Client Application Protocols

Interface Specification

Release 17.0

Document Version 1

220 Perry Parkway
Gaithersburg, MD USA 20877
Tel +1 301.977.9440
Fax +1 301.977.8846

WWW.BROADSOFT.COM

BroadWorks® Guide

Document Revision History

Release	Version	Reason for Change	Date
14.0	1	Created new document by merging all CAP protocols into this document (for example CCC2 and CAP-C). Made other improvements. Note that the document name was changed. Specifically, “Protocol” was changed to “Protocols” in the document name.	November 14, 2007
14.0	1	Added acdState and joinCallCenter to more profileUpdate variants.	November 16, 2007
14.0	1	Edited and published document.	November 20, 2007
14.0	2	Fixed bugs in *profileUpdate* (added *locationCode* in initial update for *AttendantConsole*, and fixed an example).	December 5, 2007
14.0	2	Added Release 14.sp4 changes.	December 14, 2007
14.0	2	Edited changes and published document.	December 19, 2007
14.0	3	Added the recallType data element to the callUpdate message.	December 21, 2007
14.0	3	Edited changes and published document.	January 3, 2007
14.0	4	Added Release 14.sp5 changes.	June 19, 2008
15.0	1	Added Release 15.0 changes.	June 19, 2008
15.0	1	Edited changes and published document.	July 29, 2008
16.0	1	Added Release 16.0 changes.	May 20, 2009
16.0	1	Added new “CallUpdated” *updateReason* in *queueUpdate* message.	July 6, 2009
16.0	1	Edited changes and published document.	July 9, 2009
17.0	1	Added Release 17.0 changes.	March 4, 2010
17.0	1	Edited changes and published document.	April 5, 2010

Table of Contents

Table of Figures

1 Summary of Changes

This section describes the changes to this document for each release and document version.

1.1 Changes for Release 17.0, Document Version 11

This version of the document includes the following changes:

- Changed CAP version from Release 16.0 to 17.0.
- Clarified that the OCS supports Session Data Replication in section *3.1 CAP-C Architecture*.
- New *supervisorId* element for the registerAuthentication, responseAuthentication, registerRequest, registerResponse, and acknowledgement messages.
- Added the following profileUpdate elements: *hotelingHost*, *unavailabilityCode* and *wrapUpCallCenterCallId*.
- Added the following callUpdate elements: *holdReminder* and *callCenter* (with its several sub-elements).
- Added the *silentlyMonitored* element of the sessionUpdate message.
- Added several new call center actions, as well as the callActionResponse message.
- Added the "Promote" queueAction *actionType* and the associated *priority* element.
- In the queueUpdate message, added several *queueCall* sub-elements, as well as the "CallPromoted" *updateReason*.
- Added two more messages: queueInfoRequest and queueInfoResponse.

1.2 Changes for Release 16.0, Document Version 1

This version of the document includes the following changes:

- Changed CAP version from Release 15.0 to 16.0.
- The callUpdate message's *releaseCause* field can now take value 14 when a Prepaid user's call is released due to insufficient credits.
- For Click-To-Dial calls, the call ID is now the same for both call legs (the BroadWorks Originator call leg, and the originator call leg). This affects section *7.2.1 Interaction 1 – Placing a Call*.
- Added the *remoteTelUri* field to the callUpdate message, for both CallClient and AttendantConsole user types.
- Added the *callStartTime* and *callAnswerTime* fields to the callUpdate message, for both CallClient and AttendantConsole user types.
- Added the *acdStateTime* and the *cwt* fields to the profileUpdate messages. This affects the initial and subsequent updates for both CallClient and AttendantConsole users.
- Added the *mandatoryEntrance*, *bouncedCall*, *reorderedCall,* and *preservedWaitTime* fields to the queueUpdate message.
- Added the "CallUpdated" *updateReason* to the queueUpdate message.
- Added the queueProfileUpdate message.

1.3 Changes for Release 15.0, Document Version 1

This version of the document includes the following changes:

- Changed CAP version from 14.0 to 15.0.
- Removed the (unused) *bphone* element from the sessionUpdate message.

1.4 Changes for Release 14.sp5

This version of the document includes the following changes:

- Added the *simring* field to the profileUpdate messages. This affects the initial and subsequent updates for the CallClient user type (but not AttendantConsole).

1.5 Changes for Release 14.sp4

This version of the document includes the following changes:

- Added *countryCode* and *nationalPrefix* fields to the profileUpdate messages. This affects the initialUpdate for both CallClient and AttendantConsole, but not subsequent updates.
- Added the *cfa* element and the *cfaDestination* attribute to the (subsequent) AttendantConsole profileUpdate message.
- Added the *CallPark*, *GroupCallPark*, and *CallParkRetrieve* actions to the callAction command. Additionally, the "UserId" value is added for the *actionParamName* data element, for use by the *CallPark* and *CallParkRetrieve* actions.
- Added the *recallType* data element to the callUpdate message for CallClient users.

1.6 Changes for Release 14.sp3

This version of the document includes the following changes:

- Added *applicationId* field to the following messages:
  - serverStatusRequest (optional)
  - callAction
  - externalNotify
  - queueUpdate
  - queueAction
- Corrected "datagram" command description that does not require the *userUid* attribute. It uses *id* instead. This command was also marked as deprecated.
- Merged in CCC2 protocol. Reorganized and added new material.
- Added the following elements to the profileUpdate message:
  - acdState
  - callCenter element
  - id (for Call Center)
  - joinCallCenter
- Added the following elements to the callUpdate message:

- callCenterUserId
- callType

1.7 Changes for Release 14.sp2

This version of the document includes the following changes:

- Added “PrivateUser” failure reason to MonitoringUsersResponse.
- Added “monitoredUserId” element to the callUpdate, profileUpdate, and sessionUpdate messages.

1.8 Changes for Release 14.sp1

This version of the document includes the following changes:

Starting with Release 14.sp1 (with patch AP.as.14.sp1.194.ap47396) and greater, the client-originated messages can use the *id* attribute to uniquely identify the user, instead of *userUid*. Both are accepted, but the *id* is preferred (the *userUid* has no reason to be exposed on the Client Application Protocol (CAP) interface, and is likely to disappear in a future release). The following messages are affected:

- acknowledgement
- unRegister
- callAction
- externalNotify
- monitoringUsersRequest
- queueAction

1.9 Changes for Release 14.0

This version of the document includes the following changes:

- Changed CAP version from 13.0 to 14.0.
- Added optional “nWayCall” to the “CallClient” user’s “profileUpdate” message.
- Added “ConfAdd” as a new actionType in the “callAction” message.
- Added new failure causes “WASConnectivityError” and “WASProcessingError” to responseAuthentication.
- Added new failure causes “WASConnectivityError”, “WASProcessingError”, and “ExtAuthHostNotInACLError” to registerResponse.
- Added the *id* attribute to the following server-originated messages: “unRegister”, “sessionUpdate”, “profileUpdate”, “callUpdate”, “callControlInfo”, “monitoringUsersResponse” and “queueUpdate”.

1.10 Changes for Release 13.0

This version of the document includes the following changes:

- Changed CAP version from 12.0 to 13.0.
- The value -1 has been added as a valid value to the appearance data element in the “CallClient” user’s “callUpdate” message.

- Added optional “linePort” to the “CallClient” user’s “callUpdate” message.
- Added optional “appearance” tag to the conference element in the “CallClient” user’s “sessionUpdate” message.
- Added optional “threeWayCall” to the “CallClient” user’s “profileUpdate” message.
- Added optional “callTransfer” to the “CallClient” user’s “profileUpdate” message.
- The number of allowed “callIn” tags in the conference element in the “CallClient” user’s “sessionUpdate” message is changed from 2 – N to 0 – N.
- Added new “queueUpdate” and “queueAction” messages for the “CallClient” users.
- Added new “datagram” message for both the “CallClient” and “AttendantConsole” users.
- Added "XferCC" as a new actionType in the “callAction” message.
- Added new values 3 - 10 to “localAltType” data element in the “CallClient” user’s “callUpdate” message.

1.11 Changes for Release 12.0

This version of the document includes the following changes:

- Changed CAP version from 11.1 to 12.0.
- Added the “callLogs” tag to the “CallClient” user’s “profileUpdate” message.
- Added the "locationCode" tag and the "enterpriseUser" tag to the user's initial "profileUpdate" message.
- Added the new “monitoringUsersRequest” and “monitoringUsersResponse” messages for the “AttendantConsole” users.
- Added the “applicationId” tag to the existing messages “sessionUpdate”, “profileUpdate”, “callUpdate”, and “callControlInfo” to eliminate the occasional duplicated messages.
- The “serverStatusRequest” now is being sent from Application Server to its CAP connections serving as a ping message to keep the CAP connections alive.
- Fixed the “externalNotify” message example.
- Updated DTD.

1.12 Changes for Release 11.1

This version of the document includes the following changes:

- Changed CAP version from 11.0 to 11.1.
- Added the new voice mail messages flag (both BroadWorks and Third Party) to the profileUpdate messages for CallClient user type.
- Added externalNotify message for clients to set the message waiting indicator for their endpoint device.
- Added the last redirection number, country code, name, and reason to the callUpdate message for a CallClient user.
- Updated DTD.

1.13 Changes for Release 11.0

This version of the document includes the following changes:

- Changed CAP version from 2.0 to 11.0.
- Added new control types “Hold” and “Retrieve” to the callControlInfo message.
- Added an unsuccessful responseAuthentication message that can be sent from the Open Client Server to the client.
- Added two new failure causes, “UnauthorizedPhoneStatusMonitoring” and “UnauthorizedClientLicense” to the registerResponse message.
- Added new user logout reasons “ForceLogoutC” and “ForceLogoutL” to the unRegister message.
- Added a new serverStatusRequest message.
- Updated DTD.

1.14 Changes for Release 10.0

This version of the document includes the following changes:

- Changed CAP version from 1.0 to 2.0.
- Removed the non-secure password login support. The registerAuthentication and responseAuthentication messages are required for login if the password authentication is required.
- Broke the profileUpdate message for Call Client User into two sections: initial updates and subsequent updates. (Document change only.)
- Added feature setting to the Call Client User’s initial profileUpdate message for the following features: Voice Messaging User, Voice Messaging Group, and Third-Party Voice Mail Support.
- Added feature setting to the Attendant Console User’s initial profileUpdate message for the following features: Voice Messaging Group and Third-Party Voice Mail Support.
- Added feature setting for the CommPilot Express feature to the Attendant Console User’s subsequent profileUpdate message.
- Added External Tracking Id (extTrackingId) to the callUpdate message for both the Call Client User and Attendant Console User.
- Added “Redial” as an option to the callAction message. (Documentation change only.)
- Updated DTD.
- Added new CAP message, callControlInfo. Updated the DTD. Amended the Limitations section for the “Answer” call action. This section was initially applicable to Dumb MGCP Endpoints only.

2 Introduction

The Client Application Protocols (CAP-based protocols) are protocols that expose an external Call Control and monitoring interface to BroadWorks. Custom or third-party client applications can use these protocols to leverage BroadWorks call client functionality. BroadWorks uses these protocols between its back-end servers, CommPilot Call Manager, Attendant Console, and Call Center client applications.

CAP protocols are eXtensible Markup Language (XML)-based protocols. Messages are exchanged as XML documents. This provides a standard messaging interface between external applications and BroadWorks, enabling an open mechanism for any third-party application to perform Call Control actions via BroadWorks. This document is the specification of the following CAP protocols:

- **Client Application Protocol-Client (CAP-C)** - Client Call Control Protocol for individual desktop clients
- **CCC2** - Client Call Control Protocol for third-party client-server farms

The following sections describe each specific protocol in greater detail.

3 CAP-C Protocol

The CAP-C protocol is the primary Call Control and Monitoring Protocol used by individual desktop clients. It is used for the following applications:

- Call Control
- User Monitoring
- Call Center Queue Management

3.1 CAP-C Architecture

The BroadWorks CAP-C interface functions in a three-tier architecture. The first tier is the BroadWorks Execution Server that provides call functionality. The second tier is the BroadWorks Open Client Server (OCS) that acts as a proxy between the Execution Server and tier-3 clients. Tier-3 clients include, for instance, BroadWorks Call Manager clients, Attendant Call Console clients, and third-party clients.

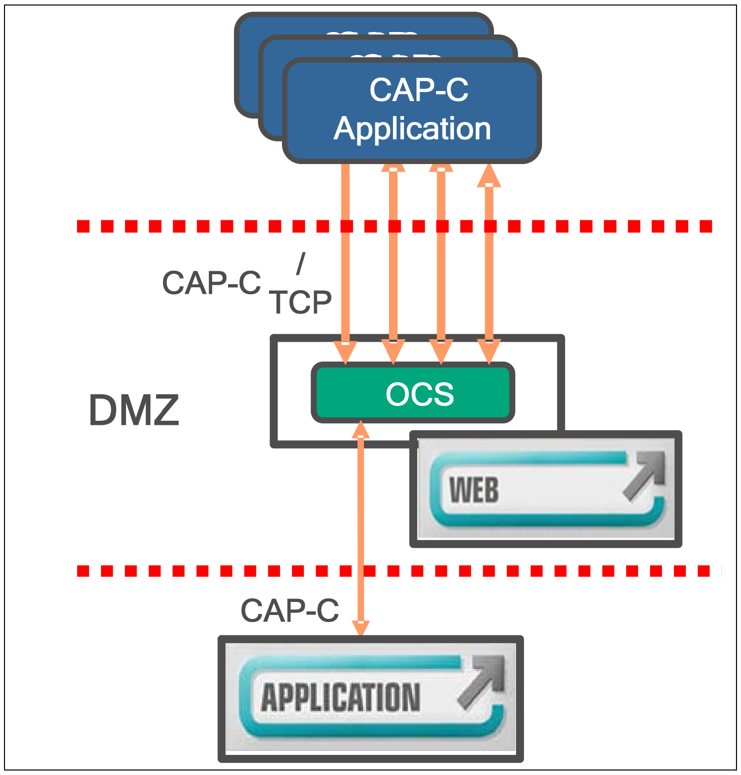

Figure 1 CAP-C Architecture

The primary Execution Server is used actively and the secondary is used as a hot standby (not shown).

The Open Client Server (OCS) performs the following:

- Opens a Transmission Control Protocol/Internet Protocol (TCP/IP) connection to each Execution Server using port 2206. Each server is allowed up to two connections by default. Both the port and number of connections can be configured in BroadWorks.
- Handles all client connections.

- Maps users to a specific server-side connection. A user can use only one connection. This allows performing basic load balancing.
- Acts as a proxy for messages between tier-3 clients and BroadWorks. Messages to and from BroadWorks are required to be in the CAP-C XML format, as specified in this document.
- Handles Execution Server Session Data Replication (SDR). When SDR is enabled, it avoids logging clients out when an Execution Server fails and accepts CAP messages to/from both Execution Servers. When SDR is disabled, clients are logged out if an Execution Server (or its connection) fails.

The OCS resides on the same machine as the Web Server.

Third-party clients connect to the Open Client Server and perform a presentation function to the end user.

3.2 CAP-C Applications

Three types of applications typically make use of the CAP-C Protocol. They are described in the following subsections.

Note that it is possible for a client application to combine several types of functionality. For example, a Call Center client application, in addition to managing the Call Center queue, could also perform user (Call Center agent, in this case) monitoring and Call Control for the Call Center supervisor. When such is the case, however, multiple CAP-C logins have to be done parallel (one per application), since one CAP-C dialog cannot handle all three types of applications simultaneously.

Section *5 Message to Protocol Mapping* contains a table indicating which message applies to which application (and specific protocol). Also, sections *7 Call Control Message Flows*, *8 User Monitoring Message Flows*, and *9 Call Center Queue Management Message Flows* show call flows for each application type CAP-C supports.

3.2.1 Call Control

Call Control is one possible use of the CAP-C Protocol. It allows a client running for a particular user to control the user's own calls, that is, to receive status information about new or ongoing calls, as well as feature configuration (for example, in *callUpdate*, *profileUpdate,* and *sessionUpdate* messages), and to control calls, using the *callAction* message. Actions that can be performed include dialing, transferring, holding, conferencing, and releasing calls.

This is the type of application that is most often used by ordinary end users.

Call Control clients, such as those for other CAP-C applications, must authenticate and register before updates and actions can be sent and received. Call Control clients use the "CallClient" userType in all of their messages. Each Call Control connection is made on behalf of a single user, using that user's user ID, and each command only applies to that user.

3.2.2 User Monitoring

User monitoring applications involve monitoring the status of other users, in the same group or enterprise as the monitoring user. This allows an Attendant Console type of application. As a matter of fact, for users monitoring CAP-C dialogs, the userType is "AttendantConsole" in all messages.

User monitoring clients, such as those for other CAP-C applications, must authenticate and register before updates can be received. This registration is performed using the monitoring user's user ID. Once the connection has been established (authenticated and registered) however, the client must also register for other user monitoring. This is accomplished by issuing the monitoringUsersRequest command, specifying the user IDs of the users to monitor. The list of user IDs is typically obtained via Open Client Interface-Provisioning (OCI-P), but this is outside the scope of this document.

The monitoring of a group or enterprise member's status can be prevented if the target user has the Privacy feature enabled for phone status privacy (unless the monitoring user is in the target user's "exclude" list). For more information, see the *User Managed Privacy Service Feature Description* [2].

3.2.3 Call Center Queue Management

The third type of applications enabled by CAP-C is Call Center queue management. It allows a Call Center client to monitor (by receiving *queueUpdate* and *queueProfileUpdate* messages) and control (by sending *queueAction* CAP-C commands) the Call Center queue, to reorder calls inside the queue or transfer a queued call to a specific number.

To access Call Center queue management functionality, a CAP-C client must authenticate and register with a userType of "CallClient" and use the Call Center virtual user's ID.

3.3 Access Control and Authorization

The Execution Server must be configured to accept CAP-C connections from remote hosts (the OCS). This is done using the Application Server command line interface (CLI) to configure the CAP access control list.

The access control list is configured in the AS_CLI/System/NetworkAccessLists/CAP context.

3.4 Configure Number of Connections

The port that the BroadWorks CAP-C server listens on, as well as the number of TCP/IP connections the OCS can open to it (depending on how many OCS should be allowed to connect simultaneously), is configurable via the Application Server command line interface (CLI), in the AS_CLI/Interfaces/CAP context. Changes are only applied on BroadWorks startup.

3.5 Licensing

The *Client Call Control* Service must be assigned to a user attempting to create a CAP-C session with the Execution Server. If a user attempts to register without this service, the registration is rejected with the reason *Unauthorized*. This is required for third-party clients, and is not necessary for BroadWorks-provided clients.

3.6 User Identification

Although the *userUid* attribute was used to uniquely identify the user in client-to-server messages up to and including Release 14.0, starting with Release 14.sp1, the *id* attribute (which is the same as the user's login ID) is supported and is now the preferred way of identifying the user. However, the *userUid* is still supported in this release.

Clients who perform Call Center queue management should used the user ID of the Call Center virtual subscriber.

4 CCC2 Protocol

The Client Call Control 2 (CCC2) Protocol (associated with the service of the same name) is another Client Application Protocol in many ways similar to CAP-C. It provides an interface with enhanced messaging for third-party client-server farms.

CCC2 is used for the following applications:

- Call Control
- Call Center Queue Management

Because this protocol is used between BroadWorks and a third-party server (trusted), user authentication and registration is not necessary.

The Client Call Control 2 service can be used in a Voice over IP (VoIP) environment or an Intelligent Network Service Control (INSC) environment. However, there are differences in how enhanced messaging is handled between the two environments. These differences are noted later in this document.

4.1 CCC2 Architecture

The CCC2 architecture is simply one or more connections between the BroadWorks Application Server and third-party servers. What happens behind the third-party servers is beyond the scope of this document (there could be third-party clients, and so on).

