Quadro "CallControl" Interface

Revision: 2.4

Abstract: **This document gives a detailed description of the Quadro's 3-rd party call control (3pCC) interface functions.**

Table of Contents:

1 Introduction 5
2 Accompanying Material 5
3 Terms Used in This Document 5
4 Functional Description 5
5 Syntax Used for Call Destinations 6
5.1 General Functions 7
5.1.1 GetSystemInfo (uniqueID, softVersion) 7
5.2 Authentication 8
5.2.1 Authenticate(username, password) 8
5.3 Incoming Call Handling 9
5.3.1 IncomingCallHandling(extensionName, enable) 9
5.3.2 CallArrived(callID, from, to) 9
5.3.3 CallOriginated(callID, from, to, pattern) 11
5.4 Control Initiated Call Handling 14
5.4.1 CreateCall(callID, from, to, displayName, userName) 14
5.4.2 CloseCall(callID) 15
5.4.3 TransferCall(called, transferParty) 15
5.4.4 CallStateChanged(callID, callState, errorCode) 16
5.4.5 JoinCalls(callID1, callID2) 17
5.4.6 UnjoinCalls(callID1, callID2) 18
5.4.7 PlayMessage(callID, messageFileName, playCount) 18
5.4.8 MessagePlayed(callID) 19
5.4.9 EnableDtmfDetection(callID, enable) 20
5.4.10 DtmfDigitDetected(callID, dtmfdigit) 20
5.4.11 InjectDTMF(callID, DTMFString) 21
5.4.12 DTMFInjected(callID) 22
5.4.13 GetWAVFiles(extensionName, listWAV) 22
5.4.14 DeleteFile (extensionName, fileName) 23
5.4.15 TransferFile (extensionName, fileName, fileSize) 24
5.4.16 UpdateConfiguration (configFileName, size) 24
5.5 Subscription for Event Notifications 25
5.5.1 Subscribe(subID, subType, params) 25
5.5.2 Unsubscribe(subID) 27
5.5.3 MailArrived(subID, mailSettings) 28

5.5.4 CDRArrived(subID, cdrType, callDetails, mediaDetails) 29
5.5.5 DialogArrived(subID, dialogParamsList) 33
5.5.6 ExtListArrived(subID, extListParams)..... 35
5.5.7 PresenceArrived(subID, dnd)..... 36

Document Revision History

Rev.	Date	Description	Valid for SW	Valid for models
1.7	02-Apr-07	Initial release	4.1.22 and higher	Quadro IP PBXs
1.8	02-Aug-07	Values for the CDR type corrected (*CDRArrived*).	4.1.22 and higher	Quadro IP PBXs
1.9	01-Aug-07	The following functions added *GetSystemInfo,* *DialogArrived,* *PresenceArrived.*	4.1.22 and higher	Quadro IP PBXs
1.10	11-Oct-07	Descriptions of the following methods updated. *TransferCall,* *JoinCalls.* Notes added to sections 4 and 5.2.	4.1.35 and higher	Quadro IP PBXs
1.11	13-Nov-07	The following functions added: *CallOriginated,* *GetWAVList,* *WAVFileDelete,* *FileTransfer,* *ExtListArrived.* *Ext* parameter is added to *CallOriginated* function. Note is added to Section 5.3.3.	4.1.35 and higher	Quadro IP PBXs
2.1	04-Mar-09	The following functions changed: *GetWAVFiles,* *FileDelete,* *TransferFile,* The following function added: *UpdateConfiguration*	5.1.13 and higher	Quadro IP PBXs
2.2	23-Mar-09	The *authenticatedBy* parameter added to *CDRArrived* function	5.1.16 and higher	Quadro IP PBXs
2.3	20-Jun-09	*The ErrorCode parameter added to CallStateChanged* function.	5.1.16 and higher	Quadro IP PBXs
2.4	18-Nov-09	*The userName parameter added to CreateCall* function.	5.1.26 and higher	Quadro IP PBXs

1 Introduction

This document describes the Quadro's "CallControl" interface functions. Third-party software developers should use this document to write applications to control calls to/from the Quadro and receive notifications of the occurrence of certain Quadro events.

2 Accompanying Material

This document comes bundled with the following software and documents all of which can be found under the Downloads section of Epygi's Technical Support Center:

- **"CallControl" ActiveX Control –** the setup file for the "CallControl" ActiveX Control.
- **Quadro "CallControl" ActiveX Control** - a guide that describes the Quadro "CallControl" ActiveX control.
- **CallControl Pack** - the setup file for a package of sample applications that demonstrate the usage of the "CallControl" ActiveX control and 3pCC interface functions in different development environments.
- **Using Quadro's "CallControl" Interface** - a guide that describes the functionality of the sample applications included in the CallControl Pack.

3 Terms Used in This Document

The following abbreviations are used in this document:

CM – Call Manager of the Quadro

PUA – Presence User Agent of the Quadro

3pCC – Third-party call control (application)

4 Functional Description

The "CallControl" ActiveX Control is intended to enhance the Quadro's functionality. It allows a third-party call control application to remotely initiate calls from the Quadro, handle various calls in the Quadro and subscribe to certain event notifications from the Quadro.

The interface can be logically divided into the following four parts:

- Authentication
- Incoming Call Handling
- Control Initiated Call Handling
- Subscription for Event Notifications

The Quadro "CallControl" Interface is designed as an XML RPC server. All interface functions can be called by sending a corresponding XML RPC request to the server. For all XML RPC requests, an appropriate response will be sent back. The response will either contain the return value of the requested method or indicate an error occurred as described in the specification of XML RPC (http://www.xmlrpc.com/spec).

Please note:

- The "CallControl" interface uses XML RPC over TCP, not over HTTP as described in the specification.
- Connections can also be established using SSL (secure socket layer), in which case it is required that the secure connection mode for 3pCC applications is enabled on the Quadro.

For more information on how to enable secure connection mode on the Quadro, please refer to the Quadro's on-line help.

