# Cisco Finesse Web Services Developer Guide for Release 8.5(3)

API Documentation—December 2011

## Contents:

# 1. Introduction

[contents]

This document is the official reference for the Cisco Finesse Application Programming Interface. The Finesse APIs support the Finesse agent desktop, enabling it to provide agent desktop functionality, which includes call control, state changes, configuration information, and so on.

There are two categories of APIs for Finesse: desktop APIs and configuration APIs.

The Finesse APIs support the following capabilities:

- User Sign In/Sign Out
- Agent States
- Configurations
- Subscriptions
- Call Control
- Reason Codes
- Wrap-up Reasons

This guide explains each API and the notification messages returned by the APIs, and begins with a section that assists developers in running and validating the APIs in a lab environment.

## 1.1. Communication with the Cisco Finesse Web Service

[contents]

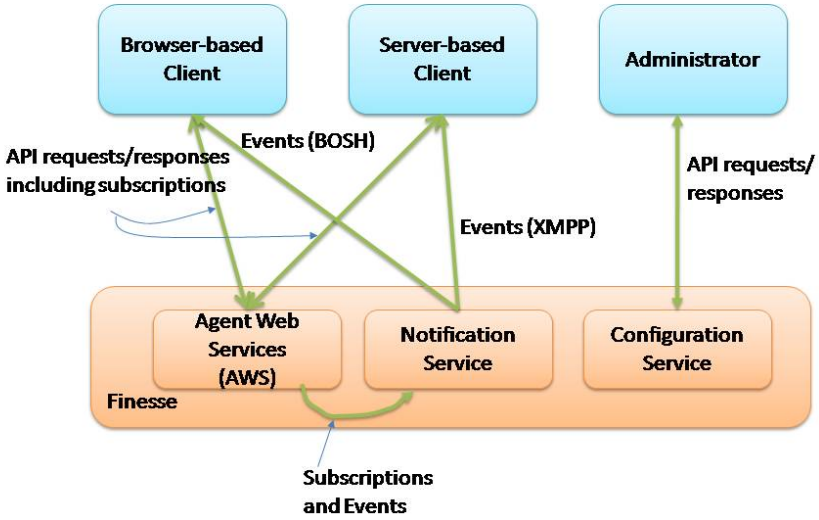


Figure 1: Finesse API and Event Flow

**Note:** Finesse Release 8.5(3) supports receiving updates through BOSH only.



## 1.1.1. Making Requests

[contents]

Cisco Finesse Desktop operations can be performed using one of the many available REST-like HTTP requests described in this guide. Clients should make all HTTP requests to port 80.

Most, but not all, Finesse Desktop APIs conform to the following format:

```
http://<host>:<port>/finesse/api/<object>
```

Operations on specific objects are performed using the ID of the object in the REST URL. For example, the URL to view a single object would be:

```
http://<host>:<port>/finesse/api/<object>/<objectID>
```

All Finesse APIs use HTTP BASIC authentication, which requires the credentials to be sent in the "Authorization" header. The credentials contain the username and password, separated by a single colon (:), within a BASE64-encoded string. For example, the Authorization header would contain the following string:

```
"Basic YWdlbnRiYXJ0b3dza2k6Y2FybWljaGFlbA=="
```

where "YWdlbnRiYXJ0b3dza2k6Y2FybWljaGFlbA==" is the Base64-encoded string of "agentbartowski:carmichael" (agentbartowski being the

username and carmichael being the password).

Finesse configuration APIs require the application user ID and password, which are established during installation, for authentication purposes.

Finesse APIs use the following HTTP methods to make requests:

- GET—Retrieve a single object or list of objects (for example, a single user or list of users).
- PUT—Replace a value in an object (for example, to change the state of a user from NOT_READY to READY).
- POST—Create a new entry in a collection (for example, to create a new reason code or wrap-up reason).
- DELETE—Remove an entry from a collection (for example, to delete a reason code or wrap-up reason).

Finesse uses the standard HTTP status codes (for example, 200, 400, 500, and so on) in the response. These status codes indicate overall success or failure of the request.

If an API operation fails, a detailed error is returned in the HTTP response message body. The error, in XML format, appears as follows:

```
<ApiErrors>
    <ApiError>
        <ErrorType>type</ErrorType>
        <ErrorMessage>message</ErrorMessage>
        <ErrorData>data</ErrorData>
  </ApiError>
</ApiErrors>
```

All requests must pass through a servlet filter (ServerStateFilter). Each request queries the Dependency Manager for the state of the system. The Dependency Manager collects the states of its dependencies (such as the state of the Cisco Finesse Notification Service) and reports these states to external entities.

If the Cisco Finesse Notification Service is down, Finesse is out of service. Finesse rejects any API requests and returns an HTTP 503 error. The error appears as follows:

```
<ApiErrors>
  <ApiError>
    <ErrorType>Service Unavailable</ErrorType>
    <ErrorData></ErrorData>
    <ErrorMessage>SERVER_OUT_OF_SERVICE</ErrorMessage>
  </ApiError>
</ApiErrors>
```

### 1.1.2. Receiving Events

[contents]

Real-time events (such as call events, state events, and so on) are sent by the Cisco Finesse Notification Service, using the XEP-0060 Publish-Subscribe extension of the XMPP (Extensible Messaging and Presence Protocol) protocol. Applications that need to communicate with the Notification Service must use XMPP over the BOSH (Bidirectional-streams Over Synchronous HTTP) transport.

BOSH is an open technology for real-time communication and is useful for maintaining a long-lived, bidirectional TCP connection between two entities (such as client and server). See documentation at the XMPP Standards Foundation for details about both XMPP and BOSH (XEP-0124).

Client applications can communicate with the Cisco Finesse Notification Service through BOSH, using the binding URI http://<host>:7071/http-bind. Developers can create their own BOSH library or use any that are available publicly, as documented on the Cisco Developer Network.

After creating the connection, applications can receive notification events of feeds to which they are subscribed. Users are currently subscribed to a few feeds by default (subject to change). Other feeds require an explicit subscription (see section 7.1.2 Subscription Management).

# 2. Working with the Cisco Finesse Web Services APIs ("Hello World")

[contents]

This section explains how to work with the Cisco Finesse Web Services APIs to validate your lab development environment.

## 2.1. Environment and Tools

[contents]

The topics in this section are for use as a learning exercise and are not meant to be used in real deployments.

To complete these exercises, you need the following:

- A user who is configured as an agent in Unified CCE (with an agent ID, password, and extension).
  Make the agent a member of a team and of a queue. (A queue is a Unified CCE Skill Group.)
- Three phones that are configured in Cisco Unified Communications Manager: one for the agent, one for the caller, and one to use for conferencing and transfer APIs. These can be Cisco IP "hard phones" or Cisco IP Communicator softphones.
- Two tools: Poster and Pidgin.

**Note:** Poster and Pidgin are meant to aid in development; however, they are not officially supported.

### 2.1.1. Poster

[contents]

Poster is a utility that allows you to send HTTP requests to the Finesse Web Service by entering the URI for an API and checking the response. All APIs are accessible by URI and follow a request/response paradigm. There is always a single response for any request.

You can download Poster from https://addons.mozilla.org/en-US/firefox/addon/2691/.

Note that although the Finesse Desktop is not supported on Firefox (it is supported on Internet Explorer 8 only), the Poster tool is a Firefox plug-in.

After Poster is added to Firefox, press **Ctrl-Alt-P** to launch it.

To test an API in Poster, follow these steps:

1. Copy and paste the URI for the API request from this Developer Guide into a text editor. For example, to enter the URI for signing in, copy the URI from the Sign In API. Examine the pasted code for case sensitivity and format and remove any carriage returns.
2. Update the URI with the IP address and port of your Cisco Finesse Web Services server.
3. Add any mandatory parameters for the request.
4. Enter the username and password for the agent you set up for these exercises.
5. For Content Type, enter **application/xml**.
6. Click the appropriate action (GET, PUT, or POST).

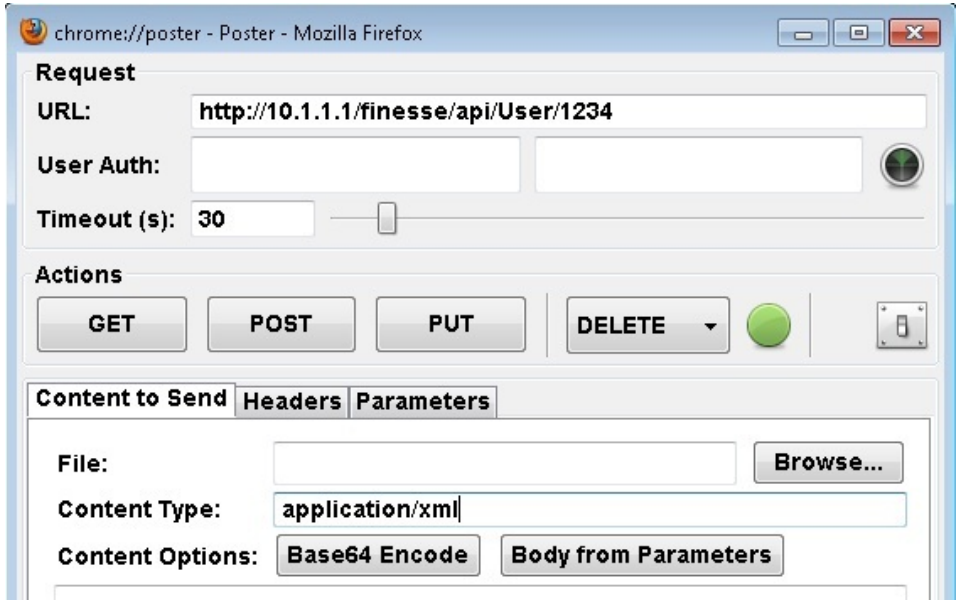


Figure **2**: Poster Request

The object response appears in the Poster window.


Figure **3**: Poster Response

## 2.1.2. Pidgin

[contents]

Pidgin is a multiplatform instant messaging client that supports many common messaging protocols, including XMPP. You can use Pidgin to establish an XMPP connection and view XMPP messages published by the Cisco Finesse Notification Service.

Notifications that result from API requests made in Poster appear in the XMPP Console tool of the Pidgin application. For example, if you use Poster to subscribe to call events and then use your Cisco IP Phone to place a call to an agent, you can see the call delivered event in the Pidgin XMPP Console window.

**Note:** Make sure that you use the same username and resource values in both Poster and Pidgin.

You can download Pidgin from http://www.pidgin.im/download/.

Perform the following steps to configure XMPP:

1. In Pidgin, go to **Tools > Plugins** to open the Plugins dialog box.
2. Check the **XMPP Console** and **XMPP Service Discovery** check boxes.

Perform the following steps to configure Pidgin:

1. Add an account for your XMPP server. Go to **Pidgin > Accounts > Manage Accounts > Add Account**. The Add Account dialog box opens.
2. For Protocol, select **XMPP**.
3. For Username, enter the username for the agent that you added.
4. For Domain, enter the fully-qualified domain name of the Cisco Finesse server.
5. For Resource, enter any text.
6. For Password, enter the password of the agent.

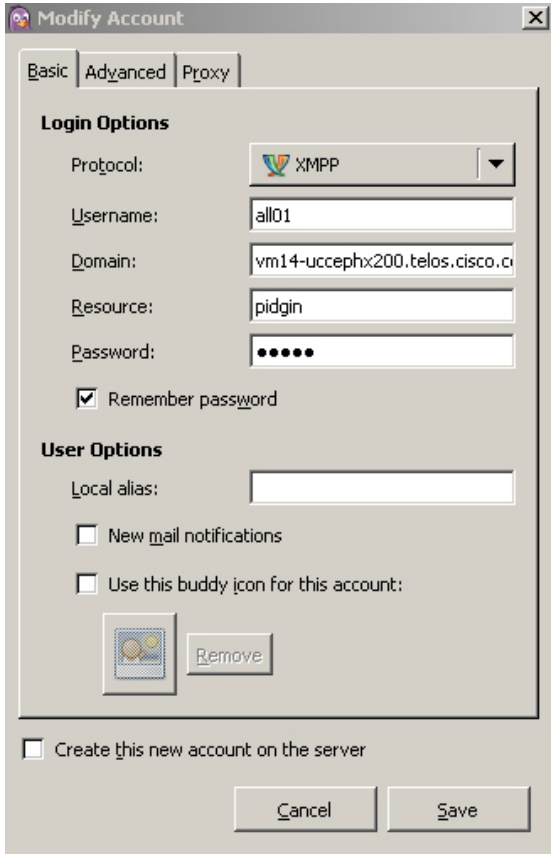


Figure **4**: The Pidgin Interface

7. Click **Save**.
8. Click the **Advanced** tab.
9. Check the **Allow plaintext auth over unencrypted streams** check box.
10. For Connect Server, enter the IP address of the Finesse server.
11. If the Connection Security drop-down menu is present, choose **Use encryption if available**.
12. Click **Save**.

**Note:** Connect port and File transfer proxies should be filled in automatically (5222 should appear in the Connect port field).

The XMPP logo next to the agent's name becomes active (is no longer dimmed). To see event messages in Pidgin, open the XMPP Console.


Figure **5**: Open XMPP Console in Pidgin

**Note:** The agent must be signed in to Finesse through Poster or the browser interface to be signed in to the XMPP account on Pidgin.

The XMPP Console window immediately begins updating every few seconds with iq type statements. The window does not display an event message until an event occurs. If the XMPP Console window fills with iq type notifications and becomes difficult to navigate, close and reopen it to refresh with a clean window.

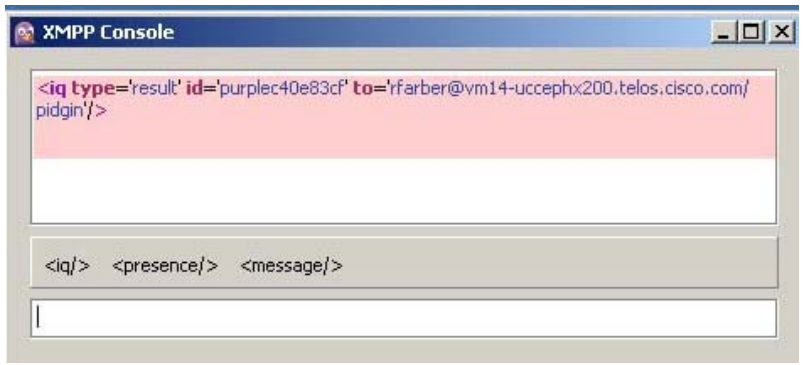


Figure **6**: The XMPP Console Window

## 2.2. Using the Cisco Finesse APIs

[contents]

APIs that control actions on the Finesse Desktop and call control make use of two objects:

- User object—The User object represents agent and supervisor data and actions. This object is used to get information about a single user or list of users, to sign in or out of the Finesse Desktop, and change agent state.
- Dialog object—The Dialog object represents a dialog with participants. For media type "voice", this object represents a call. A participant can represent an internal user (such as an agent) or an external user (for example, a customer). A participant can belong to only one dialog but a user can be a participant in several dialogs. The Dialog object is used for call control and call data.

The following sections provide instructions and examples for using the APIs with Poster and Pidgin.

## 2.2.1. Signing In to Finesse

[contents]

Use the User—Sign In to Finesse API to sign the agent in.

This example uses the following information:

- Finesse server IP address: 172.16.204.26
- Agent name: John Smith
- Agent ID: 1234
- Agent password: jsmith
- Agent extension: 1001

**1.** Access Poster (Ctrl + Alt +P from the Mozilla Firefox browser) and enter the following string in the URL field:

```
http://172.16.204.26/finesse/api/User/1234
```

**2.** Enter the agent's ID (1234) and extension (1001) in the two User Auth fields directly under the URL field.

**3.**In the Content Type field, enter application/XML.

**4.**In the area under Content Options, enter the following:

```
<User>
<state>LOGIN</state>
<extension>1001</extension>
</User>
```

**5.**Click **PUT**.

Poster returns the following response:

```
PUT on http://172.16.204.26/finesse/api/User/1234
Status 202: Accepted
```

Finesse returns a user notification, which you can view in Pidgin:

```
<Update>
  <data>
    <user>
      <dialogs>/finesse/api/User/93964892/Dialogs</dialogs>
      <extension>1001</extension>
      <firstName>John</firstName>
      <lastName>Smith</lastName>
      <loginId>1234</loginId>
      <loginName>jsmith</loginName>
      <roles>
        <role>Agent</role>
      </roles>
      <state>NOT_READY</state>
      <teamId>1</teamId>
      <teamName>Default</teamName>
      <uri>/finesse/api/User/1234</uri>
    </user>
  </data>
  <event>PUT</event>
  <requestId></requestId>
  <source>/finesse/api/User/1234</source>
</Update>
```

The agent is now signed in and in NOT_READY state.

### 2.2.2. Changing the Agent State

[contents]

Use the User—Change Agent State API to change the agent state to Ready.

This example uses the same agent information as the previous example.

**1.** In Poster, enter the following string in the URL field:

```
http://172.16.204.26/finesse/api/User/1234
```

**2.** Enter the agent's ID (1234) and extension (1001) in the two User Auth fields directly under the URL field.

**3.**In the Content Type field, enter application/XML.

**4.**In the area under Content Options, enter the following:

```
<User>
<state>READY</state>
</User>
```

**5.** Click **PUT**.

Poster returns the following response:

```
PUT on http://172.16.204.26/finesse/api/User/1234
Status 202: Accepted
```

Finesse returns the following user notification:

```
<Update>
  <data>
    <user>
      <dialogs>/finesse/api/User/1234/Dialogs</dialogs>
      <extension>1001</extension>
      <firstName>John</firstName>
      <lastName>Smith</lastName>
      <loginId>1234</loginId>
      <loginName>jsmith</loginName>
      <roles>
        <role>Agent</role>
      </roles>
      <state>READY</state>
      <teamId>1</teamId>
      <teamName>Default</teamName>
      <uri>/finesse/api/User/93964892</uri>
    </user>
  </data>
  <event>PUT</event>
  <requestId></requestId>
  <source>/finesse/api/User/93964892</source>
</Update>
```

# 3. The Cisco Finesse Desktop APIs

[contents]



Cisco Finesse comprises of a set of web APIs that allow Unified Contact Center Enterprise (Unified CCE) to send and receive information about:

- Current system configuration
- Agents and agent states
- Calls and call states
- Teams

The individual Web APIs are method calls sent as GET or POST requests using a REST-like interface over HTTP. Methods follow a request/response paradigm, and there is always a single response for any request.

All APIs must provide BASIC authentication credentials, as described in Making Requests.

## 3.1. User APIs

[contents]

The User object is a container element that holds agent and supervisor objects. Elements common to the subobjects can be at the user level. A user can have more than one role.