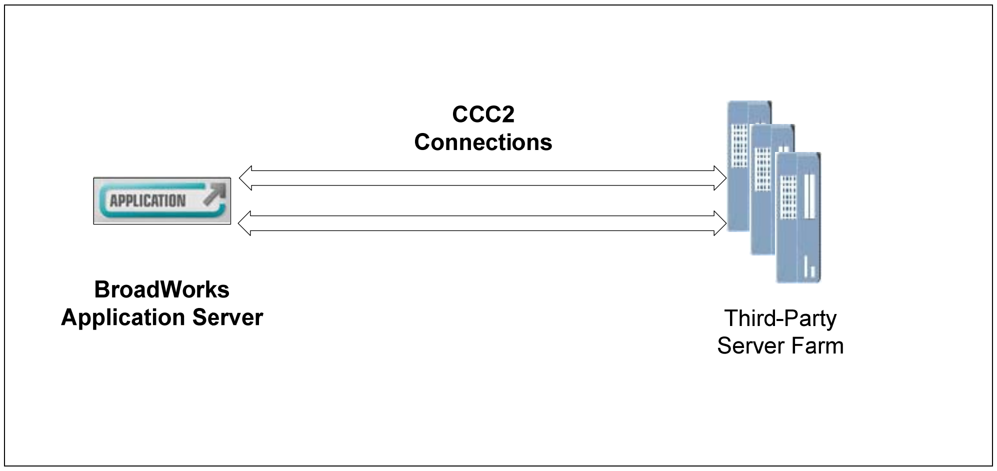

Figure 2 CCC2 Architecture

The third-party server farm opens TCP/IP connections to the Execution Server using port 2212 (this is the default port for Client Call Control 2 connections). Each server is allowed up to two connections by default. Both the port and number of connections can be configured in BroadWorks.

4.2 CCC2 Applications

Two types of applications typically make use of the CCC2 protocol. They are described in the following sub-sections.

Section *5 Message to Protocol Mapping* contains a table indicating which message applies to which application (and specific protocol). Also, sections *7 Call Control Message*

Flows and *9 Call Center Queue Management Message Flows* show call flows for each application type CCC2 supports.

4.2.1 Call Control

Call Control is one possible use of the CCC2 Protocol. The third-party server receives status information (for example, in *callUpdate*, *profileUpdate,* and *sessionUpdate* messages) for every user having the Client Call Control 2 feature. The server can in turn control calls using the *callAction* message. Actions that can be performed include dialing, transferring, holding, conferencing, and releasing calls.

Call Control applications use the “CallClient” userType in all of their messages. The *id* attribute is used to identify the user.

4.2.2 Call Center Queue Management

The second type of application enabled by CCC2 is Call Center queue management. It allows monitoring (by receiving *queueUpdate* and *queueProfileUpdate* messages) and control (by sending *queueAction* CCC2 commands) of the Call Center queue, to reorder calls inside the queue, or transfer a queued call to a specific number.

Call Center queue management applications use the “CallClient” userType in all of their messages. The *id* attribute is set to the Call Center virtual subscriber’s user ID.

4.3 Access Control and Authorization

The Execution Server must be configured to accept CCC2 connections from remote hosts. This is done using the Application Server command line interface (CLI) to configure the CAP access control list (the CAP access control list is shared between CAP-C and CCC2).

The access control list is configured in the AS_CLI/System/NetworkAccessLists/CAP context.

Client Call Control 2 connections do not require any authorization of users. All connections are logged in directly without any user registration. Outgoing messages are sent to connections in a round-robin fashion if more than one CCC2 connection exists. The third-party server farm must dispatch the correct messages to users.

4.4 Configure Number of Connections

The port that the BroadWorks Application Server for CCC2 listens on, as well as the number of TCP/IP connections a third-party server can open to it, is configurable via the Application Server command line interface (CLI), at the AS_CLI/Interfaces/CAP level. The following values can be modified for the Client Call Control 2 service:

- ccc2ServerPort – Any integer value from 1025 to 65535
- ccc2NumClientConnections – Any integer value from 0 to 10

Changes are applied when BroadWorks is restarted.

If a third-party server opens more than one connection to a given Application Server, the CCC2 messages are not all sent on the same connection. The third-party server must be able to receive the information via any one connection.

4.5 Licensing

The Client Call Control 2 feature must be assigned to a user attempting to use a CCC2 session with the Execution Server. If a user does not have this service assigned, no CCC2 messages are sent to the server farm for the user. This also applies to a Call Center virtual subscriber, for Call Center queue management.

4.6 User Identification

Although the *userUid* attribute was used to uniquely identify the user in client-to-server messages up to and including Release 14.0, starting with Release 14.sp1, the *id* attribute (which is the same as the user's login ID) is supported and is now the preferred way of identifying the user. However, the *userUid* is still supported.

For Call Center queue management, the user ID to use is that of the Call Center virtual subscriber.

5 Message to Protocol Mapping

The following table maps each Client Application Protocol message, indicating which of the three protocols it is part of, along with the type of application(s) each message is for.

Message	CAP-C	CCC2
registerAuthentication	Call Control User Monitoring CC Queue	-
responseAuthentication	Call Control User Monitoring CC Queue	-
registerRequest	Call Control User Monitoring CC Queue	-
registerResponse	Call Control User Monitoring CC Queue	-
acknowledgement	Call Control User Monitoring CC Queue	-
unRegister	Call Control User Monitoring CC Queue	-
sessionUpdate	Call Control User Monitoring	Call Control
profileUpdate	Call Control User Monitoring	Call Control
callUpdate	Call Control User Monitoring	Call Control
callAction	Call Control	Call Control
callActionResponse	Call Control	Call Control
callControlInfo	Call Control	Call Control
externalNotify	Call Control	Call Control
infoRequest	Other[1]	Other[1]
infoResponse	Other[2]	Other[2]
serverStatusRequest	Other[3]	Other[3]
monitoringUsersRequest	User Monitoring	-
monitoringUsersResponse	User Monitoring	-
queueUpdate	CC Queue	CC Queue
queueAction	CC Queue	CC Queue
queueProfileUpdate	CC Queue	CC Queue
queueInfoRequest	CC Queue	CC Queue
queueInfoResponse	CC Queue	CC Queue

[1] For information on the infoRequest message, see section *7.14 infoRequest*.
[2] For information on the infoResponse message, see section *7.15 infoResponse*.
[3] For information on the serverStatusRequest message, see section *7.16 serverStatusRequest*.

Message	CAP-C	CCC2
datagram (deprecated)	Other[4]	Other[4]

[4] For information on the datagram message, see section *7.22 datagram (this command is deprecated)*.

6 CAP Messages

CAP messages, which are described in following sub-sections, are defined using the format shown in the following table.

6.1 CAP Message Format

Description	Description of Message, Time Sent, Meaning
Protocols	Specifies the protocol(s) using this message. One or several of CAP-C or CCC2.
Response message	Whether a response message is required (to the message) and what is the response message.
Multi-format	Whether the message can appear in different formats, with different sets of tags, and so on, based on the type of client receiving the message or the context in which the message is sent.
Message direction	Whether the message is sent from a client to a server, vice-versa, or both.
Tags	All tags that can occur in the message, listed in a tabular format. For multi-format messages, the tags in each format are listed in separate tables.
Sample	A sample XML message. For multi-format messages, a sample is provided for each format.
Allowed tag values	For each tag/attribute in the tag's listing that takes defined values, the allowed values are listed. Note that this section may not exist for all messages. Most messages list allowed values for tags directly in the Tags section.

6.2 registerAuthentication

Description	Sent by a client to log in a BroadWorks subscriber. This is the first message that a client using CAP must send to BroadWorks. The server responds to the register request with a **responseAuthentication** that includes a private key used for password authentication.
Protocols	CAP-C
Response message	Yes, the response message is responseAuthentication.
Multi-format	No
Message direction	Client to server
Tags	Tag names and allowed values are as follows:

Tag Name	Allowed Values	Required	Comments
user	Elements	Yes	Indicates the start of BroadWorks user registration information.
userType	"CallClient" or "AttendantConsole"	Yes	The *userType* attribute indicates the type of client functionality the registering user has from a BroadWorks perspective. In this message format, the value "CallClient" means that the given user has call client functionality and "AttendantConsole" means that the given user has attendant console functionality.

Tag Name	Allowed Values	Required	Comments
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType run simultaneously.
id	Text	Yes	The user ID of the BroadWorks subscriber registering or signing in.
supervisorId	Text	No	Id of the supervisor who is trying to register as a call center to monitor it. It is part of the user element.

The following is an example for a normal user logging in:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="registerAuthentication">
      <commandData>
        <user userType="CallClient">
          <id>joepublic</id>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

The following is an example of a supervisor logging in as a call center to monitor a queue. The optional supervisorId element is present in this case:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="registerAuthentication">
    <commandData>
      <user userType="CallClient">
        <id>sales@broadsoft.com</id>
        <applicationId>supervisor@broadsoft.com</applicationId>
        <supervisorId>supervisor@broadsoft.com</supervisorId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.3 responseAuthentication

Description	Sent by a BroadWorks server to a client in response to a previously received **registerAuthentication** message. This message contains a unique private key and algorithm that the client uses to compute a message digest for the user's password. This message must be followed by a **registerRequest** to complete the login process.
Protocols	CAP-C
Response message	Yes, the response message is: registerRequest.
Multi-format	Yes (successful, unsuccessful)
Message direction	Server to client
Tags	Tag names and allowed values are as follows:

6.3.1 Successful responseAuthentication

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	Indicates the start of BroadWorks user registration information.
userType	“CallClient” or “AttendantConsole”	Yes	The *userType* attribute indicates the type of client functionality the registering user has from a BroadWorks perspective. In this message format, a value of “CallClient” means that the given user has call client functionality and “AttendantConsole” means that the given user has attendant console functionality.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously.
Id	Text	Yes	The user ID of the BroadWorks subscriber registering or signing in.
nonce	Text	Yes	The private key that BroadWorks sends to the client to be used to generate the password digest.
algorithm	Text	Yes	The algorithm the client should use to generate the password digest to be used during the log in process
supervisorId	Text	No	Id of the supervisor who is trying to register as a call center to monitor it. It is It is part of the user element.

The following is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="responseAuthentication">
    <commandData>
      <user userType="CallClient">
        <id>joepublic</id>
        <applicationId>broadSoftExample</applicationId>
      </user>
      <nonce>1051713840192</nonce>
      <algorithm>MD5</algorithm>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.3.2 Unsuccessful responseAuthentication

This message is sent by the Open Client Server to the client application.

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	Indicates the start of BroadWorks user registration information.

Name and Attributes	Allowed Values	Required	Comments
userType	“CallClient” or “AttendantConsole”	Yes	The *userType* attribute indicates the type of client functionality the registering user has (from a BroadWorks perspective). In this message format, a value of “CallClient” means that the given user has call client functionality and “AttendantConsole” means that the given user has attendant console functionality.
failure	Elements	Yes	This tag means that a failure occurred while registering the BroadWorks subscriber.
failureCause	“UnknownID” “NetworkServerConnectivityFailure” “WASConnectivityError” “WASProcessingError”	Yes	The reason for the registration failure. “UnknownID” means that the Network Server cannot locate the user in this Application Server cluster. “NetworkServerConnectivityFailure” means that the server (Open Client Server) cannot connect to the Network Server. “WASConnectivityError” means that BroadWorks tried to use external authentication through a Web-based Authentication Server (WAS), but could not connect to the WAS. “WASProcessingError” means that BroadWorks tried to use external authentication through a Web-based Authentication Server (WAS), but the WAS indicated a processing error occurred.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously.
id	Text	Yes	The user ID of the BroadWorks subscriber registering or signing in.
supervisorId	Text	No	Id of the supervisor who is trying to register as a call center to monitor it. It is part of the user element.

The following is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="responseAuthentication">
    <commandData>
      <user userType="CallClient">
        <failure failureCause="NetworkServerConnectivityFailure"/>
        <id>joepublic</id>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.4 registerRequest

Description	Sent by a client to log in a BroadWorks subscriber. If the password is required, the *securePassword* attribute should be used. The server responds to the register request with a **registerResponse**. If the client registration fails, the response contains a reason indicating the cause of failure.
Protocols	CAP-C
Response message	Yes, the response message is: registerResponse.
Multi-format	No
Message direction	Client to server
Tags	Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	Indicates the start of BroadWorks user registration information.
userType	"CallClient" or "AttendantConsole"	Yes	The *userType* attribute indicates the type of client functionality the registering user has from a BroadWorks perspective. In this message format, a value of "CallClient" means that the given user has call client functionality and "AttendantConsole" means that the given user has attendant console functionality.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are intended to be run simultaneously.
id	Text	Yes	The user ID of the BroadWorks subscriber registering or signing in.
securePassword	Text (32 characters)	No	The password message digests for the registering BroadWorks user. Note that this field is optional. If it is not given, it is assumed that the user is not using a secure login scheme.
supervisorId	Text	No	Id of the supervisor who is trying to register as a call center to monitor it. It is part of the user element.

The following is an example of a secured login:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="registerRequest">
      <commandData>
        <user userType="CallClient">
          <id>joepublic</id>

<securePassword>dc70779bf8461b5a1e6aea58f636d5c0</securePassword>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

6.5 registerResponse

Description	Sent by a BroadWorks server to a client in response to a previously received **registerRequest** message. This message contains the user unique identifier (uid) that all subsequent messages between the server and client contain, until an **unRegister** message is sent or received. As such, this uid value represents a session between the client and server. This message must be acknowledged by the client application. Only on receipt of an **acknowledgement** does the server issue the remaining CAP messages required to complete registration.
Protocols	CAP-C
Response message	Yes, the response message required is: acknowledgement.
Multi-format	Yes (successful call client user registration, successful attendant console user registration, unsuccessful user registration)
Message direction	Server to client
Tags	Tag names and allowed values are as follows:

6.5.1 Successful Call Client User Registration

Tag names and allowed values for *Successful Call Client User Registration* are as follows*:*

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber that registered successfully.
userType	“CallClient”	Yes	The *userType* attribute indicates the type of client functionality the registering user has from a BroadWorks perspective. In this message format, a value of “CallClient” means that the given user has call client functionality.
userUid	Text	Yes	The *userUid* attribute value is to be used in all further communication to or from the server, uniquely identifying this user within BroadWorks.
applicationId	Text	Yes	The ID of the application that registered via a previous registerRequest message.
id	Text	Yes	The user ID of the BroadWorks subscriber that registered via a previous registerRequest message.
supervisorId	Text	No	Id of the supervisor who is trying to register as a call center to monitor it. It is part of the user element.

The following is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="registerResponse">
    <commandData>
      <user userType="CallClient" userUid="169729071">
        <id>joepublic</id>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.5.2 Successful Attendant Console User Registration

Tag names and allowed values for Successful Attendant Console User Registration are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber that registered successfully.
userType	"AttendantConsole"	Yes	The *userType* attribute indicates the type of client functionality the user has from a BroadWorks perspective. In this message format, the value of "AttendantConsole" means that the user has attendant console functionality.
userUid	Text	Yes	The *userUid* attribute value is to be used in all further communication to or from the server, uniquely identifying this user within BroadWorks.
id	Text	Yes	The user ID of the BroadWorks subscriber that registered via a previous registerRequest message.
applicationId	Text	Yes	The ID of the application that registered via a previous registerRequest message.
numMonitoredUsers	Text	Yes	The number of users whose status this attendant console user monitors. This corresponds to the number of initial profileUpdate and sessionUpdate messages that are sent by BroadWorks to this attendant console, each message providing information about a single user being monitored.
callDetailsEnabled	"True" or "False"	Yes	Indicates whether call detail viewing is enabled for this user. If True, callUpdate messages containing detailed call information for monitored users are sent from the BroadWorks server to the client.

The following is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="registerResponse">
    <commandData>
      <user userType="AttendantConsole" userUid="108215935">
        <id>janedoe</id>
        <applicationId>broadSoftExample</applicationId>
        <numMonitoredUsers>50</numMonitoredUsers >
        <callDetailsEnabled>True</callDetailsEnabled>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.5.3 Unsuccessful User Registration

Tag names and allowed values for *Unsuccessful User Registration* are as follows*:*

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber that was unsuccessful in registering.
userType	"CallClient" | "AttendantConsole"	Yes	The *userType* attribute indicates the type of client functionality the registering user submitted in the registerRequest message.
failure	Elements	Yes	This tag means that a failure occurred while registering the BroadWorks subscriber.
failureCause	"UnknownID" "IncorrectPasword" "UnsupportedVersion" "UnauthorizedCommPilotCallManager" "UnauthorizedAttendant Console" "UnauthorizedClientCall Control" "UnauthorizedPhoneStatusMonitoring" "UnauthorizedClientLicense" "WASConnectivityError" "WASProcessingError" "ExtAuthHostNotInACLError"	Yes	The reason for the registration failure. "UnknownID" means that the user ID in a registerRequest is not a valid BroadWorks user. "IncorrectPassword" means that the password supplied in the registerRequest is not correct for the registering user. "UnsupportedVersion" means that the version of the protocol is unsupported by BroadWorks. The five "Unauthorized" reasons mean that the user ID in the registerRequest message does not have the given userType functionality as specified in the registerRequest. This situation can occur when a valid user is attempting to register as an attendant console or call client, but has not been assigned attendant console or call client functionality. "WASConnectivityError" means that BroadWorks tried to use external authentication through a Web-based Authentication Server (WAS), but could not connect to the WAS. "WASProcessingError" means that BroadWorks tried to use external authentication through a Web-based Authentication Server (WAS), but the WAS indicated a processing error occurred. "ExtAuthHostNotInACLError" means that the Application Server refused the login because the OCS was not in the Access Control List (ACL) list.
applicationId	Text	Yes	The ID of the application that registered via a previous registerRequest message.
id	Text	Yes	The user ID of the BroadWorks subscriber for whom the registration failed.

Name and Attributes	Allowed Values	Required	Comments
supervisorId	Text	No	Id of the supervisor who is trying to register as a call center to monitor it. It is part of the user element.

The following is an example:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="registerResponse">
      <commandData>
        <user userType="AttendantConsole">
          <failure failureCause="UnauthorizedAttendantConsole"/>
          <id>joepublic</id>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

6.6 acknowledgement

Description	Sent between BroadWorks and a client acknowledging message reception, whether successful or unsuccessful. This message is sent when a client receives a successful **registerResponse** message; it must issue an **acknowledgement**. Only on receipt of this **acknowledgement** does BroadWorks issue the remaining messages required to complete registration.
Protocols	CAP-C
Response message	No
Multi-format	No
Message direction	Client to server
Tags	Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber who is performing the call action.
userType	“CallClient” or “AttendantConsole”	Yes	The *userType* attribute indicates the type of client application sending this message.
userUid	Text	No	The *userUid* attribute uniquely identifies the BroadWorks user. It identifies the BroadWorks subscriber who is acknowledging the registerResponse. Either the *userUid* or the *id* attribute is required (the *id* is preferred).
id	Text	No	The user ID of the BroadWorks subscriber who is acknowledging the registerResponse. Either the *userUid* or the *id* attribute is required (the Id is preferred).
message	Empty	Yes	This tag indicates the message that is being acknowledged.

Name and Attributes	Allowed Values	Required	Comments
messageName	"registerResponse"	Yes	The *messageName* attribute identifies the specific CAP message that is being acknowledged. Currently, only the registerResponse message requires acknowledgement.
failure	Empty	No	This tag identifies the failure if the acknowledgement is unsuccessful. The absence of this tag implies a successful acknowledgement.
failureCause	Empty	No	The reason why the message was rejected. Currently there are no cases in which a client issues a failed acknowledgement.
applicationId	Text	Yes	The ID of the application that registered via a previous registerRequest message.
supervisorId	Text	No	Id of the supervisor who is trying to register as a call center to monitor it. It is part of the user element.

The following is an example of a successful acknowledgement of a registerResponse:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="acknowledgement">
    <commandData>
      <user userType="CallClient" userUid="108215935" id="joe">
        <message messageName="registerResponse"/>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.7 unRegister

Description	Sent by a client when logging out. Also sent by the BroadWorks server to a client to indicate a forced log out. Once this message has been sent from BroadWorks or received from a client, that user does not get a CAP message until a successful registration occurs.
Protocols	CAP-C
Response message	No
Multi-format	No
Message direction	Client to server or server to client
Tags	Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber that has to unregister.