- All requests sent to the Quadro must end with double CRLF.
- The listening port of the Quadro for call control requests is *4849*. The default connection mode is nonsecure.

Please note: If a third-party call control application connects to the Quadro from the WAN interface, a filtering rule for the corresponding host should be created on the **Internet Uplink→Filtering Rules→*Call Control Access*** page to allow the application a remote access. Creating a filtering rule is not required if the firewall is not setup on the Quadro. For more information on how to create filtering rules on the Quadro, please refer to the Quadro's on-line help.

5 Syntax Used for Call Destinations

Parameters used in the methods to indicate a call destination must conform to the following syntax:

[CallType]:[CalledParty]

The following is where the [CallType] must be used with: "pbx", "sip", "pstn", "tel", "ar" or "vm".

- **pbx:** If the [CallType] is "pbx", the [CalledParty] must specify a valid extension of the Quadro.

  For example, pbx:11 specifies extension 11.

- **pstn:** If the [CallType] is "pstn", the [CalledParty] must specify a valid PSTN number prefixed by a FXO port number from which the call is to be placed: 0, 1, 2, etc. where 0 indicates any free port.

  For example, pstn:118005555555 specifies the PSTN destination 18005555555 that should be placed from FXO port 1.

- **sip:** If the [CallType] is "sip", the [CalledParty] must specify a valid SIP URI.

  For example, sip:someuser@sip.epygi.com specifies the SIP destination for a user registered on the sip.epygi.com SIP server.

- **tel:** If the [CallType] is "tel", the [CalledParty] must specify a valid routing number.

- **ar:** If the [CallType] is "ar", the [CalledParty] must be a valid extension of the Quadro. The call type "ar" specifies the extension's auto responder.

  For example, ar:11 specifies the auto responder of extension 11.

- **vm:** If the [CallType] is "vm", the [CalledParty] must be a valid extension of the Quadro. The call type "vm" specifies the extension's mailbox. Additionally, a particular voice mail ID can be specified in the [CalledParty] member followed by the extension and a semicolon.

  For example, vm:11 specifies the mailbox of extension 11. vm:12;5 specifies the mail with an ID 5 in the mailbox of extension 12.

The call types "ar" and "vm" are special call types.

- When the call type is "ar" (auto responder), CM makes a call to the specified extension's auto responder that by default plays a welcome message and stores a voice message.
- When the call type is "vm" (voice mailbox), CM makes a call to the specified extension's voice mail (similar to dialing *0) and plays the voice message specified by a unique ID. If no ID is specified, the CM will play the voice mail help to guide the caller thorough its hierarchy.

5.1 General Functions

5.1.1 GetSystemInfo (uniqueID, softVersion)

This function retrieves the system information from a specified Quadro.

Method	Services.CallProcessingService.getsysteminfo
Description	This method retrieves the system information from a specified Quadro.
Return value	String: Quadro's Unique identifier String: Quadro's software version
Parameters	String Unique Identifier
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.SystemService.getsysteminfo</methodName> <params> <param> <value><string>3771559303139</string></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>3771559303139</string></value> </param> <param> <value><string>02-4.1.31-028100</string></value> </param> <param> <value><string> 4.1.31 - Release</string></value> </param> </params> </methodResponse>

5.2 Authentication

This group of functions is used by a 3pCC application to receive authorization from the Quadro before calling any interface function. If the 3pCC is not authorized by the Quadro, any call to an interface function will be rejected and the TCP connection will be closed. Note that both the administrator and extensions are authorized to use the interface.

5.2.1 Authenticate(username, password)

Method	Services.CallProcessingService.authenticate
Description	This method authenticates a 3pCC application to use the “CallControl” interface.
Return value	Boolean: Authenticated or not String: In case of an error, the error message
Parameters	String: Authentication user name String: User password
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.authenticate</methodName> <params> <param> <value><string>admin</string></value> </param> <param> <value><string>123456789</string></value> </param> </params> </methodCall>
Response examples	<?xml version="1.0"?> <methodResponse> <params> <param> <value><boolean>1</boolean></value> </param> <param> <value><string>Authenticated.</string></value> </param> </params> </methodResponse>
	<?xml version="1.0"?> <methodResponse> <params> <param> <value><boolean>0</boolean></value> </param> <param> <value><string>Wrong username or password.</string></value> </param> </params> </methodResponse>

5.3 Incoming Call Handling

This group of functions allows handling incoming calls to a desired extension of the Quadro.

The following functions are included in this group.

5.3.1 IncomingCallHandling(extensionName, enable)

This function is called by a 3pCC application to enable or disable incoming call handling for an extension specified by the *extensionName* parameter. The *enable* parameter either enables or disables the call handling.

Method	Services.CallProcessingService.incomingcallhandling
Description	This method enables/disables incoming call handling for an extension of the Quadro.
Return value	None
Parameters	String: Extension number Boolean: Enable/disable flag (value 1 – enable, value 0 – disable)
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.incomingcallhandling</methodName> <params> <param> <value><string>11</string></value> </param> <param> <value><boolean>1</boolean></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>0</string></value> </param> </params> </methodResponse>

5.3.2 CallArrived(callID, from, to)

The CM calls this function when a new call is received from a specified called party (the *from* parameter) on a specified extension (the *to* parameter).

