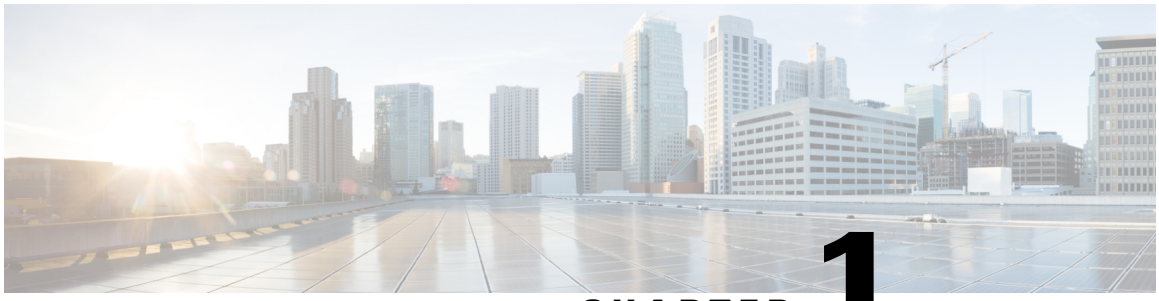
Cisco Finesse Web Services Developer (API) Guide Release 10.0(1)

First Published: December 12, 2013

Last Modified: May 16, 2014

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



CHAPTER 1

Introduction

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, such as call control, state changes, and configuration information.

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
- Teams
- Queues
- Mobile Agents
- Workflows

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.

JavaScript Library and Sample Gadgets

Finesse provides a JavaScript library (finesse.js) and a number of sample gadgets to help jumpstart your gadget development. The JavaScript library provides a substantial amount of fundamental code infrastructure that you would otherwise need to write yourself. The JavaScript library and sample gadgets are available on the Cisco Developer Network at the following link: http://developer.cisco.com/web/finesse/documentation

Important If you are developing third-party gadgets for Finesse, you must ensure that your new and existing gadgets use the Finesse JavaScript library that came with the release of Finesse that you use.

New in This Release

The following sections provide an overview of changes to this guide for Cisco Finesse Release 10.0(1).

Unified Contact Center Express Support

Cisco Finesse Release 10.0(1) is supported in Unified Contact Center Express (Unified CCX). The deploymentType parameter was added to the SystemInfo object to identify the platform in which Finesse is deployed. The possible values for the deploymentType parameter are UCCE or UCCX. For more information, see SystemInfo - Get SystemInfo, on page 104.

For APIs that may behave differently depending on which platform Finesse is deployed, a new “Platform-based API differences” section has been added to explain the behavior.

Call Recording

A new dialog API was added to allow a user to record an active call. This API is supported only for Finesse deployments with Unified CCX. For more information, see Dialog — Start Recording, on page 90.

Workflows and Workflow Actions

New APIs were added to allow administrators to configure workflows and workflow actions for agents. For more information, see the following sections:

- Workflow APIs, on page 180
- WorkflowAction APIs, on page 197

Also, new Team APIs were added to allow administrators to view workflows for and assign workflows to specific teams. For more information, see the following sections:

- Team — Get List of Workflows, on page 177
- Team — Update List of Workflows, on page 179

For information about subscribing to workflow action events, see Workflow Action Event, on page 265.

Gadget Caching

In this release, gadget caching is enabled on the Finesse container. For more information, see Gadget Caching, on page 259.

Third-Party Gadget Changes

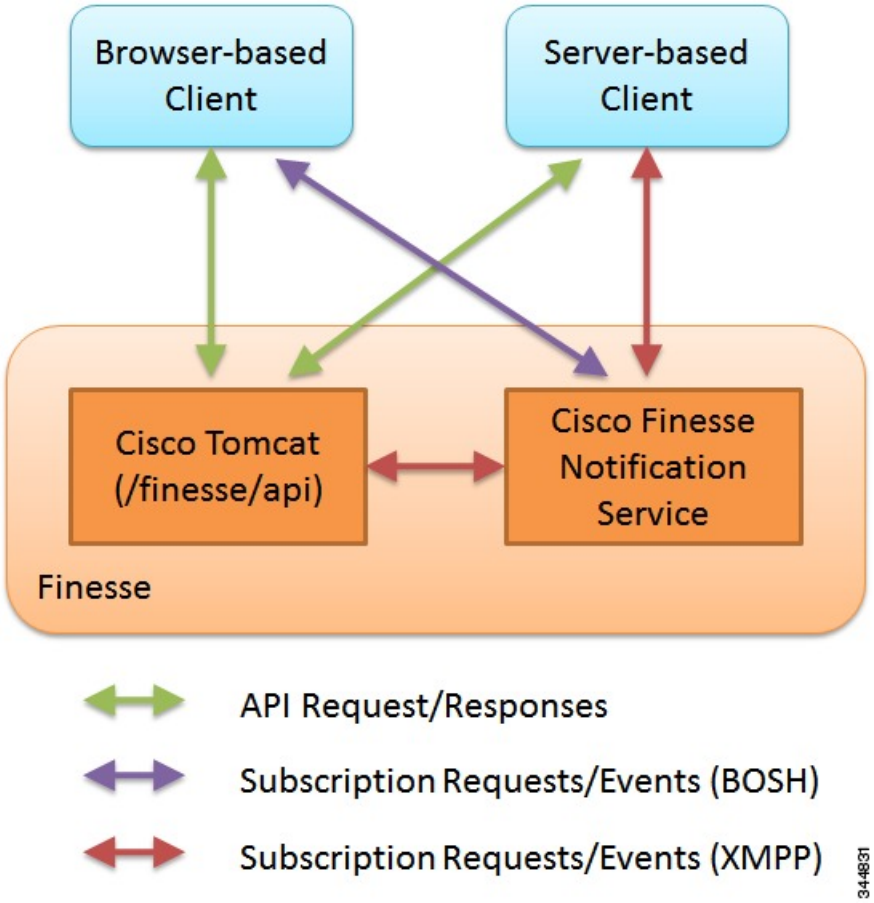
The CLI command to set or reset the 3rdpartygadget account has changed. For more information, see Password for 3rdpartygadget Account, on page 269.

Third-party gadgets are now migrated across upgrades and included in DRS backup and restore. For more information, see the following sections:

- Migration, on page 271
- Backup and Restore, on page 271

Communication with Cisco Finesse Web Service

Figure 1: Finesse API and Event Flow



The service names in the preceding diagram are specific to Unified CCE deployments. In a Unified CCX deployment, the service names are Cisco Finesse Tomcat and Cisco Unified CCX Notification Service.

Note The Finesse desktop supports receiving updates through BOSH only.

Client Requests

Cisco Finesse supports both HTTP and secure HTTP (HTTPS) requests from clients. Cisco Finesse Desktop operations can be performed using one of the many available REST-like HTTP/HTTPS requests described in this guide.

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 (HTTP) would be:

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

The URL to view a single object (HTTPS) would be:

```
https://<FQDN>:<port>/finesse/api/<object>/<objectID>
```

FQDN is the fully-qualified domain name of the Finesse server.

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, and 500) 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>
```

Finesse has a Dependency Manager that collects the state of internal dependencies for Finesse (such as the state of the Cisco Finesse Notification Service) and reports these states to external entities.

If any of these dependencies are down, Finesse is out of service. If the Cisco Tomcat is running, 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>
```

If the Cisco Tomcat service is not running, Finesse returns a Connection Timeout error.

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).

If an administrator changes the password for an agent or supervisor on the secondary Administration & Data server (if configured) while the primary distributor process on Unified CCE is down, the agent or supervisor can still use the old password and access all REST APIs except the sign-in request. To ensure this does not happen, the primary distributor must be up and running when the administrator changes the password.

HTTP Requests

In a Unified CCE deployment, clients should make all HTTP requests to port 80. In a Unified CCX deployment, clients should make all HTTP requests to port 8082.

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

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

HTTPS Requests

Clients should make all HTTPS requests to port 8443 in a Unified CCE deployment and port 8445 in a Unified CCX deployment. Most, but not all, Finesse desktop APIs conform to the following format:

```
https://<FQDN>:<port>/finesse/api/<object>
```

This document uses the HTTP request for all URIs and example URIs. If you want to use HTTPS requests, make the following changes to the URIs:

- Replace *http* with *https*.
- Use the fully-qualified domain name (FQDN) of the Finesse server instead of the IP address to avoid address mismatch errors. (The SSL certificate uses the Finesse hostname.)
- If Finesse is deployed with Unified CCE, use port 8443.
- If Finesse is deployed with Unified CCX, use port 8445.