Name and Attributes	Allowed Values	Required	Comments
id	Text	No	The *id* attribute represents the user ID of the BroadWorks subscriber that registered via a previous registerRequest message. Either the *userUid* or the *id* attribute is required (the *id* is preferred) for client generated messages.
userLogoutReason	"ClientLogout" "ForceLogoutR" "ForceLogoutD" "ForceLogoutU" "ForceLogoutL" "ForceLogoutC"	Yes	The *userLogoutReason* attribute value gives the reason for the un-registration. "ClientLogout" indicates that the client logged out. "ForceLogoutR" indicates that BroadWorks logged the user out because the user was moved to another BroadWorks server in a redundant server cluster. "ForceLogoutD" indicates that BroadWorks logged the user out because the user has logged in on another terminal (duplicate log in). "ForceLogoutU" indicates that the service is no longer assigned to the user. "ForceLogoutL" indicates the Application Server is locked. "ForceLogoutC" is sent by the Open Client Server and indicates that the connection between the Open Client Server and the Application Server is lost.
userType	"CallClient" or "AttendantConsole"	Yes	The *userType* attribute indicates the type of client functionality the user has (from a BroadWorks perspective).
userUid	Text	No	The *userUid* attribute value uniquely identifying the user within BroadWorks. Either the *userUid* or the *id* attribute is required (the *id* is preferred) for client-generated messages.
applicationId	Text	Yes	The ID of the application that registered via a previous registerRequest message.

The following is an example of a client-originated message:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="unRegister">
    <commandData>
      <user userLogoutReason="ClientLogout" userType="CallClient"
          id="joe@abc.com">
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.8 sessionUpdate

Description	Sent by BroadWorks to a client providing the client's session information. Session information is data/state not local to a call but valid for the state of the call client application as a whole. This message is sent in the following scenarios: From BroadWorks to a call client after successful **acknowledgement** of a **registerResponse** (CAP-C). From BroadWorks to a call client when its session state changes, for example, hook status, and conference state. From BroadWorks to an attendant console when a monitored user's status changes (CAP-C). This message is **sent for VoIP users**. It is **not sent for INSC users**.
Protocols	CAP-C, CCC2
Response message	No
Multi-format	Yes (call client user, attendant console user)
Message direction	Server to client

6.8.1 Call Client User (CAP-C and CCC2)

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks user for whom call client session information is being communicated.
id	Text	Yes	The *id* attribute represents the user ID of the BroadWorks subscriber.
userType	"CallClient"	Yes	The *userType* attribute indicates the type of client functionality the user has from a BroadWorks perspective.
userUid	Text	Yes	The *userUid* attribute value uniquely identifying the user within BroadWorks.
offHook	"True" or "False"	No	This tag indicates the user's phone/endpoint off-hook status. "True" means the endpoint went off hook and "False" means it is or went on hook. This tag is present in this message only when the off-hook state changes or when this message is sent as part of registration.
numCalls	Text	No	The number of active calls that this user has. For each active call that a user has, a callUpdate message is issued by BroadWorks, giving state information about that active call. This tag appears in the sessionUpdate issued as part of the registration process only. sessionUpdates issued during normal processing do not have this tag.

Name and Attributes	Allowed Values	Required	Comments
maxCalls	Text	No	Gives the current maximum calls allowed for the given call client application.
epControl	"True" or "False"	No	Indicates if the user currently is using an endpoint that performs its own Call Control (such as a SIP phone). "True" indicates yes; "False" indicates no.
conference	Elements	No	This tag gives the state of a conference (if any) that a user is involved in. This tag occurs only when a conference is established/exists when a client is registering. If no conference exists, this tag does not appear.
conferenceState	"Active" "Released" "Held"	No	The *conferenceState* attribute value gives the state of the conference.
callIn	Null	No	This tag identifies the calls in the conference. There can be 0 to N such tags in this message, but these only occur when a conference tag occurs.
callInId	Text	No	The *callInId* attribute value is the call ID of a call in the conference.
appearance	"1" – N	No	The appearance of the call. This is the line appearance on the phone or device this call is on.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously. The value is left blank for CCC2.
silentlyMonitored	Empty	No	Part of the conference element. If present, indicates that the user is silently monitoring a conference. This is only present for the user who performed the silent monitor request.

NOTE: In the sessionUpdate message that is sent to the client as part of the registration (for CAP-C), if the user is not involved in a call, then the value of "epControl" is always set to "True" and the maxCalls is always set to "–1", regardless of the type of phone used.

This is an example of the registration process with an active conference (CAP-C):

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="sessionUpdate">
    <commandData>
      <user id="joe@abc.com" userType="CallClient" userUid="139557133">
        <offHook>True</offHook>
        <numCalls>2</numCalls>
        <maxCalls>-1</maxCalls>
        <epControl>True</epControl>
        <conference conferenceState="Active">
          <callIn callInId="09289589"/>
          <callIn callInId="09289000"/>
```

```
            <appearance>1</appearance>
          </conference>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

Following is an example of a conference state change (CAP-C):

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="sessionUpdate">
      <commandData>
        <user id="joe@abc.com" userType="CallClient" userUid="139557133">
          <conference conferenceState="Held">
          </conference>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

Following is the same example of a conference state change (CCC2). The only difference is that the *applicationId* field is blank. Note that the protocol name (CAP) is the same in CCC2:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="sessionUpdate">
      <commandData>
        <user id="joe@abc.com" userType="CallClient" userUid="139557133">
          <conference conferenceState="Held">
          </conference>
          <applicationId></applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

6.8.2 Attendant Console User (CAP-C)

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks attendant console user to whom this message is being sent.
id	Text	Yes	The *id* attribute represents the user ID of the BroadWorks subscriber that registered via a previous registerRequest message.
userType	“AttendantConsole”	Yes	The *userType* attribute indicates the type of client functionality the user has from a BroadWorks perspective. In this case, a value of AttendantConsole means that this message is destined for an attendant console client.

Name and Attributes	Allowed Values	Required	Comments
userUid	Text	Yes	The *userUid* attribute value uniquely identifying the attendant console user within BroadWorks.
monitoredUserUid	Text	Yes	This tag identifies the user who the attendant console is monitoring, and whose status change is being communicated via this message.
monitoredUserId	Text	Yes	The *monitoredUserId* attribute identifies the ID of the user being monitored by the attendant console.
offHook	"True" or "False"	Yes	This tag indicates the user's phone/endpoint off-hook status. "True" means the endpoint went off hook and "False" means it is or went on hook.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously.

The following is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="sessionUpdate">
    <commandData>
      <user id="ac@abc.com" userType="AttendantConsole"
          userUid="141466199">
        <monitoredUserUid>169729071</monitoredUserUid>
        <monitoredUserId>mary@abc.com</monitoredUserId>
        <offHook>False</offHook>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.9 profileUpdate

Description	This message is sent in the following scenarios: ▪ From BroadWorks to a call client after successful **acknowledgement** of a **registerResponse** (CAP-C). This message contains data required by a call client application on initialization. ▪ From BroadWorks to a client when a user's configuration has been changed in the database (CAP-C, CCC2). ▪ From BroadWorks to an attendant console after a successful **acknowledgement** to a **registerResponse** (CAP-C). This message contains data required by the attendant console client on initialization. ▪ From BroadWorks to an attendant console when a monitored user's state changes (CAP-C). This message is **sent for VoIP users**. It is **not sent for INSC users**.
Protocols	CAP-C or CCC2
Response message	No
Multi-format	Yes. (call client user initial, call client user update, attendant console user initial, attendant console user update)
Message direction	Server to client

6.9.1 Call Client User – Initial Message, as Part of Registration Process (CAP-C)

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber that this profileUpdate is being sent to.
id	Text	Yes	The *id* attribute represents the user ID of the BroadWorks subscriber that registered via a previous registerRequest message.
userUid	Text	Yes	The *userUid* attribute value uniquely identifies the user.
userType	"CallClient"	Yes	The *userType* attribute indicates which user type's profile is being updated.
initialUpdate	Empty	Yes	If this element is part of the message, it indicates this message is an initial update. If the element is not part of the message, it indicates this message is a subsequent update.
firstName	Text	Yes	The first name of this user.
lastName	Text	Yes	The last name of this user.
phone	Text	Yes	The phone number of this user.
extension	Text	Yes	The phone extension of this user.
locationCode	Text	Yes	The enterprise location dialing code of the user's group
countryCode	Text	Yes	The country code of this user. Can be empty if the user has no primary phone number.

Name and Attributes	Allowed Values	Required	Comments
nationalPrefix	Text	Yes	The national prefix associated with the user’s country code. Can be empty if the user has no primary phone number, or if the user’s country code has no national prefix.
enterpriseUser	“On” “Off”	Yes	"On" means that the service provider the user belongs to is an enterprise. "Off" means the service provider is not an enterprise.
email	Text	Yes	The e-mail address of this user.
groupName	Text	Yes	The ID of the group this user belongs to.
serviceProviderName	Text	Yes	The ID of the service provider the user (and the user’s group) belongs to.
supportEmail	Text	Yes	The e-mail address this call client should use to report support issues to.
threeWayCall	Empty “On”	Yes	If no value exists, it indicates that the Three-Way Call feature is not assigned to the user. “On” means the Three-Way Call feature is assigned to the user and active.
nWayCall	Empty “On”	Yes	If no value exists, it indicates that the N-Way Call feature is not assigned to the user. “On” means the N-Way Call feature is assigned to the user and active.
callTransfer	Empty “On”	Yes	If no value exists, it indicates that the Call Transfer feature is not assigned to the user. “On” means the Call Transfer feature is assigned to the user and active.
cfa	Empty “On” “Off”	Yes	If no value exists, it indicates that Call Forwarding Always (CFA) is not assigned to this user. “On” means that CFA is assigned and active; “Off” means it is assigned but inactive.
cpe	Empty Text	Yes	If no value exists, it indicates that CommPilot Express (CPE) is not assigned to this user. Otherwise, the value is the name of the currently active profile.
dnd	Empty “On” “Off”	Yes	If no value exists, it indicates that Do Not Disturb (DND) is not assigned to this user. “On” means that DND is assigned and active; “Off” means it is assigned but inactive.

Name and Attributes	Allowed Values	Required	Comments
lnrd	Empty “On”	Yes	If value is null, it indicates Last Number Redial (LNRD) is not assigned to the user. “On” indicates that LNRD is assigned and active.
oi	Empty “On”	Yes	If value is null, it indicates Outlook Integration (OI) is not assigned or is inactive for the user. “On” indicates that OI is assigned and active.
ro	Empty “On” “Off”	Yes	If no value exists, it indicates that Remote Office (RO) is not assigned to this user. “On” means that RO is assigned and active; “Off” means it is assigned but inactive.
li	Empty “On”	Yes	If no value exists, it indicates that LDAP Integration (LI) is not assigned to this user. “On” means that LI is assigned and active.
voiceMessaging	Empty “On” “Off”	Yes	If no value exists, it indicates that the Voice Messaging User feature is not assigned to the user. “On” means the Voice Messaging User feature is assigned to the user and is turned on; “Off” means the Voice Messaging User feature is assigned to the user but is not turned on.
voiceMessagingGroup	Empty “On”	Yes	If no value exists, it indicates that the Voice Messaging Group feature is not assigned to the user’s group. “On” means the Voice Messaging Group feature is assigned to the user’s group and active.
thirdPartyVoiceMessaging	Empty “On”	Yes	If no value exists, it indicates that the Third-Party Voice Mail Support feature is not assigned to the user. “On” means the Third-Party Voice Mail Support feature is assigned to the user and active.
thirdPartyVMGroup	Empty “On” “Off”	Yes	If no value exists, it indicates that the Third-Party Voice Mail Support feature is not authorized to the user’s group. “On” means the Third-Party Voice Mail Support feature is authorized to the user’s group and active. “Off” means the Third-Party Voice Mail Support feature is authorized to the user’s group but not active.
hasVMMessages	“True” | “False”	No	Indicates if a user has any new voice mail messages when the BroadWorks Voice Messaging feature is used.

Name and Attributes	Allowed Values	Required	Comments
hasThirdPartyVMMessages	“True” | “False”	No	Indicates if a user has any new third-party voice mail messages when a Third-Party Voice Messaging feature is used.
callLogs	Empty “On”	Yes	If value is empty, it indicates that neither Basic Call Logs nor the Enhance Call Logs feature is assigned to the user. “On” indicates that at least one of the call logs feature is assigned to the user.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously.
acdState	“Sign-in” “Sign-out” “Available” “Unavailable” “Wrap-up”	No	The ACD state of the user (for Call Center agents). This state applies to the user across all Call Centers it is a member of.
callCenter	Elements	No	This tag identifies a Call Center, which is identified by the *id* attribute. This tag is only present for Call Center agents.
id	Text	No	The *id* attribute (under the *callCenter* tag) represents the ID of the Call Center.
joinCallCenter	“True” | “False”	No	The *joinCallCenter* field indicates whether the user is currently participating in the Call Center given by ID.
simring	Empty “On” “Off”	No	If the value is empty, it indicates that the SImultaneous Ringing service is unassigned to the user. The “On” value indicates that the service is assigned and enabled, and “Off” indicates that the service is assigned but disabled.
acdStateTime	Text	No	The time (in milliseconds since January 1, 1970 GMT) at which the ACD state was last changed.
cwt	Empty “On” “Off”	Yes	If no value exists, it indicates that Call Waiting (CW) is not assigned to this user. ▪ “On” means that Call Waiting is assigned and active. ▪ “Off” means that Call Waiting is assigned but inactive.
hotelingHost	Text	No	The hoteling host the user is associated with if one exists or empty to indicate no host/guest association.
unavailabilityCode	Text	No	The unavailability code set for the user or empty if one does not exist.

Name and Attributes	Allowed Values	Required	Comments
wrapUpCallCenterCallId	Text	No	The call id of the last ACD incoming or outgoing call. Only present when the ACD state has changed to Wrap Up.