Method	Services.CallProcessingService.process
Description	This method determines how an incoming call should be handled based on a 3pCC’s call filtering rules and the current contextual state of the call. The method takes a single Struct, which is treated as an associative array of SIP header names and values. The “To”, “From” and “Contact” fields are mandatory. Their absence will generate an XML-RPC error. The method also takes the unique ID of the call, which must be used in the

	response message. The method returns an array of strings, the first of which indicates the desired outcome of the call. It is called the Action string. The semantics of subsequent strings in the array depend on the value of the Action string. Action strings are case insensitive and can be one of the following: • *Redirect*: subsequent strings are the destinations to redirect to. Additionally, a new caller ID and a display name can be specified to overwrite the respective parameters of the original call. • *Decline:* An optional reason may be present in the next string of the array. • *Accept:* Additionally, a new caller ID, and a display name can be specified to overwrite the original ones. Also, a boolean parameter “remotecontrol” can be specified if an incoming call must be accepted virtually for further handling by a 3pCC (e.g. to play a message for that call, to transfer the call, etc.).
Return value	Struct: Describes the action on an incoming call String: Call identifier
Parameters	String: Call identifier Struct: Treated as a dictionary of SIP header names and values

Request example	`<?xml version="1.0"?>`
	`<methodCall>`
	`<methodName>Services.CallProcessingService.process</methodName>`
	`<params>`
	`<param>`
	`<value><struct>`
	`<member>`
	`<name>From</name>`
	`<value><string>pbx:12</string></value>`
	`</member>`
	`<member>`
	`<name>To</name>`
	`<value><string>sip:720096011@sip.epygi.com:5060</string></value>`
	`</member>`
	`<member>`
	`<name>Contact</name>`
	`<value><string>pbx:12</string></value>`
	`</member>`
	`</struct></value>`
	`</param>`
	`<param>`
	`<value><string>54373701851916480</string></value>`
	`</param>`
	`</params>`
	`</methodCall>`

Response examples	`***** Redirect *****`
	`<?xml version="1.0"?>`
	`<methodResponse>`
	`<params>`
	`<param>`
	`<value><struct>`
	`<member>`
	`<name>action</name>`
	`<value><string>redirect</string></value>`
	`</member>`
	`<member>`
	`<name>target</name>`
	`<value><array><data>`

```
<value><string>sip:someuser@sip.epygi.com</string></value>
</data></array></value>
</member>
<member>
<name>callerid</name>
<value><string>3pCC</string></value>
</member>
<member>
<name>displayname</name>
<value><string>Accepted by 3pcc</string></value>
</member>
</struct></value>
</param>
<param>
<value><string>4972246272364275287</string></value>
</param>
</params>
</methodResponse>
***** Decline *****
<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value><struct>
<member>
<name>action</name>
<value><string>decline</string></value>
</member>
</struct></value>
</param>
<param>
<value><string>54375999659615371</string></value>
</param>
</params>
</methodResponse>
***** Accept *****
<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value><struct>
<member>
<name>action</name>
<value><string>accept</string></value>
</member>
<member>
<name>callerid</name>
<value><string>3pCC</string></value>
</member>
<member>
<name>remotecontrol</name>
<value><boolean>0</boolean></value>
</member>
</struct></value>
</param>
<param>
<value><string>54377923804855751</string></value>
</param>
</params>
</methodResponse>
```


5.3.3 CallOriginated(callID, from, to, pattern)

The CM calls this function when a new routing call is received from a specified calling party (the *from* parameter) to a specified destination (the *to* parameter). The *from* and *to* parameter conform to the syntax described in Syntax Used for Call Destinations.

The *ext* parameter contains extension number from which the call was made.

The *pattern* parameter contains the call routing pattern (for example, *8** or *[0-7]?*) that was used when setting up the call.

Please note: In order to receive notification from the Quadro concerning to the routing call, you should enable the "Check with 3PCC" option on the second page of the Call Routing Wizard, accessible from the Call Routing page in Telephony menu. Also you should subscribe for CR (call routing) event notifications (Please refer 5.5.1).

Method	Services.CallProcessingService.calloriginated
Description	This method determines how a routing call should be handled based on a 3pCC's call filtering rules and the current contextual state of the call. The "To", "From" and "Pattern" fields are mandatory. Their absence will generate an XML-RPC error. The method also takes the unique ID of the call, which must be used in the response message. The XML response returns an array of strings: In *Decline* case - the caller ID (see below) In *Accept* case - caller ID, subscriptionID (see below).
Return value	Struct: Describes the action on a routing call, in case of accept subscription id String: Call identifier
Parameters	String: Call identifier Struct: ext, from, to, pattern
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.calloriginated</methodName> <params> <param> <value><struct> <member> <name>Ext</name> <value><string>11</string></value> </member> <member> <name>From</name> <value><string>sip:someuser@sip.epygi.com</string></value> </member> <member> <name>To</name> <value><string>sip:quadroextension11@sip.epygi.com</string></value> </member> <member> <name>Pattern</name> <value><string>[0-7]?</string></value> </member> </struct></value> </param> <param> <value><string>15838606732868341</string></value> </param> </params> </methodCall>

Response examples	***** Decline ***** <?xml version="1.0"?> <methodResponse> <params> <param> <value><struct> <member> <name>action</name> <value><string>decline</string></value> </member> </struct></value> </param> <param> <value><string>15838606732868341</string></value> </param> </params> </methodResponse> ***** Accept ***** <?xml version="1.0"?> <methodResponse> <params> <param> <value><struct> <member> <name>action</name> <value><string>accept</string></value> </member> <member> <name>subscrid</name> <value><string>18446744073632098529</string></value> </member> </struct></value> </param> <param> <value><string>15838606732868341</string></value> </param> </params> </methodResponse>

5.4 Control Initiated Call Handling

These methods can be used to create and manipulate calls of the CM.

The following functions are included in this group.

5.4.1 CreateCall(callID, from, to, displayName, userName)

This function is called by a 3pCC to create a new call using the settings of a specified extension (the *from* parameter) to a specified destination (the *to* parameter).

The *from* parameter must specify an existing extension or an attendant or a "system" reserved extension.

The *to* parameter must conform to the syntax described in Syntax Used for Call Destinations.

The *displayName* parameter specifies the alternative display name of the *from* parameter.

The *userName* parameter specifies the alternative callerID of the *from* parameter.