Real-Time Events

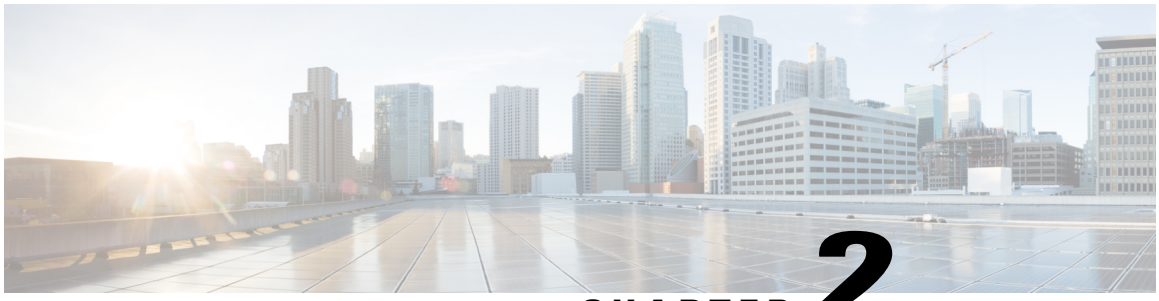
Real-time events (such as call events, state events, and so on) are sent by the Cisco Finesse Notification Service, using the XEP-0060 Publish-Subscribe extension of the XMPP (Extensible Messaging and Presence Protocol) protocol. Applications that need to communicate with the Notification Service must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

All real-time events are sent over HTTPS.

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

Client applications can communicate with the Cisco Finesse Notification Service through BOSH over HTTPS, using the binding URI https://<FQDN>:7443/http-bind. Developers can create their own BOSH library or use any that are available publicly, as documented on the Cisco Developer Network (see http://developer.cisco.com/web/cupapi/overview-of-interfaces).

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 Subscription Management).



CHAPTER 2

Lab Development Environment Validation with Cisco Finesse Web Services APIs

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

Environment and Tools

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

To complete these exercises, you need the following:

- A user who is configured as an agent in Unified CCE or Unified CCX (with an agent ID, password, and extension). Make the agent a member of a team and of a queue. (A queue is a 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.

Poster

Poster is an example of a REST client utility that allows you to send HTTP requests to a specific URL. You can use this utility in your lab to exercise the Finesse Web Service APIs 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 Poster may not work properly with HTTPS. If you want to test HTTPS requests, you can use RESTClient, which you can download from http://www.wiztools.org/. Enable HTTPS on RESTClient as follows:

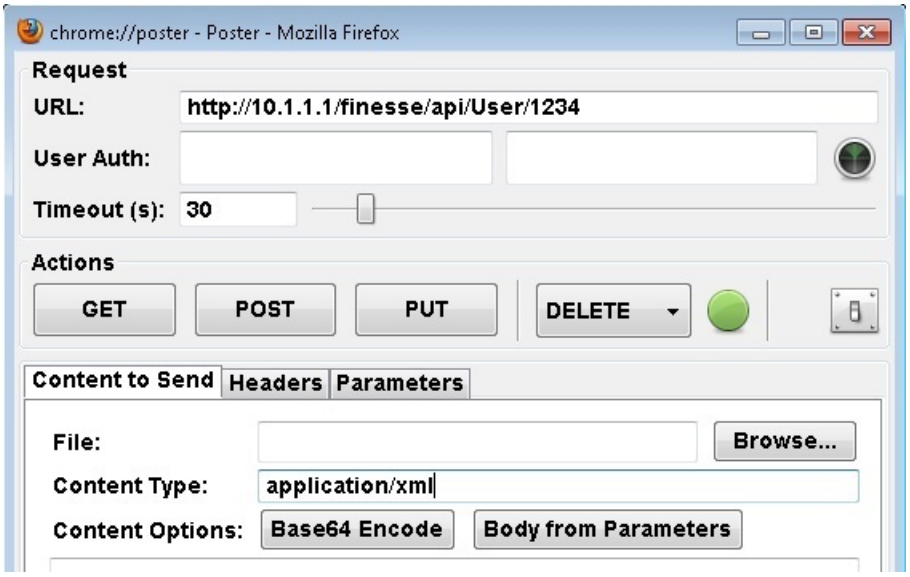
1. Click the **SSL** tab.
2. Click the **Etc** tab.
3. Ensure that the **Trust self-signed certificate** check box is checked.
4. Set the Hostname verifier to **Allow All**.

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 User — Sign In to Finesse API. Examine the pasted code for case sensitivity and format and remove any carriage returns.
2. Update the URI with the IP address 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

Pidgin

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.

Note You cannot be signed in to Pidgin at the same time you are signed in to Finesse as the XMPP event feed is disrupted.

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 change an agent's state, you can see the resulting agent state change 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:

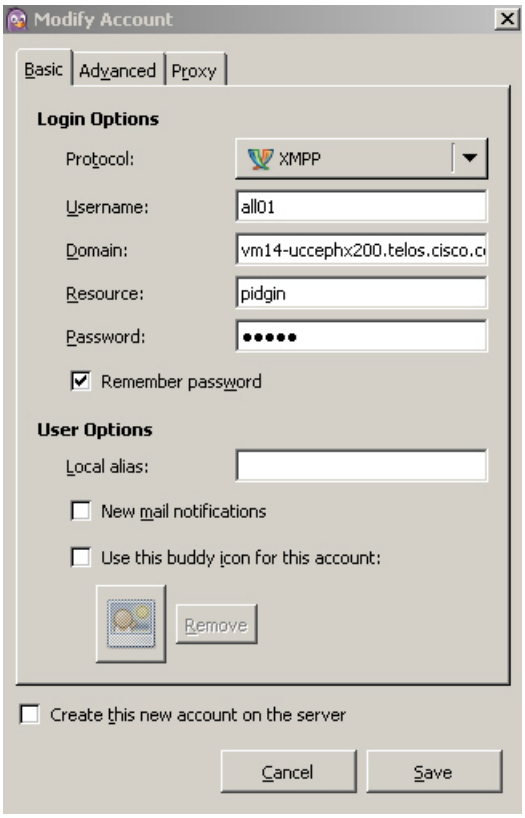
1. In Pidgin, go to **Tools > Plugins** to open the Plugins dialog box.
2. Check the **XMPP Console** and **XMPP Service Discovery** check boxes.

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).

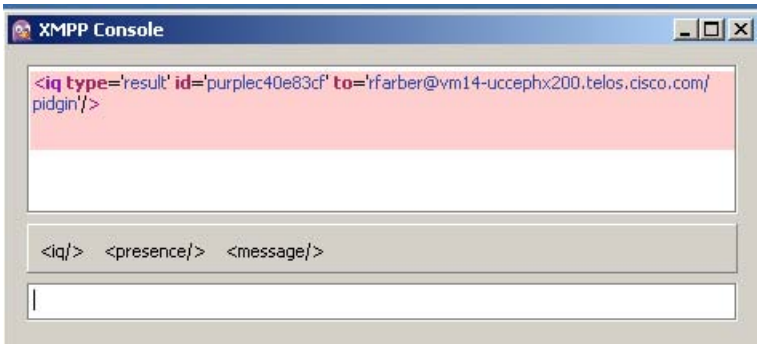
The XMPP logo next to the agent's name becomes active (is no longer dimmed). To see event messages in Pidgin, open the XMPP Console.

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 to update 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



Cisco Finesse APIs

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.

Sign In to Finesse

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

This example uses the following information:

- Finesse server FQDN: finesse1.xyz.com
- Agent name: John Smith
- Agent ID: 1234
- Agent password: 1001
- Agent extension: 1001

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

```
http://finesse1.xyz.com/finesse/api/User/1234
```

2. Enter the agent's ID (1234) and password (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://finesse1.xyz.com/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.

Change Agent State

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://finesse1.xyz.com/finesse/api/User/1234
```

2. Enter the agent's ID (1234) and password (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**.

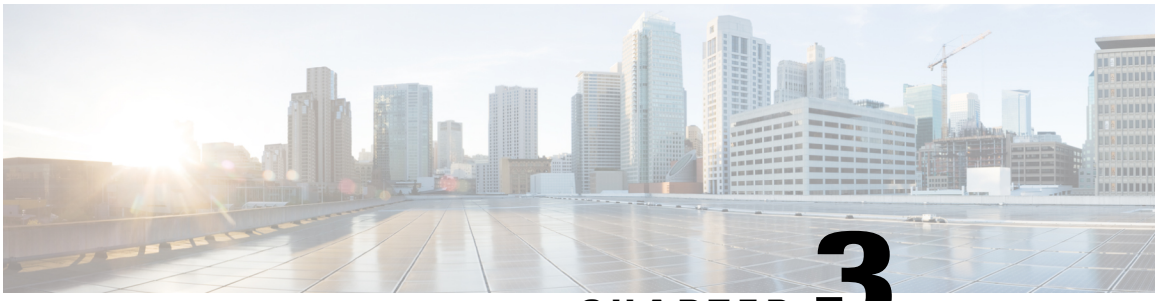
Poster returns the following response:

```
PUT on http://finesse1.xyz.com/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>
```



CHAPTER 3

Cisco Finesse Desktop APIs

Cisco Finesse comprises of a set of web APIs that communicate with Unified Contact Center Enterprise (Unified CCE) or Unified Contact Center Express (Unified CCX) to send and receive information about:

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

All Finesse web APIs are RESTful interfaces 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 HTTP Requests.

User APIs

The User object represents an agent or supervisor and includes user information, roles, state, dialogs, and more. Actions can be invoked on the User object to update the user's state.

User — Sign In to Finesse

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.

Note To sign in as a mobile agent, see User — Sign In to Finesse as a Mobile Agent, on page 18.

URI:	http://<FQDN>/finesse/api/User/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234`
Security Constraints:	Role: Agent or supervisor Limitations: Users can only act on their own User objects.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<User>`
	`  <state>LOGIN</state>`
	`  <extension>1001001</extension>`
	`</User>`

HTTP Response:	202-Successfully Accepted 400-Bad Request (for example, malformed or incomplete request, or invalid extension) 401-Unauthorized (for example, the user is not authenticated in the Web Session) 404-Not Found (for example, the user ID is not known) 503-Service Unavailable (for example, the notification service is not running)

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Invalid Authorization User Specified</ErrorType>`
	`  <ErrorData>4321</ErrorData>`
	`  <ErrorMessage>The user specified in the authentication credentials and the uri don't match</ErrorMessage>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Invalid Device • Generic Error • Authorization Failure • Invalid Authorization User Specified • User Not Found

	• Service Unavailable For descriptions and other possible error codes, see Cisco Finesse API Errors.
Notifications Triggered:	User Notifications

Platform-Based API Differences

Stand-alone Finesse with Unified CCE

Finesse does not support agent sign-in with an E.164 extension when Finesse is deployed with Unified CCE. However, agents can make calls to and receive calls from E.164 phone numbers.

Coresident Finesse with Unified CCX

Finesse does support agent sign-in with an E.164 extension when Finesse is deployed with Unified CCX. The maximum number of characters supported for an E.164 extension is 15 (a single plus sign followed by 14 digits).

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
id	String	Yes	The ID of the user	-	If the user is configured in Unified CCE, size is determined by Unified CCE. If the user is configured in Unified CCX, size is determined by Unified Communications Manager.
state	String	Yes	The new state that the user wants to be in	LOGIN	-

Parameter	Type	Required	Description	Possible Values	Validation
extension	String	Yes	The extension with which the user wants to sign in	-	The extension exists in Unified Communications Manager. If the user is configured in Unified CCE, size is determined by Unified Communications Manager. If the user is configured in Unified CCX, size is determined by Unified CCX.

User — Sign In to Finesse as a Mobile Agent

The User - Sign in to Finesse as a mobile agent API allows a user to sign in to the CTI server as a mobile agent.

Note Additional configuration is required on Unified CCE and Unified CM before a mobile agent can sign in. After using this API, you may need to perform additional steps to complete the sign-in. For more information, see the *Mobile Agent Guide for Cisco Unified Contact Center Enterprise & Hosted*.

Cisco Unified Mobile Agent (Unified MA) enables an agent using an PSTN phone and a broadband VPN connection (for agent desktop communications) to function just like a Unified CCE agent.

Note This API is supported only for Finesse deployments with Unified CCE. Unified CCX does not support mobile agents.

This API uses the existing User object with a LOGIN state only. The user must be authenticated to use this API successfully.

URI:	http://<FQDN>/finesse/api/User/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234`
Security Constraints:	Role: Agent or supervisor Limitations: Users can only act on their own User objects.

HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<User>`
	`  <state>LOGIN</state>`
	`  <extension>1001001</extension>`
	`  <mobileAgent>`
	`  <mode>CALL_BY_CALL</mode>`
	`  <dialNumber>4085551234</dialNumber>`
	`  </mobileAgent>`
	`</User>`

HTTP Response:	202-Successfully Accepted **Note** This response only indicates successful completion of the request. The request is processed and the actual response is sent as part of a User notification. 400-Invalid Input (for example, the mode provided is invalid) 400-Parameter missing (for example, the mode or dialNumber is not provided) 400-Generic error 401-Unauthorized (for example, the user is not authenticated in the Web Session) 401-Invalid authorization user specified (an authenticated user tried to make a request for another user) 404-User not found (for example, the agent is not recognized)

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Invalid Authorization User Specified</ErrorType>`
	`  <ErrorData>4321</ErrorData>`
	`  <ErrorMessage>The user specified in the authentication credentials and the uri don't match</ErrorMessage>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Invalid Input • Parameter Missing • Generic Error • Authorization Failure • Invalid Authorization User Specified • User Not Found For descriptions and other possible error codes, see Cisco Finesse API Errors.
Notifications Triggered:	User Notifications

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
id	String	Yes	The ID of the user.	-	The user is configured in Unified CCE. Size determined by Unified CCE.
state	String	Yes	The new state that the user wants to be in.	LOGIN	-
extension	String	Yes	The extension with which the user wants to sign in.	-	The extension exists in Unified CM. Size is determined by Unified CM.
mobileAgent	Object	Required for mobile agents	Indicates that the user is a mobile agent.	-	-
mode	String	Required for mobile agents	The connection mode for a mobile agent. In CALL_BY_CALL mode, the agent's phone is dialed for each incoming call. In NAILED_CONNECTION mode, the agent is called when the agent signs in. The line remains connected through multiple customer calls.	CALL_BY_CALL, NAILED_CONNECTION	
dialNumber	Integer	Required for mobile agents	The phone number that the system calls to connect with a mobile agent.	-	Validated by the Unified CM dial plan.

User — Sign out of Finesse

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

URI:	http://<FQDN>/finesse/api/User/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234`
Security Constraints:	Role: Agent or supervisor Limitations: Users can only act on their own User objects.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<User>`
	`  <state>LOGOUT</state>`
	`</User>`

HTTP Response:	202-Successfully accepted 400-Bad Request (for example, malformed or incomplete request, or invalid extension) 401-Unauthorized (for example, the user is not authenticated in the Web Session) 404-Not Found (for example, the user ID is not known) 503-Service Unavailable (for example, the notification service is not running)

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Invalid Input</ErrorType>`
	`  <ErrorData>state</ErrorData>`
	`  <ErrorMessage>Invalid State specified for user</ErrorMessage>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Invalid State • Authorization Failure • Invalid Authorization User Specified • User Not Found • Internal Server Error • Service Unavailable

	For descriptions and other possible error codes, see Cisco Finesse API Errors.
Notifications Triggered:	User Notifications

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
id	String	Yes	The ID of the user.	-	The user is configured in Unified CCE. Size determined by Unified CCE.
state		Yes	The new state that the user wants to be in.	LOGOUT	-

User — Get User

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

For a mobile agent, this API returns the full user object, including the mobileAgent node.

Note Mobile agent information is available to the Finesse node that the mobile agent signed in to. However, the other Finesse node in the cluster does not have the mobile agent data. If the agent signs in to the other node (for example, during a client failover scenario), the mobile agent information is lost and the User object does not return any mobile agent data fields. As a result, the Finesse Desktop user interface inaccurately represents the mobile agent as a regular agent, including all related features. Additionally, any other type of CTI failover results in Finesse losing the current mobile agent information.

As a workaround for this limitation, have the agent sign out and sign back in as a mobile agent. Note that the Unified Mobile Agent feature behaves as normal whether Finesse knows the agent is functioning as a mobile agent or not.

URI:	http://<FQDN>/finesse/api/User/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234`
Security Constraints:	Role: Agent, administrator

	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 401-Unauthorized (for example, the user is not authenticated in the Web Session) 404-Not Found (for example, the user ID is not known) 500-Runtime exception 503-Service Unavailable (for example, the notification service is not running)
Successful Response:	<User> <uri>/finesse/api/User/1234</uri> <roles> <role>Agent</role> <role>Supervisor</role> </roles> <loginId>{id}</loginId> <loginName>csmith</loginName> <state>NOT_READY</state> <stateChangeTime>2012-03-01T17:58:21Z</stateChangeTime> <pendingState></pendingState> <reasonCodeId>2</reasonCodeId> <extension>1001001</extension> <firstName>Chris</firstName> <lastName>Smith</lastName> <teamId>500</teamId> <teamName>Sales</teamName> <dialogs>/finesse/api/User/1234/Dialogs</dialogs> <teams> <Team> <uri>/finesse/api/Team/{id}</uri> <id>{team-id}</id> <name>First Line Support</name> </Team> <Team> <uri>/finesse/api/Team/{id}</uri> <id>{team-id}</id> <name>Second Line Support</name> </Team> <Team> <uri>/finesse/api/Team/{id}</uri> <id>{team-id}</id> <name>Third Line Support</name> </Team> ... other teams ... </teams> <Subscriptions> <subscription>/finesse/api/User/1234</subscription> </Subscriptions> </User>
Successful Response (Mobile Agent):	<User> ... Full User Object ... <mobileAgent> <mode>CALL_BY_CALL</mode>

Note Mobile Agent applies only to Unified CCE deployments.	<dialNumber>4085551234</dialNumber> </mobileAgent> </User>
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>User Not Found</ErrorType> <ErrorMessage>UNKNOWN_USER</ErrorMessage> <ErrorData>4023</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Authorization Failure • Invalid Authorization User Specified • User Not Found • Internal Server Error • Service Unavailable For descriptions and other possible error codes, see Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