### 3.1.1. User—Get User

[contents]

The Get User API allows a user to get a copy of the user object.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<id> |
| **Example URI:** | `http://host/finesse/api/User/1234` |
| **Security Constraints:** | Role—Agent, Administrator<br>Limitations—Agents can only act on their own User object. Administrators can get any User object. |
| **HTTP Method:** | GET |

| | |
|---|---|
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **HTTP Response:** | 200—Success<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>404—Not Found (for example, the user ID is not known)<br>500—Runtime exception<br>503—Service Unavailable (for example, the notification service is not running) |
| **Successful Response:** | <User><br><uri>/finesse/api/User/1234</uri><br><roles><br><role>Agent</role><br><role>Supervisor</role><br></roles><br><loginId>1234</loginId><br><loginName>csmith</loginName><br><state>NOT_READY</state><br><extension>1001001</extension><br><firstName>Chris</firstName><br><lastName>Smith</lastName><br><teamId>500</teamId><br><teamName>Sales</teamName><br><dialogs>/finesse/api/User/1234/Dialogs</dialogs><br><teams><br><Team><br><id>501</id><br><name>First Line Support</name><br></Team><br><Team><br><id>502</id><br><name>Second Line Support</name><br></Team><br><Team><br><id>503</id><br><name>Third Line Support</name><br></Team><br>... other teams ...<br></teams><br></User> |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>User Not Found</ErrorType><br><ErrorMessage>UNKNOWN_USER</ErrorMessage><br><ErrorData>4023</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure |

| | |
|---|---|
| | • Invalid Authorization User Specified<br>• User Not Found<br>• Internal Server Error<br>• Service Unavailable<br>For descriptions and other possible error codes, see API Error Codes. |

**Response Parameters**

- uri
- roles
- role
- loginId
- loginName
- state
- extension
- firstName
- lastName
- dialogs
- subscription
- Team
- teamId
- teamName
- teams
- id
- name

## 3.1.2. User—Get List of Users

[contents]

The Get List of Users API allows an administrator to get a list of users.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/Users |
| **Example URI:** | `http://host/finesse/api/Users` |
| **Security Constraints:** | Role—Administrator<br>Limitations—Any administrator can use this API. |
| | |

| | |
|---|---|
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **HTTP Response:** | 200—Success<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>500—Internal server error<br>503—Service Unavailable (for example, the notification service is not running) |
| **Successful Response:** | <Users><br><User><br>... Full User Object ...<br></User><br><User><br>... Full User Object ...<br></User><br><User><br>... Full User Object ...<br></User><br><User><br>... Full User Object ...<br></User><br><User><br>... Full User Object ...<br></User><br>... Additional Users...<br></Users> |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Unauthorized</ErrorType><br><ErrorMessage>The user is not authorized to perform this operation</ErrorMessage><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• User Not Found<br>• Internal Server Error<br>• Service Unavailable<br>For descriptions and other possible error codes, see API Error Codes. |

| | |
|---|---|

**Response Parameters**

- uri
- roles
- role
- loginId
- loginName
- state
- extension
- firstName
- lastName
- dialogs
- subscription
- Team
- teamId
- teamName
- teams
- id
- name

## 3.1.3. User—Get List of Dialogs Associated with a User

[contents]

The User—Get List of Dialogs Associated with a User API allows an administrator or agent to obtain a list of dialogs associated with a particular user. An administrator can get a list of dialogs that are associated with any user. Agents can only get a list of their own dialogs.

The structure of a *single* full Dialog object is shown in the following example:

```
<Dialog>
    <uri>/finesse/api/Dialog/12345678</uri>
    <mediaType>Voice</mediaType>
    <state>ACTIVE</state>
    <fromAddress>2002</fromAddress>
    <toAddress>2000</toAddress>
    <mediaProperties>
        <dialedNumber>2000</dialedNumber>
        <callType>AGENT_INSIDE</callType>
        <DNIS>2000</DNIS>
        <wrapUpReason>Another satisfied customer</wrapUpReason>
        <callvariables>
          <CallVariable>
            <name>callVariable1</name>
            <value>Chuck Smith</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable2</name>
            <value>Cisco Systems,Inc<value>
          </CallVariable>
          <CallVariable>
            <name>callVariable3</name>
            <value>chuckSmith@cisco.com<value>
          </CallVariable>
          ... Other CallVariables up to 10 ...
          <CallVariable>
            <name>callVariable10</name>
```

```
          <value>Preferred Customer<value>
        </CallVariable>
        <CallVariable>
          <name>ecc.user</name>
          <value>csmith<value>
        </CallVariable>
        <CallVariable>
          <name>ecc.years[0]</name>
          <value>1985<value>
        </CallVariable>
        <CallVariable>
          <name>ecc.years[1]</name>
          <value>1995<value>
        </CallVariable>
        <CallVariable>
          <name>ecc.years[2]</name>
          <value>2005<value>
        </CallVariable>
      </callvariables>
  </mediaProperties>
  <participants>
    <Participant>
      <mediaAddress>2002</mediaAddress>
      <state>ACTIVE</state>
      <stateCause></stateCause>
      <actions>
        <action>HOLD</action>
        <action>DROP</action>
      </actions>
    </Participant>
    <Participant>
      <mediaAddress>2000</mediaAddress>
      <state>HELD</state>
      <stateCause></stateCause>
      <actions>
        <action>RETRIEVE</action>
        <action>DROP</action>
      </actions>
    </Participant>
  </participants>
</Dialog>
```

The User—Get List of Dialogs API returns a list of such Dialog objects, all of which are associated with a particular user.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<id>/Dialogs |
| **Example URI:** | `http://host/finesse/api/User/1234/Dialogs` |
| **Security Constraints:** | Role—Agent, Administrator<br>Limitations—Administrators can get a list of dialogs associated with any user. Agents can only get a list of their own dialogs. |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |

| | |
|---|---|
| **HTTP Request:** | — |
| **HTTP Response:** | 200—Success<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>500—Internal server error |
| **Successful Response:** | `<Dialogs>`<br>`  <Dialog>`<br>`  ... Full Dialog Object ...`<br>`  </Dialog>`<br>`  <Dialog>`<br>`  ... Full Dialog Object ...`<br>`  </Dialog>`<br>`  <Dialog>`<br>`  ... Full Dialog Object ...`<br>`  </Dialog>`<br>`  <Dialog>`<br>`  ... Full Dialog Object ...`<br>`  </Dialog>`<br>`  <Dialog>`<br>`  ... Full Dialog Object ...`<br>`  </Dialog>`<br>`  ... Additional Dialogs...`<br>`</Dialogs>` |
| **Failure Response Example:** | `<ApiErrors>`<br>`  <ApiError>`<br>`  <ErrorType>Authorization Failure</ErrorType>`<br>`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`<br>`  <ErrorData>jsmith</ErrorData>`<br>`  </ApiError>`<br>`</ApiErrors>` |
| **Error Codes:** | • Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

For more information about Dialog objects and Dialog APIs see Section 3.2 Dialog APIs.

### Response Parameters

- uri
- mediaType
- state
- stateCause
- fromAddress
- mediaAddress
- dnis

- dialedNumber
- Participants
- Participant
- Actions
- mediaProperties
- callvariables
- CallVariable

### 3.1.4. User—Sign In to Finesse

[contents]

The User—Sign In to Finesse API allows a user to sign in to the CTI server. This API forces a sign-in. That is, if a user is already signed in, that user will be signed in again.

If the response is successful, the user is signed in and is automatically set to the NOT_READY state.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<id> |
| **Example URI:** | `http://host/finesse/api/User/1234` |
| **Security Constraints:** | Role—Agent or Supervisor<br>Limitations—Users can only act on their own User objects. |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | `<User>`<br>`  <state>LOGIN</state>`<br>`  <extension>1001001</extension>`<br>`</User>` |
| **HTTP Response:** | 202—Successfully Accepted<br>400—Bad Request (for example, malformed or incomplete request, or invalid extension)<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>404—Not Found (for example, the user ID is not known)<br>503—Service Unavailable (for example, the notification service is not running) |

| | |
|---|---|
| **Failure Response Example:** | <ApiErrors><br>  <ApiError><br>  <ErrorType>Invalid Authorization User Specified</ErrorType><br>  <ErrorData>4321</ErrorData><br>  <ErrorMessage>The user specified in the authentication credentials and the uri don't match</ErrorMessage><br>  </ApiError><br></ApiErrors> |
| **Error Codes:** | • Parameter Missing<br>• Invalid Input<br>• Invalid Device<br>• Generic Error<br>• Authorization Failure<br>• Invalid Authorization User Specified<br>• User Not Found<br>• Service Unavailable<br>For descriptions and other possible error codes, see API Error Codes. |
| **Notifications Triggered:** | User notification |

### Request Parameters

- id—Required
- state—Required
- extension—Required

## 3.1.5. User—Sign Out of Finesse

[contents]

The User—Sign Out of Finesse API allows a user to sign out of the CTI server.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<id> |
| **Example URI:** | http://host/finesse/api/User/1234 |
| **Security Constraints:** | Role—Agent or Supervisor<br>Limitations—Users can only act on their own User objects. |

| | |
|---|---|
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | <User><br><state>LOGOUT</state><br></User> |
| **HTTP Response:** | 202—Successfully accepted<br>400—Bad Request (for example, malformed or incomplete request, or invalid extension)<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>404—Not Found (for example, the user ID is not known)<br>503—Service Unavailable (for example, the notification service is not running) |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Invalid Input</ErrorType><br><ErrorData>state</ErrorData><br><ErrorMessage>Invalid State specified for user</ErrorMessage><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Parameter Missing<br>• Invalid Input<br>• Invalid State<br>• Authorization Failure<br>• Invalid Authorization User Specified<br>• User Not Found<br>• Internal Server Error<br>• Service Unavailable<br>For descriptions and other possible error codes, see API Error Codes. |
| **Notifications Triggered:** | User notification |

### Request Parameters

- id—Required
- state—Required

## 3.1.6. User—Change Agent State

[contents]

The User—Change Agent State API allows a user to change the state of an agent on the CTI server.

If the request is successful, the response of the state change is sent as part of a User notification.

The following figure shows the supported state transitions.

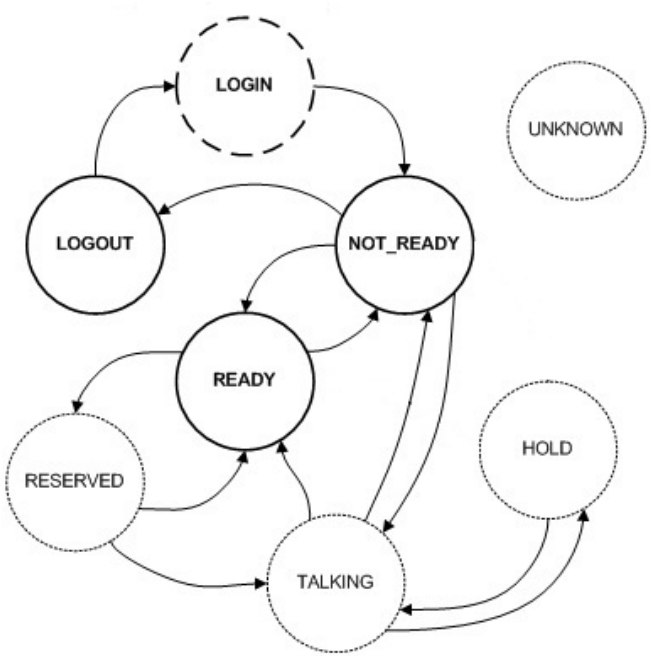


Figure **7**: Supported State Transitions

Users can set the following states using this API:

- READY
- NOT_READY
- LOGOUT

The LOGIN state is a transitive state. That is, when set, LOGIN triggers a change that results in a new state.

The following are states that users can be in while on a call. However, these states are not set by the user. For example, agents cannot change their state to TALKING. They enter the TALKING state when they answer an ACD call.

- RESERVED
- TALKING
- HOLD

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<id> |
| **Example URI:** | `http://host/finesse/api/User/1234` |
| **Security Constraints:** | Role—Agent or Supervisor<br>Limitations—Users can only act on their own User objects. |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | `<User>`<br>`  <state>READY</state>`<br>`</User>` |
| **HTTP Response:** | 202—Successfully accepted<br>400—Bad request<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>404—Not Found (for example, the user ID is not known)<br>500—Internal server error<br>503—Service Unavailable (for example, the notification service is not running) |
| **Failure Response Example:** | `<ApiErrors>`<br>`  <ApiError>`<br>`  <ErrorType>Parameter Missing</ErrorType>`<br>`  <ErrorData>state</ErrorData>`<br>`  <ErrorMessage>State Parameter missing</ErrorMessage>`<br>`  </ApiError>`<br>`</ApiErrors>` |
| **Error Codes:** | • Parameter Missing<br>• Invalid Input<br>• Invalid State<br>• Authorization Failure<br>• Invalid Authorization User Specified |

| | |
|---|---|
| | • User Not Found<br>• Internal Server Error<br>• Service Unavailable<br>For descriptions and other possible error codes, see API Error Codes. |
| **Notifications Triggered:** | User notification |

**Request Parameters**

- id—Required
- state—Required

## 3.1.7. User—Get ReasonCode

[contents]

The User—Get ReasonCode API allows a user (agent or supervisor) to get an individual NOT_READY or LOGOUT reason code, which is already defined and stored in the Finesse configuration database (and that is applicable to the user). The user can display the reason code label on their desktop when they change their state to NotReady or Logout respectively.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<id>/ReasonCode/<reasoncodeId> |
| **Example URI:** | `http://host/finesse/api/User/1234/ReasonCode/12` |
| **Security Constraints:** | Role—Agent or Supervisor<br>Limitations—A user must be signed in to get a reason code. A user cannot retrieve reason codes that belong to another user. |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:** | `<ReasonCode>`<br>`  <uri>/finesseconfig/api/ReasonCode/1</uri>`<br>`  <category>NOT_READY</category>`<br>`  <code>12</code>` |

```
    <label>Lunch</label>
    <forAll>true</forAll>
</ReasonCode>
```

| | |
|---|---|
| **HTTP Response:** | 200—Success<br>400—Bad request<br>400—Finesse API error (for example, the object does not exist, the object is stale, violation of DB constraint, and so on)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>404—Not Found (for example, the reason code does not exist or has been deleted)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>1234</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Not Found<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- category
- code
- forAll
- label
- uri

## 3.1.8. User—Get ReasonCode List

[contents]

The User—Get ReasonCode List API allows a user (an agent or supervisor) to get a list of NOT_READY or LOGOUT reason codes (that are applicable to that user), which are defined and stored in the Finesse configuration database. Users can assign one of the reason codes on the desktop when they change their state to Not Ready or Logout respectively. The required URL parameter "category" is used to specify which (Logout or Not Ready) reason codes are returned.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<id>/ReasonCodes?category=NOT_READY\|LOGOUT |

| | |
|---|---|
| **Example URI:** | `http://host/finesse/api/User/1234/ReasonCodes?category=NOT_READY` |
| **Security Constraints:** | Role—Agent or Supervisor<br>Limitations—A user must be signed in to get a list of reason codes. A user cannot retrieve reason codes that belong to another user. |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:** | `<ReasonCodes category="NOT_READY">`<br>`  <ReasonCode>`<br>`  <uri>/finesseconfig/api/ReasonCode/1</uri>`<br>`  <category>NOT_READY</category>`<br>`  <code>12</code>`<br>`  <label>Lunch</label>`<br>`  <forAll>true</forAll>`<br>`  </ReasonCode>`<br>`  <ReasonCode>`<br>`  ... Full ReasonCode Object ...`<br>`  </ReasonCode>`<br>`  <ReasonCode>`<br>`  ... Full ReasonCode Object ...`<br>`  </ReasonCode>`<br>`</ReasonCodes>` |
| **HTTP Response:** | 200—Success<br>400—Bad request<br>400—Finesse API error (for example, the object does not exist, the object is stale, violation of DB constraint, and so on)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>404—Not Found (for example, the reason code does not exist or has been deleted)<br>500—Internal server error |
| **Failure Response Example:** | `<ApiErrors>`<br>`  <ApiError>`<br>`  <ErrorType>Authorization Failure</ErrorType>` |

```
        <ErrorMessage>UNAUTHORIZED</ErrorMessage>
        <ErrorData>1234</ErrorData>
    </ApiError>
</ApiErrors>
```

| | |
|---|---|
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Not Found<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

**Note:** The ReasonCode list could be empty (for example, if there are no reason codes for the specified category in the Finesse configuration database).

**Note:** For Finesse Release 8.5(3), all reason codes that have the *forAll* parameter set to true, will apply to any user.

#### Request Parameters

category—Required

#### Response Parameters

- ReasonCodes
- category
- ReasonCode

## 3.1.9. User—Get WrapUpReason

[contents]

The User—Get WrapUpReason API allows a user to retrieve a WrapUpReason object. For more information about wrap-up reasons, see Wrap-up Reason APIs.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<userId>/WrapUpReason/<wrapUpReasonId> |
| **Example URI:** | `http://host/finesse/api/User/1234/WrapUpReason/1001` |
| **Security Constraints:** | Role—Agent, Supervisor, or Administrator |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| | |

| | |
|---|---|
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:** | <WrapUpReason><br>  <uri>/finesse/api/User/1234/WrapUpReason/205</uri><br>  <label>Product Question</label><br>  <forAll>true</forAll><br></WrapUpReason> |
| **HTTP Response:** | 200—Success<br>400—Bad request<br>400—Finesse API error (for example, the object does not exist, the object is stale, violation of DB constraint, and so on)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>404—User Not Found (the user id provided is invalid and no such agent exists in CTI)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br>  <ApiError><br>  <ErrorType>Authorization Failure</ErrorType><br>  <ErrorMessage>UNAUTHORIZED</ErrorMessage><br>  <ErrorData>1234</ErrorData><br>  </ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- uri
- label
- forAll

## 3.1.10. User—Get WrapUpReason List

[contents]

The User—Get WrapUpReason List API allows a user to get a list of all wrap-up reasons applicable to that user. For more information about wrap-up reasons, see Wrap-up Reason APIs.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<userId>/WrapUpReasons |
| **Example URI:** | `http://host/finesse/api/User/1234/WrapUpReasons` |
| **Security Constraints:** | Role—Agent, Supervisor, or Administrator |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:** | `<WrapUpReasons>`<br>`  <WrapUpReason>`<br>`  <label>Successful tech support call</label>`<br>`  <forAll>true</forAll>`<br>`  <uri>/finesse/api/User/1234/WrapUpReason/205</uri>`<br>`  </WrapUpReason>`<br>`  ... more wrap-up reasons ...`<br>`</WrapUpReasons>` |
| **HTTP Response:** | 200—Success<br>400—Bad request<br>400—Finesse API error (for example, the object does not exist, the object is stale, violation of DB constraint, and so on)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>404—User Not Found (the user id provided is invalid and no such agent exists in CTI)<br>500—Internal server error |
| **Failure Response Example:** | `<ApiErrors>`<br>`  <ApiError>`<br>`  <ErrorType>Authorization Failure</ErrorType>`<br>`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`<br>`  <ErrorData>1234</ErrorData>`<br>`  </ApiError>`<br>`</ApiErrors>` |

| | |
|---|---|
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

**Response Parameters**

- uri
- label
- forAll

## 3.1.11. User—Change Agent State (Pass NotReady or Logout Corresponding ReasonCode to CTI)

[contents]

This API allows users to change the state of the agent in the CTI server and pass along the code value of a corresponding reason code (when the state to be changed is NotReady or Logout). The user must already be authenticated (by way of the tomcat realm) to successfully use this API.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<id> |
| **Example URI:** | `http://host/finesse/api/User/1234` |
| **Security Constraints:** | Role—Agent<br>Limitations—Users can only act on their own User objects. |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | `<User>`<br>`  <state>NOT_READY</state>`<br>`  <ReasonCodeId>1001</ReasonCodeId>`<br>`</User>` |
| **HTTP Response:** | 202—Successfully accepted<br>400—Parameter Missing |

| | |
|---|---|
| | 400—Invalid Input<br>400—Invalid State<br>401—Authorization Failure (for example, the user is not authenticated in the Web Session)<br>401—Invalid Authorization User Specified (for example, the authenticated user tried to make a request as another user)<br>404—User Not Found (the agent ID provided is invalid and no such agent exists in CTI)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br>  <ApiError><br>  <ErrorType>Parameter Missing</ErrorType><br>  <ErrorData>state</ErrorData><br>  <ErrorMessage>State Parameter missing</ErrorMessage><br>  </ApiError><br></ApiErrors> |
| **Error Codes:** | • Parameter Missing<br>• Invalid Input<br>• Invalid State<br>• Authorization Failure<br>• Invalid Authorization User Specified<br>• User Not Found<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |
| **Notifications Triggered:** | User notification |

### Request Parameters

- id—Required
- ReasonCodeID
- state—Required

## 3.1.12. User—Get MediaPropertiesLayout

[contents]

The User—Get MediaPropertiesLayout API allows an agent to get a copy of the mediaProperties object, which determines how call variables and ECC variables appear on the agent desktop. For more information about the mediaProperties object, see Media Properties Layout APIs.

