>5061</transportPort>
        </item>
        <item>
          <transportName>TCP</transportName>
          <transportPort>5060</transportPort>
        </item>
      </ServiceTransportV6>
      <NoOfTransportElementsV6>2</NoOfTransportElementsV6>
      <ServiceVersion>asm7.1.0.0.151003</ServiceVersion>
      <ServiceFeatures soapenc:arrayType="ns1:FeatureData[4]" xsi:type="soapenc:Array">
        <item>
          <FeatureName>mustDualRegister</FeatureName>
          <FeatureVersion>0</FeatureVersion>
          <FeatureValue>true</FeatureValue>
        </item>
        <item>
          <FeatureName>FS-AST</FeatureName>
          <FeatureVersion>0</FeatureVersion>
          <FeatureValue>FS-Available</FeatureValue>
        </item>
        <item>
          <FeatureName>FS-PPM</FeatureName>
          <FeatureVersion>0</FeatureVersion>
          <FeatureValue>FS-Available</FeatureValue>
        </item>
        <item>
          <FeatureName>servicePriority</FeatureName>
          <FeatureVersion>0</FeatureVersion>
          <FeatureValue>1</FeatureValue>
        </item>
        <item>
          <FeatureName>failbackPolicy</FeatureName>
          <FeatureVersion>0</FeatureVersion>
          <FeatureValue>auto</FeatureValue>
        </item>
      </ServiceFeatures>
      <NoOfFeatureElements>4</NoOfFeatureElements>
    </item>
```

```
<item>
 <ServiceName>proxy-server</ServiceName>
  <ServiceFQDN>sm2.avaya.com</ServiceFQDN>
 <ServiceDomain>avaya.com</ServiceDomain>
 <PPMServiceFQDN>http://sm2.avaya.com/axis/services/PPM</PPMServiceFQDN>
 <ServiceType>CoreSM</ServiceType>
 <ServiceURI>135.9.183.74</ServiceURI>
 <PPMServiceURI>http://135.9.183.74/axis/services/PPM</PPMServiceURI>
 <ServiceTransport soapenc:arrayType="ns1:TransportData[3]" xsi:type="soapenc:Array">
  <item>
   <transportName>TLS</transportName>
   <transportPort>5061</transportPort>
  </item>
  <item>
   <transportName>TCP</transportName>
   <transportPort>5060</transportPort>
  </item>
  <item>
   <transportName>UDP</transportName>
   <transportPort>5050</transportPort>
  </item>
 </ServiceTransport>
 <NoOfTransportElements>3</NoOfTransportElements>
 <ServiceVersion>asmSwVersionUnknown</ServiceVersion>
 <ServiceFeatures soapenc:arrayType="ns1:FeatureData[4]" xsi:type="soapenc:Array">
  <item>
   <FeatureName>mustDualRegister</FeatureName>
   <FeatureVersion>0</FeatureVersion>
   <FeatureValue>true</FeatureValue>
  </item>
  <item>
   <FeatureName>FS-AST</FeatureName>
   <FeatureVersion>0</FeatureVersion>
   <FeatureValue>FS-Available</FeatureValue>
  </item>
  <item>
   <FeatureName>FS-PPM</FeatureName>
   <FeatureVersion>0</FeatureVersion>
   <FeatureValue>FS-Available</FeatureValue>
  </item>
  <item>
   <FeatureName>servicePriority</FeatureName>
   <FeatureVersion>0</FeatureVersion>
   <FeatureValue>2</FeatureValue>
  </item>
  <item>
   <FeatureName>failbackPolicy</FeatureName>
   <FeatureVersion>0</FeatureVersion>
   <FeatureValue>auto</FeatureValue>
  </item>
 </ServiceFeatures>
 <NoOfFeatureElements>4</NoOfFeatureElements>
</item>
<item>
 <ServiceName>proxy-server</ServiceName>
  <ServiceFQDN>bsm.avaya.com</ServiceFQDN>
```

```
    <ServiceDomain>avaya.com</ServiceDomain>
    <PPMServiceFQDN>http://bsm.avaya.com/axis/services/PPM</PPMServiceFQDN>
    <ServiceType>BranchSM</ServiceType>
    <ServiceURI>135.9.183.86</ServiceURI>
    <PPMServiceURI>http://135.9.183.86/axis/services/PPM</PPMServiceURI>
    <ServiceTransport soapenc:arrayType="ns1:TransportData[2]" xsi:type="soapenc:Array">
     <item>
      <transportName>TLS</transportName>
      <transportPort>5061</transportPort>
     </item>
     <item>
      <transportName>TCP</transportName>
      <transportPort>5060</transportPort>
     </item>
    </ServiceTransport>
    <NoOfTransportElements>2</NoOfTransportElements>
    <ServiceVersion>asmSwVersionUnknown</ServiceVersion>
    <ServiceFeatures soapenc:arrayType="ns1:FeatureData[4]" xsi:type="soapenc:Array">
     <item>
      <FeatureName>mustDualRegister</FeatureName>
      <FeatureVersion>0</FeatureVersion>
      <FeatureValue>true</FeatureValue>
     </item>
     <item>
      <FeatureName>FS-AST</FeatureName>
      <FeatureVersion>0</FeatureVersion>
      <FeatureValue>FS-Available</FeatureValue>
     </item>
     <item>
      <FeatureName>FS-PPM</FeatureName>
      <FeatureVersion>0</FeatureVersion>
      <FeatureValue>FS-Available</FeatureValue>
     </item>
     <item>
      <FeatureName>servicePriority</FeatureName>
      <FeatureVersion>0</FeatureVersion>
      <FeatureValue>3</FeatureValue>
     </item>
     <item>
      <FeatureName>failbackPolicy</FeatureName>
      <FeatureVersion>0</FeatureVersion>
      <FeatureValue>auto</FeatureValue>
     </item>
    </ServiceFeatures>
    <NoOfFeatureElements>4</NoOfFeatureElements>
   </item>
   <item>
    <ServiceName>presence-services</ServiceName>
    <ServiceURI>ps2.dr.avaya.com</ServiceURI>
    <ServiceType>PS</ServiceType>
    <ServiceTransport soapenc:arrayType="ns1:TransportData[1]" xsi:type="soapenc:Array">
     <item>
      <transportName>TLS</transportName>
      <transportPort>5222</transportPort>
     </item>
    </ServiceTransport>
```