This is an example – after a successful registration by a call client user (server to client):

```
<?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="profileUpdate">
      <commandData>
        <user id="joe@abc.com" userUid="139557133" userType="CallClient">
          <initialUpdate/>
          <firstName>John</firstName>
          <lastName>Doe</lastName>
          <phone>3015551000</phone>
          <extension>1000</extension>
          <locationCode>7</locationCode>
          <countryCode>1</countryCode>
          <nationalPrefix/>
          <enterpriseUser>On</enterpriseUser>
          <email>john.doe@initech.com</email>
          <groupName>AGroup</groupName>
          <serviceProviderName>MySP</serviceProviderName>
          <supportEmail>support@initech.com</supportEmail>
          <threeWayCall/>
          <nWayCall>On</nWayCall>
          <callTransfer>On</callTransfer>
          <cfa>Off</cfa>
          <cpe>Available: In Office</cpe>
          <cwt>On</cwt>
          <dnd>Off</dnd>
          <lnrd>On</lnrd>
          <oi/>
          <ro>Off</ro>
          <simring/>
          <li>On</li>
          <voiceMessaging/>
          <voiceMessagingGroup/>
          <thirdPartyVoiceMessaging>On</thirdPartyVoiceMessaging>
          <thirdPartyVMGroup>On</thirdPartyVMGroup>
          <hasVMMessages>True</hasVMMessages>
          <hasThirdPartyVMMessages>False</hasThirdPartyVMMessages>
          <callLogs>On</callLogs>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

6.9.2 Call Client User – Subsequent Updates (CAP-C, CCC2)

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber that this profileUpdate is being sent to.
id	Text	Yes	The *id* attribute (under the user tag) represents the user ID of the BroadWorks subscriber whose state has changed.
userUid	Text	Yes	The *userUid* attribute value uniquely identifies the user.
userType	"CallClient"	Yes	The *userType* attribute indicates which user type's profile is being updated.
threeWayCall	Empty "On"	No	If no value exists, it indicates that the Three-Way Call feature is not assigned to the user. "On" means the Three-Way Call feature is assigned to the user and active.
nWayCall	Empty "On"	No	If no value exists, it indicates that the N-Way Call feature is not assigned to the user. "On" means the N-Way Call feature is assigned to the user and active.
callTransfer	Empty "On"	No	If no value exists, it indicates that the Call Transfer feature is not assigned to the user. "On" means the Call Transfer feature is assigned to the user and active.
cfa	Empty "On" "Off"	No	If no value exists, it indicates that Call Forwarding Always (CFA) is not assigned to this user. "On" means that CFA is assigned and active; "Off" means it is assigned but inactive.
cpe	Empty Text	No	If no value exists, it indicates that CommPilot Express (CPE) is not assigned to this user. Otherwise, the value is the name of the currently active profile.
dnd	Empty "On" "Off"	No	If no value exists, it indicates that Do Not Disturb (DND) is not assigned to this user. "On" means that DND is assigned and active; "Off" means it is assigned but inactive.
lnrd	Empty "On"	No	If value is null, it indicates Last Number Redial (LNRD) is not assigned to the user. "On" indicates that LNRD is assigned and active.
ro	Empty "On" "Off"	No	If no value exists, it indicates that Remote Office (RO) is not assigned to this user. "On" means that RO is assigned and active; "Off" means it is assigned but inactive.
hasVMMessages	"True" | "False"	No	This indicates whether a user has any new voice mail messages if the BroadWorks Voice Messaging feature is used.

Name and Attributes	Allowed Values	Required	Comments
hasThirdPartyVMMessages	"True" | "False"	No	This indicates whether a user has any new third party voice mail messages if the Third-Party Voice Messaging feature is used.
applicationId	Text	Yes	This is the ID of the application when multiple clients of the same userType are run simultaneously.
acdState	"Sign-in" "Sign-out" "Available" "Unavailable" "Wrap-up"	No	This is the ACD state of the user (for Call Center agents). This state applies to the user across all Call Centers the user is a member of.
callCenter	Elements	No	This tag identifies a Call Center, which is identified by the *id* attribute. This tag is only present when the *joinCallCenter* flag has been modified for Call Center agents.
id	Text	No	The *id* attribute (under the *callCenter* tag) represents the ID of the Call Center for which the *joinCallCenter* flag value changes.
joinCallCenter	"True" | "False"	No	The *joinCallCenter* field indicates whether the user is currently participating in the Call Center given by ID.
simring	Empty "On" "Off"	No	If the value is empty, it indicates that the SImultaneous Ringing service is unassigned to the user. The "On" value indicates that the service is assigned and enabled, and "Off" indicates that the service is assigned but disabled.
acdStateTime	Text	No	The time (in milliseconds since January 1, 1970 GMT) at which the ACD state was last changed.
cwt	Empty "On" "Off"	No	If no value exists, it indicates that Call Waiting (CW) is not assigned to this user. ▪ "On" means that Call Waiting is assigned and active. ▪ "Off" means that Call Waiting is assigned but inactive.
hotelingHost	Text	No	The hoteling host the user is associated with if one exists or empty to indicate no host/guest association.
unavailabilityCode	Text	No	The unavailability code set for the user or empty if one does not exist.
wrapUpCallCenterCallId	Text	No	The call id of the last ACD incoming or outgoing call. Only present when the ACD state has changed to Wrap Up.

This is a CAP-C example when a user has changes to the user's CommPilot Express (CPE) settings:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="profileUpdate">
      <commandData>
        <user id="joe@abc.com" userUid="139557133" userType="CallClient">
          <cpe>Unavailable</cpe>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

For CCC2, the above example would be similar, except for an empty applicationId.

Here is a call center example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="profileUpdate">
    <commandData>
      <user id="agent@agc.com" userType="CallClient" userUid="101567640">
        <acdState>WrapUp</acdState>
        <acdStateTime>1235684031000</acdStateTime>
        <wrapUpCallCenterCallId>localhost:30</wrapUpCallCenterCallId>
        <hotelingHost/>
         <unavailabilityCode>lunch</unavailabilityCode>
        <applicationId>BroadWorks Supervisor</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.9.3 Attendant Console User – Initial Message, Part of Registration Process (CAP-C)

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber that this profileUpdate is being sent to.
id	Text	Yes	The *id* attribute represents the user ID of the BroadWorks subscriber that registered via a previous registerRequest message.
userUid	Text	Yes	The *userUid* attribute value uniquely identifies the user.
userType	“AttendantConsole”	Yes	The *userType* attribute indicates which user type’s profile is being updated.
monitoredUserUid	Text	Yes	The UID of the user being monitored by the attendant console.
monitoredUserId	Text	Yes	The monitoredUserId element identifies the ID of the user being monitored by the attendant console.
initialUpdate	Empty	Yes	If this element is part of the message, it indicates this message is an initial update. If the element is not part of the message, it indicates this message is a subsequent update.

Name and Attributes	Allowed Values	Required	Comments
firstName	Text	Yes	The first name of this user.
lastName	Text	Yes	The last name of this user.
phone	Text	Yes	The phone number of this user.
extension	Text	Yes	The phone extension of this user.
locationCode	Text	Yes	The enterprise location dialing code of the user’s group.
countryCode	Text	Yes	The country code of this user. Can be empty if the user has no primary phone number.
nationalPrefix	Text	Yes	The national prefix associated with the user’s country code. Can be empty if the user has no primary phone number, or if the user’s country code has no national prefix.
mobile	Text	Yes	The mobile number of this user.
pager	Text	Yes	The pager number for this user.
email	Text	Yes	The e-mail address for this user.
department	Text	Yes	The department for this user.
title	Text	Yes	The title of this user.
voiceMessaging	Empty “On” “Off”	Yes	If no value exists, it indicates that the Voice Messaging User feature is not assigned to the user. “On” means the Voice Messaging User feature is assigned to the user and is turned on; “Off” means the Voice Messaging User feature is assigned to the user but is not turned on.
voiceMessagingGroup	Empty “On”	Yes	If no value exists, it indicates that the Voice Messaging Group feature is not assigned to the user’s group. “On” means the Voice Messaging Group feature is assigned to the user’s group and active.
thirdPartyVoiceMessaging	Empty “On”	Yes	If no value exists, it indicates that the Third-Party Voice Mail Support feature is not assigned to the user. “On” means the Third-Party Voice Mail Support feature is assigned to the user and active.
thirdPartyVMGroup	Empty “On” “Off”	Yes	If no value exists, it indicates that the Third-Party Voice Mail Support feature is not authorized to the user’s group. “On” means the Third-Party Voice Mail Support feature is authorized to the user’s group and active. “Off” means the Third-Party Voice Mail Support feature is authorized to the user’s group but not active.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously.
unavailabilityCode	Text	No	The unavailability code set for the user or empty if one does not exist.

The following is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="profileUpdate">
    <commandData>
      <user id="ac@abc.com" userUid="141466199"
          userType="AttendantConsole">
        <initialUpdate/>
        <monitoredUserUid>147184435</monitoredUserUid>
        <monitoredUserId>john@abc.com</monitoredUserId>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
        <phone>3015551000</phone>
        <extension>1000</extension>
        <locationCode>7</locationCode>
        <countryCode>1</countryCode>
        <nationalPrefix/>
        <mobile>3012221098</mobile>
        <pager>3012221098</pager>
        <email>john.doe@initech.com</email>
        <voiceMessaging>Off</voiceMessaging>
        <department>Support</department>
        <title>Engineer</title>
        <voiceMessaging>Off</voiceMessaging>
        <voiceMessagingGroup/>
        <thirdPartyVoiceMessaging>On</thirdPartyVoiceMessaging>
        <thirdPartyVMGroup>On</thirdPartyVMGroup>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.9.4 Attendant Console User – Subsequent Updates (CAP-C)

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber that this profileUpdate is being sent for.
id	Text	Yes	The *id* attribute represents the user ID of the BroadWorks subscriber that registered via a previous registerRequest message.
userUid	Text	Yes	The *userUid* attribute value uniquely identifies the user.
userType	“AttendantConsole”	Yes	The *userType* attribute indicates which user type profile is being updated.
monitoredUserUid	Text	Yes	The UID of the user being monitored by the attendant console, whose state has changed. Currently only the DND state for a monitored user is sent via this message.
monitoredUserId	Text	Yes	The monitoredUserId element identifies the ID of the user being monitored by the attendant console.

Name and Attributes	Allowed Values	Required	Comments
dnd	"On" "Off"	No	"On" means that Do Not Disturb (DND) is assigned and active; "Off" means it is assigned but inactive or not assigned to the user.
cpe	Empty Text	No	If no value exists, it indicates that CommPilot Express (CPE) is not assigned to this user. Otherwise, the value is the name of the currently active profile.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously.
acdState	"Sign-in" "Sign-out" "Available" "Unavailable" "Wrap-up"	No	The ACD state of the user (for Call Center agents). This state applies to the user across all Call Centers it is a member of.
callCenter	Elements	No	This tag identifies a Call Center, which is identified by the *id* attribute. This tag can only be present for call center agents.
id	Text	No	The *id* attribute (under the *callCenter* tag) represents the ID of the Call Center for which the *joinCallCenter* flag value is reported.
joinCallCenter	"True" | "False"	No	The *joinCallCenter* field indicates whether the user is currently participating in the Call Center given by ID.
cfa	"On", "Off"	No	"On" means that Call Forwarding Always (CFA) is assigned and active; "Off" means it is assigned but inactive or not assigned to the user. When "On", the *cfaDestination* attribute is present.
cfaDestination	Text	No	The *cfaDestination* attribute (under the *cfa* tag and only present when *cfa* is "On") represents the Call Forwarding Always destination number or SIP URI.
acdStateTime	Text	No	The time (in milliseconds since January 1, 1970 GMT) at which the ACD state was last changed.
cwt	Empty "On" "Off"	No	If no value exists, it indicates that Call Waiting (CW) is not assigned to this user. ▪ "On" means that Call Waiting is assigned and active. ▪ "Off" means that Call Waiting is assigned but inactive.
unavailabilityCode	Text	No	The unavailability code set for the user or empty if one does not exist.

This is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="profileUpdate">
    <commandData>
```

```
        <user id="ac@abc.com" userUid="141466199"
            userType="AttendantConsole">
          <dnd>Off</dnd>
          <monitoredUserUid>147184435</monitoredUserUid>
          <monitoredUserId>john@abc.com</monitoredUserId>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

This is an example of Call Forwarding Always:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="profileUpdate">
      <commandData>
        <user id="ac@abc.com" userUid="141466199"
            userType="AttendantConsole">
          <cfa cfaDestination="3015551000">On</cfa>
          <monitoredUserUid>147184435</monitoredUserUid>
          <monitoredUserId>john@abc.com</monitoredUserId>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

6.10 callUpdate

Description	Sent by BroadWorks to a client to indicate that an active call's state has changed. This message is also used to indicate an incoming call for a registered user. This message may appear during successful registration of a client (for CAP-C) if there are active calls already for that Call Client user. This message is **sent for both VoIP users** and **INSC users**.
Protocols	CAP-C, CCC2
Response message	No
Multi-format	Yes (Call Client update, Attendant Console call updates)
Message direction	Server to client

6.10.1 Call Client User

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber for whom call state information is being sent.
id	Text	Yes	The *id* attribute of the user element represents the user ID of the BroadWorks subscriber.
userUid	Text	Yes	The *userUid* attribute value uniquely identifies the user within BroadWorks.

Name and Attributes	Allowed Values	Required	Comments
userType	“CallClient”	Yes	The *userType* attribute indicates which user type profile is being updated. CAP-C and CCC2 only.
call	Elements	Yes	This tag identifies the call whose state information is being updated.
callId	Text	Yes	The *callId* attribute is the call identifier, which uniquely identifies the active call. The *callId* is unique within one Application Server.
extTrackingId	Text	Yes	For VoIP users, extTrackingId is the unique ID used to identify a call that comes into BroadWorks, even after being forwarded to different users. For INSC users, extTrackingId contains the callId used over the Wholesale Protocol. It is a globally unique value.
callCenterUserId	Text	No	When present, this field represents the Call Center ID that was called before the call was redirected to the Call Center agent. When this field is not present, it means that either the user is not a Call Center agent, the agent was called directly, or that the agent originated the call.
state	“0” (idle) “1” (alerting) “2” (active) “3” (held) “4” (remote held) “5” (released) “6” (detached) “7” (client alerting)	Yes	The state of the call.
appearance	“–1”, “1” – N	Yes	The appearance of the call. -1 means that the call is not currently using a call appearance. For VoIP users, this is the line appearance on the phone or device this call is on. For INSC users, this identifies the instance when there are multiple simultaneous call activities for the user. For example, if there are two simultaneous call terminations for a user, one has appearance 1 and the other has appearance 2.
personality	“0” (BroadWorks Originator) “1” (Originator) “2” (Terminator)	Yes	The personality of the call. The personality indicates whether this user originated this call or this call was placed to the user.

Name and Attributes	Allowed Values	Required	Comments
callType	"0" (unknown) "1" (intra-group) "2" (enterprise) "3" (network) "4" (network URL) "5" (emergency) "6" (repair) "7" (error) "8" (city-wide centrex) "9" (private dial plan)	Yes	The call type for this call leg.
releaseCause	See Allowed Tag Values.	Yes	The release cause of the call. This tag has meaning when the call is released. Until then, it has a default value of "0" that can be ignored.
remoteCountryCode	Text	Yes	The country code of the remote phone number in the call.
remoteNumber	Text	Yes	The remote phone number in the call. "Unavailable" if the remote number is not available. "Private" if privacy restrictions are in place. Remote phone number if neither one of the two above applies.
remoteName	Text or Null	Yes	The remote name (if available) in the call.
localAltType	See Allowed Tag Values.	Yes	The local alternate address type. This indicates if the local number is a BroadWorks main or alternate number for the user.
linePort	Empty/Text	No	The linePort that uniquely identifies the location the call is associated with when Multiple Call Arrangement is enabled for a user.
localNumber	Text	Yes	This is the local number of the user involved in the call. It should correspond with the localAltType. CCC2 only.
redirectToNum	Text	No	This is the number to which the call is being redirected. CCC2 only.
redirectToReason	Text	No	This is the reason for the redirectToNumber. CCC2 only.
redirectFromCountryCode	Empty/Text	No	This parameter applies to VoIP users only. This is the country code of the redirectFromNumber.
redirectFromNumber	Empty/Text	No	This parameter applies to VoIP users only. This identifies the party where the call was redirected from. Presentation indicator rules apply.

Name and Attributes	Allowed Values	Required	Comments
redirectFromName	Empty/Text	No	This parameter applies to VoIP users only. This is the name associated with the redirectFromNumber. Presentation indicator rules apply.
redirectFromReason	Empty/Text	No	This parameter applies to VoIP users only. This is the reason the call was redirected by the redirectFromNumber.
redirectFromCounter	Text	No	This parameter applies to INSC users only. This parameter is copied from the redirectCounter in the Wholesale Protocol serviceRequest message.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously. The value is left blank for CCC2.
origCalledNum	Text	No	This parameter applies to INSC users only. This parameter is copied from the origCalledNum in the Wholesale Protocol serviceRequest message.
origRedirReason	Text	No	This parameter applies to INSC users only. This parameter is copied from the origRedirReason in the Wholesale Protocol serviceRequest message.
recallType	"BCO", "Transfer", "CallPark", "AHR"	No	The recallType indicates the type of recall the call is for and is only present when the call is a recall/callback for one of the following: "BCO" – For Busy Camp-On Callback "Transfer" – for Call Transfer Recall "CallPark" – For Call Park Recall "AHR" – For Automatic Hold/Retrieve Recall
remoteTelUri	Text	No	This parameter indicates the remote party phone number in E.164 format. An extension can be denoted with ";ext=…", and the phone context with ";phone-context=…", when present. Value can be "Private" when presentation is private for the remote party or "Unavailable" when the remote user is unknown.
callStartTime	Text	No	The time (in milliseconds since January 1, 1970 GMT) the call was initiated or received by the user.
callAnswerTime	Text	No	The time (in milliseconds since January 1, 1970 GMT) the call was answered.

Name and Attributes	Allowed Values	Required	Comments
holdReminder	Empty	No	If present, indicates the hold reminder timer has expired on the server.
callCenter	Elements	No	This tag identifies the call center information for the call IF the call is from a call center.
Id	Text	No	The *id* attribute (under the *callCenter* tag) represents the ID of the Call Center.
DNISName	Text	No	DNIS name if available.
DNISNumber	Text	No	DNIS number if available
callWaitTime	Text	No	Amount of time the call has been waiting in the queue in milliseconds
longestCallWaitTime	Text	No	Longest wait time of a call in the queue in milliseconds
numCallsInQueue	Text	No	Number of calls remaining in the queue

Following is a CAP-C example of the first message after placing a call from the CommPilot Call Manager:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="callUpdate">
      <commandData>
        <user id="joe@abc.com" userType="CallClient" userUid="141466199">
          <call callId="localHost220335:0" extTrackingId="102:1">
            <appearance>1</appearance>
            <personality>0</personality>
            <state>1</state>
            <releaseCause>0</releaseCause>
            <remoteCountryCode>1</remoteCountryCode>
            <remoteNumber>3015554000</remoteNumber>
            <remoteName></remoteName>
            <remoteTelUri>tel:+13015554000;ext=4000</remoteTelUri>
            <localAltType>0</localAltType>
            <linePort>2403649001@192.168.1.76<linePort>
            <redirectFromCountryCode>1</redirectFromCountryCode>
            <redirectFromNumber>3015554000</redirectFromNumber>
            <redirectFromName></redirectFromName>
            <redirectFromReason>do-not-disturb</redirectFromReason>
            <callStartTime>1235684031000</callStartTime>
          </call>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

Following is a CAP-C example of a call center call:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callUpdate">
    <commandData>
      <user id="north00@mtlasdev84.net" userType="CallClient"
```

```
     userUid="101567640">
        <call callId="localHost2615:0" extTrackingId="17:1">
          <appearance>-1</appearance>
          <personality>1</personality>
          <state>5</state>
          <releaseCause>0</releaseCause>
          <remoteCountryCode>1</remoteCountryCode>
          <remoteNumber>603</remoteNumber>
          <remoteName/>
          <localAltType>0</localAltType>
          <callType>2</callType>
          <callStartTime>1235684031000</callStartTime>
          <callAnswerTime>1235684062235</callAnswerTime>
          <callCenterUserId>sales_cc@broadsoft.com</callCenterUserId>
           <callCenter id="sales_cc@broadsoft.com">
             <DNISName>Sales Call Center</DNISName>
             <DNISNumber>7034556766</localAltType>
             <callWaitTime>2000</callWaitTime >
             <longestWaitTime>60000</longestWaitTime>
             <numCallsInQueue>4</numCallsInQueue>
           </callCenter>
           <holdReminder/>
        </call>
        <applicationId>BroadWorks Supervisor</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is a CAP-C example of an existing call that was released:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="callUpdate">
      <commandData>
        <user userType="CallClient" userUid="141466199">
          <call callId="localHost220352:0" extTrackingId="102:1">
            <appearance>1</appearance>
            <personality>1</personality>
            <state>5</state>
            <releaseCause>0</releaseCause>
            <remoteCountryCode>1</remoteCountryCode>
            <remoteNumber>3015554000</remoteNumber>
            <remoteName>Jane Doe</remoteName>
            <remoteTelUri>tel:+13015554000</remoteTelUri>
            <localAltType>0</localAltType>
            <callStartTime>1235684031000</callStartTime>
            <callAnswerTime>1235684038934</callAnswerTime>
          </call>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

Following is a CCC2 example of a call forwarded to another number because of a busy condition:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
```

```
    <command commandType="callUpdate">
      <commandData>
        <user id="joe@abc.com" userType="CallClient" userUid="141466199">
          <call callId="localHost220335:0" extTrackingId="102:1">
            <appearance>1</appearance>
            <personality>0</personality>
            <state>6</state>
            <releaseCause>0</releaseCause>
            <remoteCountryCode>1</remoteCountryCode>
            <remoteNumber>3015554000</remoteNumber>
            <remoteName></remoteName>
            <localAltType>0</localAltType>
            <localNumber>2405551000</localNumber>
            <redirectToNumber>3016661234</redirectToNumber>
            <redirectToReason>user-busy</redirectToReason>
            <callStartTime>1235684031000</callStartTime>
            <callAnswerTime>1235684038934</callAnswerTime>
          </call>
          applicationId></applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

6.10.2 Attendant Console User (CAP-C)

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber that this profileUpdate is being sent for.
id	Text	Yes	The *id* attribute of the user element represents the user ID of the BroadWorks subscriber that registered via a previous registerRequest message.
userUid	Text	Yes	The *userUid* attribute value uniquely identifies the console user.
userType	“AttendantConsole”	Yes	The *userType* attribute indicates which user type should receive the call update.
monitoredUserUid	Text	Yes	The UID of the user being monitored by the attendant console.
monitoredUserId	Text	Yes	The monitoredUserId element identifies the ID of the user being monitored by the attendant console.
call	Elements	Yes	This tag identifies the call whose state information is being updated.
callId	Text	Yes	The *callId* attribute is the call identifier, which uniquely identifies the call.
extTrackingId	Text	Yes	The unique ID used to identify a call that comes into BroadWorks, even after it is forwarded to different users.

Name and Attributes	Allowed Values	Required	Comments
state	"0" (idle) "1" (alerting) "2" (active) "3" (held) "4" (remote held) "5" (released) "6" (detached) "7" (client alerting)	Yes	The state of the call. For the Attendant Console, it is restricted to either 2 (active) or 5 (released).
remoteNumber	Text	Yes	The remote phone number in the call.
remoteName	Text	Yes	The remote name (if available) in the call.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously.
remoteTelUri	Text	No	This parameter indicates the remote party phone number in E.164 format. An extension can be denoted with ";ext=…", and the phone context with ";phone-context=…", when present. Value can be "Private" when presentation is private for the remote party or "Unavailable" when the remote user is unknown.
callStartTime	Text	No	The time (in milliseconds since January 1, 1970 GMT) the call was initiated or received by the user.
callAnswerTime	Text	No	The time (in milliseconds since January 1, 1970 GMT) the call was answered.
holdReminder	Empty	No	If present, indicates the hold reminder timer has expired on the server.
callCenter	Elements	No	This tag identifies the call center information for the call IF the call is from a call center.
id	Text	No	The *id* attribute (under the *callCenter* tag) represents the ID of the Call Center.
DNISName	Text	No	DNIS name if available.
DNISNumber	Text	No	DNIS number if available
callWaitTime	Text	No	Amount of time the call has been waiting in the queue in milliseconds
longestCallWaitTime	Text	No	Longest wait time of a call in the queue in milliseconds
numCallsInQueue	Text	No	Number of calls remaining in the queue

Following is an example of a call update for a new call for a monitored user:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callUpdate">
    <commandData>
      <user id="ac@abc.com" userType="AttendantConsole"
          userUid="142192203">
        <monitoredUserUid>139537371</monitoredUserUid>
        <monitoredUserId>joe@abc.com</monitoredUserId>
        <call callId="localHost221957:0" extTrackingId="154:1">
          <state>2</state>
          <remoteNumber>2403645132</remoteNumber>
          <remoteName>John Smith</remoteName>
          <remoteTelUri>tel:+12403645132</remoteTelUri>
          <callStartTime>1235684031000</callStartTime>
          <callAnswerTime>1235684038934</callAnswerTime>
        </call>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Allowed tag values are as follows:

<releaseCause>: “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9” | “10” | “11” | “12” | “13” | “14”

0: Normal

1: Flash

2: Abnormal

3: Busy

4: Forbidden

5: Route Failure

6: Global Failure

7: Request Failure

8: Server Failure

9: Translation Failure

10: Temporarily Unavailable

11: User Not Found

12: Request Timeout

13: Dial Tone Timeout

14: Insufficient Credits

<localAltType>: “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9” | “10”

0: Main

1: Alt1

2: Alt2

3: Alt3

4: Alt4

5: Alt5

6: Alt6

7: Alt7

8: Alt8

9: Alt9

10: Alt10

- <redirectToReason> is “unknown” | “user-busy” | “unavailable” | “no-answer” | “unconditional” | “time-of-day” | “do-not-disturb” | “deflection” | “follow-me” | “out-of-service” | “away” | “transfer” | “voicemail”
- <redirectFromReason> is “unknown” | “user-busy” | “unavailable” | “no-answer” | “unconditional” | “time-of-day” | “do-not-disturb” | “deflection” | “follow-me” | “out-of-service” | “away” | “transfer” | “voicemail” | “fax-deposit” | “BW-ImplicitID” | “BW-ExplicitID” | “ic-user-outdial” | “hunt-group” | “call-center”

6.11 callAction

Description	Sent by a client to indicate that an action needs to be taken on an existing active call. Also sent by a client to place an outgoing call. A **callUpdate** message is issued by BroadWorks to indicate the change on the active call caused by this **callAction** message.
Protocols	CAP-C, CCC2
Response message	Optionally
Multi-format	No
Message direction	Client to server

Tag names and allowed values are as follows:

Name and Attributes	**Allowed Values**	**Required**	**Comments**
User	Elements	Yes	This tag identifies the BroadWorks subscriber who is performing the call action.
userType	“CallClient”	Yes	The *userType* attribute indicates what type of client application is sending this message.
userUid	Text	No	The *userUid* attribute uniquely identifies the BroadWorks user. Either the userUid or the *id* attribute is required (the id is preferred).
id	Text	No	The *user id* attribute uniquely identifies the BroadWorks subscriber who is sending the message. Either the userUid or the *id* attribute is required (the id is preferred).
action	Elements	Yes	This tag identifies the call action being taken.

Name and Attributes	Allowed Values	Required	Comments
actionType	“Dial” “Redial” “Hold” “Release” “Answer” “Xfer” "XferCC" “XferConsult” “XferVM” “ConfStart” “ConfHold” “ConfAnswer” “ConfRelease” “ConfAdd” “CallPark” “GroupCallPark” “CallParkRetrieve” “MakeCallAsCC” “MakePersonalCall” “MakeAgentEscalationCall” “MakeAgentEmergencyCall” “SilentMonitorCall” “ConferenceUnmute” “TagCallWithDispositionCode” “CustOriginatedTraceCall”	Yes	The *actionType* action attribute identifies the specific call action that is to be taken. All actionTypes are applicable to VoIP users. The following actionTypes are applicable to INSC users: ▪ Release ▪ Answer ▪ Xfer ▪ XferVM
actionParam	Text	No	This element, placed inside the action element, gives the various data items required with different call actions. This tag’s value depends on its name, which is given by its attribute *actionParamName*. Depending on the actionType value, zero to N actionParam tags might exist.
actionParamName	“Number” “CallId” “UserId” “DNISNumber” “DispositionCode”	Yes	This attribute identifies the name of the actionParam tag. The CallId should be used in all cases when the call has already been established. The Number should be used for initiating new calls including transfers. The UserId is allowed to be of the form “userId” if using the default system domain or “userId@domain” for any valid domain on the system. Note that the UserId value is currently only valid for the “CallPark” and “CallParkRetrieve” *actionType* values. The DNISNumber is the DNIS number of the call center to be used in the CLID for the call, when actionType is MakeCallAsCC. The DispositionCode should be used when tagging a call with a disposition code.

Name and Attributes	Allowed Values	Required	Comments
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously. CAP-C only.
responseRequested	"True", "False"	No	Attribute of the action element. Default value is False. Only supported for actionType TagCallWithDispositionCode and CustOriginatedTraceCall. Results in a CallActionResponse message from server to client.
id	Text	No	This action attribute is optional and if included in the request is returned in the response. It can be used to correlate action requests and responses. Only applicable when responseRequested is True.

Here are the parameters each actionType takes:

- Dial
  - Number – The number to dial
- Redial
  - (none)
- Hold
  - CallId – The call id to hold
- Release
  - CallId – The call id to release
- Answer
  - CallId – The call id to answer
- Xfer
  - CallId – The call id to transfer
  - Number – The number to transfer the call to
- XferCC
  - CallId – The call id to transfer
  - Number – The number to transfer the call to
- XferConsult
  - CallId – The first call id to transfer
  - CallId – The second call id to transfer
- XferVM
  - CallId – The call id to transfer
  - Number – The number to transfer the call to

- ConfStart
  - CallId – The first call id to conference
  - CallId – The second call id to conference
- ConfHold
  - (none)
- ConfAnswer
  - (none)
- ConfRelease
  - (none)
- ConfAdd
  - CallId – The call id to add to the conference
- CallPark
  - CallId – The call id to park
  - Number – The number against which to park the call
- GroupCallPark
  - CallId – The call id to park
- CallParkRetrieve
  - UserId – The user id to retrieve the parked call from
- MakeCallAsCC
  - Number – The number to call
  - DNISNumber – The call center number to make the call as
- MakePersonalCall (this call action uses the user's clid instead of the DNIS clid for the outgoing call)
  - Number – The number to call
- MakeAgentEscalationCall
  - Number (optional) – the supervisor number for the escalation
- MakeAgentEmergencyCall
  - CallId – The call id for the emergency call
  - Number (optional) – the supervisor number for the emergency call
- SilentMonitorCall
  - Number – The number of the call center agent to monitor
- ConferenceUnmute (his call action is used when a user wants to escalate a silently monitored conference to a barge-in)
  - (none)
- TagCallWithDispositionCode (this call action supports the responseRequested attribute)
  - CallId (optional) – if not included, applies to last incoming and outgoing ACD call

  - DispositionCode – The disposition code to tag to the call
- CustOriginatedTraceCall (this call action supports the responseRequested attribute)
  - CallId (optional) – The call id of the call to trace

Following is a CAP-C example of “Dial”, which places an outgoing call (for CCC2, the applicationId would not be present):

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <action actionType="Dial">
          <actionParam
            actionParamName="Number">3015554000</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is a CAP-C example of “Redial”, which places an outgoing call using the last number dialed:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <action actionType="Redial"/>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is a CAP-C example of “Hold”, which places an existing call on hold:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <action actionType="Hold">
          <actionParam
            actionParamName="CallId">localHost220431:0</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is a CAP-C example of “Release”, which releases an existing call:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
```

```
      <commandData>
        <user userType="CallClient" id="joe@abc.com">
          <action actionType="Release">
            <actionParam
              actionParamName="CallId">localHost220391:1</actionParam>
          </action>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

Following is a CAP-C example of “Answer”, which accepts an incoming call, or takes a held call off hold:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="callAction">
      <commandData>
        <user userType="CallClient" id="joe@abc.com">
          <action actionType="Answer">
            <actionParam
              actionParamName="CallId">localHost220562:0</actionParam>
          </action>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

Following is a CAP-C example of “Xfer”, which transfers an existing call to another number (blind transfer):

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="callAction">
      <commandData>
        <user userType="CallClient" id="joe@abc.com">
          <action actionType="Xfer">
            <actionParam
              actionParamName="CallId">localHost220741:0</actionParam>
            <actionParam
              actionParamName="Number">3015552000</actionParam>
          </action>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

Following is a CAP-C example of “XferCC”, which transfers an existing call to a Call Center number (blind transfer). The call is transferred to the top of the queue if there are queued calls in the Call Center.

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="callAction">
      <commandData>
        <user userType="CallClient" id="joe@abc.com">
          <action actionType="XferCC">
            <actionParam
              actionParamName="CallId">localHost220741:0</actionParam>
            <actionParam
              actionParamName="Number">3015552000</actionParam>
          </action>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

Following is a CAP-C example of “XferConsult”, which transfers an existing call to another existing call (transfer with consultation):

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <action actionType="XferConsult">
          <actionParam
            actionParamName="CallId">localHost220391:0</actionParam>
          <actionParam
            actionParamName="CallId">localHost220562:0</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is a CCC2 example of “XferVM”, which transfers an existing call to a given voice mail destination:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <action actionType="XferVM">
          <actionParam
            actionParamName="CallId">localHost220562:0</actionParam>
          <actionParam actionParamName="Number">8985</actionParam>
        </action>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is a CAP-C example of “ConfStart”, which initiates a conference with two existing calls:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <action actionType="ConfStart">
          <actionParam
            actionParamName="CallId">localHost220391:0</actionParam>
          <actionParam
            actionParamName="CallId">localHost220431:0</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is a CAP-C example of “ConfHold”, which holds the existing conference this user is in:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <action actionType="ConfHold"/>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is a CAP-C example of “ConfAnswer”, which takes the hold off the existing conference this user is in:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <action actionType="ConfAnswer"/>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is a CAP-C example of “ConfRelease”, which releases or ends the existing conference this user is in:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <action actionType="ConfRelease"/>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is a CAP-C example of “ConfAdd”, which adds an existing call to a conference:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <action actionType="ConfAdd">
          <actionParam
            actionParamName="CallId">localHost220431:0</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

The following CAP-C example shows the “CallPark” action sent in a callAction command using an *actionParamName* of “Number”:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="u1">
        <action actionType="CallPark">
          <actionParam
actionParamName="CallId">localHost6:0</actionParam>
          <actionParam actionParamName="Number">9002</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

The following CAP-C example shows the “CallPark” action sent in a callAction command using an *actionParamName* of “UserId”:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="u1">
        <action actionType="CallPark">
          <actionParam
actionParamName="CallId">localHost6:0</actionParam>
          <actionParam actionParamName="UserId">u2</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

The following CAP-C example shows the “GroupCallPark” action sent in a callAction command:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="u1">
        <action actionType="GroupCallPark">
          <actionParam
actionParamName="CallId">localHost6:0</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

The following CAP-C example shows the “CallParkRetrieve” action sent in a callAction command using an *actionParamName* of “Number”:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="u1">
        <action actionType="CallParkRetrieve">
          <actionParam actionParamName="Number">9002</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

The following CAP-C example shows the “CallParkRetrieve” action sent in a callAction command using an *actionParamName* of “UserId”:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="u1">
        <action actionType="CallParkRetrieve">
          <actionParam actionParamName="UserId">u2</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

The following CAP-C example shows the “MakeCallAsCC” action sent in a callAction command:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="callAction">
      <commandData>
        <user userType="CallClient" id="agentid@broadsoft.com">
          <action actionType="MakeCallAsCC">
            <actionParam
              actionParamName="Number">3015554000</actionParam>
            <actionParam
              actionParamName="DNISNumber">2403641000</actionParam>
          </action>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

The following CAP-C example shows the “MakePersonalCall” action sent in a callAction command:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="callAction">
      <commandData>
        <user userType="CallClient" id="agentid@broadsoft.com">
            <action actionType="MakePersonalCall">
              <actionParam
                actionParamName="Number">3015554000</actionParam>
            </action>
            <applicationId>broadSoftExample</applicationId>
          </user>
        </commandData>
      </command>
    </BroadsoftDocument>
```