**Note:** Finesse Release 8.5(3) supports a single default instance only.

Finesse supports the following media properties:

- call variables (callVariable1 through callVariable10)

- ECC variables

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<userId>/MediaPropertiesLayout/default |
| **Example URI:** | `http://host/finesse/api/User/1234/MediaPropertiesLayout/default` |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:** | <MediaPropertiesLayout><br>  <header><br>  <entry><br>  <displayName>Customer Name</displayName><br>  <mediaProperty>callVariable1</mediaProperty><br>  </entry><br>  </header><br>  <column><br>  <entry><br>  <displayName>Customer Name</displayName><br>  <mediaProperty>callVariable1</mediaProperty><br>  </entry><br>  <entry><br>  <displayName>Customer Acct#</displayName><br>  <mediaProperty>user.cisco.acctnum</mediaProperty><br>  </entry><br>  </column><br>  <column><br>  <entry><br>  <displayName>Support contract</displayName><br>  <mediaProperty>callVariable2</mediaProperty><br>  </entry><br>  <entry><br>  <displayName>Product calling about</displayName><br>  <mediaProperty>callVariable3</mediaProperty><br>  </entry><br>  </column><br></MediaPropertiesLayout> |
| **HTTP Response:** | 200—Success<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors> |

| | |
|---|---|
| | <pre><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors></pre> |
| **Error Codes:** | • Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- header
- column
- entry
- displayName
- mediaProperty

## 3.2. Dialog APIs

[contents]

The Dialog object represents a dialog with participants. For the media type "voice", this object represents a call. A participant represents an internal or external user's CallConnection, or that user's leg of the call.



### 3.2.1. Dialog—Get Dialog

[contents]

The Dialog—Get Dialog API allows users to get a copy of the Dialog object.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/Dialog/<id> |
| **Example URI:** | <pre>http://host/finesse/api/Dialog/12345678</pre> |
| **Security Constraints:** | Role—Agent, Administrator<br>Limitations—Agents can only act on their own Dialog object. Administrators can get any Dialog object. |

| | |
|---|---|
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **HTTP Response:** | 200—Success<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>401—Invalid Authorization (for example, the authenticated user tried to get a Dialog for which they are not a participant)<br>404—Not found (for example, the dialog id is invalid)<br>500—Internal server error |
| **Successful Response:** | <Dialog><br><uri>/finesse/api/Dialog/12345678</uri><br><mediaType>Voice</mediaType><br><state>ACTIVE</state><br><fromAddress>2002</fromAddress><br><toAddress>2000</toAddress><br><mediaProperties><br><dialedNumber>2000</dialedNumber><br><callType>AGENT_INSIDE</callType><br><DNIS>2000</DNIS><br><wrapUpReason>Another satisfied customer</wrapUpReason><br><callvariables><br><CallVariable><br><name>callVariable1</name><br><value>Chuck Smith</value><br></CallVariable><br><CallVariable><br><name>callVariable2</name><br><value>Cisco Systems,Inc<value><br></CallVariable><br><CallVariable><br><name>callVariable3</name><br><value>chuckSmith@cisco.com<value><br></CallVariable><br>... Other CallVariables up to 10 ...<br><CallVariable><br><name>callVariable10</name><br><value>Preferred Customer<value><br></CallVariable><br><CallVariable><br><name>ecc.user</name><br><value>csmith<value><br></CallVariable><br><CallVariable><br><name>ecc.years[0]</name><br><value>1985<value><br></CallVariable><br><CallVariable><br><name>ecc.years[1]</name><br><value>1995<value><br></CallVariable><br><CallVariable><br><name>ecc.years[2]</name> |

```
            <value>2005<value>
          </CallVariable>
        </callvariables>
    </mediaProperties>
    <participants>
       <Participant>
          <mediaAddress>2002</mediaAddress>
          <state>ACTIVE</state>
          <stateCause></stateCause>
          <actions>
             <action>HOLD</action>
             <action>DROP</action>
          </actions>
       </Participant>
       <Participant>
          <mediaAddress>2000</mediaAddress>
          <state>HELD</state>
          <stateCause></stateCause>
          <actions>
             <action>RETRIEVE</action>
             <action>DROP</action>
          </actions>
       </Participant>
    </participants>
</Dialog>
```

**Failure Response Example:**

```
<ApiErrors>
  <ApiError>
    <ErrorType>Not Found</ErrorType>
    <ErrorMessage>Invalid dialogId specified for dialog</ErrorMessage>
  </ApiError>
</ApiErrors>
```

**Error Codes:**

- Authorization Failure
- Invalid Authorization User Specified
- Dialog Not Found
- Internal Server Error

### Response Parameters

- uri
- mediaType
- state
- stateCause
- fromAddress
- mediaAddress
- dialedNumber
- dnis
- wrapUpReason
- Participants
- Participant
- Actions

- mediaProperties
- callType
- callvariables
- CallVariable

### 3.2.2. Dialog—Take Action on a Participant within a Dialog

[contents]

The Dialog—Take Action on a Participant within a Dialog API allows a user to take an action on a participant within a dialog. Agents must be the participant they are targeting with an action.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/Dialog/<id> |
| **Example URI:** | `http://host/finesse/api/Dialog/54321` |
| **Security Constraints:** | Role—Agent<br>Limitation—Agents can only act on a participant of a dialog when they are that participant. |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | `<Dialog>`<br>`  <targetMediaAddress>1001001</targetMediaAddress>`<br>`  <requestedAction>ANSWER</requestedAction>`<br>`</Dialog>` |
| **HTTP Response:** | 202—Successfully accepted<br>400—Parameter missing (for example, the targetMediaAddress or action is not provided)<br>400—Invalid input<br>401—Authorization failure<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>404—Dialog not found<br>500—Internal server error |
| **Failure Response Example:** | `<ApiErrors>` |

| | |
|---|---|
| | <ApiError><br><ErrorType>Invalid Input</ErrorType><br><ErrorData>requestedAction</ErrorData><br><ErrorMessage>Invalid 'requestedAction' specified for dialog</ErrorMessage><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Invalid Input<br>• Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |
| **Notifications Triggered:** | • Dialog Notification<br>• Dialog CTI Error Notification (if a CTI error occurs) |

#### Request Parameters

- targetMediaAddress—Required
- requestedAction—Required

## 3.2.3. Dialog—Update Call Variable Data

[contents]

The Dialog—Update Call Variable Data API allows a user to set or change call variables (including named variables or ECC variables) on a particular dialog. If the user is an agent, that user must be the participant they are targeting with the action. The corresponding event is only published if any of the values of the call variables or named variables are updated.

**Note:** Cisco Finesse Release 8.5(3) only supports Latin1 characters for ECC variables. Other Unicode characters are not supported. For example, if a user tries to use this API to update an ECC variable that contains Chinese characters, Finesse may not return the correct value in the subsequent dialog update it sends to the client.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/Dialog/<id> |
| **Example URI:** | http://host/finesse/api/Dialog/54321 |
| **Security Constraints:** | Role—Agent<br>Limitation—Agents can only act on a participant of a dialog when they are that participant. |
| **HTTP Method:** | PUT |

| | |
|---|---|
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | <Dialog><br><requestedAction>UPDATE_CALL_DATA</requestedAction><br><mediaProperties><br><wrapUpReason>{39 byte description of the call}</wrapUpReason><br><callvariables><br><CallVariable><br><name>{name of the call variable/named variable}</name><br><value>{value to be changed}</value><br></CallVariable><br><CallVariable><br>... Other call variables to be modified ...<br></CallVariable><br></callvariables><br></mediaProperties><br></Dialog> |
| **HTTP Response:** | 202—Successfully accepted<br>400—Parameter missing (for example, the mediaProperties, callvariables, callvariable, or action is not provided)<br>400—Invalid input (the callvariable name or action is not recognized or is invalid or there are duplicate call variable names)<br>401—Authorization failure<br>401—Invalid Authorization User Specified (the authenticated user tried to make a request of another dialog that is not their own)<br>404—Dialog not found<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Parameter missing<br>• Invalid Input<br>• Authorization Failure<br>• Invalid Authorization User Specified<br>• Dialog not found<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

| | |
|---|---|
| **Notifications Triggered:** | • Dialog Notification<br>• Dialog CTI Error Notification (if a CTI error occurs) |

### Request Parameters

- mediaProperties—Required
- wrapUpReason—Either wrapUpReason or callvariables must be present
- callvariables—Either wrapUpReason or callvariables must be present
- CallVariable—Required if the callvariables tag is present
- requestedAction—Required

**Note:** If both call variables and a wrap-up reason are present in the request to update call data, the values for the wrap-up reason and call variables must all pass validation for Finesse to send the request to Unified CCE.

### ECC and Call Variable Error Handling

When a client makes an invalid update request for a ECC or call variable, that request is sent to Finesse and then to Unified CCE. Unified CCE logs certain errors but does not return events for them. In these cases, Finesse does not return an error. Clients must be aware of this behavior and follow the appropriate Unified CCE documentation.

A client can also send an update request for an ECC or call variable that contains both valid and invalid data (that is, some of the ECC or call variable updates in the request payload are valid while others are invalid). See the following table to determine the response from Finesse in these error scenarios.

| Error Scenario | Unified CCE Response | Finesse Response |
|---|---|---|
| 1. A request was sent that generates an error from Unified CCE to Finesse.<br>2. The request payload contained no valid ECC or call variables. | Unified CCE sends an error to Finesse. | Finesse forwards the error to the client. |
| 1. A request was sent that generates an error from Unified CCE to Finesse.<br>2. The request payload contained a mix of valid and invalid ECC or call variables. | 1. Unified CCE sends an error to Finesse.<br>2. Unified CCE does not send an UPDATE_CALL_DATA event to Finesse (that is, Unified CCE fails the entire request). | 1. Finesse forwards the error to the client.<br>2. The client does not receive an UPDATE_CALL_DATA event. |
| 1. A request was sent that does not generate an error from Unified CCE to Finesse.<br>2. The request payload contained no | Unified CCE does not respond. | Finesse does not respond. |

| | | |
|---|---|---|
| valid ECC or call variables. | | |
| 1. A request was sent that does not generate an error from Unified CCE to Finesse.<br>2. The request payload contained a mix of valid and invalid ECC or call variables. | 1. Unified CCE does not send an error to Finesse.<br>2. Unified CCE sends an UPDATE_CALL_DATA event to Finesse for the valid ECC and call variables. | 1. Finesse does not forward an error to the client.<br>2. Finesse forwards the UPDATE_CALL_DATA event to the client. |

**Note:** When the size of the value of an ECC variable name exceeds its maximum length, Unified CCE silently truncates the value and updates the variable. As a result, Finesse does not receive a maximum length error.

Users of this API must ensure that the variables they are trying to update exist. Users must follow the exact format of each variable and ensure that the maximum size is not exceeded.

### 3.2.4. Dialog—Create a New Dialog (Make a Call)

[contents]

The Dialog—Create a New Dialog API allows users to make a call. Making a call is accomplished by creating a new Dialog and specifying the fromAddress (the caller's extension) and the toAddress (the destination target), and posting it to the Dialog collection for that user.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<id>/Dialogs |
| **Example URI:** | `http://host/finesse/api/User/1234/Dialogs` |
| **Security Constraints:** | Role—Agent<br>Limitations—Users can only create dialogs with a fromAddress that they are currently signed in to. |
| **HTTP Method:** | POST |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | `<Dialog>`<br>`<requestedAction>MAKE_CALL</requestedAction>`<br>`<fromAddress>1001001</fromAddress>`<br>`<toAddress>1001002</toAddress>` |

| | |
|---|---|
| | <Dialog><br>...<br></Dialog> |
| **HTTP Response:** | 202—Successfully accepted<br>Note: The 202 Successfully accepted response only indicates successful completion of the request. The request is processed and the actual response of the action is sent as part of a Dialog notification.<br>400—Parameter missing (for example, the fromAddress or toAddress is not provided)<br>400—Invalid input<br>400—Invalid destination<br>401—Authorization failure (for example, the user is not yet authenticated in the Web Session)<br>401—Invalid Authorization (for example, the authenticated user tried to use a fromAddress that is not their own)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Parameter Missing<br>• Invalid Input<br>• Invalid Destination<br>• Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error |
| **Notifications Triggered** | If a CTI error occurs, Finesse sends a Dialog CTI Error Notification |

### Request Parameters

- requestedAction
- fromAddress
- toAddress

## 3.2.5. Dialog—Make a Consult Call Request

[contents]

The Dialog—Make a Consult Call Request API allows an agent to make a consult call request. After the consult call request succeeds, that call can be completed as a transfer or conference. The requestedAction for a consult call is CONSULT_CALL. This request is sent to the Dialog URL of an existing active call, from where the call is initiated. The ID in the URL represents the Dialog ID of the active call.

Finesse Release 8.5(3) supports the transfer or conference of any held call to the current active call, as long as the agent performing the transfer or conference is a participant in both the held and active calls. Finesse Release 8.5(3) does not support single-step transfer. Furthermore, Finesse does not support blind transfer or blind conference through the API or desktop.

Blind transfer is defined as follows: An agent has an active call and initiates a consult call to a destination, and then completes the transfer while the call is ringing at the destination.

Blind conference is defined as follows: An agent has an active call and initiates a consult call to a destination, and then starts a conference while the call is ringing at the destination.

Note on call variables in transfer or conference: Finesse maintains a copy of the call variables (including call peripheral variables and ECC variables) for every call (Dialog object) in the system. The next time Unified CCE sets the call variables to values that are not NULL (through CTI events like CALL_DATA_UPDATE_EVENT), the call variables maintained by Finesse are updated with these (not NULL) values. In this way, Finesse ensures that a client always receives the latest data for call variables sent by Unified CCE. Because an empty string is considered a valid value, when call variables are set to empty strings by Unified CCE, Finesse updates its version of the same call variables to empty strings, and then updates the client.

**Note:** An agent or supervisor who signs in after being on an active conference call with other devices (which are not associated with any other agent or supervisor) may experience unpredictable behavior with the Finesse Desktop due to incorrect Dialog notification payloads. These limitations also encompass failover scenarios where failover occurs while the agent or supervisor is participating in a conference call. For example, an agent is on a conference call when the Finesse server fails. When that agent is redirected to the other Finesse server, that agent could see unpredictable behavior on the desktop. Examples of unpredictable behavior include, but are not limited to, the following:

- The desktop does not reflect all participants in a conference call.
- The desktop does not reflect that the signed-in agent or supervisor is in an active call.
- Dialog updates contain inconsistent payloads.

Despite these caveats, users may continue to perform normal operations on their phones. Desktop behavior will return to normal after the agent or supervisor drops off the conference call.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/Dialog/<id> |
| **Example URI:** | `http://host/finesse/api/Dialog/1234` |
| **Security Constraints:** | Role—Agent |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |

| | |
|---|---|
| **HTTP Request:** | <Dialog><br><requestedAction>CONSULT_CALL</requestedAction><br><toAddress>1001002</toAddress><br><targetMediaAddress>1001001<targetMediaAddress><br></Dialog> |
| **HTTP Response:** | 202—Successfully accepted<br>Note: This response only indicates a successful completion of the request. The request is processed and the actual response is sent as part of a Dialog notification.<br>400—Parameter missing (for example, the toAddress or requestedAction is not provided)<br>400—Invalid input (for example, the toAddress or requestedAction is invalid)<br>400—Invalid destination (for example, the toAddress is the same as the caller's extension)<br>401—Authorization failure<br>401—Invalid authorization (for example, a user tried to use a fromAddress that did not belong to that user)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Parameter Missing<br>• Invalid Input<br>• Invalid Destination<br>• Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error |
| **Notifications Triggered** | • Dialog Notification<br>• Dialog CTI Error Notification (if a CTI error occurs) |

### Request Parameters

- requestedAction—Required
- targetMediaAddress—Required
- toAddress—Required

[contents]

### 3.2.6. Dialog—Make a Silent Monitoring Call

The Dialog—Make a Silent Monitoring Call API allows a supervisor to silently monitor an agent on an active call, who is in Talking state. A new dialog is created, specifying the fromAddress (supervisor's extension) and the toAddress (agent's extension), and posting the call to the supervisor's dialog collection.

**Note:** Phones of agents to be monitored must support silent monitoring and must be configured in Cisco Unified Communications Manager as follows:

- The correct device type must be configured.
- The device must have Bridge Monitoring enabled.
- The correct permissions must be configured (under User Management > End User > PG User, in the Permissions area, select Standard CTI Allow Call Recording, and then click Add to User Group).

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/User/<id>/Dialogs |
| **Example URI:** | `http://host/finesse/api/User/1234/Dialogs` |
| **Security Constraints:** | Role—Supervisor, Administrator<br>Limitations—A supervisor must be signed in to the fromAddress (extension) being used to create the silent monitoring call. Agents to be monitored must be assigned to a team that supervisor is responsible for. An administrator can silently monitor any call, except a "silent monitored" call.<br>If the agent transfers the call that the supervisor is monitoring, the silent monitoring session ends. |
| **HTTP Method:** | POST |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | `<Dialog>`<br>`  <requestedAction>SILENT_MONITOR</requestedAction>`<br>`  <fromAddress>1001001</fromAddress>`<br>`  <toAddress>2001002</toAddress>`<br>`</Dialog>` |
| **HTTP Response:** | 202—Successfully accepted<br>Note: This response only indicates a successful completion of the request. The request is processed and the actual response is |

| | |
|---|---|
| | sent as part of a Dialog notification.<br>400—Parameter missing (the fromAddress, toAddress, or requestedAction is not provided)<br>400—Invalid input (the fromAddress, toAddress, or requestedAction is invalid)<br>400—Invalid destination (for example, the fromAddress and toAddress are the same)<br>400—Invalid state (the supervisor is already silent monitoring or in an active call)<br>401—Authorization failure (the user is not authenticated in the web session yet or is not a supervisor or administrator)<br>401—Invalid authorization user specified (a user tried to use the ID of another user in the request URL or tried to monitor an agent that did not belong to a team that user supervises)<br>401—Unauthorized (a supervisor, who is not an administrator, tried to use a fromAddress that did not belong to that supervisor to request a silent monitor call)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Parameter Missing<br>• Invalid Input<br>• Invalid Destination<br>• Invalid State<br>• Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error |
| **Notifications Triggered** | Dialog Notification |

### Request Parameters

- requestedAction—Required
- fromAddress—Required
- toAddress—Required

## 3.2.7. Dialog—End a Silent Monitoring Call

[contents]

The Dialog—End a Silent Monitoring Call API allows a supervisor to drop a silent monitoring call that was initiated by that supervisor. The Dialog is updated by specifying a requestedAction of DROP and targetMediaAddress of the extension of the supervisor who initiated the silent monitoring call.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/Dialog/<dialogid> |
| **Example URI:** | `http://host/finesse/api/Dialog/32458` |
| **Security Constraints:** | Role—Supervisor, Administrator<br>Limitations—A supervisor can only end a silent monitoring call that was initiated by that supervisor. An administrator can end any silent monitoring call. |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | `<Dialog>`<br>`  <requestedAction>DROP</requestedAction>`<br>`  <targetMediaAddress>1001002</targetMediaAddress>`<br>`</Dialog>` |
| **HTTP Response:** | 202—Successfully accepted<br>Note: This response only indicates a successful completion of the request. The request is processed and the actual response is sent as part of a Dialog notification.<br>400—Parameter missing (the targetMediaAddress or requestedAction is not provided)<br>400—Invalid input (the targetMediaAddress or requestedAction is invalid)<br>401—Authorization failure (the user is not authenticated in the web session yet or is not an administrator or supervisor)<br>401—Invalid authorization user specified (for example, a supervisor tried to use a targetMediaAddress that did not belong to that supervisor)<br>404—Not found (the dialog specified by the dialogid does not exist)<br>500—Internal server error |
| **Failure Response Example:** | `<ApiErrors>`<br>`  <ApiError>`<br>`  <ErrorType>Authorization Failure</ErrorType>`<br>`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`<br>`  <ErrorData>jsmith</ErrorData>`<br>`  </ApiError>`<br>`</ApiErrors>` |

| | |
|---|---|
| **Error Codes:** | • Parameter Missing<br>• Invalid Input<br>• Authorization Failure<br>• Invalid Authorization User Specified<br>• Not Found<br>• Internal Server Error |
| **Notifications Triggered** | Dialog Notification |

**Request Parameters**

- requestedAction—Required
- targetMediaAddress—Required

## 3.3. Team APIs

[contents]

The team object represents a team and contains the URI, team ID, team name, and the users associated with that team.