uri	String	The URI to get a new copy of the User object.
roles	String	A list of roles assigned to a user.
role	String	One of the roles assigned to a user (for example, agent, supervisor, administrator).
loginId	String	The login ID of the user.
loginName	String	The login name of the user.
state	String	The state of the user. Possible values: LOGOUT, NOT_READY, READY, RESERVED, RESERVED_OUTBOUND, RESERVED_OUTBOUND_PREVIEW, TALKING, HOLD, WORK, WORK_READY, UNKNOWN For more information about WORK and WORK_READY states, see WORK and WORK_READY User States. For more information about RESERVED_OUTBOUND state, see RESERVED_OUTBOUND User State. For more information about RESERVED_OUTBOUND_PREVIEW state, see RESERVED_OUTBOUND_PREVIEW User State.

Parameter	Type	Description
stateChangeTime	String	The time that the state of the user changed to the current state. Format: YYYY-MM-DDTHH:MM:SSZ **Note** This parameter is empty when the time of the state change is not available (if no agent state change event was received yet).
pendingState	String	The state the user will transition to next. In a Unified CCE deployment, this parameter is empty. In a Unified CCX deployment, if an agent is in TALKING state and a Finesse failover/reconnect occurs, this parameter is set to LOGOUT. The parameter indicates that the agent will transition to LOGOUT state when the call ends.
reasonCodeId	Integer	The ID of the reason code associated with the current state of the agent (if applicable). If there is no reason codes defined in Finesse, the value of this parameter is -1. The value is also -1 when a user first signs in to Finesse because the system does not have a reason code for the initial Not Ready state.
extension	String	The extension the user is currently using.
firstName	String	The first name of the user.
lastName	String	The last name of the user.
dialogs	String	The URI for the list of dialogs in which the user is a participant.
Team	XML	One set of team information.
teamId	String	The ID of the team to which the user belongs.
teamName	String	The name of the team to which the user belongs.
teams	XML	A list of teams that a user supervises (applies only to users who are assigned a role of supervisor).
id	String	The unique identifier for a user or team.
name	String	The name of the team.
mobileAgent	Object	Indicates that the user is a mobile agent. This parameter is returned for mobile agents only.

Parameter	Type	Description
mode	String	The connection mode for a mobile agent (CALL_BY_CALL or NAILED_CONNECTION). This parameter is returned for mobile agents only.
dialNumber	Integer	The phone number the system calls to connect to a mobile agent. This parameter is returned for mobile agents only.

User — Get List of Users

The User - Get list of users API allows an administrator to get a list of users.

URI:	http://<FQDN>/finesse/api/Users
Example URI:	`http://finesse1.xyz.com/finesse/api/Users`
Security Constraints:	Role: Administrator Limitations: Any administrator can use this API
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-
HTTP Response:	200-Success 401-Unauthorized (for example, the user is not authenticated in the Web Session) 500-Internal server error 503-Service Unavailable (for example, the notification service is not running)

Successful Response:	<Users> <User> ... Full User Object ... </User> <User> ... Full User Object ... </User> <User> ... Full User Object ... </User> <User> ... Full User Object ... </User> <User> ... Full User Object ... </User> ... Additional Users... </Users>
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Unauthorized</ErrorType> <ErrorMessage>The user is not authorized to perform this operation</ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	• Authorization Failure • Invalid Authorization User Specified • User Not Found • Internal Server Error • Service Unavailable For descriptions and other possible error codes, see Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
uri	String	The URI to get a new copy of the User object.
roles	String	A list of roles assigned to a user.
role	String	One of the roles assigned to a user (for example, agent, supervisor, administrator).
loginId	String	The login ID of the user.
loginName	String	The login name of the user.

Parameter	Type	Description
state	String	The state of the user. Possible values: LOGOUT, NOT_READY, READY, RESERVED, RESERVED_OUTBOUND, RESERVED_OUTBOUND_PREVIEW, TALKING, HOLD, WORK, WORK_READY, UNKNOWN For more information about WORK and WORK_READY states, see WORK and WORK_READY User States. For more information about RESERVED_OUTBOUND state, see RESERVED_OUTBOUND User State. For more information about RESERVED_OUTBOUND_PREVIEW state, see RESERVED_OUTBOUND_PREVIEW User State.
stateChangeTime	String	The time that the state of the user changed to the current state. Format: YYYY-MM-DDTHH:MM:SSZ **Note** This parameter is empty when the time of the state change is not available (if no agent state change event was received yet).
pendingState	String	The state the user will transition to next. In a Unified CCE deployment, this parameter is empty. In a Unified CCX deployment, if an agent is in TALKING state and a Finesse failover/reconnect occurs, this parameter is set to LOGOUT. The parameter indicates that the agent will transition to LOGOUT state when the call ends.
reasonCodeId	Integer	The ID of the reason code associated with the current state of the agent (if applicable).
extension	String	The extension the user is currently using.
firstName	String	The first name of the user.
lastName	String	The last name of the user.
dialogs	String	The URI for the list of dialogs in which the user is a participant.
Team	XML	One set of team information.
teamId	String	The ID of the team to which the user belongs.
teamName	String	The name of the team to which the user belongs.

Parameter	Type	Description
teams	XML	A list of teams that a user supervises (applies only to users who are assigned a role of supervisor).
id	String	The unique identifier for a user or team.
name	String	The name of the team.
mobileAgent	Object	Indicates that the user is a mobile agent. This parameter is returned for mobile agents only.
mode	String	The connection mode for a mobile agent (CALL_BY_CALL or NAILED_CONNECTION). This parameter is returned for mobile agents only.
dialNumber	Integer	The phone number the system calls to connect to a mobile agent. This parameter is returned for mobile agents only.

User — Get List of Dialogs Associated with User

The User - Get list of dialogs associated with 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://<FQDN>/finesse/api/User/<id>/Dialogs
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/Dialogs`
Security Constraints:	Role: Agent, administrator 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 401-Unauthorized (for example, the user is not authenticated in the Web Session) 500-Internal server error

Successful Response:	<Dialogs> <Dialog> ... Full Dialog Object ... </Dialog> <Dialog> ... Full Dialog Object ... </Dialog> <Dialog> ... Full Dialog Object ... </Dialog> <Dialog> ... Full Dialog Object ... </Dialog> <Dialog> ... Full Dialog Object ... </Dialog> ... Additional Dialogs... </Dialogs>
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Authorization Failure • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

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

Response Parameters

Parameter	Type	Description
uri	String	The URI to get a new copy of the Dialog object.
mediaType	String	The type of media under which a dialog is classified. Possible values: voice, email, chat.
state	String	The last state of this dialog. For a list of possible values, see State (Dialog) Parameter Values.
stateCause	String	The cause for the last participant state in a dialog. This parameter is normally associated with a FAILED participant state. Possible values: BUSY, BAD_DESTINATION, OTHER
fromAddress	String	The calling line ID of the caller.

Parameter	Type	Description
toAddress	String	The destination for the call.
mediaAddress	String	Point of contact for this participant. 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.
dnis	String	The DNIS provided with the call. For routed calls, the DNIS is the route point.
dialedNumber	String	The number dialed.
Participants	XML	A list of all participants, both internal and external, involved in a dialog.
Participant	XML	A list of information about a participant in a dialog.
actions	XML	A list of actions that are allowed for the participant as a result of the dialog update. For a list of possible values, see Actions Parameter Values.
mediaProperties	XML	Includes all of the properties for a call that corresponds to the dialog.
callvariables	Collection	A list of up to 10 call variables and ECC variables.

Parameter	Type	Description
CallVariable	XML	Contains the name and value of a call variable or ECC variable belonging to the dialog. The name indicates whether the variable is a call variable or ECC variable, based on naming conventions. Call variable names start with callVariable#, where # is 1-10. 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. The following call variables provide further details about an Outbound Option call: BACampaign, BAAccountNumber, BAResponse, BAStatus, BADialedListID, BATimeZone, BABuddyName. The BAStatus variable contains one of the following values: • PREDICTIVE_OUTBOUND (if the agent is on a Predictive Outbound Option call) • PROGRESSIVE_OUTBOUND (if the agent is on a Progressive Outbound Option call) • PREVIEW_OUTBOUND_RESERVATION (if the agent is reserved for a Preview Outbound Option call) • PREVIEW_OUTBOUND (if the agent is on a Preview Outbound Option call)

User — Change Agent State

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

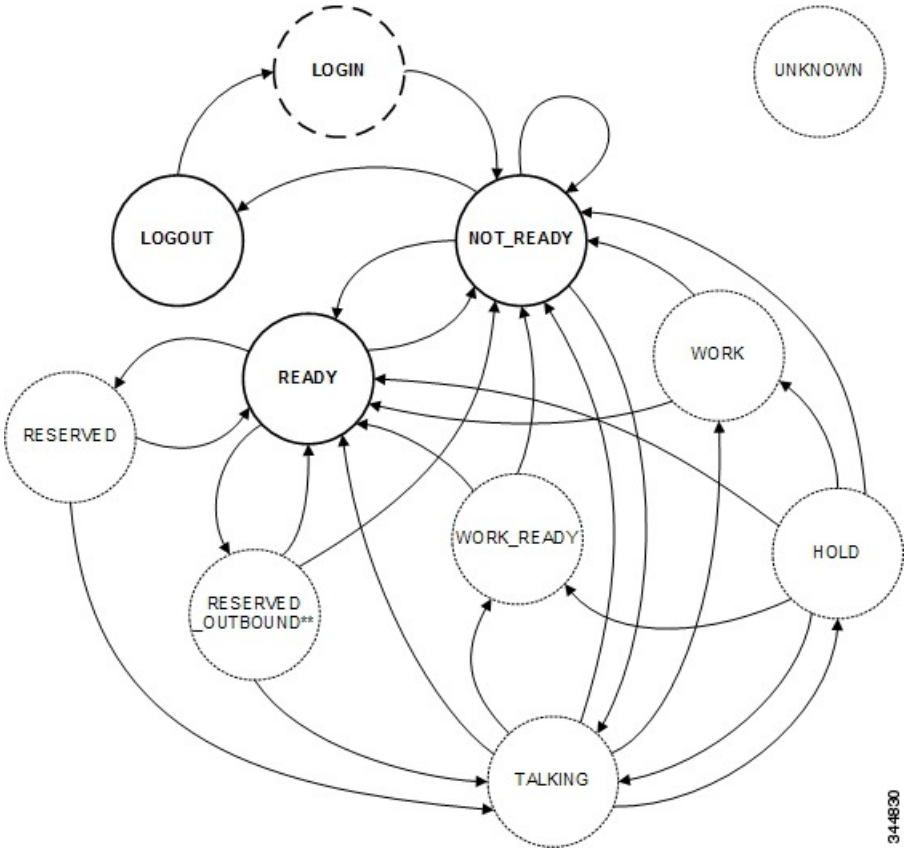
This API also allows supervisors to change the state of agents or other supervisors who belong to their teams.

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 by agents for Unified CCE.

Note This diagram contains only logical state transitions. Because the underlying system determines the state, an agent can transition from any state to any state, especially under a failover condition. This diagram describes the typical state changes that occur in the system.

Figure 7: Supported State Transitions by Agent (Unified CCE)



Note In the preceding diagram, RESERVED_OUTBOUND can represent RESERVED_OUTBOUND or RESERVED_OUTBOUND_PREVIEW state.

From	To	Description
*	UNKNOWN	If the agent state is unknown, the state is UNKNOWN. This scenario is unlikely.
LOGOUT	LOGIN	To sign in to Finesse, the agent sets the state to LOGIN. LOGIN is a transient state and transitions to NOT_READY.

From	To	Description
LOGIN	NOT_READY	After a successful LOGIN, the agent transitions to NOT_READY.
NOT_READY	LOGOUT	To sign out of Finesse, the agent sets the state to LOGOUT. An agent can set the state to LOGOUT only if that agent is in NOT_READY state.