Method	Services.CallProcessingService.createcall
Description	This method creates a new call to a specified destination using the settings of a specified extension.
Return value	String: Call identifier
Parameters	String: Unique identifier of the call (can be used later in other methods to specify a call) String: Extension String: Called destination String (optional):Extension's alternative display name.
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.createcall</methodName> <params> <param> <value><string>18446744072624214637</string></value> </param> <param> <value><string>00</string></value> </param> <param> <value><string>pbx:11</string></value> </param> </params> </methodCall> //with display name parameter <methodCall> <methodName>Services.CallProcessingService.createcall</methodName> <params> <param> <value><string>18446744072624214637</string></value> </param> <param> <value><string>00</string></value> </param> <param> <value><string>pbx:11</string></value> </param> <param> <value><string>test name</string></value> </param> <param> <value><string>12345</string></value>

	</param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>18446744072624214637</string></value> </param> </params> </methodResponse>

5.4.2 CloseCall(callID)

This function is called by a 3pCC to close a call specified by its unique ID (the *callID* parameter).

Method	Services.CallProcessingService.closecall
Description	This method closes a call previously created by the **CreateCall** method.
Return value	String: Call identifier (see below)
Parameters	String: Call identifier returned by the **CreateCall** method
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.closecall</methodName> <params> <param> <value><string>18446744072624214637</string></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>18446744072624214637</string></value> </param> </params> </methodResponse>

5.4.3 TransferCall(called, transferParty)

This function is called by a 3pCC to transfer a call specified by its unique ID (the *callID* parameter) to another destination specified in the *transferParty* parameter. The syntax of the *to* parameter is described in Syntax Used for Call Destinations.

Please note: Before transferring a call, ensure that all current playbacks are stopped; otherwise call transfer will fail (the method will return FALSE).

Method	Services.CallProcessingService.transfercall
Description	This method transfers a call previously created by the **CreateCall** method.
Return value	String: Call identifier (see below)
Parameters	String: Call identifier returned by the **CreateCall** method String: Transfer destination
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.transfercall</methodName> <params>

	<param> <value><string>1844674407343615499 3</string></value> </param> <param> <value><string>pbx:12</string></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1844674407343615499 3</string></value> </param> </params> </methodResponse>

5.4.4 CallStateChanged(callID, callState, errorCode)

The CM calls this function when the state of a call has changed.

The *callState* parameter contains the new state of the call. It can take any of the following values:

0 (*Trying*) – The CM is trying to make a call.
1 (*Ringing*) – The remote party is ringing the phone.
2 (*Confirmed*) – The remote party has accepted the call.
3 (*Closed*) – The call has been closed by the remote party.
4 (*Joined*) – The call has been joined to another call.
5 (*Unjoined*) – The call has been un-joined from another call.
6 (*Error*) – The call caused an error for some reason (refer to *errorCode*)

The *errorCode* parameter contains the error reason, if *callState* is “Error”. It can take any of the following values

0 (LineBusy) – The remote party is busy.
1 (TemporaryUnavailable) – The remote party temporarily can’t accept the call.
2 (UserNotFound) – The remote party not found.
3 (*MediaNotSupported*) – Codec of the caller and callee are incompatible.
4 (InappropriateCommand) – Inappropriate command arrived.
5 (Generic) – A generic error occurred.

Method	Services.CallProcessingService.callstatechanged
Description	This method informs the 3pCC about a change in the call state. The second parameter of the method specifies the new state of the call. The third parameter of the method specifies the error reason if callState parameter is “Error”, otherwise should be ignored
Return value	String: Call identifier (see below)
Parameters	String: Call identifier returned by the **CreateCall** method Integer: The new state of the call Integer: The error code in “Error” callState

Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.callstatechanged</methodName> <params> <param> <value><string>1844674407343615499 3</string></value> </param> <param> <value><int>2</int></value> </param> </param> <param> <value><int>0</int></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1844674407343615499 3</string></value> </param> </params> </methodResponse>

5.4.5 JoinCalls(callID1, callID2)

This function is called by a 3pCC to join two calls. Calling this function forces the CM to connect called parties of the specified calls with each other.

Please note: Before joining calls, ensure that all current playbacks are stopped; otherwise call transfer will fail (the method will return FALSE).

Method	Services.CallProcessingService.joincalls
Description	This method joins two calls previously created by the **CreateCall** method.
Return value	String: Call identifier (see below)
Parameters	String: First call identifier returned by the **CreateCall** method String: Second call identifier returned by the **CreateCall** method
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.joincalls</methodName> <params> <param> <value><string>1245795375183</string></value> </param> <param> <value><string>1245826141213</string></value> </param> </params> </methodCall>

Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1245795375183</string></value> </param> </params> </methodResponse>

5.4.6 UnjoinCalls(callID1, callID2)

This function is called by a 3pCC to un-join two previously joined calls.

Method	Services.CallProcessingService.unjoincalls
Description	This method separates two calls previously joined by the **JoinCalls** method.
Return value	String: Call identifier (the first ID described below)
Parameters	String: First call identifier returned by the **CreateCall** method String: Second call identifier returned by the **CreateCall** method
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.unjoincalls</methodName> <params> <param> <value><string>1245795375183</string></value> </param> <param> <value><string>1245826141213</string></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1245795375183</string></value> </param> </params> </methodResponse>

5.4.7 PlayMessage(callID, messageFileName, playCount)

This function is called by a 3pCC to start playing a voice message for a call.

Method	Services.CallProcessingService.playmessage
Description	This method plays a voice message to the called party. The *playCount* optional parameter specifies the number of times to play the message. The **messageFileName** parameter specifies the file name of either a system voice message or a custom voice message of an attendant on behalf of which the call was initiated (The parameter *from* in the **CreateCall** method).
Return value	String: Call identifier (see below)

Parameters	String: Call identifier returned by the **CreateCall** method String: Message file name Integer: Number of playbacks **Please note:** To stop the current playback, call the method with the *messageFilneName* set to the null string and *playCount* set to *0*.
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.playmessage</methodName> <params> <param> <value><string>1245795375183</string></value> </param> <param> <value><string>holdmusic.wav</string></value> </param> <param> <value><int>2</int></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1245795375183</string></value> </param> </params> </methodResponse>

5.4.8 MessagePlayed(callID)

The CM calls this function when the playback of a message for a call is complete.