### 3.3.1. Team—Get Object

[contents]

The Team—Get Object API allows a user to get the configuration information for a specific team, which includes the Team ID and a list of agents that are a member of that team.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/Team/<id> |
| **Example URI:** | `http://host/finesse/api/Team/5004` |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |

| | |
|---|---|
| **HTTP Response:** | 200—Success<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>404—Not found (for example, the team id is invalid) |
| **Successful Response:** | `<Team>`<br>`<id>5004</id>`<br>`<name>5004</name>`<br>`<uri>/finesse/api/Team/5004</uri>`<br>`<users>`<br>`<User>`<br>`<firstName>John</firstName>`<br>`<lastName>Brown</lastName>`<br>`<loginId>1001053</loginId>`<br>`<state>NOT_READY</state>`<br>`<extension>1001</extension>`<br>`<uri>/finesse/api/User/1001053</uri>`<br>`</User>`<br>`<User>`<br>`<firstName>Jason</firstName>`<br>`<lastName>Smith</lastName>`<br>`<loginId>1001052</loginId>`<br>`<state>TALKING</state>`<br>`<extension>1002</extension>`<br>`<uri>/finesse/api/User/1001052</uri>`<br>`</User>`<br>`</users>`<br>`</Team>` |
| **Failure Response Example:** | `<ApiErrors>`<br>`<ApiError>`<br>`<ErrorType>Authorization Failure</ErrorType>`<br>`<ErrorMessage>UNAUTHORIZED</ErrorMessage>`<br>`<ErrorData>jsmith</ErrorData>`<br>`</ApiError>`<br>`</ApiErrors>` |
| **Error Codes:** | • Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- uri
- id
- name
- users
- User
- loginId
- firstName
- lastName

- state

## 3.4. System APIs

[contents]

The SystemInfo object represents the Finesse system and includes the current system state, the XMPP server and pubSub domains configured for the system, and the hostnames or IP addresses of the primary and secondary (if configured) Finesse nodes.

### 3.4.1. SystemInfo—Get SystemInfo

[contents]

The SystemInfo—Get SystemInfo API allows a user to get information about the current status of the system, the XMPP server domain, the XMPP pubSub domain, and hostnames or IP addresses of the primary and secondary nodes.

| | |
|---|---|
| **URI:** | http://<server>/finesse/api/SystemInfo |
| **Example URI:** | `http://host/finesse/api/SystemInfo` |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **HTTP Response:** | 200—Success<br>500—Internal server error |
| **Successful Response:** | `<SystemInfo>`<br>`<status>OUT_OF_SERVICE</status>`<br>`<xmppDomain>xmppserver.cisco.com</xmppDomain>`<br>`<xmppPubSubDomain>pubsub.xmppserver.cisco.com</xmppPubSubDomain>`<br>`<primaryNode>`<br>`<host>172.16.204.25</host>`<br>`</primaryNode>`<br>`<secondaryNode>`<br>`<host>172.16.204.26</host>`<br>`</secondaryNode>`<br>`</SystemInfo>` |
| **Failure Response Example:** | `<ApiErrors>`<br>`<ApiError>`<br>`<ErrorType>Internal Server Error</ErrorType>`<br>`<ErrorMessage>Runtime Exception</ErrorMessage>` |

| | |
|---|---|
| | <ErrorData></ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- status
- xmppDomain
- xmppPubSubDomain
- primaryNode - host
- secondaryNode - host

# 4. The Cisco Finesse Configuration APIs

[contents]

Administrators use these APIs to configure the Finesse system (for example, the primary and backup CTI server settings).

The Configuration APIs require administrator credentials to be passed into the basic authorization header.



## 4.1. System Configuration APIs

[contents]

The SystemConfig object is a container element that holds the Finesse system configuration, including details about the primary and backup CTI servers.



### 4.1.1. SystemConfig—Get

[contents]

The SystemConfig—Get API allows an administrative user to get a copy of the SystemConfig object.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/SystemConfig |
| **Example URI:** | http://host/finesseconfig/api/SystemConfig |

| | |
|---|---|
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:** | <SystemConfig><br><uri>/finesseconfig/api/SystemConfig</uri><br><cti><br><host>10.1.1.1</host><br><port>42027</port><br><backupHost>10.1.1.2</backupHost><br><backupPort>42027</backupPort><br><peripheralId>5000</peripheralId><br></cti><br></SystemConfig> |
| **HTTP Response:** | 200—Success<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- cti - host
- cti - port
- cti - peripheralId

- cti - backupHost
- cti - backupPort

### 4.1.2. SystemConfig—Set

[contents]

The SystemConfig—Set API allows an administrative user to configure the system settings.

**Note:** If the backupHost and backupPort are not specified in the XML body during a PUT, and they were configured at an earlier time, the PUT operation removes these values from the database.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/SystemConfig |
| **Example URI:** | `http://host/finesseconfig/api/SystemConfig` |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | <SystemConfig><br><cti><br><host>10.1.1.1</host><br><port>42027</port><br><backupHost>10.1.1.2</backupHost><br><backupPort>42027</backupPort><br><peripheralId>5000</peripheralId><br></cti><br></SystemConfig> |
| **HTTP Response:** | 200—Success (the new settings were successfully written to the database)<br>400—Bad request<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Invalid Input</ErrorType><br><ErrorMessage>port</ErrorMessage><br><ErrorData>65536</ErrorData><br></ApiError><br></ApiErrors> |

| | |
|---|---|
| **Error Codes** | • Parameter Missing<br>• Invalid Input<br>• Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Request Parameters

- cti - host—Required
- cti - port—Required
- cti - peripheralId—Required
- cti - backupHost—Required if backupPort is present in the request
- cti - backupPort—Required if backupHost is present in the request

## 4.2. Cluster Configuration APIs

[contents]

The ClusterConfig object is a container element that holds Finesse cluster configuration. This container supports the addition of a single, secondary Finesse node. After the secondary Finesse node is installed and ready, it becomes part of the cluster.

This feature also reports replication status. Replication status determines whether a user is allowed to or restricted from changing the value of the secondary node.

The Finesse server interacts with the VOS database to get and set information about the secondary node.

- ClusterConfig—Get
- ClusterConfig—Set

### 4.2.1. ClusterConfig—Get

[contents]

The ClusterConfig—Get API allows a user to get a copy of the ClusterConfig object.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/ClusterConfig |
| **Example URI:** | `http://host/finesseconfig/api/ClusterConfig` |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |

| | |
|---|---|
| **HTTP Request:** | — |
| **HTTP Response:** | 200—Success<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>500—Internal server error |
| **Successful Response Example:** | <ClusterConfig><br><uri>/finesseconfig/api/ClusterConfig</uri><br><secondaryNode><br><host>10.1.1.1</host><br></secondaryNode><br></ClusterConfig> |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes** | • Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- secondaryNode - host

## 4.2.2. ClusterConfig—Set

[contents]

The ClusterConfig—Set API allows an administrative user to configure the cluster settings.

**Note:** If the host value is blank or is not specified in the XML body during a PUT, the PUT operation removes the host value from the database.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/ClusterConfig |
| **Example URI:** | http://host/finesseconfig/api/ClusterConfig |
| **HTTP Method:** | PUT |

| | |
|---|---|
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | <pre><ClusterConfig><br>  <secondaryNode><br>  <host>10.1.1.1</host><br>  </secondaryNode><br></ClusterConfig></pre> |
| **HTTP Response:** | 200—Success (the new settings were successfully written to the database)<br>400—Bad request<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>500—Internal server error |
| **Failure Response Example:** | <pre><ApiErrors><br>  <ApiError><br>  <ErrorType>Datastore Error</ErrorType><br>  <ErrorData>5527</ErrorData><br>  <ErrorMessage>Error reading/writing ClusterConfig from the database</ErrorMessage><br>  </ApiError><br></ApiErrors></pre> |
| **Error Codes** | • Parameter Missing<br>• Invalid Input<br>• Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

#### Request Parameters

- secondaryNode - host—Required

## 4.3. Database Configuration APIs

[contents]

The EnterpriseDatabaseConfig object is a container element that holds the properties required by Finesse to connect to the Admin Workstation database (AWDB) server for user authentication.

- EnterpriseDatabaseConfig—Get
- EnterpriseDatabaseConfig—Set

[contents]

### 4.3.1. EnterpriseDatabaseConfig—Get

The EnterpriseDatabaseConfig—Get API allows a user to get a copy of the EnterpriseDatabaseConfig object.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/EnterpriseDatabaseConfig |
| **Example URI:** | `http://host/finesseconfig/api/EnterpriseDatabaseConfig` |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:** | `<EnterpriseDatabaseConfig>`<br>`<uri>/finesseconfig/api/EnterpriseDatabaseConfig</uri>`<br>`<host>10.1.1.1</host>`<br>`<backupHost></backupHost>`<br>`<port></port>`<br>`<databaseName>ucce8x_awdb</databaseName>`<br>`<domain>example.com</domain>`<br>`<username>Admin</username>`<br>`<password>password</password>`<br>`</EnterpriseDatabaseConfig>` |
| **HTTP Response:** | 200—Success<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>500—Internal server error |
| **Failure Response Example:** | `<ApiErrors>`<br>`<ApiError>`<br>`<ErrorType>Authorization Failure</ErrorType>`<br>`<ErrorMessage>UNAUTHORIZED</ErrorMessage>`<br>`<ErrorData>jsmith</ErrorData>`<br>`</ApiError>`<br>`</ApiErrors>` |
| **Error Codes:** | • Authorization Failure<br>• Internal Server Error |

| | |
|---|---|
| | For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- host
- backupHost
- port
- databaseName
- domain
- username
- password

## 4.3.2. EnterpriseDatabaseConfig—Set

[contents]

The EnterpriseDatabaseConfig—Set API allows an administrative user to configure the enterprise database settings.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/EnterpriseDatabaseConfig |
| **Example URI:** | `http://host/finesseconfig/api/EnterpriseDatabaseConfig` |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | <EnterpriseDatabaseConfig><br>  <uri>/finesseconfig/api/EnterpriseDatabaseConfig</uri><br>  <host>10.1.1.1</host><br>  <backupHost>10.1.1.2</backupHost><br>  <port>1433</port><br>  <databaseName>ucce8.x_awdb</databaseName><br>  <domain>example.com</domain><br>  <username>Admin</username><br>  <password>password</password><br></EnterpriseDatabaseConfig> |
| **HTTP Response:** | 200—Success (the new settings were successfully written to the database)<br>400—Bad request<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>500—Internal server error |

| | |
|---|---|
| **Failure Response Example:** | <pre><ApiErrors><br>  <ApiError><br>  <ErrorType>Invalid Input</ErrorType><br>  <ErrorMessage>host</ErrorMessage><br>  <ErrorData>10.1.1.1</ErrorData><br>  </ApiError><br></ApiErrors></pre> |
| **Error Codes** | • Parameter Missing<br>• Invalid Input<br>• Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Request Parameters

- host—Required
- backupHost—Optional (**Note:** If you do not specify the backupHost in the XML body during a PUT but it was configured at an earlier time, the PUT operation resets the value to blank.)
- port—Required
- databaseName—Required
- domain—Required
- username—Required
- password—Required

## 4.4. Layout Configuration APIs

[contents]

The LayoutConfig object is a container element that enables an administrator to customize the layout of the Finesse Desktop by uploading an XML file.

The LayoutConfig object is structured as follows:

```
<LayoutConfig>
   <uri>/finesseconfig/api/LayoutConfig/default</uri>
   <layoutxml><?xml version="1.0" encoding="UTF-8"?>
      <finesseLayout xmlns="http://www.cisco.com/vtg/finesse">
        <layout>
        <role>Agent</role>
        <page>
                <gadget>http://localhost/desktop/gadgets/CallControl.xml</gadget>
        </page>
                <tabs>
                        <tab>
                                <id>home</id>
                                <label>Home</label>
                        </tab>
                        <tab>
                                <id>manageCall</id>
                                <label>Manage Call</label>
                        </tab>
                </tabs>
        </layout>
```

```
        <layout>
        <role>Supervisor</role>
        <page>
                <gadget>http://localhost/desktop/gadgets/CallControl.xml</gadget>
        </page>
                <tabs>
                        <tab>
                                <id>home</id>
                                <label>Home</label>
                                <gadgets>
                                        <gadget>http://localhost/desktop/gadgets/TeamPerformance.xml</gadget>
                                </gadgets>
                        </tab>
                        <tab>
                                <id>manageCall</id>
                                <label>Manage Call</label>
                        </tab>
                </tabs>
        </layout>
    </finesseLayout>
</layoutxml>
</LayoutConfig>
```

- LayoutConfig—Get
- LayoutConfig—Set

## 4.4.1. LayoutConfig—Get

[contents]

The LayoutConfig—Get API allows a user to get a copy of the default LayoutConfig object.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/LayoutConfig/default |
| **Example URI:** | `http://host/finesseconfig/api/LayoutConfig/default` |
| **Security Constraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to get a copy of the LayoutConfig object. |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:** | `<LayoutConfig>`<br>`<uri>/finesseconfig/api/LayoutConfig/default/</uri>`<br>`<layoutxml><?xml version="1.0" encoding="UTF-8"?>`<br>`<finesseLayout xmlns="http://www.cisco.com/vtg/finesse">`<br>`<layout>` |

```
        <role>Agent</role>
        <page>
                <gadget>http://localhost/desktop/gadgets/CallControl.xml</gadget>
        </page>
                <tabs>
                        <tab>
                                <id>home</id>
                                <label>Home</label>
                        </tab>
                        <tab>
                                ...
                        </tab>
                </tabs>
        </layout>
        <layout>
        <role>Supervisor</role>
        <page>
                <gadget>http://localhost/desktop/gadgets/CallControl.xml</gadget>
        </page>
                <tabs>
                        ...
                </tabs>
        </layout>
    </finesseLayout>
  </layoutxml>
</LayoutConfig>
```

| | |
|---|---|
| **HTTP Response:** | 200—Success<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>500—Internal server error |
| **Failure Response Example:** | `<ApiErrors>`<br>`  <ApiError>`<br>`  <ErrorType>Authorization Failure</ErrorType>`<br>`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`<br>`  <ErrorData>jsmith</ErrorData>`<br>`  </ApiError>`<br>`</ApiErrors>` |
| **Error Codes:** | • Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

layoutxml

## 4.4.2. LayoutConfig—Set

[contents]

The LayoutConfig—Set API allows an administrator to update the default layout setting for the Finesse Desktop.

**Note:** The XML data is verified to ensure it is valid XML syntax and that it conforms to the Finesse schema.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/LayoutConfig/default |

| | |
|---|---|
| **Example URI:** | `http://host/finesseconfig/api/LayoutConfig/default` |
| **Security Constraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to update the LayoutConfig object. |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | <LayoutConfig><br><layoutxml><?xml version="1.0" encoding="UTF-8"?><br><finesseLayout xmlns="http://www.cisco.com/vtg/finesse"><br><layout><br><role>Agent</role><br><page><br><gadget>http://localhost/desktop/gadgets/CallControl.xml</gadget><br></page><br><tabs><br><tab><br><id>home</id><br><label>Home</label><br><gadgets><br><gadget>http://localhost/desktop/gadgets/HelloWorld.xml</gadget><br><gadget>http://localhost/desktop/gadgets/GoodByeWorld.xml</gadget><br></gadgets><br></tab><br><tab><br>...<br></tab><br></tabs><br></layout><br><layout><br><role>Supervisor</role><br><page><br><gadget>http://localhost/desktop/gadgets/CallControl.xml</gadget><br></page><br><tabs><br>...<br></tabs><br></layout><br></finesseLayout><br></layoutxml><br></LayoutConfig> |
| **HTTP Response:** | 200—Success (the new settings were successfully written to the database)<br>400—Parameter missing (the XML file was not provided)<br>400—Invalid input (the submitted XML is invalid or does not conform to the Finesse layout schema)<br>401—Authorization failure (for example, the user is not yet authenticated in the web session)<br>500—Internal server error |

| | |
|---|---|
| **Failure Response Example:** | <ApiErrors><br>  <ApiError><br>  <ErrorType>Invalid Input</ErrorType><br>  <ErrorMessage>layoutxml</ErrorMessage><br>  </ApiError><br></ApiErrors> |
| **Error Codes** | • Parameter Missing<br>• Invalid Input<br>• Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

**Request Parameters**
layoutxml—Required

## 4.5. Reason Code APIs

[contents]

The ReasonCode object represents a reason code that can be applied when an agent is changing state. There are two categories of reason codes: Not Ready reason codes and Logout reason codes. The ReasonCode APIs are for administrator operations.

The structure of a ReasonCode object is as follows:

```
<ReasonCode>
    <uri>/finesseconfig/api/ReasonCode/{id}</uri>
    <category>NOT_READY</category>
    <code>10</code>
    <label>Team Meeting</label>
    <forAll>true</forAll>
</ReasonCode>
```

**Note:** If you provide two or more duplicate tags in the XML body for a POST or PUT operation, the value of the last duplicate tag is processed and all other duplicate tags are ignored.

Administrators can create, edit, or delete not ready and sign out reason codes using either the reason code APIs or the Finesse Administration Console. Not ready reason codes can be configured using the Not Ready Reason Code Management gadget in the Administration Console or using the reason code APIs, with the category set to NOT_READY. Similarly, sign out reason codes can be configured using the Sign Out Reason Code Management gadget in the Administration Console or using the reason code APIs with the category set to LOGOUT.

- ReasonCode—Get
- ReasonCode—Get List
- ReasonCode—Create
- ReasonCode—Update
- ReasonCode—Delete

### 4.5.1. ReasonCode—Get

[contents]

The ReasonCode—Get API allows the user to retrieve a full ReasonCode object.

| | |
|---|---|

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/ReasonCode/<id> |
| **Example URI:** | `http://host/finesseconfig/api/ReasonCode/476` |
| **Security Constraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to get a reason code. |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:** | `<ReasonCode>`<br>`  <uri>/finesseconfig/api/ReasonCode/{id}</uri>`<br>`  <category>NOT_READY</category>`<br>`  <code>10</code>`<br>`  <label>Team Meeting</label>`<br>`  <forAll>true</forAll>`<br>`</ReasonCode>` |
| **HTTP Response:** | 200—Success<br>400—Bad request<br>400—Finesse API error (for example, the object does not exist, the object is stale, violation of DB constraint, and so on)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>404—Not Found (the resource cannot be found, for example, it might have been deleted)<br>500—Internal server error |
| **Failure Response Example:** | `<ApiErrors>`<br>`  <ApiError>`<br>`  <ErrorType>Authorization Failure</ErrorType>`<br>`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`<br>`  <ErrorData>jsmith</ErrorData>`<br>`  </ApiError>`<br>`</ApiErrors>` |

| | |
|---|---|
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- category
- code
- label
- forAll

## 4.5.2. ReasonCode—Get List

[contents]

The ReasonCode—Get List API allows an administrator to get a list of Not Ready or Logout reason codes. The required URL parameter *category* specifies whether to retrieve Logout reason codes or Not Ready reason codes. If this URL parameter is missing, the API returns an error.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/ReasonCodes?category=NOT_READY\|LOGOUT |
| **Example URI:** | `http://host/finesseconfig/api/ReasonCodes?category=NOT_READY` |
| **Security Constraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to get a list of reason codes. |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:**<br>— | `<ReasonCodes>`<br>`  <category>NOT_READY</category>`<br>`  <ReasonCode>`<br>`  ... Full ReasonCode Object ...`<br>`  </ReasonCode>` |

| | |
|---|---|
| | <ReasonCode><br>... Full ReasonCode Object ...<br></ReasonCode><br><ReasonCode><br>... Full ReasonCode Object ...<br></ReasonCode><br></ReasonCodes> |
| **HTTP Response:** | 200—Success<br>400—Bad request<br>400—Finesse API error (for example, the object does not exist, the object is stale, violation of DB constraint, and so on)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### URL Request Parameter

category—Required

### Response Parameters

- ReasonCodes
- category
- ReasonCode

## 4.5.3. ReasonCode—Create

[contents]

The ReasonCode—Create API allows an administrator to create a new reason code. The administrator specifies the category, code, label, and forAll attributes for the reason code.