The following CAP-C example shows the “MakeAgentEscalationCall” action sent in a callAction command:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="agentid@broadsoft.com">
        <action actionType="MakeAgentEscalationCall">
          <actionParam
            actionParamName="Number">3015554000</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

The following CAP-C example shows the “MakeAgentEmergencyCall” action sent in a callAction command:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="agentid@broadsoft.com">
        <action actionType="MakeAgentEmergencyCall">
          <actionParam
            actionParamName="Number">3015554000</actionParam>
          <actionParam
            actionParamName="CallId">localHost220741:0</actionParam>
        </action>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

The following CAP-C example shows the “SilentMonitorCall” action sent in a callAction command:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callAction">
    <commandData>
      <user userType="CallClient" id="supervisorid@broadsoft.com">
        <action actionType="SilentMonitorCall">
          <actionParam
           actionParamName="Number">3013334322</actionParam>
        </action>
```

```
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

The following CAP-C example shows the “ConferenceUnmute” action sent to escalate a slient monitor conference to a barge-in:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="callAction">
      <commandData>
        <user userType="CallClient" id="supervisorid@broadsoft.com">
          <action actionType="ConferenceUnmute"/>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

The following CAP-C example shows the “TagCallWithDispositionCode” action sent in a callAction command, requesting a response and with the optional “id” attribute used by the client for correlation purpose:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="callAction">
      <commandData>
        <user userType="CallClient" id="agentid@broadsoft.com">
          <action actionType="TagCallWithDispositionCode"
responseRequested="True" id="1">
            <actionParam
              actionParamName="CallId">localHost220741:0</actionParam>
            <actionParam
              actionParamName="DispositionCode">promotionA</actionParam>
          </action>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

The following CAP-C example shows the “CustOriginatedTraceCall” action sent in a callAction command, requesting a response (in this case without the optional “id” attribute):

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="callAction">
      <commandData>
        <user userType="CallClient" id="agentid@broadsoft.com">
          <action actionType="CustOriginatedTraceCall"
responseRequested="True">
            <actionParam
              actionParamName="CallId">localHost220741:0</actionParam>
          </action>
          <applicationId>broadSoftExample</applicationId>
        </user>
      </commandData>
    </command>
  </BroadsoftDocument>
```

6.12 CallActionResponse

Description	Sent by the server to the client application in response to the **callAction** message if a response was requested. If there is an error, it is indicated in the response message.
Protocols	CAP-C, CCC2
Response message	No
Multi-format	No
Message direction	Server to client

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks user who requested the original callAction.
id	Text	Yes	The *id* attribute represents the user ID of the BroadWorks subscriber that sent the request.
userType	“CallClient”	Yes	The *userType* attribute indicates the type of client functionality the user has from a BroadWorks perspective. Only “CallClient” is supported here.
userUid	Text	Yes	The *userUid* attribute value uniquely identifying the user within BroadWorks.
action	Element	Yes	The action Element.
actionType	Same values as in callAction message	Yes	This action attribute is the actionType the result response is for.
id	Text	No	This action attribute is included if it was present in the request. It can be used to correlate action requests and responses.
result	Element	Yes	This tag indicates the result of the callAction request.
success	“True” “False”	Yes	This result attribute identifies whether the call action succeeded or not. “True” indicates the request is successful and “False” indicates there is an error.
message	Text	No	This element (inside the result element) contains an error message if the action was not successful.

The following CAP-C example shows a successful response to the “TagCallWithDispositionCode” action:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callActionResponse">
```

```
    <commandData>
      <user userType="CallClient" userUid="2343444398">
        <id>agentid@broadsoft.com</id>
        <action actionType="TagCallWithDispositionCode" id="1">
          <result success="True"/>
        </action>
        <applicationId>broadsoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

The following CAP-C example shows a failure response to the "TagCallWithDispositionCode" action:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callActionResponse">
    <commandData>
      <user userType="CallClient" userUid="2343444398">
        <id>agentid@broadsoft.com</id>
        <action actiontype="TagCallWithDispositionCode" id="1">
          <result success="False">
            <message>Invalid disposition code</message>
          </result>
        </action>
        <applicationId>broadsoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.13 callControlInfo

Description	Sent by BroadWorks to a call client specifying various allowed per-Call Control functionalities. For example, such a message would be sent if the device on which the call was received allows a call client to answer the call. This message is **sent for VoIP users**. It is **not sent for INSC users**.
Protocols	CAP-C, CCC2
Response message	No
Multi-format	No
Message direction	Server to client

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber for whom Call Control information is being sent.
id	Text	Yes	The *id* attribute represents the user ID of the BroadWorks subscriber.
userUid	Text	Yes	The *userUid* attribute value uniquely identifies the user within BroadWorks.

Name and Attributes	Allowed Values	Required	Comments
userType	“CallClient”	Yes	The *userType* attribute indicates which user type profile is being updated.
call	Elements	Yes	This tag identifies the call whose control information is being communicated.
callId	Text	Yes	The *callId* attribute is the call identifier, which uniquely identifies the active call.
extTrackingId	Text	Yes	The unique ID used to identify a call that comes into BroadWorks, even after it is forwarded to different users.
control	“True” “False”	Yes	This tag identifies the specific Call Control ability being enabled or disabled. The *controlType* attribute is to be used to distinguish various control abilities.
controlType	“Answer” “Hold” “Retrieve”	Yes	The specific Call Control ability. “Answer” means that the call client application is allowed to answer an incoming call. “Hold” means that the call client application is allowed to put a call on hold. “Retrieve” means that the call client application is allowed to retrieve a held call. The call client application may respond to this message with a callAction CAP message.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously. The value is left blank for CCC2.

The following is a CAP-C example. For CCC2, the applicationId would be blank, but the message would otherwise be the same:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="callControlInfo">
    <commandData>
      <user id="joe@abc.com" userType="CallClient" userUid="141466199">
        <call callId="localHost220335:0" extTrackingId="102:1">
          <control controlType="Answer">True</control>
          <control controlType="Hold">True</control>
          <control controlType="Retrieve">False</control>
        </call>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.14 externalNotify

Description	Sent by a client to set the message waiting indicator on the user’s phone. This message is **sent for VoIP users**. It is **not sent for INSC users**.

Protocols	CAP-C, CCC2
Response message	No
Multi-format	No
Message direction	Client to server

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
User	Elements	Yes	This tag identifies the BroadWorks subscriber who is performing the call action.
userType	“CallClient”	Yes	The *userType* attribute indicates the type of client application sending this message.
userUid	Text	No	The *userUid* attribute uniquely identifies the BroadWorks user. Either the *userUid* or the *id* attribute is required (the id is preferred).
id	Text	No	The *user id* attribute uniquely identifies the BroadWorks subscriber who is sending the message. Either the *userUid* or the *id* attribute is required (the *id* is preferred).
thirdPartyMWI	“on” | “off”	No	If this tag is present, the third-party voice message MWI is set according to the value.
clearVMMWI	Empty	No	If this tag is present, the voice messaging MWI is cleared.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType are run simultaneously. CAP-C only.

The following is a CAP-C example. The example would be similar for CCC2, except that the *applicationId* attribute would not be present.

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="externalNotify">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <thirdPartyMWI>off</thirdPartyMWI>
        <clearVMMWI></clearVMMWI>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.15 infoRequest

Description	Sent by a client requesting information about the BroadWorks CAP, such as a version number.
Protocols	CAP-C, CCC2
Response message	Yes

Multi-format	No
Message direction	Client to server

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
capVersionRequest	Empty	Yes	This tag identifies that the client is requesting the CAP version number.

The following is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="infoRequest">
    <commandData>
      <capVersionRequest/>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.16 infoResponse

Description	Sent by a server in response to a client's request about the BroadWorks CAP information, such as version number. The same tag may appear multiple times, for example, BroadWorks can return two "*capVersionResponse*" with values 1.1 and 1.0 respectively. It means that the current server supports both CAP version 1.1 and CAP version 1.0.
Protocols	CAP-C, CCC2
Response message	No
Multi-format	No
Message direction	Server to client

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
capVersionResponse	Text	Yes	This tag identifies the CAP version number that the current BroadWorks server supports.

The following is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="infoResponse">
    <commandData>
       <capVersionResponse>17.0</capVersionResponse>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.17 serverStatusRequest

Description	Sent by the client application to verify that the client-to-server connection is still active. It is also sent by the Application Server to all its CAP-C/CCC2

	connections to keep/verify that the connection is still active. There is no response to this message.
Protocols	CAP-C, CCC2
Response message	No
Multi-format	No
Message direction	Client to server or server to client

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	No	This tag identifies the BroadWorks subscriber for whom this message is being sent.
id	Text	No	The *id* attribute represents the user ID of the BroadWorks subscriber.
userUid	Text	No	The *userUid* attribute value uniquely identifies the user within BroadWorks.
userType	“CallClient” “AttendantConsole”	No	The *userType* attribute indicates the type of client application sending this message.
applicationId	Text	No	The ID of the application when multiple clients of the same userType are run simultaneously. Mandatory when the user tag is present.

The following is an example of when sent from client to server:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="serverStatusRequest">
    <commandData>
      <user userType="CallClient" userUid="141466199">
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

The following is an example of when sent from server to client:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="serverStatusRequest">
    <commandData/>
  </command>
</BroadsoftDocument>
```

6.18 monitoringUsersRequest

Description	Sent by the client application to server to modify the list of monitored users.
Protocols	CAP-C
Response message	Yes
Multi-format	No
Message direction	Client to server

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks user who is requesting the changes to the monitored user list.
userType	"AttendantConsole"	Yes	The *userType* attribute indicates the type of client functionality the user has from a BroadWorks perspective. Only "AttendantConsole" is supported here.
userUid	Text	No	The *userUid* attribute value uniquely identifying the user within BroadWorks. Either the userUid or the *id* attribute is required (the id is preferred).
id	Text	No	The *user id* attribute uniquely identifies the BroadWorks subscriber who is sending the message. Either the *userUid* or the *id* attribute is required (the *id* is preferred).
monitoring	Element	Yes	This tag indicates the requested change to the monitored user's list.
monType	"Add" "Delete" "Replace"	Yes	This attribute identifies the specific action that is to be taken on the monitored user's list.
monUser	Text	Yes	The login ID of the user to be added to/removed from the monitored user list. There can be more than one such attribute in the message.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType run simultaneously.