```
      <NoOfTransportElements>1</NoOfTransportElements>
      <ServiceVersion>swVersionUnknown</ServiceVersion>
      <ServiceFeatures soapenc:arrayType="ns1:FeatureData[2]" xsi:type="soapenc:Array">
       <item>
        <FeatureName>XMPP_IM</FeatureName>
        <FeatureVersion>0</FeatureVersion>
        <FeatureValue>FS-Available</FeatureValue>
       </item>
       <item>
        <FeatureName>servicePriority</FeatureName>
        <FeatureVersion>0</FeatureVersion>
        <FeatureValue>1</FeatureValue>
       </item>
      </ServiceFeatures>
      <NoOfFeatureElements>2</NoOfFeatureElements>
     </item>
   </ServicesList>
   <NoOfServiceElements>4</NoOfServiceElements>
   <FNUList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:FNUFeature[2]" xsi:type="soapenc:Array">
     <item>
      <FNUType>avaya-cm-fnu=off-hook</FNUType>
      <App>true</App>
      <Media>false</Media>
      <CMVersion>R013x.00.0.300</CMVersion>
      <NoOfFNUDataElements>0</NoOfFNUDataElements>
     </item>
     <item>
      <FNUType>avaya-cm-fnu=transfer-to-voicemail</FNUType>
      <App>false</App>
      <Media>false</Media>
      <CMVersion>R013x.00.0.300</CMVersion>
      <NoOfFNUDataElements>0</NoOfFNUDataElements>
     </item>
   </FNUList>
   <NoOfFNUElements>2</NoOfFNUElements>
  </ServerCapabilities>
</ns1:getHomeCapabilitiesResponse>
```

Error Conditions/Faults:

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
“DatabaseNotAccessible”	There was a problem with the database
“NotIMUser”	Requesting user does not exist in the Enterprise database
“DataNotAvailable”	PPM unable to find the requested data
“InternalError”	Unexpected exception in PPM processing

3.4.2 getHomeServer

The getHomeServer method is used to retrieve the location and domain of the primary server that the endpoint is using.

Note: This method is still supported, but is deprecated. It has intentionally not been enhanced to support IPv6 addresses, and SIP Endpoints should use getHomeCapabilities instead.

getHomeServer (Handle)

Request Parameter	Type	Length / Value	Description
Handle	String	URI max 256 characters	The handle of the user in the form of "user@domain". Also see the note in Section 3.8 Handles.

getHomeServerResponse(ServerInfo)

Response Parameter	Type	Description
ServerInfo	HomeServerInfo	Container of HomeServer Information

HomeServerInfo	Type	Value	Description
PpmServer	String	HTTP URL (IPv4 only)	The PPM URL for the proxy server the gHS request was sent to.. If the user is a REMO user, then the IP address in the URL will be transformed as per Section 6: REMO User Support. If the endpoint can't connect to the SM "core", then this will contain the address of the Branch SM (BSM).
SipServer	String	Host (IPv4 only)	The IP address for the proxy server the getHomeServer request was sent to.If the user is a REMO user, then the IP address will be transformed as per Section 6: REMO User Support. If the endpoint can't connect to the SM "core", then this will contain the address of the Branch SM (BSM).
SipDomain	String	The SIP domain as provisioned in SMGR admin	The end point should always send the register for user@domain, where

			domain is the SIP domain and not the hostname of the SIP server.
TransportDataInfo	TransportListInfo		Transport information

TransportListInfo	Type	Description
NumOfElements	Integer	Number of elements in transport data list
TransportList	ArrayOfTransportData	Array of transport data types. Derived from the "Listen Port" table for the SIP Entity. The TransportData element is only included if the Endpoint column is checked for one or more Listen Port entries.

TransportData	Type	Value	Description
TransportName	String	{tls, tcp, udp}	Name of the transport.The transport ordering in the response will be 1.tls 2.tcp 3.udp
TransportPort	int	Examples{5060, 5061}	The port of the transport type

Example getHomeServer request:

```
<ns1:getHomeServer xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <handle>70004@avaya.com</handle>
</ns1:getHomeServer>
```

Example getHomeServer Response:

```
<ns1:getHomeServerResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ServerInfo>
    <ppmServer>http://135.9.183.69/axis/services/ppm</ppmServer>
    <sipServer>135.9.183.69</sipServer>
    <sipDomain>avaya.com</sipDomain>
    <transportDataInfo>
      <NoOfElements>2</NoOfElements>
      <TransportList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:TransportData[2]" xsi:type="soapenc:Array">
        <item>
          <transportName>TLS</transportName>
          <transportPort>5061</transportPort>
        </item>
        <item>
          <transportName>TCP</transportName>
          <transportPort>5060</transportPort>
        </item>
```

```
      </TransportList>
    </transportDataInfo>
  </ServerInfo>
</ns1:getHomeServerResponse>
```

Error Conditions/Faults:
The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
"NotIMUser"	Requesting user does not exist in the Enterprise database
"InternalError"	Unexpected exception in PPM processing.
"DataNotAvailable"	PPM unable to find the requested data

3.4.3 getPermissionsType
Some Avaya phones currently use this message as part of their login sequence but it is not required and is not returning any useable data.

3.4.4 setVolumeSettings
This operation allows an endpoint to set its Ringer Volume, Speaker Volume, Ringer Cadence, and Receiver Volume settings. getVolumeSettings is done within the context of the getAllEndpointConfiguration request/response but setVolumeSettings is a separate operation initiated by the endpoint.

setVolumeSettings(Handle, Identity, VolumeSettings)

Request Parameter	Type	Length / Value	Description
Handle	String	URI max 256 characters	The handle of the user in the form of "user@domain". Also see the note in Section 3.8 Handles.
Identity	DeviceIdentity		The identity of the specific device to apply changes to. If not supplied the changes will be stored as a single entry for all devices owned by Handle.
VolumeSettings	VolumeSet		Container for volume settings

DeviceIdentity	Type	Length / Value	Description
DeviceHandle	String (optional)	nn:nn:nn:nn:nn:nn max 128 characters	MAC address of the device. "nn" values are 6 pairs of hexadecimal digits separated by

			hyphens or colons (as shown here).

VolumeSet	Type	Length / Value	Description
RingerVolume	Short (optional)	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	Relative volume on a scale of 1 – 10
SpeakerVolume	Short (optional)	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	Relative volume on a scale of 1 - 10
ReceiverVolume	Short (optional)	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	Relative volume on a scale of 1 - 10
RingerCadence	Short (optional)	{1, 2, 3 ,4 ,5 ,6 ,7 , 8, 9, 10}	Ringer cadences on a scale of 1 - 10

setVolumeSettingsResponse(Result)

Result Parameter	Type	Description
Result	String	PPM_SUCCESS or a fault message

Example setVolumeSettings Request:

```
<ns1:setVolumeSettings
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>70004@avaya.com</Handle>
  <VolumeSettings>
    <RingerVolume>5</RingerVolume>
    <ReceiverVolume>5</ReceiverVolume>
    <SpeakerVolume>5</SpeakerVolume>
    <RingerCadence>1</RingerCadence>
  </VolumeSettings>
</ns1:setVolumeSettings>
```

Example PPM Response:

```
<ns1:setVolumeSettingsResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <PPMResponse>PPM_Success</PPMResponse>
</ns1:setVolumeSettingsResponse>
```

Error Conditions/Faults:

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
“Invalid Value ”	If value to be set is not of enum type VolumeSet then this fault will be send to endpoints
“NotIMUser”	Requesting user does not exist in the Enterprise database
“InternalError”	Unexpected exception in PPM processing
“DatabaseError”	PPM cannot access the database for reading or writing

3.4.5 setOneTouchDialList

The One Touch Dial List is a set of buttons and associated addresses which contain data that has been provisioned on the Communication Manager. The one touch dial list can be managed on the endpoint using the setOneTouchDialList method. Note that the button location is determined from the administration of an autodial button in the station form on Communication Manager and cannot be changed from the endpoint.

This information is initially retrieved when an endpoint registers with PPM and is in the ListOfOneTouchDial information element of the getAllEndpointConfigurationResponse. If an address (CM extension) has been provisioned for the button on the Communication Manager, PPM will indicate that this button is read only [i.e. ReadOnly = true] in the getAllEndpointConfigurationResponse. Read only buttons may only have their labels customized by the end user, and attempts to change the address will result in a fault. Otherwise, buttons may have their address and label customized by the end user (via setOneTouchDialList).

In SM 6.2 Feature Pack 4, a new capability was added on the System Manager so that the System Administrator can update an Endpoint/user’s Label field for one or more One Touch Dial / Autodial (the CM name for these buttons) buttons. Autodial button address configuration has been available on the SMGR and CM for a long time. There is no restriction that the Endpoint/user cannot change the One Touch Dial button Label field if the System Administrator had set it previously, as there is with the One Touch Dial button Address field.

setOneTouchDialList(Handle, ListOfOneTouchDialData)

Request Parameters	Type	Length / Value	Description
Handle	String	URI max 256 characters	The handle of the user in the form of “user@domain”. Also see the note in Section 3.8 Handles.
ListOfOneTouchDialData	OneTouchDialListInfo	Container for the one touch dial list	ListOfOneTouchDialData

OneTouchDialListInfo	Type	Length / Value	Description
NoOfElements	Integer		Number of elements in the array
OneTouchDialList	Array of OneTouchButton		Array of OneTouchButton elements

OneTouchButton	Type	Length / Value	Description
ButtonLocation	Short	1-96	The location of button
Address	String	max 256 characters	Number (or general SIP URI) assigned to the button for One Touch Dial. Cannot be modified by the endpoint when the ReadOnly flag is set to true in the getAllEndpointConfigurationResponse. This string can contain special characters sequences: *, #, ~m (mark), ~p (pause), ~s (suppress), ~w (wait) which are validated by the SMGR.
Label	String (optional)	max 255 characters	User or System Administrator assigned button label. This is the label that can be displayed for the button. If no label has been set (via setOneTouchDialList), PPM will not return a one Touch Dial button Label to the Endpoint.

setOneTouchDialListResponse(Result)

Response Parameter	Type	Description
Result	String	PPM_Success or fault message

Example setOneTouchDialList request:

```
<ns1:setOneTouchDialList
 xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
  <Handle>78001@avaya.com</Handle>
  <ListOfOneTouchDialData>
   <NoOfElements>2</NoOfElements>
   <OneTouchDialList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenc:arrayType="ns1:OneTouchButton[2]" xsi:type="soapenc:Array">
     <item>
```

```
        <ButtonLocation>7</ButtonLocation>
        <Address>70004</Address>
        <Label>Bob</Label>
      </item>
    </OneTouchDialList>
  </ListOfOneTouchDialData>
</ns1:setOneTouchDialList>
```

Example setOneTouchDialList Response:

```
<ns1: setOneTouchDialListResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <PPMResponse>PPM_Success</PPMResponse>
</ns1: setOneTouchDialListResponse>
```

Error Conditions/Faults:

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
"Invalid Value"	If an attempt is made to change the address field and the readOnly flag is set to 'true', then this fault will be sent to endpoints
"NoListAvailable"	The request from the endpoint did not contain any data
"NotIMUser"	Requesting user does not exist in the Enterprise database
"InternalError"	Unexpected exception in PPM processing
"DatabaseError"	PPM cannot access the database for reading or writing
"DatabaseNotAccessible"	PPM cannot access the database
"ButtonNotFound"	The request from the endpoint is for a button that has been deleted on CM.

3.5 CALL HISTORY

In version SM 6.2 Feature Pack 4, support for a centralized call journal was added to SM. When enabled on the SMGR for a user, SM will store call logs to this journal for the user. PPM supports downloading and deleting call logs from the journal.

A restriction in Feature Pack 4 was that call logs are stored only on the primary SM for the user. Attempts to download or delete call logs through PPM not on the primary SM, would result in a SOAP fault with a fault code of *OperationNotSupported*. Upon receiving that fault, the endpoint should continue using its call logs which are stored locally. However, upon success, the endpoint should replace its locally maintained call logs with the call logs in the PPM response. As of SM 8.0.0, this restriction was removed from the user's secondary SM, since getCallHistory and deleteCallHstory operations were now supported on the user's secondary SM as well.

An endpoint should only attempt the call history operations if supported by PPM. The endpoint should examine the *ppm-features* section in the getHomeCapabilites response to determine if the *getCallHistory* and *deleteCallHistory* methods are supported on the user's primary and secondary SMs. See the example under 3.4.1.

3.5.1 getCallHistory

This operation returns the call logs from the centralized call journal database stored on either the user's Primary or the secondary SM.

getCallHistory(Handle, CallHistoryCriteria)

Request Parameter	Type	Length / Value	Description
Handle	String	SIP URI max 256 characters	The handle of the user in the form of "user@domain". Also see the note in Section 3.8 Handles.
CallHistoryCriteria	CallHistoryCriteria (optional)		Criteria for selection of specific call log records. If omitted,"all" records are returned.

CallHistoryCriteria	Type	Length / Value	Description
Bridged	Boolean (optional)		If "false", then only records for non-bridged calls are returned. Otherwise, all call logs are returned.

If the CallHistoryCriteria isn't specified, then all call log records are returned. If it's specified with an empty list, then no call log records are returned as there is no selection

criteria specified. This would only be useful if an endpoint wanted to know if call history is available on the endpoint without actually download any records.

getCallHistory Response(CallHistoryList)

Response Parameter	Type	Description
CallHistoryList	CallHistoryList	Container for call log records

ContactHistoryList	Type	Description
NoOfElements	Integer	Number of elements in the array
CallHistoryInfo	ArrayOfCallLogs	Array of call log records. There is a system limit of 100 records per user.

ArrayOfCallLogs	Type	Description
CallHistoryDataList	CallHistoryData	Array of individual call log records time ordered from oldest to newest.

CallHistoryData	Type	Description
DialogId	String	The dialog-id value provided in the DSE notification.
Number	String	The remote party id of the call.
LocalNumber	String (optional)	The local party id of the call. For bridged calls, this represents the principal user associated with the call. For non-bridged calls, this field isn't returned.
Name	String (optional)	The name of the remote party if available.
Type	String	The values are "answered", "outgoing" or "missed".
Bridged	Boolean	Is the call bridged or not.
Time	dateTime	The time that the call started.
Duration	duration	The duration of the call. Granularity is never less than seconds.
Privacy	Boolean	If true, endpoints should enable logic for displaying privacy calls.

The **DialogId** and **Number** elements are unique for a user and are used when deleting call logs. The **Name** element isn't returned in the response if it is unknown. The **LocalNumber** element isn't returned for non-bridged calls. Note that both **Duration** and

Time elements use standard w3.org types. Please refer to the xmlschema under the www.w3.org website for details.

If the **Privacy** flag is true, the phones should suppress displaying name and number information when the LOCALLY_ENFORCE_PRIVACY setting is enabled.

Example getCallHistory request:

```
<ns1:getCallHistory xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
   <Handle xsi:type="xsd:string">70004</Handle>
</ns1:getCallHistory>
```

Example getCallHistory request for non-bridged calls only:

```
<ns1:getCallHistory xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
   <Handle xsi:type="xsd:string">70004</Handle>
  <CallHistoryCriteria>
     <Bridged>false</Bridged>
  </CallHistoryCriteria>
</ns1:getCallHistory>
```

Example PPM Response:

```
<ns1:getCallHistoryResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <CallHistoryList>
      <NoOfElements>4</NoOfElements>
      <CallHistoryInfo soapenc:arrayType="ns1:CallHistoryData[17]" xsi:type="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
         <item>
            <DialogId>538440967</DialogId>
            <Number>13035382200@avaya.com</Number>
           <Type>outgoing</Type>
           <Bridged>false</Bridged>
           <Time>2014-05-29T15:29:55Z</Time>
           <Duration>PT1H49M16S</Duration>
           <Privacy>false</Privacy>
         </item>
         <item>
            <DialogId>569540967</DialogId>
            <Number>17204382159@avaya.com</Number>
            <Name>Johnson, Kelly</Name>
            <Type>answered</Type>
            <Bridged>false</Bridged>
            <Time>2014-05-29T17:56:09Z</Time>
            <Duration>PT1M44S</Duration>
            <Privacy>false</Privacy>
         </item>
         <item>
            <DialogId>608640967</DialogId>
            <Number>13034428590@avaya.com</Number>
            <Name>Merritt, John</Name>
            <Type>missed</Type>
            <Bridged>false</Bridged>
```

```
            <Time>2014-05-29T21:55:04Z</Time>
            <Duration>PT18S</Duration>
            <Privacy>false</Privacy>
         </item>
         <item>
            <DialogId>365640967</DialogId>
            <Number>13035388432@avaya.com</Number>
            <Name>Reed, Patricia</Name>
            <Type>answered</Type>
            <Bridged>true</Bridged>
            <LocalNumber>13035381234@avaya.com</LocalNumber>
            <Time>2014-06-05T8:33:28</Time>
            <Duration>PT16M3S</Duration>
            <Privacy>false</Privacy>
         </item>
      </CallHistoryInfo>
   </CallHistoryList>
</ns1:getCallHIstoryResponse>
```

Error Conditions/Faults:

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
"DatabaseError"	PPM cannot access the database for reading or writing.
"OperationNotSupported"	The SM does not support call history operations for the endpoint user, i.e. the feature is not enabled for this user. As of SM 8.0.0, Call History logs are stored on both the user's primary and secondary SMs, and the getCallHistory operation is supported on the user's secondary SM as well. So this Fault string is only applicable on SM 7.1.x and prior when a getCallHistory request was received on the user's secondary SM.
"NotIMUser"	Requesting user does not exist in the Enterprise database
"InternalError"	Unexpected exception in PPM processing

3.5.2 deleteCallHistory

This operation allows the client to delete specific call log records, or all call log records for a user from the centralized call journal database stored on the SM.

deleteCallHistory(Handle, DeleteCriteriaList)

Request Parameter	Type	Length / Value	Description
Handle	String	SIP URI max 256 characters	The handle of the user in the form of "user@domain". Also see the note in Section 3.8 Handles.
DeleteCriteriaList	DeleteCriteriaList (optional)		Criteria for selection of specific call log records to delete. If the DeleteCriteriaList isn't specified, then all call log records are deleted. If it's specified with an empty list, then no call log records are deleted as there is no selection criteria specified.

DeleteCriteriaList	Type	Length / Value	Description
DeleteCriteriaList	DeleteCriteria		Array of DeleteCriteria records.

DeleteCriteria	Type	Length /Value	Description
DialogId	String (optional)	max 64 characters	The dialog-id value provided in the Dialog State Event notification.
Number	String (optional)	max 256 characters	The remote party id of the call.

deleteCallHistory Response()

Response Parameter	Type	Description
Result	String	PPM_SUCCESS or fault message

Example deleteCallHistory request for a specific call log record:

```
<ns1:deleteCallHistory xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
   <Handle xsi:type="xsd:string">70004</Handle>
   <DeleteCriteriaList>
```

```
      <item>
         <DialogId>10432360</DialogId>
         <Number>3038465718@avaya.com</Number>
      </item>
   </DeleteCriteriaList>
</ns1: deleteCallHistory >
```

Example deleteCallHistory request for all call log records:

```
<ns1:deleteCallHistory xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004">
   <Handle xsi:type="xsd:string">70004</Handle>
</ns1: deleteCallHistory >
```

Example PPM Response:

```
<ns1:deleteCallHistoryResponse
xmlns:ns1="http://xml.avaya.com/service/ProfileManagement/112004" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <PPMResponse>PPM_Success</PPMResponse>
</ns1:deleteCallHistoryResponse>
```

Error Conditions/Faults:

The following faults are sent by PPM to indicate an error condition in an HTTP 500 response. Note that this list may not be exhaustive. PPM can also return an HTTP 503 response with a Retry-After header which is described in section 3.6 Fault Message.

Fault	Description
CallLogNotFound	No call log records matched the criteria passed in the request.
“DatabaseError”	PPM cannot access the database for reading or writing.
“OperationNotSupported”	The SM does not support call history operations for the endpoint user, i.e. the feature is not enabled for this user. As of SM 8.0.0, Call History logs are stored on both the user’s primary and secondary SMs, and the deleteCallHistory operation is supported on the user’s secondary SM as well. So this Fault string is only applicable on SM 7.1.x and prior when a deleteCallHistory request was received on the user’s secondary SM.
“NotIMUser”	Requesting user does not exist in the Enterprise database
“InternalError”	Unexpected exception in PPM processing

3.6 FAULT MESSAGE

When an error condition is encountered, PPM responds with a PPMOperationFault. The fault messages associated with the various responses are located within each section.

PPMOperationFault

Result Parameter	Type	Description
Fault	PPMOperationFault	PPM error message

PPMOperationFault	Type	Description
PPMMethod	String	Name of PPM method where the fault occurred
Fault	String	Fault code generated by PPM

3.6.1 PPMOperationFault Format

When PPM throws a PPMOperationFault, the response to the client is a SOAP-ENV fault wrapper with the contents of the PPMOperationFault contained in the detail element of the response. For the faults it throws, PPM creates the body under the wrapper.
Here is an example of what PPM will send: Note that data contained in the <detail > tag will not strictly follow the posted PPM WSDL. In particular, the <detail> section will contain an extra <Fault> element and an extra <exceptionName> element which are not in the WSDL. There is currently no intention to fix this as changes could disrupt processing on endpoints that have come to rely on this format.

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>...<SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
        <faultcode>. . . </faultcode>
        <faultstring>. . .</faultstring>
           <detail>
            . . .
          </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.6.2 503 with Retry-After header

PPM can handle a varying number of concurrent SOAP requests from its clients depending on the type of Session Manager (SM) it is running on. If PPM is running on a

Core SM, it can handle 100 concurrent requests. If PPM is running on a Branch SM, it can handle either 75 or 38 concurrent requests depending on if the Branch SM is a “big” (S8800) or “small” BSM (S8300D). So, when an incoming PPM request encounters a server busy condition due to all its worker threads being used up, PPM sends back a 503 response with a Retry-After header that contains a retry time in seconds. This retry time is also enclosed in a <wait /> tag in the SOAP fault response. The retry values sent in the SOAP response are rotated across a range of seconds, in 3 second increments. This ensures messages to PPM are spread out over time to avoid flooding the server.

Where possible, the value in the Retry-After header should be used as opposed to the value provided in the <wait> element. The <wait> element is provided only for compatibility with older endpoints.

The example below shows a SOAP fault message where the retry time is 19 seconds:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV='http://schemas.xmlsoap.org/soap/envelope/'>
  <SOAP-ENV:Header>
    Retry-After: 19
    ...
  <SOAP-ENV:Header>
 <SOAP-ENV:Body>
  <SOAP-ENV:Fault>
    <faultcode>SOAP-ENV:soapenv:Server.Busy</faultcode>
    <faultstring>Server is busy - try later</faultstring>
    <detail>
      <wait xmlns='http://schemas.xmlsoap.org/soap/envelope/'>19</wait>
    </detail>
  </SOAP-ENV:Fault>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.6.3 500 with ConfigurationMismatch

A ConfigurationMismatch fault is returned when PPM cannot find necessary database entries for the handle indicated in a client request. The example below shows a SOAP fault message indicating a ConfigurationMismatch:

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>...<SOAP-ENV:Header>
 <SOAP-ENV:Body>
   <SOAP-ENV:Fault >
     <faultcode>SOAP-ENV:Server.generalException</faultcode>
      <faultstring>ConfigurationMismatch</faultstring>
      <detail>
        <Fault>
```

```
          <Fault>
            <PPMMethod>getAllEndpointConfiguration</PPMMethod>
            <fault>ConfigurationMismatch</fault>
            <usePPMFault>true</usePPMFault>
          </Fault>
          <exceptionName>com.avaya.ccs.ppm.domain.PPMOperationFault</exceptionName>
        </Fault>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.7 IDENTITY LIST

The SIP user 70004@avaya.com has the following communication addresses administered using the SMGR.

Handle	Domain	HandleType	HandleSubtype
70004	avaya.com	sip	shortform (CM extension)
3035370004	avaya.com	sip	username
+13035370004	avaya.com	sip	E.164
otherSip70004@yahoo.com	other	sip	
jabber70004@jab.com	other	xmpp	Jabber
lotus70004@lotus.com	other	ibm	Lotus Notes
xchg70004@xchg.com	other	smtp	MS exchange
sip:ocs70004@ocssip.com	other	sip	msrtc
yahoo70004@yahoo.com	other	smtp	

This is the Identities object in the gAEC response. Handles with a handle type of ‘sip’, ‘smtp’, ‘xmpp’, and ‘ibm’ are sent. If the SIP handle is the same as the user’s CM extension, the identity type is set to ‘shortform’. If the handle type is ‘sip’ and the subtype is null, the identity type is set to ‘otherSip’.

If the handle type is ‘smtp’ and the handle subtype is ‘MS exchange’, the identity type is set to ‘msexchange.’ If there are other types of ‘smtp’ addresses, they are given an identity type of ‘otherSmtp’.