**Note:** The forAll parameter determines whether a reason code is global (true) or non-global (false). Cisco Finesse Release 8.5(3) only supports global reason codes. Non-global reason codes are not supported in this release. If an administrator uses this API to create a reason code and sets the forAll parameter to false, that reason code will not appear in the Agent Desktop or the Administrative Console.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/ReasonCode/ |
| **Example URI:** | `http://host/finesseconfig/api/ReasonCode/` |
| **Security Constraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to create a reason code. |
| **HTTP Method:** | POST |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | `<ReasonCode>`<br>`  <category>NOT_READY</category>`<br>`  <code>24</code>`<br>`  <label>Lunch Break</label>`<br>`  <forAll>true</forAll>`<br>`</ReasonCode>` |
| **HTTP Response:** | 200—Success; the Finesse server has successfully created the new ReasonCode. The response contains an empty response body, and a "location:" header denoting the absolute URL of the newly created ReasonCode object<br>400—Bad request<br>400—Finesse API error (for example, the reason code already exists)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>500—Internal server error |
| **Failure Response Example:** | `<ApiErrors>`<br>`  <ApiError>`<br>`  <ErrorType>Authorization Failure</ErrorType>`<br>`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`<br>`  <ErrorData>jsmith</ErrorData>`<br>`  </ApiError>`<br>`</ApiErrors>` |
| **Error Codes:** | • Authorization Failure |

- Invalid Authorization User Specified
- Internal Server Error

For descriptions and other possible error codes, see API Error Codes.

### Request Parameters

- category—Required
- code—Required
- label—Required
- forAll—Required

## 4.5.4. ReasonCode—Update

[contents]

The ReasonCode—Update API allows an administrator to modify an existing reason code. The administrator specifies an existing reason code and category, along with the value of the field to update.

**Note:** The forAll parameter determines whether a reason code is global (true) or non-global (false). Cisco Finesse Release 8.5(3) only supports global reason codes. Non-global reason codes are not supported in this release. If an administrator uses this API to update a reason code and sets the forAll parameter to false, that reason code will not appear in the Agent Desktop or in the Administrative Console.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/ReasonCode/<id> |
| **Example URI:** | `http://host/finesseconfig/api/ReasonCode/476` |
| **Security Restraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to update a reason code. |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | `<ReasonCode>`<br>`  <code>1001</code>`<br>`  <label>Lunch break</label>`<br>`  <forAll>true</forAll>`<br>`</ReasonCode>` |

| | |
|---|---|
| **HTTP Response:** | 200—Success (The Finesse server successfully updated the reason code)<br>400—Bad request<br>400—Finesse API error (for example, duplicate reason code)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>404—Not found<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Request Parameters

- code
- label
- forAll

**Note:** You do not need to include the attributes (code, label, or forAll) that you do not need to change. For example, if you want to change only the label for an existing reason code from "In Meeting" to "Attend Meeting", you can send the following request:

```
<ReasonCode>
   <label>Attend Meeting</label>
</ReasonCode>
```

## 4.5.5. ReasonCode—Delete

[contents]

The ReasonCode—Delete API allows an administrator to delete an existing reason code.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/ReasonCode/<id> |
| **Example URI:** | http://host/finesseconfig/api/ReasonCode/4235 |

| | |
|---|---|
| **Security Constraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to delete a reason code. |
| **HTTP Method:** | DELETE |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **HTTP Response:** | 200—Success (The Finesse server successfully deleted the specified reason code)<br>400—Bad request<br>400—Finesse API error (for example, the database operation failed due to a table being locked)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

## 4.6. Wrap-Up Reason APIs

[contents]

The WrapUpReason object represents the reason that an agent can apply to a call during call wrap-up.

The WrapUpReason object is structured as follows:

```
<WrapUpReason>
    <uri>/finesseconfig/api/WrapUpReason/{id}</uri>
    <label>Issue/Complaint</label>
    <forAll>true</forAll>
```

```
</WrapUpReason>
```

**Note:** If you provide two or more duplicate tags in the XML body for a POST or PUT operation, the value of the last duplicate tag is processed and all other duplicate tags are ignored.



## 4.6.1. WrapUpReason—Get

[contents]

The WrapUpReason—Get API allows a user to retrieve a WrapUpReason object.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/WrapUpReason/<id> |
| **Example URI:** | `http://host/finesseconfig/api/WrapUpReason/31` |
| **Security Restraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to get a wrap-up reason. |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:** | `<WrapUpReason>`<br>`  <uri>/finesseconfig/api/WrapUpReason/{id}</uri>`<br>`  <label>Product Question</label>`<br>`  <forAll>true</forAll>`<br>`</WrapUpReason>` |
| **HTTP Response:** | 200—Success<br>400—Bad request (the request body is invalid) |

| | |
|---|---|
| | 400—Finesse API error (for example, object does not exist or is stale)<br>401—Authorization failure<br>401—Invalid Authorization User Specified (for example, an authenticated user tried to use the identity of another user)<br>404—Not found (for example, the wrap-up reason was deleted)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- uri
- label
- forAll

## 4.6.2. WrapUpReason—Get List

[contents]

The WrapUpReason—Get List API allows an administrator to retrieve a list of WrapUpReason objects.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/WrapUpReasons |
| **Example URI:** | http://host/finesseconfig/api/WrapUpReasons |
| **Security Constraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to get a list of wrap-up reasons. |
| **HTTP Method:** | GET |
| | |

| | |
|---|---|
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **Successful Response:** | <WrapUpReasons><br>  <WrapUpReason><br>  ... Full WrapUpReason Object ...<br>  </WrapUpReason><br>  <WrapUpReason><br>  ... Full WrapUpReason Object ...<br>  </WrapUpReason><br>  <WrapUpReason><br>  ... Full WrapUpReason Object ...<br>  </WrapUpReason><br></WrapUpReasons> |
| **HTTP Response:** | 200—Success<br>400—Bad request (the request body is invalid)<br>400—Finesse API error (for example, object does not exist or is stale)<br>401—Authorization failure<br>401—Invalid Authorization User Specified (for example, an authenticated user tried to use the identity of another user)<br>404—Not found (for example, the wrap-up reason was deleted)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br>  <ApiError><br>  <ErrorType>Authorization Failure</ErrorType><br>  <ErrorMessage>UNAUTHORIZED</ErrorMessage><br>  <ErrorData>jsmith</ErrorData><br>  </ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- uri
- label

- forAll

### 4.6.3. WrapUpReason—Create

[contents]

The WrapUpReason—Create API allows an administrator to create a new wrap-up reason. The administrator specifies the label and forAll attributes for the wrap-up reason.

**Note:** The forAll parameter determines whether a wrap-up reason is global (true) or non-global (false). Cisco Finesse Release 8.5(3) only supports global wrap-up reasons. Non-global wrap-up reasons are not supported in this release. If an administrator uses this API to create a wrap-up reason and sets the forAll parameter to false, that wrap-up reason will not appear in the Agent Desktop or the Administrative Console.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/WrapUpReason/ |
| **Example URI:** | `http://host/finesseconfig/api/WrapUpReason/` |
| **Security Constraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to create a wrap-up reason. |
| **HTTP Method:** | POST |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | `<WrapUpReason>`<br>`  <label>Recommendation</label>`<br>`  <forAll>true</forAll>`<br>`</WrapUpReason>` |
| **HTTP Response:** | 200—Success<br>The Finesse server successfully created the new wrap-up reason. The response contains an empty response body, and a "location:" header denoting the absolute URL of the newly created WrapUpReason object.<br>400—Bad request (for example, one of the required parameters was not provided or is invalid)<br>400—Finesse API error (for example, the wrap-up reason already exists)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>500—Internal server error |

| | |
|---|---|
| **Failure Response Example:** | <ApiErrors><br><ApiError><br><ErrorType>Authorization Failure</ErrorType><br><ErrorMessage>UNAUTHORIZED</ErrorMessage><br><ErrorData>jsmith</ErrorData><br></ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Request Parameters

- label—Required
- forAll—Required

## 4.6.4. WrapUpReason—Update

[contents]

The WrapUpReason—Update API allows an administrator to modify an existing wrap-up reason. The administrator references an existing wrap-up reason by its ID and specifies the values of the fields to update.

**Note:** The forAll parameter determines whether a wrap-up reason is global (true) or non-global (false). Cisco Finesse Release 8.5(3) only supports global wrap-up reasons. Non-global wrap-up reasons are not supported in this release. If an administrator uses this API to update a wrap-up reason and sets the forAll parameter to false, that wrap-up reason will not appear in the Agent Desktop or the Administrative Console.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/WrapUpReason/<id> |
| **Example URI:** | http://host/finesseconfig/api/WrapUpReason/23 |
| **Security Restraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to update a wrap-up reason. |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |

| | |
|---|---|
| **HTTP Request:** | <WrapUpReason><br>  <label>Sales call</label><br>  <forAll>true</forAll><br></WrapUpReason> |
| **HTTP Response:** | 200—Success (The Finesse server successfully updated the wrap-up reason)<br>400—Bad request (if a required parameter is not provided or is invalid)<br>400—Finesse API error (for example, duplicate wrap-up reason or wrap-up reason does not exist)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>404—Not found (for example, the wrap-up reason was deleted)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br>  <ApiError><br>  <ErrorType>Authorization Failure</ErrorType><br>  <ErrorMessage>UNAUTHORIZED</ErrorMessage><br>  <ErrorData>jsmith</ErrorData><br>  </ApiError><br></ApiErrors> |
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Request Parameters

- label
- forAll

**Note:** You do not need to include the attributes (label or forAll) that you do not need to change. For example, if you want to change only the label for an existing wrap-up reason from "Wrong Number" to "Wrong Department", you can send the following request:

```
<WrapUpReason>
  <label>Wrong Department</label>
</WrapUpReason>
```

## 4.6.5. WrapUpReason—Delete

[contents]

The WrapUpReason—Delete API allows an administrator to delete an existing wrap-up reason. The administrator references an existing WrapUpReason object by its ID.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/WrapUpReason/<id> |

| | |
|---|---|
| **Example URI:** | `http://host/finesseconfig/api/WrapUpReason/475` |
| **Security Constraints:** | Role—Administrator<br>Limitations—A user must be signed in as an administrator to delete a wrap-up reason. |
| **HTTP Method:** | DELETE |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
| **HTTP Response:** | 200—Success (The Finesse server successfully deleted the specified wrap-up reason)<br>400—Bad request<br>400—Finesse API error (for example, the database operation failed due to a table being locked)<br>401—Authorization failure<br>401—Invalid Authorization User Specified<br>404—Not Found (for example, the wrap-up reason was deleted)<br>500—Internal server error |
| **Failure Response Example:** | `<ApiErrors>`<br>`  <ApiError>`<br>`  <ErrorType>Authorization Failure</ErrorType>`<br>`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`<br>`  <ErrorData>jsmith</ErrorData>`<br>`  </ApiError>`<br>`</ApiErrors>` |
| **Error Codes:** | • Authorization Failure<br>• Invalid Authorization User Specified<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

[contents]

## 4.7. Media Properties Layout APIs

The MediaPropertiesLayout object represents the appearance of media properties carried in the dialog objects in the call control gadget on the agent or supervisor desktop. Administrators can use these APIs to customize the layout of media properties.

The MediaPropertiesLayout is structured as follows:

```
<MediaPropertiesLayout>
       <header>
              <entry>
                     <displayName>Customer Name</displayName>
                     <mediaProperty>callVariable1</mediaProperty>
              </entry>
       </header>
       <column>
              <entry>
                     <displayName>Customer Name</displayName>
                     <mediaProperty>callVariable1</mediaProperty>
              </entry>
              <entry>
                     <displayName>Customer Acct#</displayName>
                     <mediaProperty>user.cisco.acctnum</mediaProperty>
              </entry>
       </column>
       <column>
              <entry>
                     <displayName>Support contract</displayName>
                     <mediaProperty>callVariable2</mediaProperty>
              </entry>
              <entry>
                     <displayName>Product calling about</displayName>
                     <mediaProperty>callVariable3</mediaProperty>
              </entry>
       </column>
</MediaPropertiesLayout>
```

**Note:** Finesse Release 8.5(3) supports the following media properties: call variables (up to 10) and ECC variables.

- MediaPropertiesLayout—Get
- MediaPropertiesLayout—Set

### 4.7.1. MediaPropertiesLayout—Get

[contents]

The MediaPropertiesLayout—Get API allows a user to get a copy of the default MediaPropertiesLayout object.

Finesse Release 8.5(3) supports only a single default instance.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/MediaPropertiesLayout/default |
| **Example URI:** | `http://host/finesseconfig/api/MediaPropertiesLayout/default` |
| **HTTP Method:** | GET |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |

| | |
|---|---|
| **HTTP Request:** | — |
| **Successful Response:** | `<MediaPropertiesLayout>`<br>`... Full MediaPropertiesLayout Object ...`<br>`</MediaPropertiesLayout>` |
| **HTTP Response:** | 200—Success<br>401—Unauthorized (for example, the user is not authenticated in the Web Session)<br>500—Internal server error |
| **Failure Response Example:** | `<ApiErrors>`<br>`  <ApiError>`<br>`  <ErrorType>Authorization Failure</ErrorType>`<br>`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`<br>`  <ErrorData>jsmith</ErrorData>`<br>`  </ApiError>`<br>`</ApiErrors>` |
| **Error Codes:** | • Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Response Parameters

- header
- column
- entry
- displayName
- mediaProperty

## 4.7.2. MediaPropertiesLayout—Set

[contents]

The MediaPropertiesLayout—Set API allows an administrator to configure the default call variable layout.

| | |
|---|---|
| **URI:** | http://<server>/finesseconfig/api/MediaPropertiesLayout/default |
| **Example URI:** | `http://host/finesseconfig/api/MediaPropertiesLayout/default` |

| | |
|---|---|
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | <MediaPropertiesLayout><br>  <header><br>  <entry><br>  <displayName>Customer Name</displayName><br>  <mediaProperty>callVariable1</mediaProperty><br>  </entry><br>  </header><br>  <column><br>  <entry><br>  <displayName>Customer Name</displayName><br>  <mediaProperty>callVariable1</mediaProperty><br>  </entry><br>  <entry><br>  <displayName>Customer Acct#</displayName><br>  <mediaProperty>user.cisco.acctnum</mediaProperty><br>  </entry><br>  </column><br>  <column><br>  <entry><br>  <displayName>Support contract</displayName><br>  <mediaProperty>callVariable2</mediaProperty><br>  </entry><br>  <entry><br>  <displayName>Product calling about</displayName><br>  <mediaProperty>callVariable3</mediaProperty><br>  </entry><br>  </column><br></MediaPropertiesLayout> |
| **HTTP Response:** | 200—Success (the new settings were successfully written to the database)<br>400—Parameter missing (at least one of the required parameters was not provided)<br>400—Invalid input (at least one of the parameters provided is not valid)<br>401—Authorization failure (for example, the user is not yet authenticated in the Web Session)<br>500—Internal server error |
| **Failure Response Example:** | <ApiErrors><br>  <ApiError><br>  <ErrorData>The entry contained an invalid media property: callVariable11</ErrorData><br>  <ErrorType>Invalid Input</ErrorType><br>  <ErrorMessage>HTTP Status code: 400 (Bad Request)<br>  Api Error Type: Invalid Input<br>  Error Message: Invalid media property name 'callVariable11'<br>  </ErrorMessage><br>  </ApiError><br></ApiErrors> |
| **Error Codes** | • Parameter Missing |

| | |
|---|---|
| | • Invalid Input<br>• Authorization Failure<br>• Internal Server Error<br>For descriptions and other possible error codes, see API Error Codes. |

### Request Parameters

- header—Optional
- column—Optional
- entry—Optional
- displayName—Required
- mediaProperty—Required

# 5. API Parameter Reference

[contents]



## 5.1. Parameter Types and Data Types

[contents]

The tables in the API Request and Response Parameter sections include a column named **Parameter Type/Data Type**.

### 5.1.1. Parameter Types

[contents]



#### Body Parameter

A body parameter (also known as a complex parameter) appears in the body of the message. Body parameters are used in the Set Call Data API only. In this example, the callVariables and the callerEnteredDigits are body parameters:

```
{
    "call": {
       "data": {
           "callVariable1": "X",
           "callVariable2": "Y",
           "callVariable3": "Z",
           "callerEnteredDigits": "765747",
```

#### Path Parameter

A path parameter is included in the path of the URI. In this example, *callId* is a path parameter:

```
http://host:80/webservices/CallService/Call/<callId>
/consultCall?dialedNumber=1002&consultType=1
```

### Query Parameter

Query parameters are passed in a query string on the end of the URI you are calling and are preceded by a question mark. Multiple query parameters are connected by an ampersand. In this URI, extension and forcedFlag are query parameters.

```
http://host:80/webservices/ConnectionService/Connection/
signIn?extension=1012&forcedFlag=1
```

## 5.1.2. Data Types

[contents]

Data types used in API parameters and event message fields are listed in the following table:

| Data Type | Description |
|---|---|
| Boolean | A logical data type that has one of two values: true or false. |
| Integer | A 32-bit wide integer. |
| Long | A long integer that represents a whole number whose range is greater than or equal to that of a standard integer. |
| String | A variable-length string variable. If a maximum length exists, it is listed with the parameter description. |
| Time | Date and Time |

## 5.2. API Header Parameters

[contents]

| Name | Parameter Type / Data Type | Description | Used as a Request Parameter by |
|---|---|---|---|
| **password** | Path \| String | The password that allows a user to sign in | User Sign In to Finesse |
| **username** | Path \| String | The name that allows a user to sign in | User Sign In to Finesse |

## 5.3. API Request Parameters

[contents]

| A - B| C | D - F | H - L | M - P | R - S | U - W |

| Name | Parameter Type / Data Type | Description | Used as a Request Parameter by |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **backupHost** | Body \| String | The hostname or IP address of the backup AWDB server<br>Validation: No special characters allowed except "." and "-" | EnterpriseDatabaseConfig - Set |
| **callvariables** | Collection | A list of call variables to be modified<br>Validation: Either wrapUpReason or callvariables must be present. Cannot be missing or empty | Dialog - Update Call Variable Data |
| **CallVariable** | XML | Contains the name and value of a call variable belonging to this particular dialog that needs to be updated<br>Size:<br>• CallVariable - 40 bytes<br>• ECC/Named Variables - Sum of all Names, Values and index(if array) <= 2000 bytes. Each ECC variable value cannot exceed the length defined by CTI Admin<br>Validation:<br>• There should be at least one call variable to be modified<br>• The name must be present and not empty<br>• There must be no duplicate names<br>• The Value tag must be specified (but it can be empty)<br>• Call variables must match up to one of the pre-defined call variable names (for example, callVariable1...callVariable10)<br>• For ECC/Named Variables:<br>◦ The variable starts with the prefix "user"<br>◦ If the variable is a Named Array, the variable should end with a number that is enclosed in square brackets. That is, an opening bracket '[' followed by a number, followed by a closing bracket ']' (for example, *user.myarray[2]*). If the variable includes the square brackets in any other order, it will not be considered to be a named array variable but will instead be considered as a named scalar variable, and will be allowed to be sent to the CTI server. For example, the variable *user.my[array]* would be considered to be a named scalar variable<br>◦ ECC Variable names should match those defined in the CTI Server administration UI, in name and in length<br>◦ All ECC Variable validations as applied in a CTI server are applicable, but, these validations are performed by the CTI server | Dialog - Update Call Variable Data |

| | | | |
|---|---|---|---|
| | | that receives these ECC Variables<br>Finesse Release 8.5(3) only supports Latin1 characters for ECC variables. Other Unicode characters are not supported. | |
| **category** | Query \| String | The category of the reason code.<br>Possible Values: NOT_READY, LOGOUT<br>Validation: The combination of category, code, and label must be unique. | ReasonCode—Get List<br>ReasonCode - Create<br>User—Get ReasonCode List |
| **code** | Integer | The value of the reason code.<br>Possible values: 0–65535<br>Validation: The combination of category, code, and label for a reason code must be unique. | ReasonCode - Create<br>ReasonCode - Update |
| **column** | List of entry objects | Grouping of media properties for agent and supervisor desktops.<br>Validation: Valid entry tags<br>Finesse Release 8.5(3) supports a maximum of two columns in the MediaPropertiesLayout object. Columns can contain a maximum of 10 entries and can be empty.<br>The first column specified in the PUT request appears on the left of the call control gadget on the desktop. The second column appears on the right. | MediaPropertiesLayout - Set |
| **cti - backupHost** | Body \| String | The hostname or IP address of the backup CTI server (must not be the same as the hostname or IP address of the primary CTI server)<br>Validation: No special characters allowed except "." and "-" | SystemConfig - Set |
| **cti - backupPort** | Body \| Integer | The port number of the backup CTI server<br>Validation: must be between 1 and 65535 | SystemConfig - Set |
| **cti - host** | Body \| String | The hostname or IP address of the primary CTI server<br>Default value: localhost<br>Validation: No special characters allowed except "." and "-" | SystemConfig - Set |
| **cti - peripheralId** | Body \| Integer | The ID of the CTI server peripheral<br>Default value: 5000<br>Validation: must be between 1 and 32767 | SystemConfig - Set |
| **cti - port** | Body \| Integer | The port number of the primary CTI server<br>Default value: 42027 | SystemConfig - Set |