The following is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="monitoringUsersRequest">
    <commandData>
      <user userType="AttendantConsole" id="bill@telcom.com">
        <monitoring monType="Add"/>
        <monUser>joepublic</monUser>
        <monUser>bob@telcom.com</monUser>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.19 monitoringUsersResponse

Description	Sent by the server to the client application in response to the **monitoringUsersRequest** message. If there is an error, each user that failed will be included in this response message.
Protocols	CAP-C
Response message	No
Multi-format	No
Message direction	Server to client

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks user who is requesting the changes to the monitored user list.
id	Text	Yes	The *id* attribute represents the user ID of the BroadWorks subscriber that registered via a previous registerRequest message.
userType	“AttendantConsole”	Yes	The *userType* attribute indicates the type of client functionality the user has from a BroadWorks perspective. Only “AttendantConsole” is supported here.
userUid	Text	Yes	The *userUid* attribute value uniquely identifying the user within BroadWorks.
monResult	Element	Yes	This tag indicates the result of the requested change to the monitored user’s list.
Success	“True” “False”	Yes	This attribute identifies if there is an error in modifying the monitored users list. “True” indicates the request is successful and “False” indicates there is at least one error when adding/removing the users.
failedUser	Element	No	This tag indicates a user is failed to be added or removed. There can be more than one such element in the message
Id	Text	Yes	The login ID of the user who failed to be added or removed.
failureReason	“UnknownUser” “UnauthorizedUser” “PrivateUser”	Yes	The failure reason. “UnknownUser” means the user is not found on the server. “UnauthorizedUser” means the user is not in the same group or in the same enterprise as the Attendant Console user. “PrivateUser” means that the target user has the Privacy feature enabled (for phone status).
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType run simultaneously.

Following is a sample of a successful monitoringUsersResponse:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="monitoringUsersResponse">
    <commandData>
      <user id="ac@abc.com" userType="AttendantConsole"
          userUid="139557133">
        <monResult success="True"/>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is a sample of a monitoringUsersResponse with error:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="monitoringUsersResponse">
    <commandData>
      <user id="ac@abc.com" userType="AttendantConsole"
          userUid="139557133">
        <monResult success="False"/>
        <failedUser failureReason="UnauthorizedUser"
          id="bob@telcom.com"/>
        <applicationId>broadSoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.20 queueUpdate

Description	Sent by server to a client to provide state of a call center queue. A queueUpdate is sent when the client user logs in, when the queue is modified, and if a refresh is requested. A queueUpdate containing all the calls in the queue is sent on login as a call center, on queue reorder, and on queue refresh. A queueUpdate with just the specific call information is sent when a call is added or removed from the queue.
Protocols	CAP-C, CCC2
Response message	No
Multi-format	No
Message direction	Server to client

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber for whom call state information is being sent.
id	Text	Yes	The *id* attribute represents the user ID of the BroadWorks subscriber that registered via a previous registerRequest message.

Name and Attributes	Allowed Values	Required	Comments
userUid	Text	Yes	The *userUid* attribute value uniquely identifies the call center user within BroadWorks.
userType	“CallClient”	Yes	The *userType* attribute indicates which user type profile is being updated.
queue	Elements	Yes	This tag contains the queueCalls in the queue update.
updateReason	“Initial”, “CallAdded”, ”CallAnswered”, "CallTransferred", “CallAbandoned”, “CallReordered”, “CallUpdated”, "Refresh", "QueueTimeout", "QueueOverflow", “CallPromoted”	Yes	The reason for the queue update. Initial: The Call Center user has logged in. CallAdded: A call was added to the queue. CallAnswered: An agent answered a call in the queue. CallTransferred: A call was transferred out of the queue. CallAbandoned: A call was removed from the queue. CallReordered: The queue was manually reordered. CallUpdated: An attribute of the call was updated. Refresh: Client requests a refresh of the queue. QueueTimeout: The call was removed from the queue because the cfgna timer expired. QueueOverFlow: The call was sent to voice messaging/treatment because the queue was full. CallPromoted: A call was promoted to a higher priority.
queueCall	Elements	Yes	This tag identifies a call in the Call Center queue. There can be multiple such elements. The relative order of this element with respect to other queueCall elements indicates the position of the call in the queue (when updateReason is Initial, CallReordered, or Refresh).
callId	Text	Yes	The *callId* attribute is the call identifier, which uniquely identifies the call in the queue.
remoteCountryCode	Text	No	The country code of the remote phone number in the call.
remoteNumber	Text	No	The remote phone number in the call.
remoteName	Text or Null	No	The remote name (if available) in the call.
timeAdded	Text	No	The time (in milliseconds since Jan 1, 1970 GMT) the call was added to the queue.
position	Text	No	The position of the call in the queue.

Name and Attributes	Allowed Values	Required	Comments
applicationId	Text	Yes	The ID of the application when multiple clients of the same *userType* run simultaneously. CAP-C only.
mandatoryEntrance	"Not Playing" "Playing"	No	Indicates whether or not the mandatory entrance message is currently being played for this call.
bouncedCall	"True" "False"	No	Indicates whether the call was already offered to an agent and bounced.
reorderedCall	"True" "False"	No	Indicates whether the call was previously reordered within the queue.
preservedWaitTime	Text	No	Indicates the preserved wait time in milliseconds of the call at the moment the call was actually added to the queue. The client application uses the attribute in conjunction with the *timeAdded* attribute to determine the actual wait time of the queued call.
DNISName	Text	No	The DNIS name of the call center for the call.
DNISNumber	Text	No	The DNIS number of the call center for the call.
Priority	"0", "1", "2", "3"	No	Inside the queueCall element, the priority element indicates the call priority inside the queue. 0 = Highest 1 = High 2 = Medium 3 = Low
addTimeInPriorityBucket	Text	No	The time (in milliseconds since January 1, 1970 GMT) at which the entry was added to the priority bucket.
preservedWaitTimeInPriorityBucket	Text	No	The time in milliseconds the call has accumulated when it is inserted in the priority bucket.

The position of the queueCall element indicates its order in the queue; that is, the first queueCall element is the first call in the queue, the second queueCall element is the second call in the queue, and so on.

Following is a CAP-C example of an initial queue update being sent after a Call Center supervisor has successfully logged in as a Call Center:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="queueUpdate">
    <commandData>
      <user id="cc@abc.com" userType="CallClient" userUid="141466199">
        <applicationId>supervisor</applicationId>
      </user>
      <queue updateReason="Initial">
        <queueCall callId="localHost220335:0">
          <remoteCountryCode>1</remoteCountryCode>
          <remoteNumber>3015554000</remoteNumber>
          <remoteName></remoteName>
          <timeAdded>435345345345</timeAdded>
          <mandatoryEntrance>Not Playing</mandatoryEntrance>
```

```
            <bouncedCall>False</bouncedCall>
            <reorderedCall>False</reorderedCall>
            <preservedWaitTime>86234</preservedWaitTime>
          </queueCall>
          <queueCall callId="localHost3435:0 >
            <remoteCountryCode>1</remoteCountryCode>
            <remoteNumber>3015554000</remoteNumber>
            <remoteName></remoteName>
            <timeAdded>234234234</timeAdded>
            <mandatoryEntrance>Not Playing</mandatoryEntrance>
            <bouncedCall>False</bouncedCall>
            <reorderedCall>False</reorderedCall>
            <preservedWaitTime>8299</preservedWaitTime>
          </queueCall>
        </queue>
      </commandData>
    </command>
  </BroadsoftDocument>
```

Following is a CCC2 example of a queue update for a call that is abandoned from a Call Center queue:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="queueUpdate">
      <commandData>
        <user id="cc@abc.com" userType="CallClient" userUid="141466199">
        </user>
        <queue updateReason="CallAbandoned">
         <queueCall callId="localHost3234:0 />
        <queue>
      </commandData>
    </command>
  </BroadsoftDocument>
```

Following is a CAP-C example of a queue update for a call that is transferred out from a Call Center queue:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="queueUpdate">
      <commandData>
        <user id="cc@abc.com" userType="CallClient" userUid="141466199">
          <applicationId>supervisor</applicationId>
        </user>
        <queue updateReason="CallTransferred">
          <queueCall callId="localHost3234:0 />
        <queue>
      </commandData>
    </command>
  </BroadsoftDocument>
```

Following is a CAP-C example of a queue update for a call that is transferred into the beginning of a Call Center queue:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <BroadsoftDocument protocol="CAP" version="17.0">
    <command commandType="queueUpdate">
      <commandData>
        <user id="cc@abc.com" userType="CallClient" userUid="141466199">
          <applicationId>supervisor</applicationId>
        </user>
        <queue updateReason="CallAdded">
        <queueCall callId="localHost3234:0>
          <position>1</position>
        </queueCall>
        <queue>
      </commandData>
    </command>
  </BroadsoftDocument>
```

6.21 queueAction

Description	Sent by a client to indicate that an action needs to be taken on an existing call center queue. The action can be transferring a call from the queue, reordering a call in the queue, or refreshing the queue. A queueUpdate message is usually issued by server to indicate the change on the queue caused by this queueAction message.
Protocols	CAP-C, CCC2
Response message	Yes. Only if action is for a refresh.
Multi-format	No
Message direction	Client to server

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks subscriber who is performing the call action.
userType	“CallClient”	Yes	The *userType* attribute indicates what type of client application is sending this message.
userUid	Text	No	The *userUid* attribute uniquely identifies the BroadWorks Call Center user. Either the *userUid* or the *id* attribute is required (the *id* is preferred).
id	Text	No	The user *id* attribute uniquely identifies the BroadWorks subscriber who is sending the message. Either the *userUid* or the *id* attribute is required (the *id* is preferred).
queueActionData	Elements	Yes	This tag identifies the data for the queue action.
actionType	“Transfer” “Reorder” "Refresh" “Promote”	Yes	The *actionType* attribute identifies the specific queue action that is to be taken.
queueCall	Element	No	This tag identifies a call in the call center queue. There can be only one such element.
callId	Text	No	The *callId* attribute is the call identifier, which uniquely identifies the call in the queue.

Name and Attributes	Allowed Values	Required	Comments
position	Text	No	The position of the call in the queue.
destination	Text	No	The destination number for the call being removed from the queue.
applicationId	Text	Yes	The ID of the application when multiple clients of the same *userType* run simultaneously. CAP-C only.
priority	“0”, “1”, “2”, “3”	No	Inside the queueCall element, the priority element indicates the priority the call should be promoted to, when the actionType is Promote. 0 = Highest 1 = High 2 = Medium 3 = Low

Following is a CAP-C example of a Call Center supervisor reordering a call to move it to the second position in the queue. For CCC2, the *applicationId* attribute would not be present:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="queueAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <applicationId>supervisor</applicationId>
      </user>
      <queueActionData actionType="Reorder">
        <queueCall callId="localhost233:0">
          <position>2</position>
        </queueCall>
      </queueActionData>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is an example of a Call Center supervisor removing (transferring) a call from the queue:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="queueAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <applicationId>supervisor</applicationId>
      </user>
      <queueActionData actionType="Transfer">
        <queueCall callId="localhost233:0">
          <destination>3013334232</destination>
        </queueCall>
      </queueActionData>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is an example of a Call Center supervisor requesting a refresh of the Call Center queue:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="queueAction">
    <commandData>
      <user userType="CallClient" id="joe@abc.com">
        <applicationId>supervisor</applicationId>
      </user>
      <applicationId>supervisor</applicationId>
      <queueActionData actionType="Refresh"/>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is an example of a Call Center supervisor promoting a call in a queue:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="queueAction">
    <commandData>
      <user userType="CallClient" id="supervisor@broadsoft.com">
        <applicationId>supervisor</applicationId>
      </user>
      <queueActionData actionType="Promote">
        <queueCall callId="localhost233:0">
          <priority>0</priority>
        </queueCall>
      </queueActionData>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.22 queueProfileUpdate

Description	Sent by server to a client to provide configuration of a call center queue. A queueProfileUpdate is sent when the client user logs in, and when the call center queue's configuration is modified.
Protocols	CAP-C, CCC2
Response message	No
Multi-format	No
Message direction	Server to client

Tag names and allowed values are as follows:

Name and Attributes	**Allowed Values**	**Required**	**Comments**
User	Elements	Yes	This tag identifies the BroadWorks subscriber for whom call state information is being sent.
id	Text	Yes	The *id* attribute represents the user ID of the BroadWorks subscriber who registered via a previous registerRequest message.
userUid	Text	Yes	The *userUid* attribute value uniquely identifies the Call Center user within BroadWorks.

Name and Attributes	Allowed Values	Required	Comments
userType	"CallClient"	Yes	The *userType* attribute indicates which user type profile is being updated.
initialUpdate	Empty	No	If this element is part of the message, it indicates this message is an initial update. If the element is not part of the message, it indicates this message is a subsequent update.
nightService	"Off" "On" "Manual"	Yes	▪ "Off" indicates that the Night Service is not enabled for the queue. ▪ "On" means the Night Service is enabled for the queue. ▪ "Manual" means that the queue is currently in manual-override mode. When the Night Service is enabled, the client application can obtain the Night Service schedule through the Open Client Interface-Provisioning (OCI-P) interface.
holidayService	"Off" "On"	Yes	▪ "Off" indicates that the Holiday Service is not enabled for the queue. ▪ "On" means the Holiday Service is enabled for the queue. When the Holiday Service is enabled, the client application can obtain the Holiday Service schedule through the OCI-P interface.
forcedForwarding	"Off" "On"	Yes	▪ "Off" indicates that the Forced Forwarding is not enabled for the queue. ▪ "On" means the Forced Forwarding is enabled for the queue.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType run simultaneously.

Following is an example of an initial update of the Call Center queue configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="queueProfileUpdate">
    <commandData>
      <user id="callcenter1@mtlasdev84.net" userType="CallClient"
        userUid="164007290">
        <initialUpdate/>
        <nightService>Off</nightService>
        <holidayService>On</holidayService>
        <forcedForwarding>Off</forcedForwarding>
        <applicationId>BroadWorks Supervisor</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

Following is an example of a subsequent update following a modification of the Call Center queue configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="queueProfileUpdate">
    <commandData>
      <user id="callcenter1@mtlasdev84.net" userType="CallClient"
        userUid="164007290">
        <nightService>Manual</nightService>
        <holidayService>On</holidayService>
        <forcedForwarding>Off</forcedForwarding>
        <applicationId>BroadWorks Supervisor</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.23 queueInfoRequest

Description	Sent by the call center client application to receive queue information
Protocols	CAP-C, CCC2
Response message	Yes
Multi-format	No
Message direction	Client to server

Tag names and allowed values are as follows:

Name and Attributes	**Allowed Values**	**Required**	**Comments**
user	Elements	Yes	This tag identifies the BroadWorks user who is requesting the queue information.
userType	“CallClient”	Yes	The *userType* attribute indicates the type of client functionality the user has from a BroadWorks perspective. Only “CallClient” is supported here.
userUid	Text	No	The *userUid* attribute value uniquely identifying the user within BroadWorks. Either the userUid or the *id* attribute is required (the id is preferred).
id	Text	No	The *user id* attribute uniquely identifies the BroadWorks subscriber who is sending the message. Either the *userUid* or the *id* attribute is required (the *id* is preferred).
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType run simultaneously.

Here is a CAP-C example of a queueInfoRequest:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="queueInfoRequest">
    <commandData>
      <user userType="CallClient">
        <id>agentid@broadsoft.com</id>
        <applicationId>broadsoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.24 queueInfoResponse

Description	Sent by the server to the client application in response to the **queueInfoRequest** message.
Protocols	CAP-C, CCC2
Response message	No
Multi-format	No
Message direction	Server to client

Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
user	Elements	Yes	This tag identifies the BroadWorks user who requested the queue information.
id	Text	Yes	The *id* attribute represents the user ID of the BroadWorks subscriber that sent the request.
userType	“CallClient”	Yes	The *userType* attribute indicates the type of client functionality the user has from a BroadWorks perspective. Only “CallClient” is supported here.
userUid	Text	Yes	The *userUid* attribute value uniquely identifying the user within BroadWorks.
callcenter	Elements	Yes	This tag contains information on a call center. There can be 0 to many such elements, depending on how many queues the agent has joined.
id	Text	Yes	This attribute identifies the specific call center.
numCallsInQueue	Text	No	The number of calls in the queue. Included if the response is a success.
applicationId	Text	Yes	The ID of the application when multiple clients of the same userType run simultaneously.

Here is a CAP-C example of a queueInfoResponse:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="queueInfoResponse">
```

```
    <commandData>
      <user userType="CallClient">
        <id>agentid@broadsoft.com</id>
        <callcenter id="sales@broadsoft.com">
          <numCallsInQueue>2</ numCallsInQueue >
        </callcenter>
        <callcenter id="support@broadsoft.com">
          < numCallsInQueue >3</ numCallsInQueue >
        </callcenter>
        <applicationId>broadsoftExample</applicationId>
      </user>
    </commandData>
  </command>
</BroadsoftDocument>
```

6.25 *datagram (this command is deprecated)*

Description	Sent by a client to BroadWorks. BroadWorks proxies this **datagram** to the intended CAP recipient client. This command is used so that CAP clients may relay messages to one another. Each sender and receiver is uniquely identified by the sender or receiver's user ID, user type, and application ID. ******This command is deprecated and will not be supported in a future release.******
Protocols	CAP-C, CCC2
Response message	No. It is the responsibility of the recipient client to send an acknowledgement so that the sender can validate receipt of the message.
Multi-format	No
Message direction	Client to server Server to client
Tags	Tag names and allowed values are as follows:

Name and Attributes	Allowed Values	Required	Comments
from	user Element	Yes	The *from* tag contains the user element of the user sending the datagram.
user	Elements	Yes	This tag identifies the BroadWorks subscriber who is sending the datagram.
userType	"CallClient" or "AttendantConsole"	Yes	The *userType* attribute indicates the type of client application sending this message.
id	Text	Yes	The user ID of the BroadWorks subscriber who is sending the datagram.
applicationId	Text	Yes	The ID of the application of the user sending the datagram.
to	user Element	Yes	The *to* tag contains the user element of the recipient of the datagram.
user	Elements	Yes	This tag identifies the BroadWorks subscriber who is to receive the datagram.
userType	"CallClient" or "AttendantConsole"	Yes	The *userType* attribute indicates the type of client application receiving this datagram.
id	Text	Yes	The user ID of the BroadWorks subscriber who is receiving the

Name and Attributes	Allowed Values	Required	Comments
			datagram.
applicationId	Text	Yes	The ID of the application of the destination user.
payload	Text	Yes	The data carried in the message.
payloadType	Text	No	The content of the payload. Examples are "text/plain", "text/xml", and so on.

Following is a CAP-C example of a “unicast” message relay using the datagram command:

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument protocol="CAP" version="17.0">
  <command commandType="datagram">
    <commandData>
      <from>
        <user userType="CallClient" id="agent1@broadsoft.com">
          <applicationId>CallCenterAgent</applicationId>
        </user>
      </from>
      <to>
        <user userType="CallClient" id="agent2@broadsoft.com">
          <applicationId>CallCenterAgent</applicationId>
        </user>
      </to>
      <payload payloadType="text/plain" >Hello</payload>
    </commandData>
  </command>
</BroadsoftDocument>
```

7 Call Control Message Flows

This section provides the message flow sequences between BroadWorks and a call client application for Call Control. Except for registration, it applies to both CAP-C and CCC2 in VoIP environments.

7.1 Registration (CAP-C only)

7.1.1 No Calls Active on Registration

Figure 3 shows no calls active on registration.