Method	Services.CallProcessingService.messageplayed
Description	This method is called when the playback of a message specified by the **PlayMessage** method is complete.
Return value	String: Call identifier (see below)
Parameters	String: Call identifier returned by the **CreateCall** method
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.messageplayed</methodName> <params> <param> <value><string>1246702030612</string></value> </param> </params> </methodCall>

Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1246702030612</string></value> </param> </params> </methodResponse>

5.4.9 EnableDtmfDetection(callID, enable)

This function is called by a 3pCC to switch on/off DTMF detection for a call. The *enable* parameter specifies the on/off state of DTMF detection.

Method	Services.CallProcessingService.enabledtmfdetection
Description	This method enables/disables DTMF detection for a call specified by a call identifier.
Return value	String: Call identifier (see below)
Parameters	String: Call identifier returned by the **CreateCall** method Boolean: Enable/Disable flag (value 1 – enable, value 0 – disable)
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.enabledtmfdetection</metho dName> <params> <param> <value><string>1246702030612</string></value> </param> <param> <value><boolean>1</boolean></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1246702030612</string></value> </param> </params> </methodResponse>

5.4.10 DtmfDigitDetected(callID, dtmfdigit)

The CM calls this function if a DTMF digit (button press) is detected. *dtmfdigit* contains the ASCII code of a detected digit.

Method	Services.CallProcessingService.dtmfdigitdetected
Description	This method is called when a DTMF digit is detected for a call.
Return value	String: Call identifier (see below)

Parameters	String: Call identifier returned by the **CreateCall** method Integer: Detected DTMF digit
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.dtmfdigitdetected</methodName> <params> <param> <value><string>1247273931739</string></value> </param> <param> <value><int>56</int></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1247273931739</string></value> </param> </params> </methodResponse>

5.4.11 InjectDTMF(callID, DTMFString)

This function is called by a 3pCC to start injecting dtmf sequence for a call.

Method	Services.CallProcessingService.injectdtmf
Description	This method injects a dtmf sequence to the called party. The **DTMFString** parameter specifies the DTMF sequence.
Return value	String: Call identifier (see below)
Parameters	String: Call identifier returned by the **CreateCall** method String: dtmf sequence
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.injectdtmf</methodName> <params> <param> <value><string>1245795375183</string></value> </param> <param> <value><string>01234</string></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1245795375183</string></value> </param> </params> </methodResponse>

5.4.12 DTMFInjected(callID)

The CM calls this function when the injection of a dtmf sequence for a call is complete.

Method	Services.CallProcessingService.dtmfinjected
Description	This method is called when the dtmf sequence specified by the **InjectDTMF** method has been injected.
Return value	String: Call identifier (see below)
Parameters	String: Call identifier returned by the **CreateCall** method
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.dtmfinjected</methodName> <params> <param> <value><string>1246702030612</string></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1246702030612</string></value> </param> </params> </methodResponse>

5.4.13 GetWAVFiles(extensionName, listWAV)

This function is called by a 3pCC to get the list of system messages (WAV files) available for a specified extension on the Quadro (the *extensionName* parameter).

If *extensionName* is empty, the function will return the list of universal extension recordings available on the system..

Method	Services.SystemService.getfileslist
Description	This function returns the list of system messages available at a specified extension.
Return value	Array: WAV file list (see below)
Parameters	String: Unique identifier String: extension. Leave this parameter empty when requesting a universal extension WAV file list.
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.SystemService.getfileslist</methodName> <params> <param> <value><string>18446744072156213989</string></value> </param> <param> <value><string>12</string></value> </param> </params> </methodCall>

Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>18446744072156213989</string></value> </param> <param> <value><string>12</string></value> </param> <param> <value><array><data> <value><string>first.wav</string></value> <value><string>second.wav</string></value> <value><string>third.wav</string></value> </data></array></value> </param> </params> </methodResponse>

5.4.14 DeleteFile (extensionName, fileName)

This function is called by a 3pCC to delete a specified message (WAV file, parameter *fileName*) from an extension's message list on the Quadro (according to *extensionName*). For the description of the *extensionName* parameter, see the *GetWAVFiles* function.

Method	Services.SystemService.deletefile
Description	This function deletes a specified WAV file from a specified extension’s message list.
Return value	Int: Result – 0 request failed or 1 – request success
Parameters	String: Unique identifier String: Extension. Leave this parameter empty when deleting universal extension’s file. String: File name
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.SystemService.deletefile</methodName> <params> <param> <value><string>2273583481103</string></value> </param> <param> <value><string>12</string></value> </param> <param> <value><string>inuse.wav</string></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>2273583481103</string></value> </param> <param> <value><boolean>1</boolean></value> </param> </params> </methodResponse>

5.4.15 TransferFile (extensionName, fileName, fileSize)

This function is called by a 3pCC to upload a specified WAV file from a local computer to a specified extension's message list.

Method	Services.SystemService.beginfiletransfer
Description	This method upload the WAV file from a local computer to specified extension's message list (the *extensionName* parameter). The *fileName* parameter specifies the WAV file name. The *fileSize* parameter specifies the WAV file size in bytes. After sending the request, wait for "OK" or "FAIL" string from Quadro: "OK " means the Quadro is ready to receive the WAV file, "FAIL" – error occurred. In successful case WAV file should be sent to Quadro.
Return value	Int: Result 1 – successfully uploaded and 0 - failed.
Parameters	String: Unique identifier String: Extension. Leave this parameter empty when uploading to universal extension. String: File name Int: File size
Requests example	<?xml version="1.0"?> <methodCall> <methodName>Services.SystemService.beginfiletransfer</methodName> <params> <param> <value><string>1844674407193815730</string></value> </param> <param> <value><string></string></value> </param> <param> <value><string>chord.wav</string></value> </param> <param> <value><int>97016</int></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1844674407193815730</string></value> </param> <param> <value><boolean>1</boolean></value> </param> </params> </methodResponse>

5.4.16 UpdateConfiguration (configFileName, size)

This function is called by a 3pCC to upload the content of legible configuration file from a local computer to the corresponding configuration files on the Quadro..