NOT_READY	NOT_READY	To change their Not Ready reason code, agents can set a NOT_READY state from NOT_READY.
NOT_READY	READY	To become available for incoming or Outbound Option calls, agents set their state to READY.
NOT_READY	TALKING	An agent who places a call while in NOT_READY state transitions to TALKING.
READY	RESERVED	An incoming call arrives at an agent.
READY	RESERVED _OUTBOUND	An outbound agent becomes reserved to handle an Outbound Option Progressive or Predictive call.
READY	RESERVED_OUTBOUND _PREVIEW	An outbound agent becomes reserved to handle an Outbound Option Preview call.
READY	NOT_READY	Agents can change to NOT_READY to make themselves unavailable for incoming calls.
RESERVED	READY	An agent can become RESERVED but never take a call.
RESERVED	TALKING	When an agent answers an incoming call, the agent transitions to TALKING.
RESERVED _OUTBOUND	READY	An agent can change to READY state to leave RESERVED_OUTBOUND. If the system deems it necessary, that agent may transition back to RESERVED_OUTBOUND.
RESERVED _OUTBOUND	NOT_READY	An agent can change to NOT_READY state to leave RESERVED_OUTBOUND.
RESERVED _OUTBOUND	TALKING	An agent transitions to TALKING when an Outbound Option call arrives at the agent.
RESERVED_OUTBOUND _PREVIEW	READY	An agent transitions to READY if the agent was in READY state before being reserved in an Outbound Option Preview campaign.

From	To	Description
RESERVED_OUTBOUND_PREVIEW	NOT_READY	An agent transitions to NOT_READY if that agent changes state to NOT_READY while reserved in an Outbound Option Preview campaign. This state change is a pending state change. The agent does not transition to NOT_READY until the call is complete or the Outbound Option Preview reservation is closed or rejected.
RESERVED_OUTBOUND_PREVIEW	TALKING	An agent transitions to TALKING when an Outbound Option call arrives at the agent.
TALKING	READY	If an agent is on a call that is dropped, the agent transitions to READY (if the agent was in READY state before the call).
TALKING	NOT_READY	If an agent is on a call that is dropped, the agent transitions to NOT_READY if that agent was in NOT_READY state before the call.
TALKING	WORK	If wrap-up is enabled, and the agent chooses NOT_READY while on a call, that agent enters WORK state after the call is dropped.
TALKING	WORK_READY	If wrap-up is enabled, an agent enters WORK_READY state after a call is dropped.
TALKING	HOLD	An agent puts a call on hold and transitions to HOLD state.
HOLD	READY	If an agent is connected to a held call and the call is dropped, the agent transitions to READY state (if the agent was in READY state before the call).
HOLD	NOT_READY	If an agent is connected to a held call and the call is dropped, the agent transitions to NOT_READY state (if the agent was in NOT_READY state before the call).
HOLD	WORK	If wrap-up is enabled and an agent is connected to a held call that is dropped, the agent transitions to WORK state if the agent chose to go NOT_READY during the call.
HOLD	WORK_READY	If wrap-up is enabled and an agent is connected to a held call that is dropped, the agent transitions to WORK_READY state.
HOLD	TALKING	When an agent retrieves a held call, the agent transitions to TALKING state.
WORK	READY	To leave WORK state, agents can set their state to READY.

From	To	Description
WORK	NOT_READY	To leave WORK state, agents can set their state to NOT_READY. Agents automatically transition to NOT_READY after the wrap-up timer expires.
WORK_READY	READY	To leave WORK_READY state, agents can set their state to READY. Agents automatically transition to READY after the wrap-up timer expires.
WORK_READY	NOT_READY	To leave WORK_READY state, agents can set their state to NOT_READY.

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
- RESERVED_OUTBOUND
- TALKING
- HOLD
- WORK
- WORK_READY

A user may transition from TALKING or HOLD to WORK or WORK_READY. These transitions occur when a user who is configured to wrap-up leaves a call. For more information about the WORK and WORK_READY states, see WORK and WORK_READY User States, on page 229.

Users who are configured in Outbound Option skill groups transition from the READY state to the RESERVED_OUTBOUND state when those users are reserved for Outbound Option calls. If a user wants to exit the RESERVED_OUTBOUND, that user can change to READY or NOT_READY state. If the user does nothing and then gets transferred to a customer, the user transitions to TALKING state. If the user does not get transferred to a customer, the user transitions back to READY state. For more information, see RESERVED_OUTBOUND User State, on page 229.

Note The following statements apply to a supervisor using the API to change the state of an agent or other supervisor:

- A supervisor can only change the state of an agent who is assigned to that supervisor's team.
- A supervisor can only set the state of another user to READY or LOGOUT.
- A supervisor can set the state of an agent to READY only if that agent is in RESERVED, TALKING, or HOLD state.
- A supervisor can set the state of an agent to LOGOUT only if that agent is in READY, NOT_READY, RESERVED, RESERVED_OUTBOUND, RESERVED_OUTBOUND_PREVIEW, TALKING, HOLD, WORK, or WORK_READY state.

The following table lists supported state transitions by an agent for Unified CCX.

From	To	Description
LOGIN	NOT_READY	After a successful LOGIN, the agent transitions to NOT_READY.
NOT_READY	LOGOUT	To sign out of Finesse, the agent sets the state to LOGOUT.
NOT_READY	NOT_READY	To change their Not Ready reason code, agents can set a NOT_READY state from NOT_READY.
NOT_READY	READY	To become available for incoming calls, agents set their state to READY.
READY	NOT_READY	Agents can change their state to NOT_READY to make themselves unavailable for incoming calls.
READY	LOGOUT	To sign out of Finesse, agents set their state to LOGOUT.

URI:	http://<FQDN>/finesse/api/User/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234`
Security Constraints:	Role: Agent or supervisor Limitations: Agents can only act on their own User objects. Supervisors can act on the User objects of agents who belong to their team.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<User> <state>READY</state> </User>
HTTP Response:	202-Successfully accepted 400-Bad request 401-Invalid supervisor (a supervisor tried to change the state of an agent who did not belong to that supervisor's team) 401-Unauthorized (for example, the user is not authenticated in the Web Session) 404-Not Found (for example, the user ID is not known) 500-Internal server error 503-Service Unavailable (for example, the notification service is not running)
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Parameter Missing</ErrorType> <ErrorData>state</ErrorData> <ErrorMessage>State Parameter missing</ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	• Parameter Missing • Invalid Input • Invalid State • Invalid Supervisor • Authorization Failure • Invalid Authorization User Specified • User Not Found • Internal Server Error • Service Unavailable For descriptions and other possible error codes, see Cisco Finesse API Errors.
Notifications Triggered:	User Notifications

Platform-Based API Differences

The following table describes API differences between a stand-alone Finesse deployment with Unified CCE or a coresident Finesse deployment with Unified CCX.

Scenario	Response
Change from LOGOUT to NOT_READY.	**Stand-alone Finesse with Unified CCE:** <data> <apiErrors> <apiError> <errorData>257</errorData> <errorMessage>CF_INVALID_PASSWORD_SPECIFIED</errorMessage> <errorType>Invalid State</errorType> </apiError> </apiErrors> </data> **Coresident Finesse with Unified CCX:** <data> <apiErrors> <apiError> <errorData>1010</errorData> <errorMessage>CF_INVALID_PARAMETER</errorMessage> <errorType>Invalid State</errorType> </apiError> </apiErrors> </data>
Agent receives and answers a non-ICD call.	**Stand-alone Finesse with Unified CCE:** Finesse sends a User notification with state=TALKING. **Coresident Finesse with Unified CCX:** Finesse does not send a User notification. The agent remains in NOT_READY state.
Agent puts an ICD call on hold.	**Stand-alone Finesse with Unified CCE:** Finesse sends a User notification with state=HOLD. **Coresident Finesse with Unified CCX:** Finesse does not send a User notification. The agent remains in TALKING state.
While talking on an ICD call, the agent sets a pending state of READY.	**Stand-alone Finesse with Unified CCE:** Agent transitions to READY state after the call ends. **Coresident Finesse with Unified CCX:** Unified CCX does not allow an agent to set a pending state of READY while that agent is talking on an ICD call. <data> <apiErrors> <apiError> <errorData>265</errorData> <errorMessage>CF_INVALID_AGENT_WORKMODE</errorMessage> <errorType>Invalid State</errorType> </apiError> </apiErrors> </data>

Scenario	Response
While talking on a non-ICD call (agent state can be READY or NOT_READY), the agent sets a pending state of READY.	**Stand-alone Finesse with Unified CCE:** Agent transitions to READY state after the call ends. **Coresident Finesse with Unified CCX:** Unified CCX does not allow an agent to set a pending state of READY while that agent is talking on a non-ICD call. <data> <apiErrors> <apiError> <errorData>33</errorData> <errorMessage>CF_RESOURCE_BUSY</errorMessage> <errorType>Invalid State</errorType> </apiError> </apiErrors> </data>
While talking on an ICD call, the agent attempts to change from a pending state of NOT_READY with reason code 1 to a pending state of NOT_READY with reason code 2.	**Stand-alone Finesse with Unified CCE:** Agent transitions to READY state with reason code 2 after the call ends. **Coresident Finesse with Unified CCX:** Unified CCX allows an agent to set a pending state of NOT_READY only once during a call. Unified CCX does not allow an agent to change from one Not Ready reason code to another. <data> <apiErrors> <apiError> <errorData>265</errorData> <errorMessage>CF_INVALID_AGENT_WORKMODE</errorMessage> <errorType>Invalid State</errorType> </apiError> </apiErrors> </data>

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
id	String	Yes	The ID of the user.	-	The user is configured in Unified CCE or Unified CCX.
state	String	Yes	The new state the user wants to be in.	LOGOUT, READY, NOT_READY	-

User — Change Agent State with Reason Code

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 Not Ready or Logout). Users must be authenticated to successfully use this API.

URI:	http://<FQDN>/finesse/api/User/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234`
Security Constraints:	Role: Agent Limitations: Users can only act on their own User objects.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<User>`
	`  <state>NOT_READY</state>`
	`  <reasonCodeId>1001</reasonCodeId>`
	`</User>`

HTTP Response:	202-Successfully accepted 400-Parameter Missing 400-Invalid Input 400-Invalid State 401-Authorization Failure (for example, the user is not authenticated in the Web Session) 401-Invalid Authorization User Specified (for example, the authenticated user tried to make a request as another user) 404-User Not Found (the agent ID provided is invalid and no such agent exists in CTI) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Parameter Missing</ErrorType>`