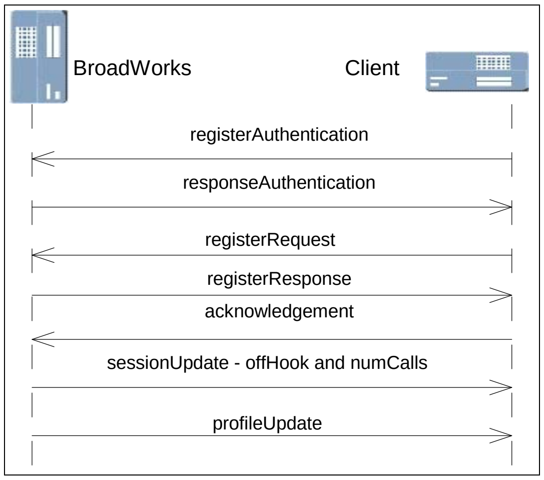

Figure 3 No Calls Active on Registration

7.1.2 One Call Active on Registration

Figure 4 shows one call active on registration (non-secured).

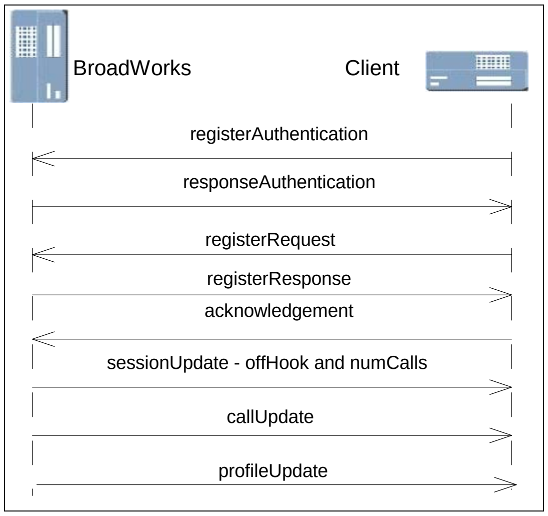

Figure 4 One Call Active on Registration

7.2 Calls (CAP-C and CCC2)

This section applies to both CAP-C and CCC2. Note that for CAP-C, registration must have previously taken place.

7.2.1 Interaction 1 – Placing a Call

Figure 5 shows placing a call as a BroadWorks originator, putting it on hold, retrieving it, and then releasing it.

Note that starting with Release 16.0, the call ID is the same for both call legs (the BroadWorks Originator call leg and the originator call leg). Prior to Release 16.0, a “callUpdate – Released” immediately followed the sessionUpdate for the first leg, and the subsequent messages used another call ID.

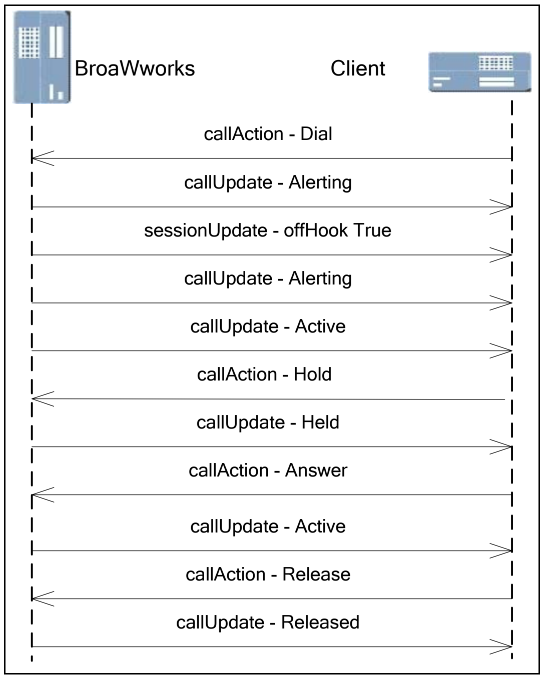

Figure 5 Interaction 1 – Placing Call

7.2.2 Interaction 2 – Transfer

Figure 6 shows transferring with consultation. Note, however, that the call setup for the first call is not shown.

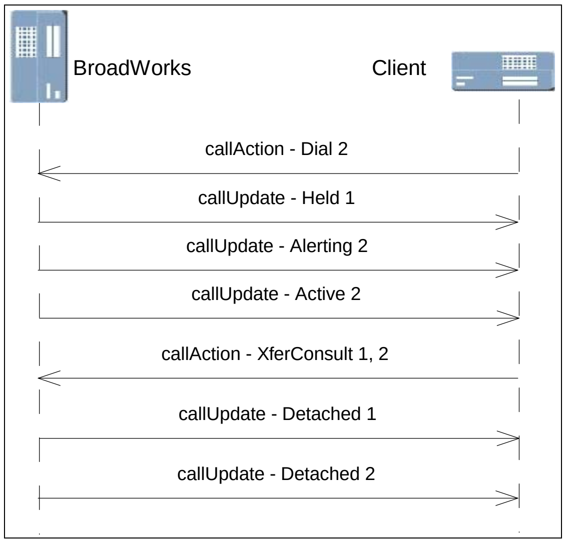

Figure 6 Interaction 2 – Transfer

7.2.3 Interaction 3 – Conference

Figure 7 shows conferencing in two calls, putting the conference on hold, retrieving it, and then releasing it. Note, however, that the call setup for the two calls is not shown.

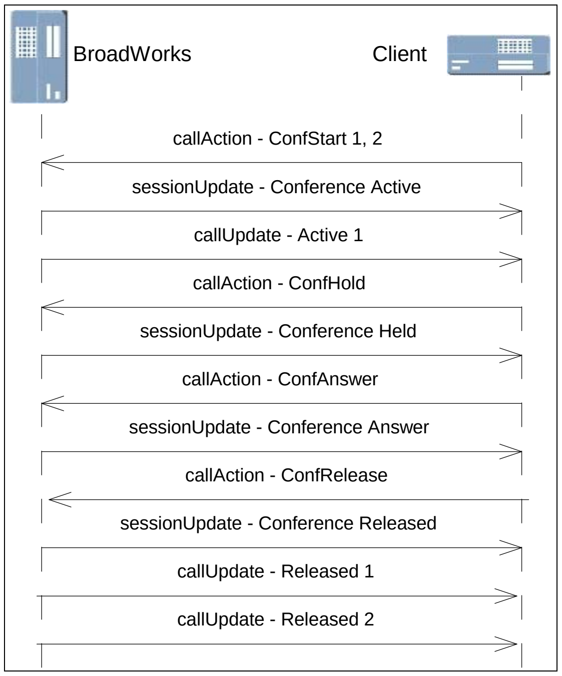
BroadWorks
Client
callAction - ConfStart 1, 2
sessionUpdate - Conference Active
callUpdate - Active 1
callAction - ConfHold
sessionUpdate - Conference Held
callAction - ConfAnswer
sessionUpdate - Conference Answer
callAction - ConfRelease
sessionUpdate - Conference Released
callUpdate - Released 1
callUpdate - Released 2

Figure 7 Interaction 3 – Conference

7.3 Logging out (CAP-C only)

Figure 8 shows what is sent when a CAP-C client logs out (or terminates).

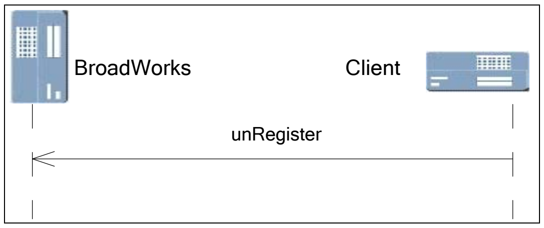

Figure 8 Logging out

8 User Monitoring Message Flows

This section provides the message flow sequences between BroadWorks and an Attendant Console application. This applies to CAP-C only.

8.1 Registration

This case shows an Attendant Console (AC) type of client registering and requesting to monitor *N* users.

The unRegister message sent when the client terminates or the user logs out is not shown in the figure.

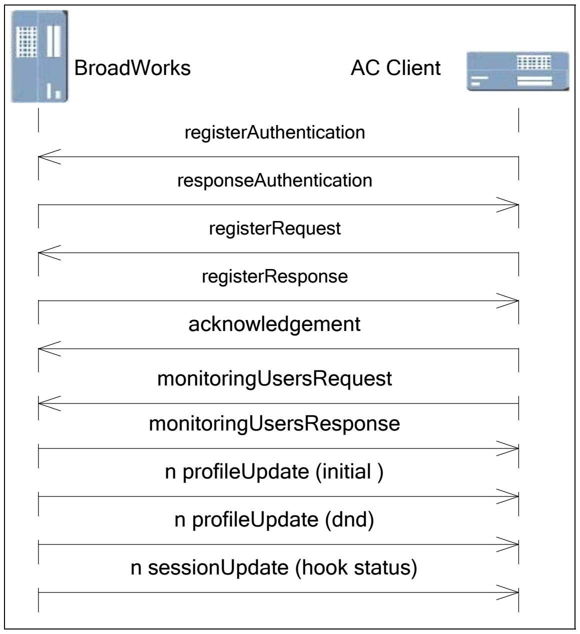

Figure 9 Registration

8.2 Real-Time Updates

8.2.1 Configuration Updates

Figure 10 illustrates that after registration, the Attendant Console client can receive real-time, Do Not Disturb (DND) status, and hook status changes for monitored users as they occur. The profile and session updates are meant to represent monitored users changing their Do Not Disturb and hook status respectively.

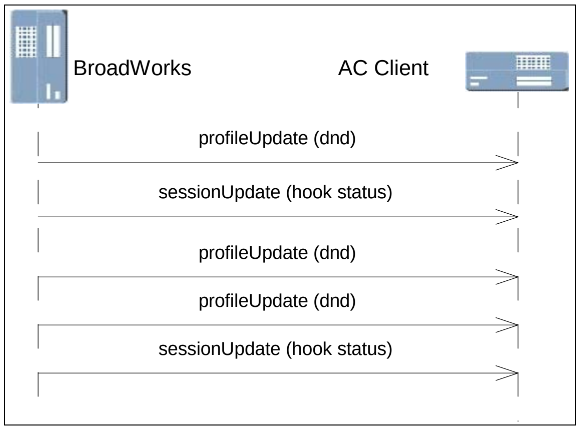

Figure 10 Real-Time Configuration Updates

8.2.2 Call Processing Updates

Figure 11 illustrates that after registration, the Attendant Console client can receive real-time call updates for the monitored users. In this scenario, a monitored user places a call and then hangs up. The monitoring user Attendant Console client receives the following messages:

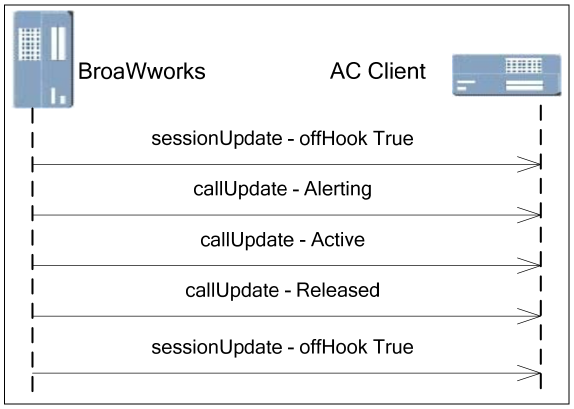

Figure 11 Real-Time Call Processing Updates

9 Call Center Queue Management Message Flows

This section provides the message flow sequences between BroadWorks and a Call Center application. This applies to CAP-C only.

9.1 Registration

9.1.1 Queue is Empty

Figure 12 shows a Call Center registration. It does not differ from other types of registrations, except for the fact that the user ID used is that of the Call Center virtual subscriber (with a userType of "CallClient"). Similar to other types of applications, client log out is done through the unRegister command. This is not shown in *Figure 12*.

In this example, the Call Center queue is initially empty.

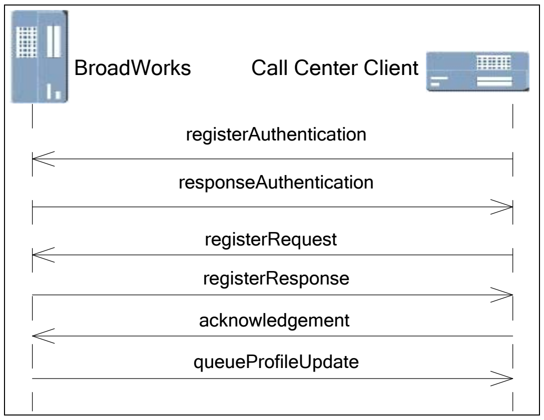

Figure 12 Call Center Registration – Empty Queue

9.1.2 Queue is Not Empty

In the example in *Figure 13*, the Call Center queue already contains calls at the time the client registers. As a result, an initial queueUpdate is sent.

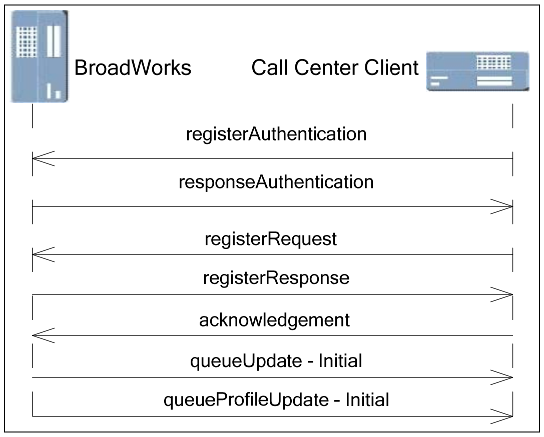

Figure 13 Call Center Registration – Queue Not Empty

9.2 Queue Management

9.2.1 Queue Updates

Figure 14 shows the messaging between the BroadWorks Application Server and the Call Center client when a Call Center receives a call that gets queued, is later answered, and then is released.

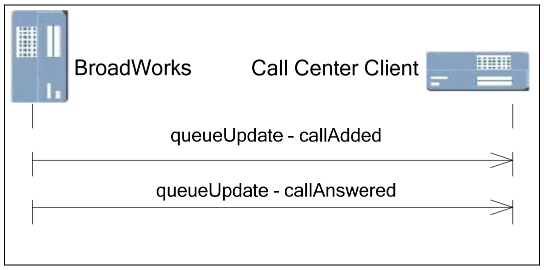

Figure 14 Call Center Queue Updates

In this case, for both the callAdded and callAnswered update reasons, the queueUpdate message only contains information about the particular call being added (or answered). Information about other calls in the queue is not included in the message.

9.2.2 Queue Manipulation

Figure 15 shows an example of Call Center queue manipulation. In this example, an existing queue is reordered by the Call Center client application. It is assumed that queueUpdate messages were previously received as the queue was populated (not shown here).

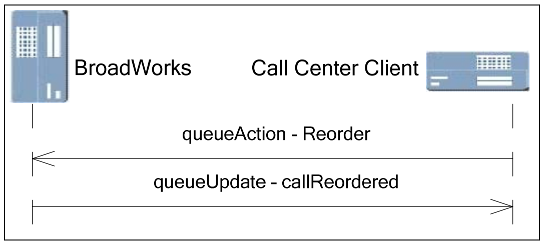

Figure 15 Call Center Queue Manipulation

The queueUpdate – callReordered message contains information about every call in the queue. This is similar to an initial queueUpdate. However, note that a queueUpdate – callAdded only has information about one particular call.