Method	Services.SystemService.configurationupdate

Description	This method uploads the content of legible configuration file from a local computer to the corresponding configuration files on the Quadro. The *configFileName* parameter specifies the configuration file name. The *size* parameter specifies the configuration file size in bytes. After sending the request, wait for “OK” or “FAIL” string from Quadro: “OK “ means the Quadro is ready to receive the configuration file, “FAIL” – error occurred. In successful case the configuration file should be sent to Quadro.
Return value	String: Unique identifier (see below) Boolean: 1 – successfully uploaded and 0 - failed
Parameters	String: Unique identifier String: Configuration file name Integer: Configuration file size in bytes
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.SystemService.configurationupdate</methodName> <params> <param> <value><string>18446744072637669126</string></value> </param> <param> <value><string>configFile.txt</string></value> </param> <param> <value><int>285</int></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>18446744072637669126</string></value> </param> <param> <value><boolean>1</boolean></value> </param> </params> </methodResponse>

5.5 Subscription for Event Notifications

These methods can be used to subscribe/unsubscribe for certain event notifications from the CM.

The following functions are included in this part.

5.5.1 Subscribe(subID, subType, params)

This function is called by a 3pCC application to subscribe for particular event notifications from the Quadro. The *subtype* parameter specifies the subscription type and the *params* parameter specifies the structure specific to the subscription type.

The following types of subscription can be specified (parameter *subType*):

1 – Specifies a subscription for a new voice mail notification. The *params* parameter for this subscription type should contain a member called “extension”, which must specify a valid extension of the Quadro.

2 – Specifies a subscription for CDR (call detailed report) notifications. The *params* parameter must be empty in this case.

3 – Specifies a subscription for notifications of call state changes. The *params* parameter for this subscription type should contain a member called “extension”, which must specify a valid extension of the Quadro.

4 – Specifies a subscription for notifications of the dnd mode changes. The *params* parameter for this subscription type should contain a member called “extension”, which must specify a valid extension of the Quadro.

5 - Specifies a subscription for CR (call routing) notifications. The *params* parameter must be empty in this case.

6 - Specifies a subscription for notifications of changes to the extensions list. The *params* parameter must be empty in this case.

If a subscription is rejected, the optional errorCode return value will describe the reason for rejection. The *errorCode* return value may take any of the following possible values:

0 – internal error
1 – incorrect subscription parameters
2 – incorrect subscription destination (extension)
3 – subscription id already exists

Method	Services.CallProcessingService.subscribe
Description	This method subscribes for particular event notifications from the Quadro.
Return value	String: Subscription identifier (see below) Boolean: Subscription confirmed or not Integer (optional): Error code if subscription is rejected
Parameters	String: Unique identifier of subscription Integer: Subscription type Struct: Subscription parameters
Request example	*** Subscription for new voice mail *** <?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.subscribe</methodName> <params> <param> <value><string>18446744071962851211</string></value> </param> <param> <value><int>1</int></value> </param> <param> <value><struct> <member> <name>extension</name> <value><string>11</string></value> </member> </struct></value> </param> </params> </methodCall> *** Subscription for CDR *** <?xml version="1.0"?> <methodCall>

	<methodName>Services.CallProcessingService.subscribe</methodName> <params> <param> <value><string>1844674407196285121 1</string></value> </param> <param> <value><int>2</int></value> </param> </params> </methodCall> *** Subscription for CR *** <?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.subscribe</methodName> <params> <param> <value><string>1844674407363209852 9</string></value> </param> <param> <value><int>5</int></value> </param> </params> </methodCall>
Response example	*** Subscription confirmed *** <?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1844674407196285121 1</string></value> </param> <param> <value><boolean>1</boolean></value> </param> <param> <value><int>0</int></value> </param> </params> </methodResponse> *** Subscription rejected *** <?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1844674407196285121 1</string></value> </param> <param> <value><boolean>0</boolean></value> </param> <param> <value><int>2</int></value> </param> </params> </methodResponse>

5.5.2 Unsubscribe(subID)

This function is called by a 3pCC to unsubscribe from event notifications.

Method	Services.CallProcessingService.unsubscribe

Description	This method unsubscribes from event notifications.
Return value	String: Unique identifier of subscription
Parameters	String: Unique identifier of subscription
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.unsubscribe</methodName> <params> <param> <value><string>1255550503495</string></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1255550503495</string></value> </param> </params> </methodResponse>

5.5.3 MailArrived(subID, mailSettings)

This function is called by the PUA to indicate that a new voice mail has arrived for a subscription specified by its unique ID (parameter subID). The *mailSettings* parameter specifies the voice mail. The following members are available in the *mailSettings* structure:

From is the voice mail sender's address that must conform to the syntax described in Syntax Used for Call Destinations.
Time is the voice mail receipt time in ISO8601 format.
Duration is the duration of the voice mail in seconds.
MailID is the unique ID of the voice mail (can be used later in the **CreateCall** method).

Method	Services.CallProcessingService.mailarrived
Description	This method notifies the subscriber of a new voice mail arrival.
Return value	String: Subscription identifier (see below)
Parameters	String: Unique identifier of subscription Struct: Voice mail settings
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.mailarrived</methodName> <params> <param> <value><string>1255550503495</string></value> </param> <param> <value><struct> <member> <name>From</name> <value><string>sip:11085@sip.epygi.loc</string></value> </member> <member> <name>Time</name>

	<value><dateTime.iso8601>2006-09-08 10:52:42</dateTime.iso8601></value> </member> <member> <name>Duration</name> <value><int>2</int></value> </member> <member> <name>MailID</name> <value><int>1</int></value> </member> </struct></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>1255550503495</string></value> </param> </params> </methodResponse>

5.5.4 CDRArrived(subID, cdrType, callDetails, mediaDetails)

This function is called by the CM to indicate that a new CDR has been created for a subscription specified by its unique ID (parameter subID). The *cdrtype* parameter (integer) specifies the type of the new CDR. The following types are available:

0x01 – CDR specifies a successful call.