| | | | |
|---|---|---|---|
| | | Validation: must be between 1 and 65535 | |
| **databaseName** | Body \| String | The name of the AWDB | EnterpriseDatabaseConfig - Set |
| **displayName** | String | Part of an entry. A label that describes the mediaProperty for that entry (for example, Account Number) that appears on the agent or supervisor desktop.<br>Each entry must have exactly one display name. HTML tags are rendered as plain text and do not retain formatting.<br>Validation: Any valid string (including an empty string). The maximum length of a displayName is 50 characters. | MediaPropertiesLayout - Set |
| **domain** | Body \| String | The domain of the AWDB | EnterpriseDatabaseConfig - Set |
| **entry** | Entry object containing a name and media property | A displayName and mediaProperty combination.<br>Validation: Each entry must have exactly one displayName and one mediaProperty. The displayName can be empty. | MediaPropertiesLayout - Set |
| **extension** | Body \| Integer | The extension with which the user wants to sign in<br>Type: number (maximum of 16 bytes)<br>Validation: the extension exists in Cisco Unified CM | User Sign In to Finesse |
| **forAll** | Boolean | The access range of the reason code or wrap-up reason.<br>Validation: must be true or false<br>Only global reason codes and wrap-up reasons (forAll parameter is set to "true") are supported for Finesse Release 8.5(3). | ReasonCode - Create<br>ReasonCode - Update<br>WrapUPReason - Create<br>WrapUpReason - Update |
| **fromAddress** | String | Used for matching the User media device. This value is the same as the extension the user is currently signed into.<br>For silent monitoring, this value represents the extension of the supervisor who initiated the silent monitoring call. | Dialog - Create a New Dialog<br>Dialog - Make a Silent Monitoring Call |
| **header** | Entry object | A single entry (combination of displayName and mediaProperty) that appears in the call header on the desktop for each call. | MediaPropertiesLayout - Set |

| | | | |
|---|---|---|---|
| **host** | Body \| String | The hostname or IP address of the AWDB server<br>Validation: No special characters allowed except "." and "-" | EnterpriseDatabaseConfig - Set |
| **id** | Path \| String | The ID of the user (maximum of 12 characters)<br>Validation: The user is configured in Unified CCE | User Sign In to Finesse<br>User Sign Out of Finesse<br>User - Change Agent State<br>User—Change Agent State (Pass NotReady or Logout Corresponding ReasonCode to CTI) |
| **label** | String | The reason code or wrap-up reason label<br>Validation for WrapUpReason APIs: The label must be unique. The label cannot be longer than 39 bytes (which is equal to 39 US English characters).<br>Validation for ReasonCode APIs: The combination of category, code, and label must be unique. The label cannot be longer than 40 characters. | ReasonCode - Create<br>ReasonCode - Update<br>WrapUpReason - Create<br>WrapUpReason - Update |
| **layoutxml** | String | The XML data that determines the layout of the Finesse desktop<br>Validation: Must be valid XML and conform with the Finesse schema | LayoutConfig - Set |
| **mediaProperties** | XML | Collection of media-specific properties related to the Dialog that need to be modified<br>Validation: Cannot be missing or empty | Dialog - Update Call Variable Data |
| **mediaProperty** | String | Part of an entry. The name of the call variable or ECC variable that is displayed to the agent or supervisor.<br>Validation: Each entry must have exactly one mediaProperty. Allowable strings are callVariable1 to callVariable10, or a valid ECC variable. The maximum length is 32 characters. | MediaPropertiesLayout - Set |
| **password** | Body \| String | The password required to sign into the database | EnterpriseDatabaseConfig - Set |
| **port** | Body \| Integer | The port of the AWDB server<br>Validation: must be between 1 and 65535 | EnterpriseDatabaseConfig - Set |
| **ReasonCodeID** | String | An empty string for no reason code. Otherwise, a database id for the ReasonCode<br>Validation: Only for a state change to Not_Ready or Logout; the value must | User—Change Agent State (Pass NotReady or Logout Corresponding ReasonCode to CTI) |

| | | | |
|---|---|---|---|
| | | correspond to a database id. Logout does *not* prevent state change with an invalid ReasonCodeId. | |
| **requestedAction** | Body \| String | The action to take on the targeted participant<br>Possible values: ANSWER, HOLD, RETRIEVE, DROP, TRANSFER, CONFERENCE, SILENT_MONITOR<br>Validation: Only actions that currently exist on a participant can be used | Dialog - Take Action on a Participant within a Dialog<br>Dialog - Make a Silent Monitoring Call<br>Dialog - End a Silent Monitoring Call |
| **requestedAction (Create a new Dialog)** | String | The way in which the Dialog is created<br>Possible values: MAKE_CALL | Dialog - Create a New Dialog |
| **requestedAction (Make a consult call request)** | String | The way in which the Dialog is created<br>Possible values: CONSULT_CALL | Dialog - Make a Consult Call Request |
| **requestedAction (Update Call Variable Data)** | String | The action to take on the dialog<br>Possible values: UPDATE_CALL_DATA | Dialog - Update Call Variable Data |
| **secondaryNode - host** | Path \| String | The hostname or IP address of the secondary Finesse node<br>Validation: No special characters allowed except "." and "-" | ClusterConfig - Set |
| **state** | Body \| String | The new state that the user wants to be in<br>Possible values: LOGIN, LOGOUT, READY, NOT_READY | User Sign In to Finesse<br>User Sign Out of Finesse<br>User - Change Agent State<br>User—Change Agent State (Pass NotReady or Logout Corresponding ReasonCode to CTI) |
| **targetMediaAddress** | Body \| String | The extension that the user is currently signed in to, which is used to locate the participant to target with the action request<br>For silent monitoring, this value represents the extension of the supervisor who initiated the silent monitoring call. | Dialog - Take Action on a Participant within a Dialog<br>Dialog - Make a Consult Call Request<br>Dialog - End a Silent Monitoring Call |

| | | | |
|---|---|---|---|
| **toAddress** | String | The destination for the call<br>In a silent monitoring call, the toAddress is the agent's extension. | Dialog - Create a New Dialog<br>Dialog - Make a Consult Call Request<br>Dialog - Make a Silent Monitoring Call |
| **username** | Body \| String | The username required to sign into the database | EnterpriseDatabaseConfig - Set |
| **wrapUpReason** | String | A description of the call<br>Size: 39 bytes (which is equal to 39 US English characters)<br>Validation: Either wrapUpReason or callvariables must be present | Dialog - Update Call Variable Data |

## 5.4. API Response Parameters

[contents]

| A - B | C | D - E | F - H | I - L | M - O | P - R | S | T- V | W - X

| Name | Parameter Type / Data Type | Description | Used as a Response Parameter by |
|---|---|---|---|
| **Actions** | XML | A list of actions that will be allowed for the participant as a result of the dialog update<br>For a list of possible values, see the Actions parameter values table | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| **backupHost** | String | The hostname or IP address of the backup AWDB server | EnterpriseDatabaseConfig - Get |
| **callType** | String | The type of call.<br>Possible values: ACD_IN, PREROUTE_ACD_IN,PREROUTE_DIRECT_AGENT,TRANSFER, OTHER_IN, OUT,AGENT_INSIDE, CONSULT, CONFERENCE, SUPERVISOR_MONITOR | Dialog - Get Dialog |
| **callvariables** | Collection | A list of up to ten call and ECC variables | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |

| | | | |
|---|---|---|---|
| **CallVariable** | XML | Contains the name and value of a call variable belonging to this particular Dialog. The name indicates whether the variable is a Call variable or an ECC variable, based on naming conventions.<br>Call variable names start with callVariable# where # is 1-10. All ECC variable names (both scalar and array) are prepended with "user". ECC variable arrays include an index enclosed within square brackets located at the end of the ECC array name. | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| **category** | String | The category of a ReasonCode.<br>Possible Values: NOT_READY, LOGOUT | ReasonCode—Get List<br>ReasonCode—Get<br>User—Get ReasonCode<br>User—Get ReasonCode List |
| **code** | Integer | The value of the reason code.<br>Possible values: 0–65535<br>Validation: The combination of category, code, and label for a reason code must be unique. | ReasonCode—Get |
| **column** | List of entry objects | Grouping of media properties for agent and supervisor desktops.<br>Finesse Release 8.5(3) supports a maximum of two columns in the MediaPropertiesLayout object. Columns can contain a maximum of 10 entries and can be empty. The first column supplied is always the left column. The second column (if any) is always the right column. | MediaPropertiesLayout - Get |
| **cti - backupHost** | String | The hostname or IP address of the backup CTI server | SystemConfig - Get |
| **cti - backupPort** | Integer | The port number of the backup CTI server<br>Possible values: 1—65535 | SystemConfig - Get |
| **cti - host** | String | The hostname or IP address of the primary CTI server | SystemConfig - Get |
| **cti - peripheralId** | Integer | The ID of the CTI server peripheral<br>Possible values: 1–32767 | SystemConfig - Get |
| **cti - port** | Integer | The port number of the primary CTI server<br>Possible values: 1–65535 | SystemConfig - Get |
| **databaseName** | String | The name of the AW database | EnterpriseDatabaseConfig |

| | | | |
|---|---|---|---|
| | | | - Get |
| **dialedNumber** | String | The number dialed | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| **dialogs** | String | The URI for the list of dialogs in which the user is a participant | User - Get User<br>User - Get List of Users |
| **displayName** | String | Part of an entry. A label that describes the mediaProperty for that entry (for example, Account Number) that appears on the agent or supervisor desktop (maximum length of 50 characters). | MediaPropertiesLayout - Get |
| **dnis** | String | The DNIS provided with the call<br>For routed calls, the DNIS is the route point. | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| **domain** | String | The domain of the AWDB | EnterpriseDatabaseConfig - Get |
| **entry** | Entry object containing a name and media property | A displayName and mediaProperty combination.<br>The displayName can be empty. | MediaPropertiesLayout - Get |
| **eventType** | String | The type of CTI event:<br>• Ringing—The agent's phone is ringing.<br>• Answered—The agent answers the phone.<br>• Dropped—The call terminates.<br>• Work Ready—The agent transitions to the Work Ready agent state.<br>• Work Not Ready—The agent transitions to the Work Not Ready agent state. | |
| **extension** | Integer | The extension the user is currently using | User - Get User<br>User - Get List of Users |
| **firstName** | String | The first name of the user | User - Get User |

| | | | |
|---|---|---|---|
| | | | User - Get List of Users<br>Team - Get Object |
| **forAll** | Boolean | Whether a reason code or wrap-up reason applies to all agents<br>Possible Values: true or false<br>Only global reason codes and wrap-up reasons (forAll parameter is set to "true") are supported for Finesse Release 8.5(3). | ReasonCode—Get<br>User—Get ReasonCode<br>User - Get WrapUpReason<br>User - Get WrapUpReason List<br>WrapUpReason - Get<br>WrapUpReason - Get List |
| **fromAddress** | String | The calling line ID of the caller. | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| **header** | Entry object | A single entry (combination of displayName and mediaProperty) that appears in the call header on the desktop for each call. | MediaPropertiesLayout - Get |
| **host** | String | The hostname or IP address of the AWDB server | EnterpriseDatabaseConfig - Get |
| **id** | String | The unique identifier for the team, user, or action | User - Get User<br>User - Get List of Users<br>Team - Get Object |
| **label** | String | The UI label for a wrap-up reason or reason code, (for example, Sales Call, Complaint) | WrapUpReason - Get<br>WrapUpReason - Get List<br>ReasonCode—Get<br>ReasonCode—Get List<br>User—Get ReasonCode<br>User - Get WrapUp Reason<br>User - Get WrapUpReason List |
| **lastName** | String | The last name of the user | User - Get User<br>User - Get List of Users<br>Team - Get Object |

| | | | |
|---|---|---|---|
| **layoutxml** | String | The XML data that determines the layout of the Finesse desktop | LayoutConfig - Get |
| **loginId** | String | The login ID of the user | User - Get User<br>User - Get List of Users<br>Team - Get Object |
| **loginName** | String | The login name of the user | User - Get User<br>User - Get List of Users |
| **mediaAddress** | String | Point of contact for this participant<br>Possible Values: Extension of an agent who is a participant on a call dialog object, ANI for a caller who is a participant on a call | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| **mediaProperties** | XML | Includes all of the properties for a Call that corresponds to the Dialog | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| **mediaProperty** | String | Part of an entry. The name of the call variable or ECC variable that is displayed to the agent or supervisor (maximum length of 32 characters). | MediaPropertiesLayout - Get |
| **mediaType** | String | The type of media under which a dialog is classified<br>Possible values: voice, email, chat | User - Get List of Dialogs Associated with a User |
| **name** | String | The name of the team | User - Get User<br>User - Get List of Users<br>Team - Get Object |
| **Participant** | XML | A participant in a dialog | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| **Participants** | XML | A list of all participants, both internal and external, involved in a dialog | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| | | | |

| | | | |
|---|---|---|---|
| **password** | String | The password required to sign in to the database | EnterpriseDatabaseConfig - Get |
| **port** | Integer | The port of the AWDB server | EnterpriseDatabaseConfig - Get |
| **primaryNode - host** | String | The hostname or IP address of the primary Finesse node | SystemInfo - Get SystemInfo |
| **ReasonCode (object)** | XML | A reason code object, (this object can have a category of NOT_READY or LOGOUT, and a descriptive label and numeric code-value) | ReasonCode—Get List<br>User—Get ReasonCode<br>User—Get ReasonCode List |
| **ReasonCodes** | XML | Represents a list of ReasonCode objects | ReasonCode—Get List<br>User—Get ReasonCode List |
| **role** | String | One of the roles assigned to a user<br>Possible values: Agent, Supervisor | User - Get User<br>User - Get List of Users |
| **roles** | String | A list of roles assigned to a user | User - Get User<br>User - Get List of Users |
| **secondaryNode - host** | String | The hostname or IP address of the secondary Finesse node | ClusterConfig - Get<br>SystemInfo - Get SystemInfo |
| **state (dialog)** | String | The last state of this dialog<br>For a list of possible values, see the state (dialog) parameter values table | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| **state (participant)** | String | The last participant state change for a dialog<br>For a list of possible values, see the state (participant) parameter values table | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| **state (user)** | String | The state of the user<br>Possible values: LOGOUT, NOT_READY, READY, RESERVED, | User - Get User<br>User - Get List of Users |

| | | | |
|---|---|---|---|
| | | TALKING, WORK, WORK_READY, UNKNOWN<br>For more information about WORK and WORK_READY states, see the section WORK and WORK_READY User States. | Team - Get Object<br>Team - Get Object |
| **stateCause (participant)** | String | The cause for the last participant state in a dialog<br>This parameter is normally associated with a FAILED participant state<br>Possible values: BUSY, BAD_DESTINATION, OTHER | Dialog - Get Dialog<br>User - Get List of Dialogs Associated with a User |
| **status** | String | The state of the system<br>Possible values:<br>IN_SERVICE (The system is in service and normal operations are accepted)<br>OUT_OF_SERVICE (The system is out of service and normal operations will result in a 503 Service Unavailable response) | SystemInfo - Get SystemInfo |
| **subscription** | String | The URI for the subscription | User - Get User<br>User - Get List of Users |
| **Team** | XML | One set of team information | User - Get User<br>User - Get List of Users |
| **teams** | XML | A list of teams that a user supervises (applies to users who have a role of supervisor only) | User - Get User<br>User - Get List of Users |
| **Team** | XML | A set of information (ID and name) for one team | User - Get User<br>User - Get List of Users |
| **teamId** | String | The ID of the team to which the user belongs | User - Get User<br>User - Get List of Users |
| **teamName** | String | The name of the team to which the user belongs | User - Get User<br>User - Get List of Users |
| **uri** | String | The URI to get a new copy of the object (for example, user, dialog, team, ReasonCode, or WrapUpReason object) | User - Get User<br>User - Get List of Users<br>Dialog - Get Dialog<br>User - Get List of Dialogs |

| | | | |
|---|---|---|---|
| | | | Associated with a User<br>Team - Get Object<br>User—Get ReasonCode<br>User - Get WrapUpReason<br>User - Get WrapUpReason List<br>WrapUpReason - Get<br>WrapUpReason - Get List |
| **User** | XML | One specific user on a team | Team - Get Object |
| **username** | String | The username required to sign into the database | EnterpriseDatabaseConfig - Get |
| **users** | XML | The list of users that belong to a team | Team - Get Object |
| **wrapUpReason** | String | A description of the call<br>Size: 39 bytes (which is equal to 39 US English characters) | Dialog - Get Dialog |
| **xmppDomain** | String | The XMPP server domain | SystemInfo - Get SystemInfo |
| **xmppPubSubDomain** | String | The XMPP server pubSub domain | SystemInfo - Get SystemInfo |

### 5.4.1. State (Dialog) Parameter Values

[contents]

The following table describes possible values for the state (dialog) response parameter:

| Dialog State | Description |
|---|---|
| ALERTING | Indicates that the call is ringing at a device |
| ACTIVE | Indicates that the dialog has at least one active participant |
| DROPPED | Indicates that the dialog has no active participants |

| | |
|---|---|
| FAILED | Indicates that the dialog has failed |
| INITIATING | Indicates that the phone is off the hook at a device |
| INITIATED | Indicates that the phone is dialing at the device |

### 5.4.2. Actions Parameter Values

[contents]

The following table describes possible values (allowable actions) for the Actions response parameter:

| Participant Allowable Action | Enabled Button on Desktop | Description |
|---|---|---|
| MAKE_CALL | Make a New Call | Enables the agent to make an outgoing call |
| ANSWER | Answer | Enables the agent to answer an incoming call |
| HOLD | Hold | Enables the agent to hold a call that is currently active on the desktop |
| RETRIEVE | Retrieve | Enables the agent to retrieve a call that was on hold on the desktop |
| DROP | End | Enables the agent to drop the participant of a call |
| UPDATE_CALL_DATA | — | Enables the agent to set call data for the call<br>Finesse Release 8.5(3) does not allow an agent to set call data from the desktop. A user can set call data through the API only. |
| CONSULT_CALL | Consult | Enables the agent to make a consult call for transfer or conference |
| TRANSFER | Transfer | Enables the agent to complete a transfer between the selected held call and the existing active call on the desktop |
| CONFERENCE | Conference | Enables the agent to start a conference between the selected held call and the existing active call on the desktop |

| | | |
|---|---|---|
| | | |

**Note:** For Finesse Release 8.5(3), the Participant Allowable Action is present where applicable for all participants on a call, including participants who are not agents. The actions for participants who are not agents are not needed by the client and may not always be accurate. These actions will be removed in a subsequent release.

### 5.4.3. State (Participant) Parameter Values

[contents]

The following table describes possible values for the state (participant) response parameter:

| Participant State | Allowable Actions for the Participant State | Call State on Finesse Desktop | Description |
|---|---|---|---|
| INITIATING | DROP, UPDATE_CALL_DATA | Off Hook | Indicates that an outgoing call, not yet active, exists on the device |
| INITIATED | DROP, UPDATE_CALL_DATA | Dialing | Indicates that the phone is dialing at a device |
| ALERTING | ANSWER | Incoming | Indicates that an incoming call is ringing on the device |
| ACTIVE | HOLD, DROP, UPDATE_CALL_DATA, CONSULT_CALL | Active | Indicates that the participant is active on the call |
| FAILED | DROP | Busy | Indicates that the call failed (BUSY) |
| FAILED | DROP | Error | Indicates that the call failed (BAD_DESINATION) |
| FAILED | DROP | Error | Indicates that the call failed (OTHER) |
| HELD | RETRIEVE, DROP, UPDATE_CALL_DATA, TRANSFER (if active call exists), CONFERENCE (if active call exists) | Hold | Indicates that the participant has held their connection to the call |
| DROPPED | — | — | Indicates that the participant has dropped from the call |
| | | | |

| | | | |
|---|---|---|---|
| WRAP_UP | UPDATE_CALL_DATA | Active | Indicates that the participant is not in active state on the call but is wrapping up after the participant has dropped from the call |
| SILENT_MONITOR | DROP, UPDATE_CALL_DATA | Active | Indicates that the participant is silently monitoring an agent |

### 5.4.4. CTI Event Mappings for Dialog and Participant State

[contents]

The following table provides a list of CTI call events and the associated Dialog and Participant states for the call. This table is specifically oriented towards the agent receiving an incoming call.