10 CAP-C Document Type Definition

The following is the CAP-C document type definition (DTD):

```
<?xml encoding="UTF-8"?>
<!ELEMENT BroadsoftDocument (command)>
<!ATTLIST   BroadsoftDocument
    protocol    (CAP) #REQUIRED
    version     (17.0) #REQUIRED>
<!ELEMENT command (commandData)>
<!ATTLIST command
    commandType
(registerAuthentication | responseAuthentication | registerRequest |
registerResponse | unRegister | sessionUpdate |  profileUpdate |
callUpdate | callAction | callActionResponse | callControlInfo |
acknowledgement | infoRequest | infoResponse | serverStatusRequest |
externalNotify | monitoringUsersRequest | monitoringUsersResponse |
queueUpdate | queueAction | queueProfileUpdate | queueInfoRequest |
queueInfoResponse | datagram)   #REQUIRED>
<!ELEMENT commandData (from?, to?, user?, queue?, queueActionData?,
capVersionRequest?, capVersionResponse?, nonce?, algorithm?, payload?)>
<!ELEMENT capVersionRequest EMPTY>
<!ELEMENT capVersionResponse (#PCDATA)>
<!ELEMENT nonce (#PCDATA)>
<!ELEMENT algorithm (#PCDATA)>
<!ELEMENT user  (id?, numMonitoredUsers?, callDetailsEnabled?, failure?,
securePassword?, offHook?, numCalls?, conference?, acdState?,
acdStateTime?, initialUpdate?, firstName?, lastName?, phone?, extension?,
locationCode?, enterpriseUser?, email?, mobile?, pager?, department?,
title?, groupName?, serviceProviderName?, supportEmail?, maxCalls?,
epControl?, threeWayCall?, nWayCall?, callTransfer?, cfa?, cpe?, cwt?,
dnd?, lnrd?, oi?, ro?, li?, simring?, call*, applicationId, action?,
message?, voiceMessaging?, voiceMessagingGroup?,
thirdPartyVoiceMessaging?, thirdPartyVMGroup?, hasVMMessages?,
hasThirdPartyVMMessages?, thirdPartyMWI?, clearVMMWI?, callLogs?,
monitoring?, monUser*, monResult?, failedUser*, monitoredUserUid?,
monitoredUserId?, callCenter*, countryCode?, nationalPrefix?,
nightService?, holidayService?, forcedForwarding?, supervisorId?,
hotelingHost?, unavailabilityCode?, wrapUpCallCenterCallId?)>
<!ATTLIST user
userType  (CallClient | AttendantConsole)  #REQUIRED
userUid    CDATA   #IMPLIED
id  CDATA   #IMPLIED
userLogoutReason
(ClientLogout|ForceLogoutU|ForceLogoutR|ForceLogoutD|ForceLogoutC|ForceLo
goutL) #IMPLIED>
<!ELEMENT failure EMPTY>
<!ATTLIST failure
failureCause  (UnknownID | IncorrectPassword | UnsupportedVersion |
UnauthorizedAttendantConsole | UnauthorizedCommPilotCallManager |
UnauthorizedClientCallControl | UnauthorizedPhoneStatusMonitoring |
UnauthorizedClientLicense | NetworkServerConnectivityFailure |
WASConnectivityError | WASProcessingError | ExtAuthHostNotInACLError)
#REQUIRED>
<!ELEMENT conference (callIn, appearance, silentlyMonitored?)>
<!ATTLIST conference
         conferenceState (Active | Released | Held) #REQUIRED>
<!ELEMENT callIn EMPTY>
<!ATTLIST callIn
         callInId       CDATA #REQUIRED>
<!ELEMENT silentlyMonitored (#PCDATA)>
```

```
<!ELEMENT acdState (#PCDATA)>
<!ELEMENT acdStateTime (#PCDATA)>
<!ELEMENT call  (appearance?, personality?, state?, releaseCause?,
remoteCountryCode?, remoteNumber?, remoteName?, remoteTelUri?,
localAltType?, linePort?, redirectFromCountryCode?, redirectFromNumber?,
redirectFromName?, redirectFromReason?, control?, callType?,
callStartTime?, callAnswerTime?, callCenterUserId?, recallType?,
callCenter?, holdReminder?) >
<!ATTLIST call
            callId            CDATA #REQUIRED
            extTrackingId     CDATA #REQUIRED>
<!ELEMENT action (actionParam*)>
<!ATTLIST action
          actionType
( Dial | Redial | Hold | Release | Answer | Xfer | XferConsult | XferVM |
ConfStart | ConfHold | ConfAnswer | ConfRelease | ConfAdd | setDial |
XferCC | CallPark | GroupCallPark | CallParkRetrieve | MakeCallAsCC |
MakePersonalCall | MakeAgentEscalationCall | MakeAgentEmergencyCall |
SilentMonitorCall | ConferenceUnmute | TagCallWithDispositionCode |
CustOriginatedTraceCall) #REQUIRED
          responseRequested (True|False) #IMPLIED "False"
          id CDATA #IMPLIED>
<!ELEMENT actionParam (#PCDATA)>
<!ATTLIST actionParam
           actionParamName      (Number | CallId | UserId | DNISNumber |
DispositionCode)  #REQUIRED>
<!ELEMENT control (#PCDATA)>
<!ATTLIST control
           controlType   (Answer | Hold | Retrieve)   #REQUIRED>
<!ELEMENT message EMPTY>
<!ATTLIST message
          messageName    (registerResponse)    #REQUIRED>
<!ELEMENT monitoring EMPTY>
<!ATTLIST monitoring
            monType  (Add | Delete | Replace)  #REQUIRED>
<!ELEMENT monUser (#PCDATA)>
<!ELEMENT monResult EMPTY>
<!ATTLIST monResult
            success  (True | False)  #REQUIRED>
<!ELEMENT failedUser EMPTY>
<!ATTLIST failedUser
            failureReason (UnknownUser | UnauthorizedUser |
                           PrivateUser)  #REQUIRED
            id            CDATA #REQUIRED>
<!ELEMENT id (#PCDATA)>
<!ELEMENT securePassword (#PCDATA)>
<!ELEMENT appearance (#PCDATA)>
<!ELEMENT personality (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT releaseCause (#PCDATA)>
<!ELEMENT remoteCountryCode (#PCDATA)>
<!ELEMENT remoteNumber (#PCDATA)>
<!ELEMENT remoteName (#PCDATA)>
<!ELEMENT remoteTelUri (#PCDATA)>
<!ELEMENT localAltType (#PCDATA)>
<!ELEMENT linePort (#PCDATA)>
<!ELEMENT numMonitoredUsers (#PCDATA)>
<!ELEMENT callDetailsEnabled (#PCDATA)>
<!ELEMENT offHook (#PCDATA)>
<!ELEMENT numCalls (#PCDATA)>
<!ELEMENT monitoredUserUid (#PCDATA)>
<!ELEMENT monitoredUserId (#PCDATA)>
```

```
<!ELEMENT callCenter (joinCallCenter?, DNISName?, DNISNumber?,
callWaitTime?, longestWaitTime?, numCallsInQueue?)>
<!ATTLIST callCenter
            id  CDATA #REQUIRED>
<!ELEMENT joinCallCenter (#PCDATA)>
<!ELEMENT DNISName (#PCDATA)>
<!ELEMENT DNISNumber (#PCDATA)>
<!ELEMENT addTimeInPriorityBucket (#PCDATA)>
<!ELEMENT preservedWaitTimeInPriorityBucket (#PCDATA)>
<!ELEMENT callWaitTime (#PCDATA)>
<!ELEMENT longestWaitTime (#PCDATA)>
<!ELEMENT numCallsInQueue (#PCDATA)>
<!ELEMENT initialUpdate EMPTY>
<!ELEMENT firstName (#PCDATA)>
<!ELEMENT lastName (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT extension (#PCDATA)>
<!ELEMENT locationCode (#PCDATA)>
<!ELEMENT countryCode (#PCDATA)>
<!ELEMENT nationalPrefix (#PCDATA)>
<!ELEMENT enterpriseUser (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT mobile (#PCDATA)>
<!ELEMENT pager (#PCDATA)>
<!ELEMENT department (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT groupName (#PCDATA)>
<!ELEMENT serviceProviderName (#PCDATA)>
<!ELEMENT supportEmail (#PCDATA)>
<!ELEMENT maxCalls (#PCDATA)>
<!ELEMENT epControl (#PCDATA)>
<!ELEMENT threeWayCall (#PCDATA)>
<!ELEMENT nWayCall (#PCDATA)>
<!ELEMENT callTransfer (#PCDATA)>
<!ELEMENT cfa (#PCDATA)>
<!ATTLIST cfa
            cfaDestination  CDATA #IMPLIED>
<!ELEMENT cfaDestination (#PCDATA)>
<!ELEMENT cpe (#PCDATA)>
<!ELEMENT cwt (#PCDATA)>
<!ELEMENT dnd (#PCDATA)>
<!ELEMENT lnrd (#PCDATA)>
<!ELEMENT oi (#PCDATA)>
<!ELEMENT ro (#PCDATA)>
<!ELEMENT li (#PCDATA)>
<!ELEMENT simring (#PCDATA)>
<!ELEMENT applicationId (#PCDATA)>
<!ELEMENT voiceMessaging (#PCDATA)>
<!ELEMENT voiceMessagingGroup (#PCDATA)>
<!ELEMENT thirdPartyVoiceMessaging (#PCDATA)>
<!ELEMENT thirdPartyVMGroup (#PCDATA)>
<!ELEMENT hasVMMessages (#PCDATA)>
<!ELEMENT hasThirdPartyVMMessages (#PCDATA)>
<!ELEMENT callLogs (#PCDATA)>
<!ELEMENT redirectFromCountryCode (#PCDATA)>
<!ELEMENT redirectFromNumber (#PCDATA)>
<!ELEMENT redirectFromName (#PCDATA)>
<!ELEMENT redirectFromReason (#PCDATA)>
<!ELEMENT callType (#PCDATA)>
<!ELEMENT callStartTime (#PCDATA)>
<!ELEMENT callAnswerTime (#PCDATA)>
<!ELEMENT callCenterUserId (#PCDATA)>
```

```
<!ELEMENT holdReminder (#PCDATA)>
<!ELEMENT recallType (#PCDATA)>
<!ELEMENT thirdPartyMWI (#PCDATA)>
<!ELEMENT clearVMMWI (#PCDATA)>
<!ELEMENT nightService(#PCDATA)>
<!ELEMENT holidayService(#PCDATA)>
<!ELEMENT forcedForwarding(#PCDATA)>
<!ELEMENT supervisorId(#PCDATA)>
<!ELEMENT hotelingHost(#PCDATA)>
<!ELEMENT unavailabilityCode(#PCDATA)>
<!ELEMENT wrapUpCallCenterCallId(#PCDATA)>
<!ELEMENT queue (queueCall+)>
<!ATTLIST queue
          updateReason  (Initial | CallAdded | CallAnswered |
CallTransferred | CallAbandoned | CallReordered | CallUpdated | Refresh |
QueueTimeout | QueueOverflow | CallPromoted)  #REQUIRED>
<!ELEMENT queueCall  (remoteCountryCode?, remoteNumber?, remoteName?,
timeAdded?, mandatoryEntrance?, bouncedCall?, reorderedCall?,
preservedWaitTime?, position?, destination?, priority?) >
<!ATTLIST queueCall
            callId            CDATA #REQUIRED>
<!ELEMENT queueActionData (queueCall?)>
<!ATTLIST queueActionData
         actionType    (Remove | Reorder | Refresh | Promote ) #REQUIRED>
<!ELEMENT timeAdded (#PCDATA)>
<!ELEMENT mandatoryEntrance (#PCDATA)>
<!ELEMENT bouncedCall (#PCDATA)>
<!ELEMENT reorderedCall (#PCDATA)>
<!ELEMENT preservedWaitTime (#PCDATA)>
<!ELEMENT position (#PCDATA)>
<!ELEMENT destination (#PCDATA)>
<!ELEMENT priority (#PCDATA)>
<!ELEMENT from (user)>
<!ELEMENT to (user)>
<!ELEMENT payload (#PCDATA)>
<!ATTLIST payload
          payloadType   CDATA   #IMPLIED>
```

11 CCC2 Document Type Definition

The following is the CCC2 document type definition (DTD):

```
<?xml encoding="UTF-8"?>
<!ELEMENT BroadsoftDocument (command)>
<!ATTLIST   BroadsoftDocument
    protocol    (CAP) #REQUIRED
    version     (17.0) #REQUIRED>
<!ELEMENT command (commandData)>
<!ATTLIST command
    commandType
(sessionUpdate | profileUpdate | callUpdate | callAction |
callActionResponse | callControlInfo | infoRequest | infoResponse |
serverStatusRequest | externalNotify | queueUpdate | queueAction |
queueProfileUpdate | queueInfoRequest | queueInfoResponse | datagram)
#REQUIRED>
<!ELEMENT commandData (from?, to?, user?, queue?, queueActionData?,
capVersionRequest?, capVersionResponse?, payload?)>
<!ELEMENT capVersionRequest EMPTY>
<!ELEMENT capVersionResponse (#PCDATA)>
<!ELEMENT user  (id?,offHook?, numCalls?, conference?, acdState?,
acdStateTime?, maxCalls?, epControl?, threeWayCall?, nWayCall?,
callTransfer?, cfa?, cpe?, cwt?, dnd?, lnrd?, ro?, simring?, call*,
action?, hasVMMessages?, hasThirdPartyVMMessages?, thirdPartyMWI?,
clearVMMWI?, callCenter*, applicationId?, nightService?, holidayService?,
forcedForwarding?, hotelingHost?, unavailabilityCode?,
wrapUpCallCenterCallId?, responseRequested?)>
<!ATTLIST user
userType  (CallClient)  #REQUIRED
userUid    CDATA   #IMPLIED
id  CDATA   #IMPLIED>
<!ELEMENT conference (callIn, appearance, silentlyMonitored?)>
<!ATTLIST conference
         conferenceState (Active | Released | Held) #REQUIRED>
<!ELEMENT callIn EMPTY>
<!ATTLIST callIn
         callInId       CDATA #REQUIRED>
<!ELEMENT silentlyMonitored (#PCDATA)>
<!ELEMENT acdState (#PCDATA)>
<!ELEMENT acdStateTime (#PCDATA)>
<!ELEMENT call  (appearance?, personality?, state?, releaseCause?,
remoteCountryCode?, remoteNumber?, remoteName?, remoteTelUri?,
localAltType?, linePort?, redirectFromCountryCode?, redirectFromNumber?,
redirectFromName?, redirectFromReason?, control?, callType?,
callStartTime?, callAnswerTime?, callCenterUserId?, localNumber?,
redirectToNum?, redirectToReason?, redirectFromCounter?, origCalledNum?,
origRedirReason?, recallType?, callCenter?, holdReminder?) >
<!ATTLIST call
           callId           CDATA #REQUIRED
           extTrackingId     CDATA #REQUIRED>
<!ELEMENT action (actionParam*)>
<!ATTLIST action
         actionType
( Dial | Redial | Hold | Release | Answer | Xfer | XferConsult | XferVM |
ConfStart | ConfHold | ConfAnswer | ConfRelease | ConfAdd | XferCC |
CallPark | GroupCallPark | CallParkRetrieve | MakeCallAsCC |
MakePersonalCall | MakeAgentEscalationCall | MakeAgentEmergencyCall |
SilentMonitorCall | ConferenceUnmute | TagCallWithDispositionCode |
CustOriginatedTraceCall ) #REQUIRED
         responseRequested (True|False) #IMPLIED "False"
         id CDATA #IMPLIED>
<!ELEMENT actionParam (#PCDATA)>
<!ATTLIST actionParam
          actionParamName      (Number | CallId | UserId | DNISNumber |
DispositionCode)  #REQUIRED>
<!ELEMENT control (#PCDATA)>
<!ATTLIST control
          controlType   (Answer | Hold | Retrieve)   #REQUIRED>
<!ELEMENT id (#PCDATA)>
<!ELEMENT appearance (#PCDATA)>
<!ELEMENT personality (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT releaseCause (#PCDATA)>
<!ELEMENT remoteCountryCode (#PCDATA)>
<!ELEMENT remoteNumber (#PCDATA)>
<!ELEMENT remoteName (#PCDATA)>
<!ELEMENT remoteTelUri (#PCDATA)>
<!ELEMENT localAltType (#PCDATA)>
<!ELEMENT linePort (#PCDATA)>
<!ELEMENT offHook (#PCDATA)>
<!ELEMENT numCalls (#PCDATA)>
<!ELEMENT callCenter (joinCallCenter?, DNISName?, DNISNumber?,
callWaitTime?, longestWaitTime?, numCallsInQueue?)>
<!ATTLIST callCenter
            id  CDATA #REQUIRED>
<!ELEMENT joinCallCenter (#PCDATA)>
<!ELEMENT DNISName (#PCDATA)>
<!ELEMENT DNISNumber (#PCDATA)>
<!ELEMENT addTimeInPriorityBucket (#PCDATA)>
<!ELEMENT preservedWaitTimeInPriorityBucket (#PCDATA)>
<!ELEMENT callWaitTime (#PCDATA)>
<!ELEMENT longestWaitTime (#PCDATA)>
<!ELEMENT numCallsInQueue (#PCDATA)>
<!ELEMENT maxCalls (#PCDATA)>
<!ELEMENT epControl (#PCDATA)>
<!ELEMENT threeWayCall (#PCDATA)>
<!ELEMENT nWayCall (#PCDATA)>
<!ELEMENT callTransfer (#PCDATA)>
<!ELEMENT cfa (#PCDATA)>
<!ELEMENT cpe (#PCDATA)>
<!ELEMENT cwt (#PCDATA)>
<!ELEMENT dnd (#PCDATA)>
<!ELEMENT lnrd (#PCDATA)>
<!ELEMENT ro (#PCDATA)>
<!ELEMENT simring (#PCDATA)>
<!ELEMENT applicationId (#PCDATA)>
<!ELEMENT hasVMMessages (#PCDATA)>
<!ELEMENT hasThirdPartyVMMessages (#PCDATA)>
<!ELEMENT redirectFromCountryCode (#PCDATA)>
<!ELEMENT redirectFromNumber (#PCDATA)>
<!ELEMENT redirectFromName (#PCDATA)>
<!ELEMENT redirectFromReason (#PCDATA)>
<!ELEMENT callType (#PCDATA)>
<!ELEMENT callStartTime (#PCDATA)>
<!ELEMENT callAnswerTime (#PCDATA)>
<!ELEMENT callCenterUserId (#PCDATA)>
<!ELEMENT holdReminder (#PCDATA)>
<!ELEMENT recallType (#PCDATA)>
<!ELEMENT localNumber (#PCDATA)>
<!ELEMENT redirectToNum (#PCDATA)>
<!ELEMENT redirectToReason (#PCDATA)>
<!ELEMENT redirectFromCounter (#PCDATA)>
```

```
<!ELEMENT origCalledNum (#PCDATA)>
<!ELEMENT origRedirReason (#PCDATA)>
<!ELEMENT thirdPartyMWI (#PCDATA)>
<!ELEMENT clearVMMWI (#PCDATA)>
<!ELEMENT nightService(#PCDATA)>
<!ELEMENT holidayService(#PCDATA)>
<!ELEMENT forcedForwarding(#PCDATA)>
<!ELEMENT hotelingHost(#PCDATA)>
<!ELEMENT unavailabilityCode(#PCDATA)>
<!ELEMENT wrapUpCallCenterCallId(#PCDATA)>
<!ELEMENT queue (queueCall+)>
<!ATTLIST queue
          updateReason  (Initial | CallAdded | CallAnswered |
CallTransferred | CallAbandoned | CallReordered | CallUpdated | Refresh |
QueueTimeout | QueueOverflow | CallPromoted)  #REQUIRED>
<!ELEMENT queueCall  (remoteCountryCode?, remoteNumber?, remoteName?,
timeAdded?, mandatoryEntrance?, bouncedCall?, reorderedCall?,
preservedWaitTime?, position?, destination?, priority?) >
<!ATTLIST queueCall
            callId            CDATA #REQUIRED>
<!ELEMENT queueActionData (queueCall?)>
<!ATTLIST queueActionData
         actionType    (Remove | Reorder | Refresh | Promote ) #REQUIRED>
<!ELEMENT timeAdded (#PCDATA)>
<!ELEMENT mandatoryEntrance (#PCDATA)>
<!ELEMENT bouncedCall (#PCDATA)>
<!ELEMENT reorderedCall (#PCDATA)>
<!ELEMENT preservedWaitTime (#PCDATA)>
<!ELEMENT position (#PCDATA)>
<!ELEMENT destination (#PCDATA)>
<!ELEMENT priority (#PCDATA)>
<!ELEMENT from (user)>
<!ELEMENT to (user)>
<!ELEMENT payload (#PCDATA)>
<!ATTLIST payload
          payloadType   CDATA   #IMPLIED>
```

12 Limitations

12.1 The "Answer" callAction

In most scenarios, it is not possible to control endpoint behavior to accept an incoming call via the callAction message "Answer" action. This message is, however, applicable in the following two scenarios:

- When an incoming call is a BroadWorks-controlled call and there is a call waiting appearance on an analog endpoint
- When a callUpdate for a terminating call in the *Alerting* state is followed by a callControlInfo CAP message allowing the call client application to enable "Answer" functionality on a call.

In all other "answer a call" scenarios (first appearance on any endpoint, Session Initiation Protocol [SIP] or Media Gateway Control Protocol [MGCP], or call waiting appearances on SIP endpoints), if the endpoint is ringing, a user must pick up the phone to answer the call. Issuing a callAction "Answer" CAP message to BroadWorks in these scenarios does not answer the phone and the phone does not stop ringing.

For example, the following interaction does not result in any behavior change unless the *Alerting* being sent is for a call waiting appearance to an analog endpoint.

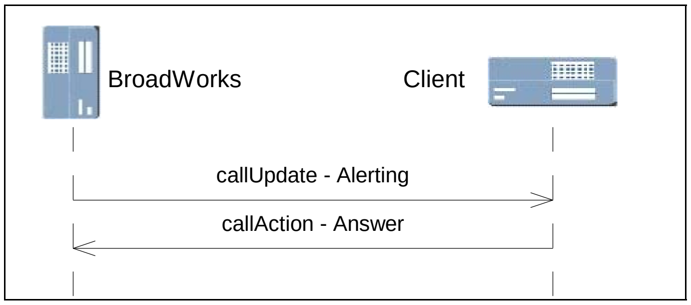

Figure 16 Analog Endpoint callAction

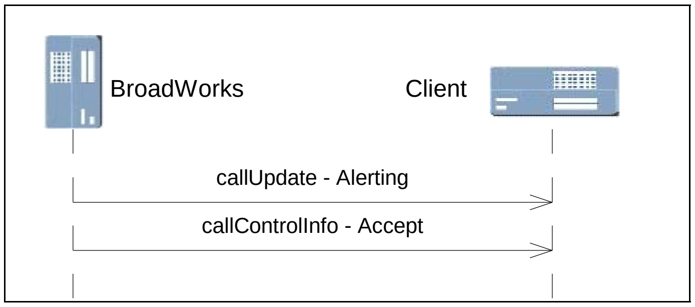

Figure 17 callControlInfo Informing Client to Allow "Accept" (a Call)

12.2 Third-Party Attendant Console

To create a third-party application with Attendant Console functionality (user presence information), the Attendant Console and Call Control services must be assigned to the user.

Appendix A: Login Password Algorithm

This section briefly explains the algorithm the client application should use when sending the user login password as part of the CAP-C registerRequest message.

12.3 Secured Password Calculation

1) Calculate the message digest of the user's plain password using the secure hash algorithm (SHA).
2) For every 4 bits in the 160-bit digest, starting from the first bit, convert it into a character in ASCII Hex format (0 – 9, a – f). The result is a 40-character string ***S1***, for example, f7a9e24777ec23212c54d7a350bc5bea5477fdbb.
3) Use the string ***S1*** to construct a new string ***S2***: ***S2*** = ***nonce*** + "***:***" + ***S1***, where ***nonce*** is the value in the responseAuthentication message from BroadWorks.
4) Calculate the message digest of ***S2*** using the MD5 algorithm.
5) For every 4 bits in the 128-bit digest from step 2, starting from the first bit, convert it into a character in ASCII Hex format (0 – 9, a – f). The result is a 32-character string, for example, dc70779bf8461b5a1e6aea58f636d5c0.
6) Use this string as the securePassword to log in.

References

[1] BroadSoft Inc. 2003. *Client Application Protocol (CAP) Overview Bulletin* 01-2003. Available from BroadSoft by request.

[2] BroadSoft Inc. 2007. *User Managed Privacy Service Feature Description, Release 14.sp2.* Available from BroadSoft Xchange at www.broadsoft.com/xchange.

[3] BroadSoft Inc. 2007. *Call Center Architecture Enhancements Feature Description,* Release 14.sp3. Available from BroadSoft Xchange at www.broadsoft.com/xchange.

[4] BroadSoft Inc. 2009. *Client Call Control 2 Message Flows*. Release 16.0. Available from BroadSoft Xchange at www.broadsoft.com/xchange.

Index