0x02 – CDR specifies a missed call.

0x04 – CDR specifies a failed outgoing call.

The parameter **callDetails** specifies a structure describing the call. The structure consists of the following members:

From(string) – call initiator’s number.

To(string) – call destination number.

Start Time(integer) – call start time (the number of seconds counted from 00:00 01/01/1970).

Duration(integer) – call duration (in seconds).

CallGUID(string) – call globally unique identifier.

Details(string) – some call info string (in case of failed call, contains error text).

AuthenticatedBy(string) – specifies information about the callers that passed an authentication on the Quadro as configured in the Local AAA Table.

CIRCaller(boolean) – specifies that the call initiator has the CIR service enabled.

The parameter **mediaDetails** specifies a structure describing the call’s media in case of successful CDR types. In all other cases **mediaDetails** is missing.

RTP_Tx_Pack(integer) – the number of transmitted RTP packets.

RTP_Tx_Codec(string) – the codec name for transmitted RTP stream.

RTP_Tx_Pack_sz(integer) – the size of transmitted RTP packets (payload).

RTP_Rx_Pack(integer) – the number of received RTP packets.

RTP_Rx_Codec(string) – the codec name for received RTP stream.

RTP_Rx_Pack_sz(integer) – the size of received RTP packet (payload).

RTP_Rx_Lost(integer) – the number of lost RTP packets from received stream.

RTP_Rx_Jitter(integer) - the inter-arrival jitter is an estimate of the statistical variance of the RTP data packet inter-arrival time, measured in timestamp units. The inter-arrival jitter is defined to be the mean deviation (smoothed absolute value) of the difference D in packet spacing at the receiver compared to the sender for a pair of packets. If Si is the RTP timestamp from packet i, and Ri is the arrival time in RTP timestamp units for packet i, then for two packets i and j, D may be expressed as:

D(i,j) = (Rj - Ri) - (Sj - Si) = (Rj - Sj) - (Ri - Si)
J(i) = J(i-1) + (|D(i-1,i)| - J(i-1))/16, where J(i) is Rx Jitter for packet i.

RTP_Rx_max_delay(integer) - maximum variance (absolute value) of actual arrival time of the RTP data packet compared to the estimated arrival time, measured in milliseconds. If Si is the RTP timestamp from packet i, and Ri is the arrival time in RTP timestamp units for packet i, then variance for packet i may be expressed as following:

V(i) = |(Ri - R1) - (Si - S1)| = |(Ri - Si) - (R1 - S1)|
Rx Maximum Delay = max V(i) / 8

RTP_Delay_Inc_Count(integer) - indicates the number of times the delay increased in jitter buffer during the call.

RTP_Delay_Dec_Count(integer) - indicates the number of times the delay decreased in jitter buffer during the call.

(The following members of the structure **mediaDetails** are non-zero only if the CDR specifies a call which contains the T.38 (fax) stream).

Fax_Tx_Pack(integer) – the number of transmitted T.38 packets.

Fax_Tx_Duble_Pack(integer) – the number of re-transmitted T.38 packets.

Fax_Rx_Pack(integer) – the number of received T.38 packets.

Fax_Rx_Lost(integer) – the number of lost T.38 packets from received stream.

Fax_Rx_Bad(integer) – the number of bad T.38 packets from received stream.

Fax_Rx_Duble_Pack(integer) – the number of duplicated T.38 packets from received stream.

Method	Services.CallProcessingService.cdrarrived
Description	This method notifies the subscriber of a new CDR creation.
Return value	String: Subscription identifier (see below)
Parameters	String: Unique identifier of subscription Integer: CDR type Struct: call details Struct: media details (optional)
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.cdrarrived</methodName> <params> <param>

```
<value><string>318264022234</string></value>
</param>
<param>
<value><int>1</int></value>
</param>
<param>
<value><struct>
<member>
<name>From</name>
<value><string>2410103@sip.epygi.loc</string></value>
</member>
<member>
<name>To</name>
<value><string>200</string></value>
</member>
<member>
<name>Start Time</name>
<value><string>1174026593</string></value>
</member>
<member>
<name>Duration</name>
<value><int>5</int></value>
</member>
<member>
<name>CallGUID</name>
<value><string>069a0536-2e5d-411d-98f5-06f4abd6b466</string></value>
</member>
<member>
<name>Details</name>
<value><string></string></value>
</member>
<member>
<name>AuthenticatedBy</name>
<value><string>1234</string></value>
</member>
<member>
<name>CIRCaller</name>
<value><boolean>0</boolean></value>
</member>
</struct></value>
</param>
<param>
<value><struct>
<member>
<name>RTP_Tx_Pack</name>
<value><int>265</int></value>
</member>
<member>
<name>RTP_Tx_Codec</name>
<value><string>PCMU</string></value>
</member>
<member>
<name>RTP_Tx_Pack_sz</name>
<value><int>160</int></value>
</member>
<member>
<name>RTP_Rx_Pack</name>
<value><int>247</int></value>
</member>
<member>
<name>RTP_Rx_Codec</name>
<value><string>PCMU</string></value>
</member>
<member>
```