	`  <ErrorData>state</ErrorData>`
	`  <ErrorMessage>State Parameter missing</ErrorMessage>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Invalid State • Authorization Failure

	• Invalid Authorization User Specified • User Not Found • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.
Notifications Triggered:	User Notifications

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
id	String	Yes	The ID of the user.	-	Maximum length of 12 characters.
reasonCodeId	Integer	Yes, if reason codes are configured for the given state. Otherwise, no.	An empty string for no reason code or a database ID for the reason code.	-	Only for a state change to NOT_READY or LOGOUT. The value must correspond to a database ID. Logout does not prevent state change with an invalid reasonCodeId.
state	String	Yes	The new state the user wants to be in.	NOT_READY, LOGOUT	-

User — Get Reason Code

The User - Get reason code API allows a user (agent or supervisor) to get an individual not ready or sign out reason code, which is already defined and stored in the Finesse configuration database (and that is applicable to the user). Users can display the reason code label on their desktop when they change their state to Not Ready or Logout respectively.

URI:	http://<FQDN>/finesse/api/User/<id>/ReasonCode/<reasoncodeId>

Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/ReasonCode/12`
Security Constraints:	Role: Agent, Supervisor, or Administrator 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>`
	`<uri>/finesse/api/ReasonCode/1</uri>`
	`<category>NOT_READY</category>`
	`<code>12</code>`
	`<label>Lunch</label>`
	`<forAll>true</forAll>`
	`</ReasonCode>`

HTTP Response:	200-Success 400-Bad request 400-Finesse API error (for example, the object does not exist, the object is stale, violation of DB constraint, and so on) 401-Authorization failure 401-Invalid Authorization User Specified 404-Not Found (for example, the reason code does not exist or has been deleted) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorType>Authorization Failure</ErrorType>`
	`<ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`<ErrorData>1234</ErrorData>`
	`</ApiError>`
	`</ApiErrors>`

Error Codes:	• Authorization Failure • Invalid Authorization User Specified • Not Found • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
category	String	The category of the reason code (NOT_READY or LOGOUT).
code	Integer	The value of the reason code.
forAll	String	Whether the reason code applies to all agents (possible values are true or false).
label	String	The UI label for the reason code (for example, Lunch, End of Shift).
uri	String	The URI to get a new copy of the ReasonCode object.

User — Get Reason Code List

The User - Get reason code list API allows a user (an agent or supervisor) to get a list of not ready or sign out 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 (NOT_READY or LOGOUT) reason codes are returned.

URI:	http://<FQDN>/finesse/api/User/<id>/ReasonCodes?category=NOT_READY|LOGOUT
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/ReasonCodes?category=NOT_READY`
Security Constraints:	Role: Agent, Supervisor, or Administrator 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:	<pre><ReasonCodes category="NOT_READY">
	  <ReasonCode>
	  <uri>/finesse/api/ReasonCode/1</uri>
	  <category>NOT_READY</category>
	  <code>12</code>
	  <label>Lunch</label>
	  <forAll>true</forAll>
	  </ReasonCode>
	  <ReasonCode>
	  ... Full ReasonCode Object ...
	  </ReasonCode>
	  <ReasonCode>
	  ... Full ReasonCode Object ...
	  </ReasonCode>
	</ReasonCodes></pre>

HTTP Response:	200-Success 400-Bad request 400-Finesse API error (for example, the object does not exist, the object is stale, violation of DB constraint, and so on) 401-Authorization failure 401-Invalid Authorization User Specified 404-Not Found (for example, the reason code does not exist or has been deleted) 500-Internal server error

Failure Response Example:	<pre><ApiErrors>
	  <ApiError>
	  <ErrorType>Authorization Failure</ErrorType>
	  <ErrorMessage>UNAUTHORIZED</ErrorMessage>
	  <ErrorData>1234</ErrorData>
	  </ApiError>
	</ApiErrors></pre>

Error Codes:	• Authorization Failure • Invalid Authorization User Specified • Not Found • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

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

Note All reason codes that have the *forAll* parameter set to true, will apply to any user.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
category	String	Yes	The category of reason code to retrieve.	NOT_READY, LOGOUT	-

Response Parameters

Parameter	Type	Description
ReasonCodes	XML	Represents a list of ReasonCode objects.
category	String	The category of the reason code (NOT_READY or LOGOUT).
ReasonCode	XML	A ReasonCode object. This object can have a category of NOT_READY or LOGOUT, a descriptive label, and a numeric code value.

User — Get Wrap-Up Reason

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

URI:	http://<FQDN>/finesse/api/User/<userId>/WrapUpReason/<wrapUpReasonId>
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/WrapUpReason/1001`
Security Constraints:	Role: Agent, Supervisor, or Administrator Limitations: A user must be signed in to get a wrap-up reason. A user cannot retrieve wrap-up reasons that belong to another user.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-

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

Response Parameters

Parameter	Type	Description
uri	String	The URI to get a new copy of the wrap-up reason object.
label	String	The user interface label for the wrap-up reason (for example, Sales Call).
forAll	Boolean	Whether a wrap-up reason applies to all agents. Possible values: true or false

User — Get Wrap-Up Reason List

The User-Get wrap-up reason 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://<FQDN>/finesse/api/User/<userId>/WrapUpReasons
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/WrapUpReasons`
Security Constraints:	Role: Agent, supervisor, or administrator Limitations: A user must be signed in to get a list of wrap-up reasons. A user cannot retrieve wrap-up reasons that belong to another user.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-

Successful Response:	`<WrapUpReasons>`
	`  <WrapUpReason>`
	`  <label>Successful tech support call</label>`
	`  <forAll>true</forAll>`
	`  <uri>/finesse/api/User/1234/WrapUpReason/205</uri>`
	`  </WrapUpReason>`
	`  ... more wrap-up reasons ...`
	`</WrapUpReasons>`

HTTP Response:	200-Success 400-Bad request 400-Finesse API error (for example, the object does not exist, the object is stale, violation of DB constraint, and so on) 401-Authorization failure 401-Invalid Authorization User Specified 404-User Not Found (the user id provided is invalid and no such agent exists in CTI) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>1234</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Authorization Failure • Invalid Authorization User Specified • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
uri	String	The URI to get a new copy of the wrap-up reason object.
label	String	The user interface label for the wrap-up reason (for example, Sales Call).
forAll	Boolean	Whether a wrap-up reason applies to all agents. Possible values: true or false

User — Get Media Properties Layout

The User - Get media properties layout 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 supports a single default instance only.

Finesse supports the following media properties:

- call variables (callVariable1 through callVariable10)
- ECC variables

URI:	http://<FQDN>/finesse/api/User/<userId>/MediaPropertiesLayout
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/MediaPropertiesLayout`
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-

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

Response Parameters

Parameter	Type	Description
header	Entry object	A single entry (combination of displayName and mediaProperty) that appears in the call header on the desktop for each call.

Parameter	Type	Description
column	List of entry objects	Grouping of media properties for agent and supervisor desktops. Finesse 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.
entry	Entry object	A displayName and mediaProperty combination. The displayName can be empty.
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. The displayName has a maximum length of 50 characters.
mediaProperty	String	The name of the variable that is displayed to the agent or supervisor (maximum length of 32 characters). Each entry must have exactly one mediaProperty. Allowed strings include callVariable1 through callVariable10, any valid ECC variable (user.*), and any of the following Outbound Option variables: • BACampaign • BAAccountNumber • BAResponse • BAStatus • BADialedListID • BATimeZone • BABuddyName

User — Get List of Phone Books for User

The User - Get list of phone books for user API allows a user to retrieve a list of phone books and associated contacts for that user with the private key <userId>.

URI:	http://<FQDN>/finesse/api/User/<userId>/PhoneBooks
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/PhoneBooks`
Security Constraints:	Role: Agent

	Limitations: Any signed-in user can retrieve a list of phone books for that user.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-
Successful Response:	<PhoneBooks> <PhoneBook> <name>PhoneBook 1</name> <type>GLOBAL</type> <Contacts> <Contact> ... Full Contact Object ... </Contact> <Contact> ... Full Contact Object ... </Contact> </Contacts> </PhoneBook> <PhoneBook> <name>PhoneBook 2</name> <type>TEAM</type> <Contacts> <Contact> ... Full Contact Object ... </Contact> <Contact> ... Full Contact Object ... </Contact> </Contacts> </PhoneBook> </PhoneBooks>
HTTP Response:	200-Success 400-Bad request (the request body is invalid) 400-Finesse API error (for example, the object does not exist, the object is stale, and so on) 401-Authorization failure (for example, the user is not yet authenticated in the web session) 404- User Not Found (the user ID provided is invalid and no such user exists in CTI) 500-Internal server error
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>1234</ErrorData> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
name	String	The name of the phone book.
type	String	The phone book type (either GLOBAL or TEAM).
uri	String	The URI of the contact.
firstName	String	The first name of the contact.
lastName	String	The last name of the contact.
description	String	A description of the contact.
phoneNumber	String	The phone number for the contact.

User — Get List of Workflows for User

The Workflow - Get list of workflows for user API allows a user to retrieve a list of workflows and actions assigned to them.

URI:	http://<FQDN>/finesse/api/User/<userID>/Workflows
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/Workflows`
Security Constraints:	Any logged-in user can access their own workflows
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--

Successful Response:	