Note: If the caller is also an agent, then the events go to the caller. If the caller is not an agent, events are not published to the caller.

| Scenario | CTI Event | Event Method | Dialog State | Participant State (Agent) | Participant State (Caller) |
|---|---|---|---|---|---|
| Start the Call | BEGIN_CALL_EVENT | POST (Caller) | INITIATING | Not a participant yet | INITIATING |
| Call arrives at Agent | CALL_DELIVERED | POST (Agent), PUT (Caller) | ALERTING | ALERTING | INITIATED |
| Agent answers call | CALL_ESTABLISHED | PUT | ACTIVE | ACTIVE | ACTIVE |
| Caller drops call | CALL_CONNECTION_CLEARED | PUT | ACTIVE | ACTIVE | DROPPED |
| Agent is dropped from call | CALL_CONNECTION_CLEARED | PUT | DROPPED | DROPPED | DROPPED |
| Call is cleared | CALL_CLEARED_EVENT | PUT | DROPPED | DROPPED | DROPPED |
| Call is removed | END_CALL_EVENT | DELETE | DROPPED | DROPPED | DROPPED |

The following table provides a list of CTI call events and their mapping to the Dialog state and Participant state for the call. This table is specifically oriented towards the caller making an outgoing call:

Note: If the recipient is also an agent, then the events go to the recipient. If not, they are not published to the recipient.

| | | | | | |
|---|---|---|---|---|---|

| Scenario | CTI Event | Event Method | Dialog State | Participant State (Caller) | Participant State (Recipient) |
|---|---|---|---|---|---|
| Start of any Call | BEGIN_CALL_EVENT | POST (Caller) | INITIATING | INITIATING | Not a participant yet |
| Caller takes phone off-hook | CALL_SERVICE_INITIATED_EVENT | POST (Caller) | INITIATING | INITIATING | Not a participant yet |
| Caller dials number | CALL_ORIGINATED_EVENT | PUT (Caller) | INITIATED | INITIATED | Not a participant yet |
| Destination is Busy | CALL_FAILED_EVENT (BUSY) | PUT (Caller) | FAILED | FAILED | Not a participant yet |
| Destination is Bad | CALL_FAILED_EVENT (BAD_DESTINATION) | PUT (Caller) | FAILED | FAILED | Not a participant yet |
| Destination is Recipient | CALL_DELIVERED | PUT (Caller), POST (Recipient) (See the note that precedes this table) | ALERTING | INITIATED | ALERTING |
| Recipient answers call | CALL_ESTABLISHED | PUT | ACTIVE | ACTIVE | ACTIVE |
| Caller drops call | CALL_CONNECTION_CLEARED | PUT | ACTIVE | DROPPED | ACTIVE |
| Recipient is dropped from call | CALL_CONNECTION_CLEARED | PUT | DROPPED | DROPPED | DROPPED |
| Call is cleared | CALL_CLEARED_EVENT | PUT | DROPPED | DROPPED | DROPPED |
| Call is removed | END_CALL_EVENT | DELETE | DROPPED | DROPPED | DROPPED |

The following table provides a list of CTI call events and their mapping to the Dialog state and Participant state for that call. This table is

specifically oriented towards holding a call.

Note : If the caller is also an agent, then the events go to the caller. If not, they are not published to the caller.

| Scenario | CTI Event | Event Method | Dialog State | Participant State (Agent) | Participant State (Caller) |
|---|---|---|---|---|---|
| Call has arrived and has been answered | — | — | — | — | — |
| Agent holds call | CALL_HELD | PUT | ACTIVE | HELD | ACTIVE |
| Caller holds call | CALL_HELD | PUT | ACTIVE | HELD | HELD |
| Agent retrieves call | CALL_RETRIEVED | PUT | ACTIVE | ACTIVE | HELD |
| Caller retrieves call | CALL_RETRIEVED | PUT | ACTIVE | ACTIVE | ACTIVE |

The following table provides a list of CTI call events and their mapping to the Dialog and Participant states for a call transfer. In this scenario, a call exists between the caller and Agent A. The transfer occurs after Agent B answers the consult call.

| Scenario | CTI Event (Original Call) | CTI Event (Consult Call) | Event Method | Dialog State | Participant State |
|---|---|---|---|---|---|
| Agent A starts consult call | CALL_HELD | — | PUT (original call only) | Original call —ACTIVE | Caller—ACTIVE<br>Agent A—HELD (original call)<br>Agent B—Not yet a participant |
| Agent A takes phone off-hook (BEGIN_CALL_EVENT assumed) | — | CALL_SERVICE_INITIATED_EVENT | PUT (consult call only) | Original call —ACTIVE<br>Consult call —INITIATING | Caller—ACTIVE<br>Agent A—INITIATING (consult call)<br>Agent B—Not yet a participant |

| | | | | | |
|---|---|---|---|---|---|
| Agent A dials number | — | CALL_ORIGINATED_EVENT | PUT (consult call only) | Original call —ACTIVE<br>Consult call — INITIATED | Caller— ACTIVE<br>Agent A— INITIATED (consult call)<br>Agent B— Not yet a participant |
| Agent B receives the call | — | CALL_DELIVERED | PUT (consult call, on Agent A<br>POST (consult call on Agent B | Original call —ACTIVE<br>Consult call — ALERTING | Caller— ACTIVE<br>Agent A— INITIATED (consult call)<br>Agent B— ALERTING |
| Agent B answers the call | — | CALL_ESTABLISHED | PUT (consult call only) | Original call —ACTIVE<br>Consult call —ACTIVE | Caller— ACTIVE<br>Agent A— ACTIVE (consult call)<br>Agent B— ACTIVE |
| Agent A completes the transfer of the caller to Agent B | CALL_TRANSFERRED_EVENT | — | DELETE (original call on Agent A)<br>DELETE (consult call on Agent A)<br>DELETE (consult call on Agent B)<br>POST (original call on Agent B) | Original call — DROPPED (Agent A), ACTIVE (Agent B)<br>Consult call — DROPPED (both Agent A and Agent B) | Caller— ACTIVE<br>Agent A— DROPPED (original and consult call)<br>Agent B— DROPPED (consult call), ACTIVE (original call) |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |

If the caller is also an agent, that caller receives a Dialog update (PUT) with an updated participant list after the transfer is complete.

### 5.4.5. WORK and WORK_READY User States

[contents]

A user is in either WORK or WORK_READY state during wrap-up. A user is placed in WORK state when Unified CCE plans to set the state of that user to NOT_READY when wrap-up ends. If Unified CCE plans to set the state of the user to READY when wrap-up ends, that user is placed in WORK_READY state.

A user transitions to WORK state for one of the following reasons:

- The user was in NOT_READY state before taking a call.
- The user set a state of NOT_READY while in TALKING state.

If wrap-up times out, the user transitions to NOT_READY state.

A user transitions to WORK_READY state for one of the following reasons:

- The user was in READY state before taking a call.
- The user set a state of READY while in TALKING state.

If wrap-up times out, the user transitions to READY state.

A user transitions out of WORK or WORK_READY state by performing one of the following actions:

- The user lets the wrap-up timer expire.
- The user sets a state of either READY or NOT_READY.

# 6. Cisco Finesse Errors

[contents]



## 6.1. HTTP Errors

[contents]

All HTTP errors are returned as HTTP 1.1 Status Codes. Errors that might be for Finesse-specific events are listed below:

- **500 Internal Server Error**: Finesse Web Services returns 500 if the CTI connection is lost but the loss is not yet detected by automated means.
  500:5526 - DB_RUNTIME_EXCEPTION (database error, but the database is thought to be operational)
  500:5525 - RUNTIME_EXCEPTION (a non-database error)
  500:5500 - AWS_SERVICE_UNAVAILABLE (AWS not operational)
- **503 Service Unavailable**: If Finesse is in PARTIAL_SERVICE or OUT_OF_SERVICE, it returns 503 for all requests. If any dependent service goes down, Finesse goes to OUT_OF_SERVICE state (for example, if the Cisco Finesse Notification Service is down).
  This error is due to a temporary outage or overloading condition. A retry after several seconds is likely to succeed.
  For example, the system returns 503 when the system is just starting up and when the system is trying to connect to the CTI server.

## 6.2. Cisco Finesse API Error Codes

[contents]

Error codes for Cisco Finesse are categorized as follows:

- 4xx - Client-related error codes
- 5xx - Server-related error codes

Each error code includes a failure response, error code, and error message. The following is an example of the Failure Message format:

```
{
 "response":"failure",
 "errorCode":xxxx,
 "errorMessage":<error message>
}
```

In addition to Cisco Finesse API errors, a response might return a CTI error or an HTTP error. HTTP errors are documented online: HTTP 1.1 Status Codes

| Status | Error Code | Description |
|---|---|---|
| 400 | Generic Error | Any generic error. |
| 400 | Invalid Destination | The toAddress and fromAddress are the same. This error occurs if users attempt to call their own extensions. |
| 400 | Invalid Device | The extension is invalid. |
| 400 | Invalid Input | One of the parameters (for example, state, fromAddress, toAddress, targetMediaAddress, or requestedAction), as part of the user input, is invalid or not recognized.<br>The submitted XML is not valid (LayoutConfig APIs). |
| 400 | Invalid State | The requested state change is not allowed (for example, a user already in LOGOUT state requests a state change to LOGOUT).<br>A supervisor who is already in an active call (in TALKING or ON_HOLD state) makes a silent monitoring request. |
| 400 | Parameter Missing | The extension, state value, or requestedAction is not provided.<br>If creating a dialog, the fromAddress or toAddress is not provided.<br>If creating a layout, the XML file is not provided. |
| 401 | Authorization Failure | Unauthorized (for example, the user is not yet authenticated in Web Session).<br>The user is not authorized to use the API (for example, an agent tries to use an API that only supervisors or administrators are authorized to use). |
| 401 | Invalid Authorization | The authenticated user tried to make a request for another user. |

| | | |
|---|---|---|
| | User Specified | The authenticated user tried to use a fromAddress or targetMediaAddress that does not belong to that user.<br>The primary and backup AWDB servers are down and Finesse cannot authenticate the user. |
| 404 | Not Found | The resource specified is invalid or does not exist. |
| 404 | Dialog Not Found | The dialog id provided is invalid or no such dialog exists. |
| 404 | User Not Found | The agent ID provided is invalid or not recognized. No such agent exists in CTI. |
| 500 | Internal Server Error | A runtime exception is caught (for example, a broken connection with the CTI server or another component). |
| 503 | Service Unavailable | If any dependent service goes down, Finesse goes to OUT_OF_SERVICE state (for example, if the Cisco Finesse Notification Service or the Cisco Finesse Database is down). |

# 7. Cisco Finesse Notifications

[contents]

## 7.1. About Cisco Finesse Notifications

[contents]

The Cisco Finesse Web Service sends notifications to any clients that are subscribed to that class of resource.

For example, a client that is subscribed to *User* notifications receives a notification when an agent signs in or signs out of the Finesse Desktop, when agent information changes, or when an agent's state changes.

**Note:** The preceding example illustrates some possible cases where notifications are sent. It is not intended to be an exhaustive list.

**Note:** The Notification payloads are XML encoded. If these payloads contain any special XML characters, you must ensure that the client decodes this information correctly before processing it further.

### 7.1.1. Notification Frequency

[contents]

Notifications are published as they occur when there is a change in the resource characteristics.

### 7.1.2. Subscription Management

[contents]

Finesse clients can interface directly with the Cisco Notification Service to send subscribe and unsubscribe requests notification feeds published to their respective nodes (such as /finesse/api/User/1000) by following the XEP-0600 standard.

Clients are automatically subscribed to receive the following notification feeds for the same User:

- User - finesse/api/User/{id}
- Dialogs - /finesse/api/User/{id}/Dialogs

To receive notifications for feeds to which they are not automatically subscribed, clients must explicitly subscribe to the node on which the

notifications are published. For example, agent state change notifications for all agents on a specific team are published to the node /finesse/api/Team/{id}/Users. Clients must request a subscription to this node to receive notifications on this feed.

The following example shows how to subscribe to agent state change notifications for a specific team:

```
<iq type='set'
   from='CharlesNorrad@finesse-server.cisco.com'
   to='pubsub.finesse-server.cisco.com'
   id='sub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscribe
        node='finesse/api/Team/TheA/Users'
        jid='ChuckieNorrad@finesse-server.cisco.com'/>
  </pubsub>
</iq>
```

The following example shows how to unsubscribe to agent state change notifications for a specific team:

```
<iq type='set'
   from='ChuckieNorrad@finesse-server.cisco.com'
   to='pubsub.finesse-server.cisco.com'
   id='unsub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <unsubscribe
        node='finesse/api/Team/TheA/Users'
        jid='userid@finesse-server.cisco.com'/>
  </pubsub>
</iq>
```

You can obtain connection details by performing a GET using the SystemInfo API (http://<server>/finesse/api/SystemInfo). The returned payload provides you with the domain and pubsub addresses used to interact with the Cisco Finesse Notification Service.

```
<SystemInfo>
   <status>IN_SERVICE</status>
   <xmppDomain>xmppserver.cisco.com</xmppDomain>
   <xmppPubSubDomain>pubsub.xmppserver.cisco.com</xmppPubSubDomain>
</SystemInfo>
```

Users are identified in the following manner: userid@xmppserver.cisco.com

Stanzas are sent to the pubsub domain (pubsub.xmppserver.cisco.com ).

Clients should ensure that any subscriptions that are no longer required are cleaned up.

### Subscription Persistence

All subscriptions are stored in a database and persist through the following shutdown events:

- Finesse experiences a CTI failover
- Cisco Notification Service restarts
- Cisco Tomcat restarts

In each of the preceding events, the client does not need to resubscribe to explicit subscriptions.

However, subscriptions do not persist across multiple Finesse servers. If a client fails over to an alternate Finesse server, that client must resubscribe to any explicit subscriptions.

## 7.2. Resources

[contents]

- User Notifications
- Dialog Notifications
- Dialog CTI Error Notifications

- Team Notifications

## 7.2.1. User Notifications

[contents]

Finesse sends a User notification when information about a user changes.

| | |
|---|---|
| **Format:** | XML |
| **Node:** | /finesse/api/User/{id} |
| **Source:** | /finesse/api/User/<id> |
| **Data:** | User |
| **Payload:** | <Update><br><event>{put\|delete}</event><br><source>/finesse/api/User/{id}</source><br><data><br><user><br><!-- full User object --><br></user><br></data><br></Update> |
| **Notification Triggers:** | • Addition of a user<br>• Deletion of a user<br>• State change<br>• First or last name change<br>• Role change |

### Notification Parameters

- event
- source
- data

### Sample Notification Payload

```
<Update>
   <event>put</event>
   <source>/finesse/api/User/csmith</source>
   <data>
      <user>
         <uri>/finesse/api/User/csmith</uri>
         <state>NOT_READY</state>
         <extension>1001001</extension>
```

```
        <firstName>Chris</firstName>
        <lastName>Smith</lastName>
      </user>
    </data>
</Update>
```

## 7.2.2. Dialog Notifications

[contents]

Finesse sends a Dialog notification when there is a change to information (or to an action) related to a call to which the user belongs.

| | |
|---|---|
| **Format:** | XML |
| **Node:** | /finesse/api/User/{id}/Dialogs |
| **Source:** | /finesse/api/User/{id}/Dialogs (when a Dialog is added or removed from the Dialogs collection for a User)<br>/finesse/api/Dialog/{id} (when a Dialog within the Dialogs collection for a User is modified) |
| **Data:** | Dialog |
| **Payload:** | `<Update><data><dialogs><Dialog><!-- full Dialog object --></Dialog></dialogs></data><event>{POST\|DELETE}</event><requestId>xxxxxxxxx</requestId><source>/finesse/api/User/{id}/Dialogs</source></Update>` |
| **Notification Triggers:** | • Incoming call<br>• Modification of participant state (for example, when a participant answers or hangs up a call)<br>• A new participant to the call<br>• Modification of the call data or actions |

### Notification Parameters

- event
- source
- data
- requestId

### Sample Notification Payload

```
<Update>
 <data>
   <dialogs>
     <Dialog>
       <fromAddress>1001002</fromAddress>
       <mediaType>Voice</mediaType>
       <state>ALERTING|ACTIVE|INACTIVE</state>
       <uri>/finesse/api/Dialog/16792694</uri>
       <mediaProperties>
         <dialedNumber>2000</dialedNumber>
         <callType>AGENT_INSIDE</callType>
         <DNIS>2000</DNIS>
         <callvariables>
           <CallVariable>
             <name>callVariable1</name>
             <value>Chuck Smith</value>
           </CallVariable>
           <CallVariable>
             <name>callVariable2</name>
             <value>Cisco Systems,Inc<value>
           </CallVariable>
           ...
           <CallVariable>
             <name>callVariable10</name>
             <value>Preferred Customer<value>
           </CallVariable>
           <CallVariable>
             <name>user.user</name>
             <value>csmith<value>
           </CallVariable>
           <CallVariable>
             <name>user.years[0]</name>
             <value>1985<value>
           </CallVariable>
           <CallVariable>
             <name>user.years[1]</name>
             <value>1995<value>
           </CallVariable>
         </callvariables>
       </mediaProperties>
       <Participants>
           <Participant>
              <state>ALERTING|INITIATING|ACTIVE|INACTIVE</state>
              <mediaAddress></mediaAddress>
              <stateCause></stateCause>
              <actions>
                 <action>...</action>
                 <action>...</action>
              </actions>
           </Participant>
        </Participants>
     </Dialog>
   </dialogs>
 </data>
 <event>POST</event>
 <requestId></requestId>
 <source>/finesse/api/User/1001001/Dialogs</source>
</Update>
```

## 7.2.3. Dialog CTI Error Notifications

[contents]

Call operations performed on a Dialog (such as MAKE_CALL, HOLD, RETRIEVE, ANSWER, END, TRANSFER, CONSULT, and CONFERENCE) may result in CTI errors. The Notification system sends these errors as an asynchronous update. The error notifications include the error type and the CTI error code and error constant. The error type is “Call Operation Failure”.

| | |
|---|---|
| **Format:** | XML |
| **Node:** | /finesse/api/User/{id}/Dialogs |
| **Source:** | /finesse/api/Dialog/[ID] |

| | |
|---|---|
| **Data:** | apiErrors |
| **Payload:** | <Update><br>  <data><br>  <apiErrors><br>  <apiError><br>  <errorData>[CTI Error Code]</errorData><br>  <errorMessage>[CTI Error Constant]</errorMessage><br>  <errorType>Call Operation Failure</errorType><br>  </apiError><br>  </apiErrors><br>  </data><br>  <event>PUT</event><br>  <requestId></requestId><br>  <source>/finesse/api/Dialog/[ID]</source><br></Update> |
| **Notification Triggers:** | The notification system delivers this error notification if call operations on a Dialog (such as MAKE_CALL, HOLD, RETRIEVE, ANSWER, END, TRANSFER, CONSULT, and CONFERENCE) result in a CTI error |

### Notification Parameters

- event
- source
- data
- requestId

### Sample Notification Payload

```
<Update>
   <data>
      <apiErrors>
         <apiError>
            <errorData>34</errorData>
            <errorMessage>CF_RESOURCE_OUT_OF_SERVICE</errorMessage>
            <errorType>Call Operation Failure</errorType>
         </apiError>
      </apiErrors>
   </data>
   <event>PUT</event>
   <requestId></requestId>
   <source>/finesse/api/Dialog/12345</source>
</Update>
```

### CTI Error Messages

The following table lists possible call control-specific error messages and their corresponding codes and descriptions.

| CTI Error Message | Description | Code |
|---|---|---|
| CF_INVALID_CONSULT_TYPE | The consult type is invalid | 273 |

| | | |
|---|---|---|
| CF_INVALID_CONNECTION_ID_FOR_ACTIVE_CALL | The active connection ID in the request is invalid | 23 |
| CF_INVALID_CALLING_DEVICE | The calling device is not valid | 5 |
| CF_INVALID_CALLED_DEVICE | The called device is not valid | 6 |