	<name>RTP_Rx_Pack_sz</name> <value><int>160</int></value> </member> <member> <name>RTP_Rx_Lost</name> <value><int>0</int></value> </member> <member> <name>RTP_Rx_Jitter</name> <value><int>4</int></value> </member> <member> <name>RTP_Rx_max_delay</name> <value><int>5</int></value> </member> <member> <name>RTP_Delay_Inc_Count</name> <value><int>0</int></value> </member> <member> <name>RTP_Delay_Dec_Count</name> <value><int>2</int></value> </member> <member> <name>Fax_Tx_Pack</name> <value><int>0</int></value> </member> <member> <name>Fax_Tx_Duble_Pack</name> <value><int>0</int></value> </member> <member> <name>Fax_Rx_Pack</name> <value><int>0</int></value> </member> <member> <name>Fax_Rx_Lost</name> <value><int>0</int></value> </member> <member> <name>Fax_Rx_Bad</name> <value><int>0</int></value> </member> <member> <name>Fax_Rx_Duble_Pack</name> <value><int>0</int></value> </member> </struct></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>318264022234</string></value> </param> </params> </methodResponse>

5.5.5 DialogArrived(subID, dialogParamsList)

This function is called by the PUA to indicate that call-related data have been changed for a subscription specified by its unique ID (parameter *subID*). The *dialogParamsList* parameter specifies the list of the *dialogParams* structures. The following members are available in the *dialogParams* structure:

The *CallID* parameter specifies the call.

The *Local* is the extension whose call state change notifications the 3pCC application is subscribed to.

The *Remote* parameter specifies the call /destination number.

The *CallType* parameter specifies the call type. The following values are available:

0 – PBX

1 – SIP

2 – PSTN

3 - H323

4 – Call Routing

The *Dir* parameter specifies the call direction. The following values are available:

0 – initiator – The call has been made by the local party.

1 – recipient – The call has been received by the local party.

2 – no direction specified – This value represents no information about the call direction and is used, for example, if the local party has gone off-hook.

The *DlgState* parameter specifies the call state**.** The following values are available:

0 – trying

1 – proceeding (ringing state)

2 – early (ringing state)

3 – terminated

4 – confirmed

5 – no state specified

The *Event* parameter specifies the call state event. The following values are available:

0 – cancelled

1 – rejected

2 – replaced

3 – local bye

4 – remote bye

5 – error

6 – timeout

7 – no event specified

All values are available only with the *terminated* value for the call state specified in the *DlgState* parameter.

The *Dur* parameter contains the amount of time, in seconds, since the call was created.

The *Replaces* parameter specifies the identifier of a new (replaced) call which was created as a result of an invitation with a Replaces header.

The *CallState* parameter specifies the current state of the call. The following values are available:

0 – no value specified

1 – the call is in the queue

2 – the call is established with an extension

3 – the call is established with a voice mail

All values are available with the *confirmed* and *early states* for the *dialogState* parameter.

The *ReferredBy* parameter is used to correlate a new call with a REFER request which triggered it.

Method	Services.CallProcessingService.dlgarrived
Description	This method notifies the subscriber of a call info changes.
Return value	String: Subscription identifier (see below)
Parameters	String: Unique identifier of subscription Struct: call parameters
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.dlgarrived</methodName> <params> <param> <value><string>3738587903410</string></value> </param> <param> <value><array><data> <value><struct> <member> <name>CallID</name> <value><string>38839191689583992</string></value> </member> <member> <name>Local</name> <value><string>28111@192.168.0.209:5060</string></value> </member> <member> <name>Remote</name> <value><string>14</string></value> </member> <member> <name>CallType</name> <value><int>0</int></value> </member> <member> <name>Dir</name> <value><int>1</int></value> </member> <member> <name>DlgState</name> <value><int>4</int></value> </member> </struct></value> </data></array></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>3738587903410</string></value> </param> </params> </methodResponse>

5.5.6 ExtListArrived(subID, extListParams)

This function is called by the PUA to indicate that the extension list has been changed for a subscription specified by its unique ID (parameter *subID*). The *extListParams* parameter specifies the list of the extensions’ settings in the following order: “Ext, Display, SipAddress, Type, Line”.

Method	Services.CallProcessingService.extlistarrived
Description	This method notifies the subscriber of changes to the extension list.
Return value	String: Subscription identifier (see below)
Parameters	String: Unique identifier of subscription Struct: Call parameters
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.extlistarrived</methodName> > <params> <param> <value><string>3698657756129</string></value> </param> <param> <value><array><data> <value><struct> <member> <name>Ext</name> <value><string>11</string></value> </member> <member> <name>Display</name> <value><string></string></value> </member> <member> <name>SipAddress</name> <value><string>716409611@sip.epygi.com:5060</string></value> </member> <member> <name>Type</name> <value><int>1</int></value> </member> <member> <name>Line</name> <value><int>1</int></value> </member> </struct></value> <value><struct> <member> <name>Ext</name> <value><string>12</string></value> </member> <member> <name>Display</name> <value><string></string></value> </member> <member> <name>SipAddress</name> <value><string>716409612@sip.epygi.com:5060</string></value> </member> <member>

	<name>Type</name> <value><int>1</int></value> </member> <member> <name>Line</name> <value><int>2</int></value> </member> </struct></value> </data></array></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>3698657756129</string></value> </param> </params> </methodResponse>

5.5.7 PresenceArrived(subID, dnd)

This function is called by the PUA to indicate a change in the dnd mode for a subscription specified by *subscrID*. The *dnd* parameter specifies whether the dnd has been enabled (TRUE) or disabled (FALSE).

Method	Services.CallProcessingService.prsarrived
Description	This method notifies the subscriber of a dnd mode changes.
Return value	String: Subscription identifier (see below)
Parameters	String: Unique identifier of subscription Boolean: dnd mode
Request example	<?xml version="1.0"?> <methodCall> <methodName>Services.CallProcessingService.prsarrived</methodName> <params> <param> <value><string>3738166609078</string></value> </param> <param> <value><boolean>0</boolean></value> </param> </params> </methodCall>
Response example	<?xml version="1.0"?> <methodResponse> <params> <param> <value><string>3738166609078</string></value> </param> </params> </methodResponse>