```
        <IdentityList xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="ns1:Identity[7]"
xsi:type="soapenc:Array">
      <item>
        <Address>70004</Address>
        <Type>shortform</Type>
      </item>
      <item>
        <Address>3035370004</Address>
        <Type>username</Type>
      </item>
      <item>
        <Address>+13035370004</Address>
        <Type>e164</Type>
      </item>
      <item>
        <Address>otherSip70004@yahoo.com</Address>
        <Type>otherSip</Type>
      </item>
      <item>
        <Address>xchg70004@xchg.com</Address>
        <Type>msexchange</Type>
      </item>
```

```
    <item>
      <Address>yahoo70004@yahoo.com</Address>
      <Type>otherSmtp</Type>
    </item>
    <item>
      <Address>sip:ocs70004@ocssip.com</Address>
      <Type>msrtc</Type>
    </item>
    <item>
       <Address>70004@avaya.com</Address>
       <Type>otherXmpp</Type>
    </item>
    <item>
        <Address> jabber70004@jab.com</Address>
        <Type>Jabber</Type>
    </item>
  </IdentityList>
</ListOfIdentities>
```

3.8 HANDLES

Handles are used to identify contacts and administered users. Handles are passed in the ContactData.Address and ContactData.Handles fields for the addContact, updateContact, and getContactList commands. They are also passed in the ListOfHandles of the addContactResponse and the updateContactResponse. Handles are also passed in the SearchResult.PrimaryHandle field, and in the ListOfIdentities data in the getAllEndpointConfigurationResponse. The contents of the ListOfIdentities data is described in section 3.6.

For administered users, the Primary Address and the Address fields will always be populated with a SIP handle. If there are multiple SIP handles, precedence is given to the handles under the default communication profile set, prioritized with the following order based on. handle of subtype:

1. username
2. e164
3. msrtc
4. otherSIP

If there are no SIP handles in the primary communication profile set, then the same algorithm is applied to the other communication profile sets.

If the user has a CM extension which is different than all their handles configured on the SMGR, also known as a short-form CM extension, it will be added to the SIP handle through the "avext" parameter. For example, a user has a SIP handle of 3035381234@avaya.com and a CM extension of 5381234, PPM will return the handle as 3035381234@avaya.com;avext=5381234.

External contacts typically do not have handles. So when a phone adds an external contact, it will generate a handle (actually an <Address> element) for this contact with the format of *phone_number@domain* where number is the phone number provided in the <PhoneNumber> element and domain is the domain of the contact owner. PPM will store this handle for the contact which is used for lookup and returned in the getContactList response. Should an external contact be created for a user on the System Manager without a SIP handle, PPM will generate a pseudo-handle of the format `ep__ppm__ci_`*number* where number is the SMGR DB id of the contact. Since this handle is prefixed with ep__, the phone will not attempt to call this address. Additionally, handles prefixed with ep__ will not be displayed by the endpoint.

For internal contacts that do not have any configured handles, PPM will generate a pseudo-handle of the format `ep__ppm__ui_`*number* where number is the SMGR DB id of the user. Since this handle is prefixed with ep__, the phone should not attempt to call this address. Additionally, handles prefixed with ep__ should not be displayed by the endpoint. If that user has a CM extension, it will be appended to the handle using the "avext" parameter described above.

Notes

- Because a user can have multiple handles and some of a user's handles can contain different domains, it is highly recommended that all PPM clients include user@domain in the Handle parameter of each PPM request. If the domain is not included, multiple domains exist within the Enterprise, and an individual user's handles do not all have the same domain, PPM will not be able to authenticate the user because PPM will resort to using the default domain of the Session Manager that the user is registered with.
- PPM does not return all the configured handles. PPM only returns the handles from the communication profile set that includes the handle the user entered when logging in (plus the CM extension, if it differs from all those handles).

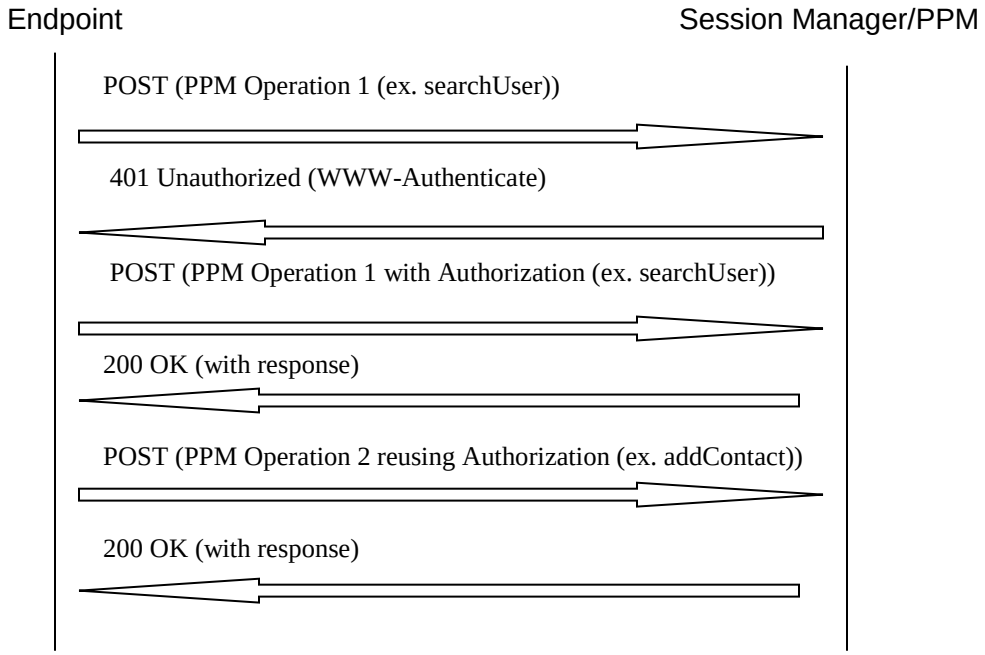
4 AUTHENTICATION

SIP endpoints communicate with the PPM to retrieve configuration information such as dialplan, buttons, and contact lists, to add or update contacts, to save device specific data, etc. Endpoints will authenticate themselves with the PPM, using the same username and password they use during SIP registration, and each PPM request to the Session Manager will be authenticated using digest authentication (see RFC 2617) over HTTP or HTTPS. This means that each request may actually consist of two round-trip requests (the original request, the response with a challenge, the subsequent request containing the challenge response, and the final response).

For each PPM message sent over HTTP, a nonce aging parameter will be maintained in PPM as each of these requests will be challenged. When a subsequent request is received by PPM before the nonce age has expired and the user's credentials are otherwise correct, the request will not be challenged. This sequence is shown with PPM Operation 1 with Authorization in the figure below. If the same nonce from that PPM operation is reused in PPM operation 2 with Authorization, that request would be challenged as that nonce is only allowed to be used once. A short timeout is associated with these nonces to accommodate the receipt of a single PPM request from a user.

For PPM messages sent over HTTPS, a nonce aging parameter will be maintained in PPM so that each request after the first one would not have to be challenged. When a subsequent request is received by PPM before the nonce age has expired and the user's credentials are otherwise correct, the request will not be challenged. This sequence is shown with PPM Operation 2 with Authorization in the figure below. A long timeout is associated with these nonces to accommodate the receipt of multiple PPM requests from the same user.

The following figure shows just one example of the communication, in this case over HTTPS, between an Avaya Advanced SIP Telephony endpoint and PPM:



5 HOW DATA IS SYNCHRONIZED BETWEEN PPM AND OTHER AURA ENTITIES /APPLICATIONS

The Session Manager and Branch Session Manager databases are populated from data replicated down to them from the System Manager database, which is the master database. The data in the System Manager database comes from several different sources in the network:

- Communication Manager (Dialplan data, Feature Access Codes, Station data, Button data, etc.)
- The System Manager GUI (Emergency numbers, Device Settings, Session Manager administration data, etc.),
- Advanced SIP Telephony (AST) phones (Contact data and Device data) via the PPM SOAP API.

PPM retrieves data from the Session Manager database, processes it, and returns it to Avaya's AST phones. Since the Session Manager database is a read-only replica of the System Manager database, PPM never writes directly into the Session Manager database. Note, the System Manager database can be backed up so none of the above mentioned data is lost.

The data from the System Manager GUI and the data saved by the AST phones via the PPM SOAP API are added directly into the System Manager's database, but that's not always true for Communication Manager (CM) data. Some CM data can be changed from the System Manager GUI and is stored directly into the System Manager database. Other CM data can be changed from the System Manager GUI via CM cut-thrus, but this data is not stored directly into the System Manager database. It's changed directly on the CM and then an automatic incremental sync is done to get the SMGR database in sync with the CM. Any CM data which is changed directly on the CM requires the System Manager to synchronize its data with that CM before that data can end up in the System Manager's database. Until the System Manager database is in sync with the CM's, users will not receive their latest button, station, dialplan, etc. data from PPM. The System Manager will automatically do a full data sync once per day with each CM. **Note:** Incremental and full CM data synchronizations can be done on demand or can be scheduled to occur one or more times per day.

When any of the data mentioned above changes in the System Manager's database, it is replicated down to the SMs and BSMs and this may take up to 60 seconds for replication to occur. A process that runs on the Session Manager is continually monitoring its database for changes to the tables which PPM utilizes. Every 2 minutes, SM will send a SIP NOTIFY to each AST phone whose PPM data has changed. These phones will then send the appropriate SOAP request to PPM to get their data caches up to date.

Because PPM takes the dialplan data, massages it into the proper format that the Avaya AST phones need, and then caches it, PPM treats the dialplan data differently. PPM does two things to make sure the dialplan data it caches is kept up to date. Once each day early in the morning, a PPM thread wakes up and rebuilds PPM's dialplan caches for each CM, whether that dialplan data has changed or not. Then every minute, another

thread in PPM wakes up to look for CM dialplan changes. If any dialplan changes are found for any of the CMs, PPM will rebuild its dialplan cache for each CM whose data has changed.

So based on the information above, when a System Administrator changes the CM dialplan tables from the System Manager GUI:

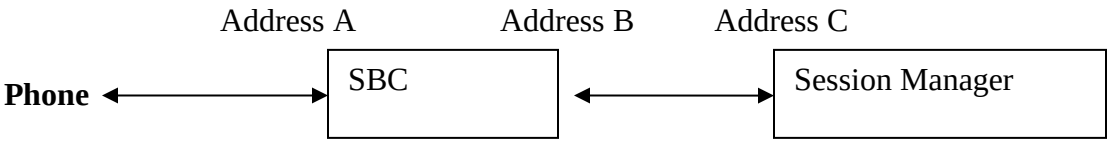
- 1. The System Manager will do an automatic incremental sync with this CM to update its database, which takes a few minutes.
- 2. Within 1 minute of the System Manager database being updated, the new dialplan data will be replicated to all the Session Managers.
- 3. PPM automatically updates its general CM data caches in response to the Session Manager database being updated.
- 4. Within 1 minute, PPM checks for any dialplan table updates, finds them for this CM, and rebuilds its dialplan cache
    - At this point, an AST phone whose station is configured on this CM could log out and log back in to get their updated dialplan data. ***(Total time to get new dialplan data, 6 minutes or less)***
- 5. The Session Manager process that monitors for and collects database changes picks up the dialplan changes and notes which users are affected. Within 2 minutes, the effected phones will be sent a SIP NOTIFY telling them to retrieve their latest Endpoint Configuration data by sending PPM a getAllEndpointConfiguration request message. PPM will return a getAllEndpointConfiguration response which includes their latest dialplan data. ***(Total time to get new dialplan data, 8 minutes or less)***
    - **Please note:** *The above mentioned times are best case and approximate. As t*hese NOTIFYs are metered out to minimize the impact on the SM, the process of sending out NOTIFYs to all affected SIP phones will take longer if the number of phones to be notified is large.

There is another scenario where the System Administrator changes the CM dialplan on the CM directly rather than from the System Manager GUI. In this case, only step #1 from above is different and the changes are highlighted in step 1' below, Due to the fact that the System Manager waits up to two minutes to do the incremental sync, the overall time to get the new dialplan data will take an additional two minutes:

- 1'. The CM sends a message to the System Manager to let it know that some changes have been made to its data. The System Manager will wait up to two minutes before retrieving the new CM data since the Administrator could be making several other changes to the CM around the same time and it wants to reduce the number incremental synchronizations it does. When the two minutes are up, and it's time to retrieve the new data, the System Manager will do an incremental sync with this CM to update its database, which as mentioned previously takes a few minutes.

6 SIP REMOTE OFFICE SOLUTION (REMO) USER SUPPORT

An Avaya Remote Office Solution (REMO) user is a SIP phone client which resides outside the Enterprise and gains access to the Enterprise via a Session Border Controller (SBC) rather than a VPN. In Release 6.2, PPM supplies the NAT (Network Address Transformation) function as a critical element for user. When REMO endpoints or users send registration requests over the internet to the public interface of an SBC at the edge of the Enterprise network, the request is proxied by the SBC and passed to the core SM. The function of PPM NAT is to expose the transformed public IP address of SM to the endpoint in the PPM response message body where applicable. The following simplified reference model illustrates the basic NAT function.



In the above REMO scenario, the SBC proxies both SIP and PPM requests to the primary and secondary Core SMs. (PPM does not perform NAT for branch SMs; you cannot configure a branch SM as an "Address C".)

The SBC, if configured with HTTP/HTTPS enabled and NAT for "A" to "C", translates the "A" address to "C" so that the PPM request arrives at the targeted SM. However, SBCs typically do not provide for embedded payload modifications in providing NAT capabilities. In order to enable the above function, when PPM receives the endpoint request, it checks the incoming IP address from the header of the incoming HTTP address, and if it is one of the configured SBC IP addresses (Address B in the above reference model), then PPM does the NAT on the address. On the other hand, when SM sends the getHomeSever and getHomeCapabilities response messages back to the endpoint, the address of A replaces the C address that PPM would normally have returned.

With the advent of IPv6 in Avaya Aura 7.1, this mapping becomes a little more complex. The following text explains the new PPM algorithm:

1. If the endpoint's address matches a Reference B address and is of type IPv4, PPM looks for an entry where the Reference C address type is IPv4.
    - If an entry is found,
        - PPM includes the Reference A address as the controller's address in the PPM message. In addition,
            1. if the Reference A address is IPv4, PPM excludes any IPv6 addresses for that same SM.
            2. if the Reference A address is IPv6, PPM excludes any IPv4 addresses for that same SM.
2. If the endpoint's address matches a Reference B address and is of type IPv6, PPM looks for an entry where the Reference C address type is IPv6.
    - If an entry is found,

- PPM includes the Reference A address as the controller's address in the PPM message. In addition,
    1. if the Reference A address is IPv4, PPM excludes any IPv6 addresses for that same SM.
    2. if the Reference A address is IPv6, PPM excludes any IPv4 addresses for that same SM.

3. If a matching entry is not found (neither IPv4 nor IPv6)
    - If the "Military Support" global flag is set
        - PPM passes that controller address without modification
    - Else
        - PPM omits that controller altogether

7 ENDPOINT CHANGE NOTIFICATIONS

Because Endpoint configuration data and user data can be modified by entities other than the Endpoints, another Session Manager application called the Endpoint Management Agent (EMA) was created to work in conjunction with PPM to inform Endpoints of these data changes.

The Endpoint Management Agent's purpose is to notify any advanced SIP telephony clients, which have subscribed to the *avaya-ccs-profile* event package, of any configuration data changes that affect them. This notification is made via a SIP NOTIFY message.

Of all the Endpoint configuration data tables and user data tables that exist on the SM, only the Device Data tables (see the Device Data section of this document) are not monitored by the EMA since they can only be changed by Endpoints.

One of two messages may be sent by the EMA. The first is a reboot NOTIFY, the second is a reload NOTIFY. The reboot NOTIFY is sent to notify the endpoint that it should reboot. Similarly, the reload NOTIFY is sent to notify the endpoint that it needs to reload some or all of its configuration data.

Here's an example of a reboot NOTIFY:

```
NOTIFY sip:1111@10.0.75.2 SIP/2.0
Call-ID: cid-1@10.0.75.2
CSeq: 2 NOTIFY
From: <sip:1111@atler.com>;tag=random2
To: <sip:1111@atler.com>;tag=random1
Via: SIP/2.0/UDP 10.0.0.100;branch-id=z9hG4bK-random-SES1
SIP/2.0/TLS 10.0.0.200;branch-id=z9hG4bK-random-cm1
Content-Length: 22
Content-Type: application/profile+xml
Contact: <sip:1111@10.0.0.200;transport=tls>
Max-Forwards: 69
User-Agent: Avaya Communication Manager v1.0
Event: avaya-ccs-profile
Subscription-State: active;expires=3600
Record-Route: <sip:10.0.0.100:5060;lr;transport=UDP>
<?xml version="1.0"?>
<event>
<eventName>Reboot</eventName>
<eventTime>[timestamp]</eventName>
</event>
```

The <eventName> element is used to indicate the type of event, "Reboot" in this example.

The reload notify contains an additional element, <eventCode>, which is used to indicate what type of data should be reloaded. The following table provides the values passed in this element as well as the action the endpoint is expected to take.

Code	Action
0	Endpoints should reload all data by sending PPM: • a getHomeServer request • a getHomeCapabilities request • a getAllEndpointConfiguration request • a getDeviceData request • one or two setDeviceData requests (as needed) • a getContactList request Note: The order in which these requests are made is entirely up the Endpoint, but the gHS and gHC requests should be the first two requests made by the AST Endpoint.
1	Endpoint should reload its configuration data by sending PPM a getAllEndpointConfiguration request with all data elements it requires as items in the Fields XML element of this request.
2	Endpoint should reload all contact data for the logged in user by sending PPM a getContactList request..
5	Endpoint should reload maintenance data by sending PPM a getAllEndpointConfiguration request with only the ListOfMaintenanceData item in the Fields XML element of this request.

Here's an example of a reload NOTIFY message:

```
NOTIFY sip:1111@10.0.75.2 SIP/2.0
Call-ID: cid-1@10.0.75.2
CSeq: 2 NOTIFY
From: <sip:1111@atler.com>;tag=random2
To: <sip:1111@atler.com>;tag=random1
Via: SIP/2.0/UDP 10.0.0.100;branch-id=z9hG4bK-random-SES1
SIP/2.0/TLS 10.0.0.200;branch-id=z9hG4bK-random-cm1
Content-Length: 22
Content-Type: application/profile+xml
Contact: <sip:1111@10.0.0.200;transport=tls>
Max-Forwards: 69
User-Agent: Avaya Communication Manager v1.0
Event: avaya-ccs-profile
Subscription-State: active;expires=3600
Record-Route: <sip:10.0.0.100:5060;lr;transport=UDP>
```

```
<?xml version="1.0"?>
<event>
<eventName>Reload</eventName>
<eventTime>[timestamp]</eventTime>
<eventCode>2</eventCode>
</event>
```

It should be noted that in 6.2, AST Endpoints have changed the way they function so that if the end user is performing any add/update/delete Contact operations, the Endpoint will not respond to any Reload Contacts SIP NOTIFYs it receives until 2 to 3 minutes after its last Contacts operation was completed. This is to prevent the Endpoint from requesting a new version of the user's contact list while the user is in the process of making additional contact data changes, but before all their Contact data changes have been replicated down to the Session Manager's local DB.

It should be further noted that recent AST Endpoints will send PPM a getContactList request within 2 to 3 minutes after the user's last Contact operation, even if it did not receive a Reload Contacts SIP NOTIFY. All existing and future AST endpoints should model this behavior.

In SM 6.2 Feature Pack 1 the notification process was optimized to stop sending notifications during synchronization operations with the Communication Manager (CM). The previous behavior resulted in numerous notifications caused by the synchronization process. As of SM 6.2 Feature Pack 1 the Endpoint Management Agent will consolidate these notifications and defer sending them until after the synchronization process has completed. The end result is fewer notifications resulting in fewer endpoint reboot/reload operations.

8 GENERAL INFORMATION REGARDING OPTIONAL VALUES

Many of the PPM operations described in this document accept optional parameter values (e.g., the contact Name parameter in the addContact operation). PPM behavior when these parameters are missing or set explicitly to "nil=true" varies depending on whether the PPM operation is an add or an update operation.

For an "add" operation (e.g., addContact) a nil parameter will not be set. Consider addContact as an example. If the contact <Name> is set to nil, the name is not set for the contact. A getContactList request will not return any information for the contact name in this case. This is true regardless of whether the <Name> is explicitly set to nil or if the <Name> is missing entirely.

> *There is a related case where the value associated with an element is an empty string. For example,* `<Name></Name>` *(or* `<Name/>`*) assigns the value of the named element to an empty string - i.e., "". This is true for both adds and updates. A query, e.g., getContact, will return the same representation encoded in XML (e.g,.* `<Name></Name>`*).*

For an "update" operation (e.g., updateContact) an explicitly nil parameter (i.e., nil="true") will not result in an update of the corresponding field. Considering updateContact as an example, an explicitly nil value for a contact's Name will leave that contact's name unchanged. So if the contact's name was never set, it will remain unset (i.e., without a value). If the contact's name was previously set to "John Doe", it will remain set to "John Doe" after the update.

For an "update" operation (e.g., updateContact) that is missing an optional parameter entirely, any existing values for that parameter will be unchanged. Consider updateContact as an example and assume that the contact <Name> has been previously set (e.g., to "John Doe"). If the <Name> parameter is missing entirely from the updateContact request, that contact's name will remain "John Doe". A subsequent getContactList request will return "John Doe" as the <Name> for that contact.

For an "update" operation (e.g., updateContact) that contains an optional parameter, but the parameter contains no value, the existing value for that parameter will be cleared. Consider updateContact as an example and assume that the contact <Name> has been previously set (e.g., to "John Doe"). If the <Name/> parameter is received in the updateContact request, that contact's name will be changed to "". A subsequent getContactList request will not return a value for <Name> for that contact. It should be noted, that if the Localized Name, Localized FirstName, and Localized LastName fields are all there but contain no value, the updateContact request will fail. This is because at least one of these fields must contain a valid value when they are sent to the SMGR.

Here are some final notes regarding explicitly setting a parameter to nil=true.

- Explicitly setting a parameter to "nil" takes the following form:

```
<SomePpmOperationParameter xsi:nil="true" xmlns/>
```

- In the example above, the "`xsi`" name space used to prefix the `nil` attribute is defined by `http://www.w3.org/2001/XMLSchema-instance`.
- The name space must be included with each parameter that is set to ''nil'', otherwise PPM will not consider that that parameter has been set to ''nil''.

Here is an example using the <Name> parameter from the updateContact operation:

```
<Name xsi:nil="true"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
```

9 GENERAL INFORMATION REGARDING CONTACTS

This section provides additional information about key contact concepts that are applicable to all contact operations (i.e., getContactList, addContact, updateContact, deleteContact, searchContact, searchContactCount, searchUser).

9.1 CONTACT LIMITS

It is recommended that no more than 250 contacts be added/configured for any user. The System Manager documented limit is 250 contacts per user, but this limit is not enforced. Exceeding this limit may result in performance degradation on both the SMGR and the SM. Since 250 is a soft-limit, endpoints should be prepared to handle more than this number.

9.2 TYPES OF CONTACTS

Contacts are classified as either enterprise (commonly referred to as **internal)** or private (commonly referred to as **external)** contacts.

Internal contacts are those contacts (in a contact list) that reference an administered User in the enterprise's System Manager. Once defined in the System Manager, a User can be added to another User's contact list.

> *As a point of interest, an **internal** contact may have phone numbers and handles associated with it that are not administered as a handle or CM extension in the System Manager (e.g., a home phone number or a home email address). This external phone number or handle would be stored as private contact information of the internal contact.*

External contacts are those contacts (in a contact list) that reference an entity or individual that is not an administered User in the System Manager. Generally, friends, family, and business contacts external to the enterprise would be classified as external contacts.

The classification of **internal** vs. **external** affects all Contact operations - add, delete, update, and read.

When a contact add or update operation is performed, there are 2 pertinent pieces of information provided in the request that are used to determine if the contact is indeed an administered/existing User:

1. The ***ContactData.Address*** field - PPM will search for a handle in the System Manager (SMGR) database that matches the provided value.

   > *Handle type is also used in determining a match. Currently the handle type must be "sip". Work is underway to extend this to "xmpp". Other types may be supported in the future.*

2. The ***ContactData.ContactPhoneData.PhoneNumber*** field - If the associated ***ContactData.ContactPhoneData.Type*** field contains a value of "work" (or "handle"), PPM will search the UPM database for a handle that matches the provided value.

If either of the above searches produces a match, the contact will be classified as an **internal** contact. Otherwise, the contact will be classified as an **external** contact.

As a further refinement of the above search, internal users can be identified by what are called long and short forms of their handles. The long form typically includes a domain name (e.g., rsmith@somecompany.com). The short form is a CM extension will not include the domain name (e.g., just rsmith).

Section 3.1, Contact Messages, (addContact) contains additional detail regarding internal and external contacts.

9.2.1 Public Contacts

Public contacts are created by a system administrator using the System Manager. Public contacts are visible to all system users in the same way other administered users are - i.e., they can be searched for and added to a user's contact list. The intent of a public contact is to create an entry in the System Manager "directory" that is visible to all users. As an example, consider the phone number for building security or building maintenance. It may be desirable to add one or both of these as public contacts so all system users can search for and add these as contacts to their respective contact lists.

Public contacts can be updated and deleted by the system administrator. Updates and deletions performed by the system administrator affect all contact lists that contain the updated/deleted public contact(s).

Users can add additional information to public contacts (e.g., home phone number). This information is private to the user who added the information.

Public contacts were introduced with release 6.2 FP1.

9.3 CONTACT DATA STRUCTURES

Contact data can be described simplistically as follows:

- **Handle**
- **ContactData**
  - Address
  - **Name**
  - **FirstName**
  - **LastName**
  - **PreferredLanguage**
  -
  - **EndpointDisplayName**
  - **FirstNameAscii**

- **LastNameAscii**
- **Alias**
- Email
- IM
- VideoCapable
- ContactPhones (**ContactPhoneData** array)
  - Phone Number
  - Label1/2
  - **Type**
  - SpeedDialEnable
- HandleDataList (**HandleData** array)
  - Handle
  - Handle type
  - Handle sub-type
- EndpointDataList (EndpointData array)

For the purposes of this discussion, the above represents only a subset of the total Contact dataset, and only a subset of these will be further described here (i.e., the items indicated by **bold font**). See section 3.1, Contact Messages for detailed information about these data structures.

9.3.1 Handle

A handle is associated with an Enterprise user. A user can have several handles (*e.g., a SIP handle, and an IM handle; an email address is also considered a handle*). A handle is associated with only a single Enterprise user. In this context, a **Handle** is one identity of the Enterprise user who "owns" the contact.

> *Strictly speaking, the handle and its associated type comprise an Enterprise user identity. For example, user@company.com/sip and user@company.com/xmpp are 2 different identities for the same Enterprise user.*

9.3.2 ContactData

ContactData is the "container" for everything known about a contact. I.e., their name(s), their phone number(s) and associated type(s), as well as any **HandleData** we may know about a contact if they are an internal contact.

9.3.2.1 Names

There are 7 different fields containing contact name information:

1. Name (Localized Full Name)
2. FirstName (Localized)
3. LastName (Localized)
4. Alias
5. EndpointDisplayName (ASCII)
6. FirstNameAscii
7. LastNameAscii

This section describes how three of their values are populated.

9.3.2.1.1 Name

Name is populated from the Localized Display Name field on the System Manager User Profile form or directly from the Endpoint. The "Localized Display Name" can be created in the following ways:

1. A concatenation of the "Last Name" and "First Name" from the User Profile form. The value takes the form of "LastName, FirstName". This is the default value if it isn't overridden by directly entering the data in the "Localized Display Name" field.
2. Entered directly in the field. This will override the default value described in the previous item.
3. Entered from the endpoint when creating a contact from the endpoint. Note that for internal and public contacts, the previous 2 methods of populating the field take precedence over the endpoint provided Name.

9.3.2.1.2 Alias

Alias is an optional field and is populated under the following circumstances:

1. When the endpoint provides a value for Name for an internal or public contact, this value is used to populate Alias (if otherwise left unspecified).
2. When explicitly specified by the endpoint

9.3.2.1.3 EndpointDisplayName

EndpointDisplayName can be populated from the System Manager User Profile form or directly from the Endpoint. The value is populated in one of the following ways

1. Directly entered into the field on the SMGR form
2. A concatenation of the "Last Name Ascii" and "First Name Ascii" fields from the User Profile form. The value takes the form of "LastNameAscii, FirstNameAscii".
3. From the Endpoint, either directly entered into this field or via a concatenation of the “LastNameAscii” and “FirstNameAscii” fields, separated by a comma, and taking the form as mentioned previously in item #2.

9.3.2.1.4 Address

When populating the Address field of an addContact request, SIP clients should do the following:

- Use the PrimaryHandle field of the contact being added from the searchUserResponse, if a searchUser request was just successfully done
    - otherwise use any manually entered PhoneNumber of type work
- If there is no PhoneNumber of type work, use any manually entered PhoneNumber of type mobile
- If there is no PhoneNumber of type work or mobile, use any manually entered PhoneNumber of type home
- If there is no manually entered PhoneNumber of type work, mobile, or home, manufacture a handle of type ep_....., as the SIP hard phones do

When populating the Address field of an updateContact or deleteContact request:

- Populate the Address field with the address for this contact from the latest getContactListResponse. Note: The avext parameter does not need to be stripped off of the address.

9.3.2.1.5 Label_2

SIP endpoints currently use the Label_2 field to mark a contact as a favorite.

9.3.3 ContactPhoneData

ContactPhoneData represents the information known about a single phone number for the contact. It's represented as an array, indicating that a contact may have multiple phone numbers associated with it. **ContactPhoneData** also includes a "Type". This is used to indicate the type of phone number such as home, work, handle, page, fax, or mobile. The "work" and "handle" types are considered equivalent if the phone number field contains the handle of an administered user

9.3.4 HandleData

HandleData will only exist for an **Internal** contact as the concept of a handle only applies to Enterprise users. It's represented as an array to indicate that an internal contact may have multiple handles (as described in the Handle section above). **HandleData** consists of:

- Handle - this is an Enterprise user's handle
- Handle type - this categorizes the handle. E.g., "sip" for telephony, "xmpp" for instant messaging, and "smtp" for email.
- Handle sub-type - this is a further categorization of the handle type. E.g., e164 is a sub-type of "sip" and represents a globally unique identifier that can be used to contact an Aura user. "username" and "msrtc" are also sub-types of "sip". "jabber" is a sub-type of "xmpp". "msexchange" is a sub-type of "smtp".

9.3.5 HandleData vs. ContactPhoneData

ContactPhoneData applies exclusively to contact information related to telephony. As such it doesn't contain any information regarding email, IM, or other communication channels.

HandleData contains information about all the administered communication addresses by which an Enterprise user can be reached. External email and external IM addresses cannot be represented by **HandleData**. In fact, the current UPM/PPM implementation cannot support external communication addresses with the exception of email which can be supported by the **ContactData.Email** field.

HandleData is only available for internal contacts. **ContactPhoneData** is available for both internal and external contacts.

> ***Sidebar:*** *Some endpoints place an icon of some type next to a phone number or handle. There is currently no standard mapping between the*

"ContactPhoneData.Type/HandleData.HandleType/HandleSubtype" and these icons. That said, the following seems like a reasonable icon mapping:

- *Video -> ContactData.VideoCapable*
- *eMail -> ContactData.Email*
- *Phone -> ContactData.ContactPhoneData; ContactData.HandleData.Type of "sip" would also be represented by a "Phone" icon.*
- *IM -> ContactData.HandleData.Type of "xmpp"*

9.3.6 Other pertinent information

For internal contacts, their **ContactPhoneData.Type** field will always contain the value "handle", even if it was originally added with the type of "work" ***as long as*** the **ContactPhoneData.PhoneNumber** matches one of the internal contact's handles. "work" will be maintained for external phone numbers (i.e., phone numbers that are not Enterprise handles or extensions).

HandleData.HandleType/HandleSubtype, while similar to, are not the same as **ContactPhoneData.Type**. As a trivial example, there is no **HandleData.Type** of "mobile" and there is no **ContactPhoneData.Type** of "sip".