### 7.2.4. Team Notifications

[contents]

Finesse sends a team notification when the agent name or agent state changes for an agent who belongs to that team.

| | |
|---|---|
| **Format:** | XML |
| **Node:** | /finesse/api/Team/{id}/Users |
| **Source:** | /finesse/api/User/{id} |
| **Data:** | Summary version of the User object |
| **Payload:** | <Update><br><event>{put}</event><br><source>/finesse/api/User/{id}</source><br><requestId>xxxxxxxxx</requestId><br><data><br><user><br><uri>/finesse/api/User/{id}</uri><br><loginId>{id}</loginId><br><firstName>Jack</firstName><br><lastName>Brown</lastName><br><state>NOT_READY</state><br></user><br></data><br></Update> |
| **Notification Triggers:** | • Agent name is changed for an agent belonging to the team<br>• Agent state is changed for an agent belonging to the team |

#### Notification Parameters

- event
- source
- data

- requestId

### Sample Notification Payload

```
<Update>
   <event>put</event>
   <source>/finesse/api/Team/1004</source>
   <requestId>xxxxxxxxx</requestId>
   <data>
      <team>
         <uri>/finesse/api/Team/1004</uri>
         <id>1004</id>
         <name>Shiny</name>
         <users>
            <User>
               <uri>/finesse/api/User/{id}</uri>
               <loginId>{id}</loginId>
               <firstName>Charles</firstName>
               <lastName>Norrad</lastName>
               <state>LOGOUT</state>
            </User>
            <User>
               <uri>/finesse/api/User/{id}</uri>
               <loginId>{id}</loginId>
               <firstName>Jack</firstName>
               <lastName>Brown</lastName>
               <state>NOT_READY</state>
            </User>
            ... other users ...
         </users>
      </team>
   </data>
</Update>
```

## 7.3. Notification Parameter Reference

[contents]

| Name | Data Type | Description | Possible Values | Used by These Notifications |
|---|---|---|---|---|
| **Data** | Object | Provides the new representation of the modified User or Team or Dialog object. This information is not provided when a user is deleted.<br>On an error notification, provides the List of ApiError objects representing the failure conditions detected by the server. | The entire User, Team, or Dialog object in its most current and updated form.<br>The Team object includes all of its agents. | User Notification<br>Dialog Notification<br>Dialog CTI Error Notification<br>Team Notification |
| **Event** | String | The type of modification that occurred to the User or Team or Dialog object. | **PUT:** A property of the User, Team, or Dialog object has been modified.<br>**DELETE:** The User, Team, or Dialog has been deleted.<br>**POST:** A User, Team, or Dialog object was added. | User Notification<br>Dialog Notification<br>Dialog CTI Error Notification<br>Team Notification |

| | | | | |
|---|---|---|---|---|
| **Source** | String | The User resource location that was modified. | /finesse/api/User/{id}<br>/finesse/api/Dialog/{id}<br>/finesse/api/Team/{id}<br>/finesse/api/User/{id}/Dialogs | User Notification<br>Dialog Notification<br>Dialog CTI Error Notification<br>Team Notification |
| **RequestId** | String | The requestId that was returned when the triggering REST API request was made. If the event was unsolicited, this tag is empty. | An opaque, unique String, used to correlate the originating request with the resulting event. | Dialog Notification<br>Dialog CTI Error Notification<br>Team Notification |

# 8. Finesse High Availability

[contents]

Finesse uses *Presence* as a mechanism to detect the availability of a particular instance of the Finesse server. The BOSH user associated with an agent is automatically subscribed to the presence of the 'finesse' BOSH user. If the Finesse server goes down or loses connection to the CTI server, a presence notification with a state of *Unavailable* is published to the BOSH user of all agents. Finesse clients can use this notification as a mechanism to detect availability of the Finesse server and decide whether to failover to the alternate Finesse server.

To receive these notifications, a client must be logged in to the Finesse server and the Notification Service on the same server.

This mechanism cannot be used to detect the availability of the Notification Service. Clients use the status of the BOSH connection to detect the availability of the Notification Service.

The following table lists whether a presence notification is sent and the presence status for Finesse under various scenarios:

| Action | Presence Notification | Presence Status | From User |
|---|---|---|---|
| Cisco Tomcat goes down | Yes | 'Unavailable' | 'finesse' |
| Finesse webapp goes down | Yes | 'Unavailable' | 'finesse' |
| Finesse loses connection to the CTI server | Yes | 'Unavailable' | 'finesse' |
| Notification Service goes down | No | 'Unavailable' | 'finesse' |

# 9. Finesse Desktop Gadget Development

[contents]



## 9.1. Notifications on the Finesse Desktop

[contents]

The Finesse Desktop contains support for OpenSocial Core Gadget Specification 1.1. OpenSocial Core Gadget Specification 1.1 supports an intergadget notification system that is based on the OpenAjax Hub 2.0 Specification.

The Finesse Desktop automatically establishes a BOSH connection to the Notification Service upon sign-in. The Finesse Desktop publishes notifications that it receives from the Notification Service to OpenAjax Hub topics. An OpenAjax topic is a string name that identifies a particular topic type to which a client can subscribe or publish. Gadgets must subscribe to these topics to receive notifications.

If the BOSH connection is disconnected, the Finesse Desktop attempts to reconnect at 30-second intervals until the connection is re-established.

We recommend that you review the OpenSocial and OpenAjax Hub specifications before you implement gadget support for notifications on the Finesse Desktop.

## 9.2. Enabling Finesse Notifications in Third-Party Containers

[contents]

There are strict requirements that must be followed to leverage the Finesse Desktop notification framework on a third-party container.

1. Clients must add a specific Finesse gadget, which establishes the BOSH connection and publishes notifications to Finesse-specific OpenAjax topics.
2. Third-party containers (that is, those other than the Finesse Desktop) must provide support for the OpenSocial Core Gadget Specification 1.1 to ensure that gadgets can subscribe to Finesse-specific notifications through the OpenAjax Hub.

## 9.3. Finesse Topics

[contents]

A gadget that is within the Finesse environment has the ability to subscribe or publish to a set of Finesse Desktop topics via OpenAjax Hub. The following sections provide details for the available topics.



### 9.3.1. Connection Information

[contents]

| **Topic Name** | finesse.info.connection |
|---|---|
| | |

| Topic Type | Gadgets subscribe to this topic. |
|---|---|

Gadgets subscribe to the finesse.info.connection topic to receive status information about the BOSH connection, which is automatically established by the Finesse Desktop or a Finesse Desktop gadget (within a non-Finesse container). Connection status information can be used to determine the state of the connection so that a gadget can act appropriately. Additionally, a resource ID is provided in the published data to allow the gadget to construct a subscribe request to the Finesse Web Services. Connection information is published every time there is a connection state change.

The published data is a JavaScript object with the following properties:

```
{
    status: string,
    resourceID: string
}
```

The *status* parameter describes the BOSH connection status. It can have any one of the following values:

- connected
- connecting
- disconnected
- disconnecting
- reconnecting
- unloading

**Note:** A BOSH connection status of "unloading" indicates that an action in the browser (such as refreshing the browser or clicking the back button) caused the BOSH connection to initiate the unloading process.

The *resourceID* parameter is a unique identifier for the BOSH connection. Although the resourceID parameter is provided with every connection status change, the ID is not available until after a BOSH connection has been successfully established. It is possible that the BOSH connection reconnects with a different resourceID.

A situation can occur where a gadget is loaded after the Finesse Desktop or gadget has already published connection information. In this case, have the gadget publish a request to a Finesse request topic, which forces the Finesse Desktop to publish the connection information again. For more information, see Finesse Requests.

### 9.3.2. Finesse Notifications

[contents]

| Topic Name | finesse.api.[resourceObject].[resourceID] |
|---|---|
| **Topic Type** | Gadgets subscribe to this topic. |

If a user has any subscriptions for a particular notification, either created by the Finesse Desktop or by an explicit subscribe request (see Subscription Management on the Finesse Desktop), the Notification Service delivers updates through the established BOSH connection. The Finesse Desktop automatically handles the management of the BOSH event connection to the Notification Service. Any notifications that are delivered through the connection are converted to JavaScript Object, and then published by the Finesse Desktop to an OpenAjax Hub topic. The name of the topic matches the node on the Finesse Notification Service on which the notification was published. However, to comply with OpenAjax topic conventions, all slashes (/) are replaced with dots (.) and the leading slash is removed.

To receive notifications, the gadgets must

1. Subscribe to the OpenAjax topic for a particular notification feed. This action ensures that no notifications are missed after sending the subscription request to Finesse Web Services.
2. If required, make a request to the Finesse Notification Service to create a subscription for the notification feed (see Subscription Management on the Finesse Desktop).

In Finesse, each notification type has an equivalent topic to which gadgets can subscribe. For a list of available Finesse notifications, see section 7 Cisco Finesse Notifications and look under the "node" property. These notifications are structured as follows:

```
{
content : Raw object payload as a String,
object : JavaScript object representation of the payload
}
```

### Sample Notification Payload

```
{
    event: "PUT"
    source: "/finesse/api/User/1000"
    data: {}
}
```

To receive notifications for User object updates, a client within the Finesse Desktop must subscribe to *finesse.api.user.1000*.

```
{
    content: "<Update>
                <data>[User Object]</data>
                <event>PUT</event>
                <source>/finesse/api/User/[ID]</source>
             </Update>"
    object: {
        Update: {
            data: [User Object],
            event: "PUT",
            source: "/finesse/api/User/[ID]
        }
    }
}
```

## 9.3.3. Finesse Requests

[contents]

| | |
|---|---|
| **Topic Name** | finesse.info.requests |
| **Topic Type** | Gadgets publish to this topic. |

Communication between gadgets and the Finesse Desktop or other gadgets is done through inter-gadget notification via OpenAjax Hub. A gadget can send an operation request to the Finesse Desktop by publishing a request object to the Finesse request topic.

The gadget must construct an object to be published to the request topic with the following structure:

```
{
    type: string,
    data: object
}
```

The *type* parameter describes the request type.

The *data* parameter provides additional information for the Finesse Desktop to respond to the request. The contents of this data depends on the type of request.

The following sections describe the different types of requests supported.

**Note:** More request types may be added in the future.

### ConnectionInfoReq

Sending an "ConnectionInfoReq" request forces the Finesse Desktop to publish a connection information object to all gadgets subscribed to the *finesse.info.connection* topic. This request allows gadgets to determine the current state of the BOSH connection and retrieve the resource ID. The gadget must be subscribed to the connectionInfo topic to receive the event.

The gadget should publish the following object to the topic *finesse.info.requests*:

```
{
    type: “ConnectionInfoReq”,
    data: { }
}
```

It is possible that the gadget may come up before the Finesse Desktop is ready to start responding to a request to send connection information. For this reason, gadgets should subscribe to the *finesse.info.connection* topic regardless. When the Finesse Desktop or gadget is ready, it starts publishing connection information immediately.

**Note:** The topic *finesse.info.connection* is shared across all subscribed gadgets. Gadgets that subscribe to this topic may receive duplicate notifications. Gadgets must be able to handle duplicate notifications appropriately.

### ConnectionReq

Sending a "ConnectionReq" forces the Finesse Desktop to attempt to establish a BOSH connection with the Notification Service. This request can only go through if either no active connection currently exists or if the current connection is in the "disconnected" state.

The gadget should publish the following object to the topic *finesse.info.requests*:

```
{
    type: "ConnectionReq",
    data: {
        id: ID,
        password: password,
        xmppDomain: xmppDomain
    },
}
```

The *id* and *password* parameters specify the ID and password of the XMPP user for which to establish a BOSH connection. The *xmppDomain* parameter specifies the domain of the XMPP server.

### SubscribeNodeReq

Sending a "SubscribeNodeReq" request causes the managed BOSH connection to send an XEP-0060 standard subscribe request (described in section 7.1 About Cisco Finesse Notifications) to subscribe to the notification feed for the specified node. The response to this request is published on the response topic finesse.info.responses.{invokeID}, where the invokeID must be generated by the gadget to identify this unique request and subscription. For more details, see Finesse Responses. The Cisco gadgets use an RFC1422v4-compliant universally unique identifier (UUID) for this invokeID. For more details, see the Finesse Software Development Kit (SDK).

To guarantee that the gadget receives the response, it must subscribe to the response topic (on the OpenAjax Hub) of its self-generated invokeID before sending the following object to the topic finesse.info.requests:

```
{
    type: "SubscribeNodeReq",
    data: {
        node: "/finesse/api/Team/{id}/Users" // the node of interest
    },
```

```
    invokeID: "xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx"
}
```

The *node* parameter specifies the node to subscribe to. The *invokeID* parameter is self-generated and is used to track this particular subscription. This parameter is also used as part of the OpenAjax topic to which the response of the request is published.

### UnsubscribeNodeReq

Sending an "UnsubscribeNodeReq" request causes the managed BOSH connection to send an XEP-0060 standard unsubscribe request (described in section 7.1 About Cisco Finesse Notifications) to unsubscribe from the specified node. The response of this request is published on the response topic finesse.info.responses.{invokeID}, where the invokeID must be generated by the gadget to identify this unique request. For more details, see Finesse Responses. The Cisco gadgets use an RFC1422v4-compliant UUID for this invokeID. For more details, see the Finesse SDK.

To guarantee that the gadget receives the response, it must subscribe to the response topic (on the OpenAjax Hub) of its self-generated invokeID before sending the following object to the topic finesse.info.requests:

```
{
    type: "UnsubscribeNodeReq",
    data: {
        node: "/finesse/api/Team/{id}/Users",
        subid: "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    },
    invokeID: "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxy"
}
```

The *node* parameter specifies the node to subscribe to. The *subid* parameter specifies the subscription to remove, which is uniquely identified by the invokeID that was used in the subscribe request. The *invokeID* parameter is self-generated and is used as part of the OpenAjax topic to which the response of the request is published.

## 9.3.4. Finesse Responses

[contents]

| | |
|---|---|
| **Topic Name** | finesse.info.responses.{invokeID} |
| **Topic Type** | Gadgets subscribe to this topic. |

Responses to requests are published to these channels. When a request is made, the gadget generates and specifies a unique invokeID as part of the request. This invokeID is used as the trailing token in the topic to which the response of the request is published.

Because this topic is only used to communicate the response of a single request and never used again, be sure to unsubscribe from the topic as part of the callback handler in the subscribe request. For example:

```
// Generate invokeID and construct request
var UUID = _util.generateUUID(),
data = {
    type: "ExampleReq",
    data: {},
    invokeID: UUID
},


// Subscribe to the response channel to ensure we don't miss the response
OAAsubid = gadgets.Hub.subscribe("finesse.info.responses."+ UUID, function (topic, data) {
    // Unsubscribe from the response topic to prevent memory leaks
    // Do this before processing the response in case the processing throws an exception
    gadgets.Hub.unsubscribe(OAAsubid);

    // Process the response here
});
```

```
// Publish the request after we have registered our response callback on the response topic
gadgets.Hub.publish("finesse.info.requests", data);
```

## 9.4. Subscription Management on the Finesse Desktop

[contents]

Because the Finesse Desktop provides a managed BOSH connection to the Cisco Finesse Notification Service, the ability to subscribe or unsubscribe to a particular notification feed is also provided as an interface using the SubscribeNodeReq and UnsubscribeNodeReq requests described in section 9.3.3 Finesse Requests.

## 9.5. Persistence of Gadget Preferences

[contents]

The Finesse Desktop can persist gadget preferences in the browser. Gadgets can set preferences using the standard OpenSocial gadget APIs, as shown in the following example:

```
var myPrefs = new gadgets.Prefs();
myPrefs.set("hello","world");
```

After a gadget sets its preferences, anytime that gadget is constructed (in the same browser), these preferences continue to be available through the same APIs.

```
var myPrefs = new gadget.Prefs(),
helloValue = myPrefs.getString("hello");
```

**Note:** Do not use preferences to persist critical application data. This data is stored on the browser and may be manually purged by the user at will. This storage is meant for preferences (similar to the type of information that is typically stored inside a cookie), and not complex application data. Additionally, when the browser runs out of the allocated storage space, this data is purged.

Additionally, if special characters are expected in the value of the preference, they should be escaped inbound and unescaped outbound, as shown in the following example:

```
var myPrefs = new gadget.Prefs(),
myPrefs.set("hello", gadgets.util.escapeString("!@#$%^&*()<>?");
…
var myPrefs = new gadget.Prefs(),
helloValue = gadgets.util.unescapeString(myPrefs.getString("hello"));
```

**Note:** Do not use special characters within the name of the preference. The use of special characters within the name of the preference is not supported.

# 10. Glossary

[contents]

Terms and Definitions are provided in this section.

BOSH - Acronym for Bidirectional-streams Over Synchronous HTTP. The BOSH protocol defines how arbitrary XML elements can be transported efficiently and reliably over HTTP in both directions between a client and server.

**Call Connections** - Refers to the parties on the call. Typically includes the agent and the calling party - another agent, customer calling into call center, another party calling into call center.

**Call Variables** - These are text fields in the desktop interface. They might appear as Var1 ... Var10, or they might be configured by the system administrator and labeled for specific purpose such as Account Number, Case Number, and so forth. Each call variable is a free-form string of up to 41 characters. The data entered in these fields is saved in the Termination Call Detail table in the database schema.

**Client** - The client is a computer application, such as a web browser, that runs on a user's local computer or workstation and connects to a server as necessary to send or receive information.

**GET (HTTP Method)** - This method requests a representation of the specified resource.

ISDN - Acronym for Integrated Services Digital Network - a set of communications standards for simultaneous digital transmission of voice, video, data, and other network services over the traditional circuits of the public switched telephone network.

JSON - Acronym for JavaScript Object Notation, a text-based open standard designed for human-readable data interchange, derived from the JavaScript programming language

**Party** - Refers to a person who is receiving a call or who is being added to a conference.

POST (HTTP Method) - The POST request method is used when the client needs to send data to the server as part of the request, such as when uploading a file or submitting a completed form.

**Queue** - The term *queue* in this guide refers to the *Skill Group* in Unified CCE. A queue is a collection of agents at a single contact center who share a common set of competencies that equip them to handle the same types of requests. Some examples of queues are a collection of agents who speak a specific language or who can assist callers with billing questions.

**Route Point** - A CTI route point designates a virtual device that can receive multiple, simultaneous calls for application-controlled redirection. For example, route point 4006 might represent the extensions of several agents in a queue.

**Unified CCE** - The Cisco Unified Contact Center Enterprise delivers intelligent contact routing, call treatment, network-to-desktop computer telephony integration (CTI), and multichannel contact management over an IP infrastructure. It combines multichannel automatic call distributor (ACD) functionality with IP telephony in a unified solution, enabling your company to rapidly deploy a distributed contact center infrastructure.

**Unmonitored device** - This might be a caller phone or an agent phone known to Unified Communications Manager that the agent has not logged into.

XMPP - Acronym for the Extensible Messaging and Presence Protocol.

**XMPP Server** - An XMPP server provides basic messaging, presence, and XML routing features. The XMPP server acts as an intelligent abstraction layer for XMPP communications. Its primary responsibilities are to manage connections from - or sessions for - other entities, in the form of XML streams, and to route appropriately-addressed XML stanzas among such entities over XML streams. OpenFire is the XMPP Server used by the Cisco Finesse.

# 11. Documents and Documentation Feedback

[contents]

**DOCUMENTS**

The Cisco Finesse Web Services Developer Guide is available from the Cisco Developer Network (CDN):

Point your browser to http://developer.cisco.com/web/finesse/overview.

Click the link for Cisco Finesse.

Sign in with your Cisco credentials.

Click **Documentation**.

If you have development questions, you can post them to the Cisco Finesse forums on the Cisco Developer Network, located at the following link: http://developer.cisco.com/web/finesse/forums

The following documents are available from the Finesse page on Cisco.com (http://www.cisco.com/en/US/products/ps11324/tsd_products_support_series_home.html):

- *Cisco Finesse Installation and Getting Started Guide*
- *User Guide for the Cisco Finesse Administration and Serviceability Consoles*
- *Release Notes for Cisco Finesse Release 8.5(3)*

**DOCUMENTATION FEEDBACK**

You can provide comments about this document by sending email to the following address: mailto:ccbu_docfeedback@cisco.com

We appreciate your comments.