```
<Workflows>
  <Workflow>
    <name>Google ring pop</name>
    <description>Pops a google web page when an agent phone
rings</description>
    <TriggerSet>
      <type>SYSTEM</type>
      <name>CALL_ARRIVES</name>
      <triggers>
        <Trigger>
          <Variable>
            <name>mediaType</name>
            <node>//Dialog/mediaType</node>
            <type>CUSTOM</type>
          </Variable>
          <comparator>IS_EQUAL</comparator>
          <value>Voice</value>
        </Trigger>
        <Trigger>
          <Variable>
            <name>callType</name>
            <node>//Dialog/mediaProperties/callType</node>
            <type>CUSTOM</type>
          </Variable>
          <comparator>IS_IN_LIST</comparator>

<value>ACD_IN,PREROUTE_ACD_IN,PREROUTE_DIRECT_AGENT,TRANSFER,
               OVERFLOW_IN,OTHER_IN,AGENT_OUT,AGENT_INSIDE,OFFERED,


CONSULT,CONSULT_OFFERED,CONSULT_CONFERENCE,CONFERENCE,
              TASK_ROUTED_BY_ICM,TASK_ROUTED_BY_APPLICATION</value>

        </Trigger>
        <Trigger>
          <Variable>
            <name>state</name>
            <node>//Dialog/participants/Participant/mediaAddress[.=

                  ${userExtension}]/../state</node>
            <type>CUSTOM</type>
          </Variable>
          <comparator>IS_IN_LIST</comparator>
          <value>ALERTING,ACTIVE,HELD</value>
        </Trigger>
        <Trigger>
          <Variable>
            <name>fromAddress</name>
            <node>//Dialog/fromAddress</node>
            <type>CUSTOM</type>
          </Variable>
          <comparator>IS_NOT_EQUAL</comparator>
          <Variable>
            <name>userExtension</name>
            <type>SYSTEM</type>
          </Variable>
        </Trigger>
      </triggers>
    </TriggerSet>
    <ConditionSet>
      <applyMethod>ALL</applyMethod>
      <conditions>
        <Condition>
          <Variable>
            <name>callVariable1</name>
            <type>SYSTEM</type>
          </Variable>
          <comparator>CONTAINS</comparator>
          <value>1234</value>
        </Condition>
        <Condition>
          <Variable>
```

```
            <name>user.foo.bar[1]</name>
            <node>//Dialog/mediaProperties/callvariables/
                  CallVariable/name[.="user.foo.bar[1]"]/..
                  /value</node>
            <type>CUSTOM</type>
          </Variable>
          <comparator>IS_NOT_EMPTY</comparator>
        </Condition>
      </conditions>
    </ConditionSet>
    <workflowActions>
      <WorkflowAction>
        <name>Google ring pop</name>
        <type>BROWSER_POP</type>
        <params>
          <Param>
            <name>windowName</name>
            <value>google</value>
          </Param>
          <Param>
            <name>path</name>
            <value>http://www.google.com?a=${CallVariable1}
                   &amp;amp;c=cat&amp;amp;${DNIS}&amp;amp;d=$
                   {user.foo.bar[1]}</value>
          </Param>
        </params>
        <actionVariables>
          <ActionVariable>
            <name>callVariable1</name>
            <type>SYSTEM</type>
            <testValue>apple</testValue>
          </ActionVariable>
          <ActionVariable>
            <name>user.foo.bar[1]</name>
            <node>//Dialog/mediaProperties/callvariables/
                  CallVariable/name[.="user.foo.bar[1]"]/
                  ../value</node>
            <type>CUSTOM</type>
            <testValue>1234</testValue>
          </ActionVariable>
        </actionVariables>
      </WorkflowAction>
      <WorkflowAction>
        <name>My Delay</name>
        <type>DELAY</type>
        <params>
          <Param>
            <name>time</name>
            <value>10</value>
          </Param>
        </params>
      </WorkflowAction>
    </workflowActions>
  </Workflow>
</Workflows>
```

HTTP Response:

	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or there is a violation of database constraints. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The resource is not found (for example, the user does not exist). 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>1234</ErrorData>`
	`  <ApiError>`
	`<ApiErrors>`

Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
uri	String	The ID in the URI maps to the primary key of the workflow entry.
name	String	The name of the workflow.
description	String	The description of the workflow.
TriggerSet	Collection	A set of events that cause the conditions to be evaluated.
ConditionSet	Collection	A set of conditions that determine if the workflow will be executed.
workflowActions	Collection	A list of existing workflow actions (WorkflowAction objects) that are executed if the trigger and its conditions are satisfied by the data in the event.

Dialog APIs

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.

Dialog — Get Dialog

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

URI:	http://<FQDN>/finesse/api/Dialog/<dialogId>
Example URI:	`http://finesse1.xyz.com/finesse/api/Dialog/12345678`
Security Constraints:	Role: Agent, administrator 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 401-Unauthorized (for example, the user is not authenticated in the Web Session) 401-Invalid Authorization (for example, the authenticated user tried to get a Dialog for which they are not a participant) 404-Not found (for example, the dialog id is invalid) 500-Internal server error

Successful Response:

```
<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>
```

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

Parameter	Type	Description
uri	String	The URI to get a new copy of the Dialog object.
mediaType	String	The type of media under which a dialog is classified (voice, email, or chat).
state	String	The last state of this dialog. For a list of possible values, see State (Dialog) Parameter Values.
fromAddress	String	The calling line ID of the caller.
toAddress	String	The destination for the call.
mediaProperties	Collection	Includes all of the properties for a call that corresponds to the dialog.
dialedNumber	String	The number dialed.
callType	String	The type of call. Possible values include ACD_IN, PREROUTE_ACD_IN, PREROUTE_DIRECT_AGENT, TRANSFER, OTHER_IN, OUT, AGENT_INSIDE, CONSULT, CONFERENCE, SUPERVISOR_MONITOR, OUTBOUND, OUTBOUND_PREVIEW, and RESERVATION.
dnis	String	The DNIS provided with the call. For routed calls, the DNIS is the route point.
wrapUpReason	String	A description of the call. Size: maximum of 39 bytes (which is equal to 39 US English characters).

Parameter	Type	Description
callvariables	Collection	A list of up to 10 call and ECC variables.
CallVariable	Collection	Contains the name and value of a call variable belonging to this dialog. The name indicates whether the variable is a call variable or an ECC variable. Call variable names start with callVariable#, where # is 1-10. 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. The following call variables provide additional details about an Outbound Option call: BACampaign, BAAccountNumber, BAResponse, BAStatus, BADialedListID, BATimeZone, BABuddyName. See the *Cisco Unified Contact Center Express CTI Protocol Developer Guide* for details of their values and usage. The BAStatus variable contains one of the following values: • PREDICTIVE_OUTBOUND (if the agent is on a Predictive Outbound Option call) • PROGRESSIVE_OUTBOUND (if the agent is on a Progressive Outbound Option call) • PREVIEW_OUTBOUND_RESERVATION (if the agent is reserved for a Preview Outbound Option call) • PREVIEW_OUTBOUND (if the agent is on a Preview Outbound Option call)
participants	Collection	A list of all participants, both internal and external, involved in a dialog.
Participant	Collection	Set of information about one participant in a dialog.
actions	Collection	A list of actions that are allowed for a participant as a result of a dialog update. For a list of possible values, see Actions Parameter Values.
mediaAddress	String	Point of contact for this participant. Possible values include the extension of an agent or ANI for a caller who are participants on a call.
state	String	The last participant state in a dialog. Possible values include INITIATING, INITIATED, FAILED, ALERTING, ACTIVE, HELD, DROPPED, and ACCEPTED.

Parameter	Type	Description
stateCause	String	The cause for the last participant state in a dialog. This parameter is normally associated with a FAILED participant state. Possible values include BUSY, BAD_DESTINATION, and OTHER.

Dialog — Take Action on a Participant Within a Dialog

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://<FQDN>/finesse/api/Dialog/<dialogId>
Example URI:	`http://finesse1.xyz.com/finesse/api/Dialog/54321`
Security Constraints:	Role: Agent 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>`
	`  <targetMediaAddress>1001001</targetMediaAddress>`
	`  <requestedAction>ANSWER</requestedAction>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted 400-Parameter missing (for example, the targetMediaAddress or action is not provided) 400-Invalid input 401-Authorization failure 401-Unauthorized (for example, the user is not authenticated in the Web Session) 404-Dialog not found 500-Internal server error

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

Platform-Based API Differences

The following table describes API differences between a stand-alone Finesse deployment with Unified CCE or a coresident Finesse deployment with Unified CCX.

Scenario	Response
A participant who is not the conference controller tries to conference in another participant.	**Stand-alone Finesse with Unified CCE:** <data> <apiErrors> <apiError> <errorData>20999</errorData> <errorMessage><ConferenceCallCommand>: Conference failed...causes include agent already has a consult call or conferencing a non-conference controller. </errorMessage> <errorType>Generic Error</errorType> </apiError> </apiErrors> </data> **Coresident Finesse with Unified CCX:** <data> <apiErrors> <apiError> <errorData>22</errorData> <errorMessage>CF_INVALID_OBJECT_STATE</errorMessage> <errorType>Invalid State</errorType> </apiError> </apiErrors> </data>

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
dialogId	Integer	Yes	The ID of the dialog.	-	-
targetMediaAddress	String	Yes	The extension that the user is currently signed in to, which is used to locate the participant to target with the action request.	-	-
requestedAction	String	Yes	The action to take on the targeted participant.	ANSWER, HOLD, RETRIEVE, DROP, TRANSFER, TRANSFER_SST, CONFERENCE	Only actions that currently exist on a participant can be used.

Dialog — Update Call Variable Data

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 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://<FQDN>/finesse/api/Dialog/<dialogId>
Example URI:	`http://finesse1.xyz.com/finesse/api/Dialog/54321`
Security Constraints:	Role: Agent 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> <requestedAction>UPDATE_CALL_DATA</requestedAction> <mediaProperties> <wrapUpReason>Happy customer!</wrapUpReason> <callvariables> <CallVariable> <name>callVariable1</name> <value>123456789</value> </CallVariable> <CallVariable> ... Other call variables to be modified ... </CallVariable> </callvariables> </callvariables> </mediaProperties> </Dialog>
HTTP Response:	202-Successfully accepted 400-Parameter missing (for example, the mediaProperties, callvariables, callvariable, or action is not provided) 400-Invalid input (the callvariable name or action is not recognized or is invalid or there are duplicate call variable names) 401-Authorization failure 401-Invalid Authorization User Specified (the authenticated user tried to make a request of another dialog that is not their own) 404-Dialog not found 500-Internal server error
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Parameter missing • Invalid Input • Authorization Failure • Invalid Authorization User Specified • Dialog Not Found • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.
Notifications Triggered:	• Dialog Notifications • Dialog CTI Error Notifications (if a CTI error occurs)

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
dialogId	Integer	Yes	The ID of the dialog.	-	-
mediaProperties	XML	Yes	Collection of media-specific properties related to the dialog that need to be modified.	-	Cannot be missing or empty.
wrapUpReason	String	No	A description of the call. Size: maximum of 39 bytes (which is equal to 39 US English characters).	-	Either wrapUpReason or callvariables must be present.
callvariables	Collection	No	A list of call variables to be modified.	-	Either wrapUpReason or callvariables must be present

Parameter	Type	Required	Description	Possible Values	Validation
CallVariable	XML	Yes, if the callvariables tag is present	Contains the name and value of a call variable belonging to this dialog. Size: • Call variable - 40 bytes • 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.	-	

Parameter	Type	Required	Description	Possible Values	Validation
					There should be at least one call variable to be modified. The name must be present and not empty. Duplicate names cannot exist. The Value tag must be specified (but it can be empty). Call variables must match up to one of the predefined call variable names (for example, callVariable1... callVariable10). For ECC/named variables: • The variable starts with the prefix "user". • If the variable is a named array, it should end with a number enclosed in square brackets (for example, *user.myarray[2]*). If the variable includes square brackets in any other order, it will not be considered a named array variable but instead considered as a named scalar variable and allowed to be sent to the CTI server. For example, the variable *user.my[array]* would be considered a named scalar variable. • ECC variable names must match those defined in the CTI server administration user interface in name and length. • All ECC variable

Parameter	Type	Required	Description	Possible Values	Validation
					validations as applied by the CTI server are applicable, but the validation is performed by the CTI server that receives the variables. **Note** Finesse supports only Latin1 characters for ECC variables. Other Unicode characters are not supported.
requestedAction	String	Yes	The action to take on the targeted participant.	ANSWER, HOLD, RETRIEVE, DROP, TRANSFER, CONFERENCE, SILENT_MONITOR	Only actions that currently exist on a participant can be used.

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 the CTI server. The CTI server 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/Unified CCX 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	CTI Server Response	Finesse Response
1 A request was sent that generates an error from the CTI server to Finesse.	The CTI server sends an error to Finesse.	Finesse forwards the error to the client.

2 The request payload contained no valid ECC or call variables.		
1 A request was sent that generates an error from the CTI server to Finesse. 2 The request payload contained a mix of valid and invalid ECC or call variables.	1 The CTI server sends an error to Finesse. 2 The CTI server does not send an UPDATE_CALL_DATA event to Finesse (that is, the CTI server fails the entire request).	1 Finesse forwards the error to the client. 2 The client does not receive an UPDATE_CALL_DATA event.
1 A request was sent that does not generate an error from the CTI server to Finesse. 2 The request payload contained no valid ECC or call variables.	The CTI server does not respond.	Finesse does not respond.
1 A request was sent that does not generate an error from the CTI serverto Finesse. 2 The request payload contained a mix of valid and invalid ECC or call variables.	1 The CTI server does not send an error to Finesse. 2 The CTI server 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. 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, the CTI server 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.

Dialog — Create a New Dialog (Make a Call)

The Dialog - Create a new dialog API allows users to make a call. Making a call is accomplished by creating a new dialog, 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://<FQDN>/finesse/api/User/<id>/Dialogs
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/Dialogs`
Security Constraints:	Role: Agent Limitations: Users can only create dialogs with a fromAddress to which they are currently signed in.

HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Dialog>`
	`  <requestedAction>MAKE_CALL</requestedAction>`
	`  <fromAddress>1001001</fromAddress>`
	`  <toAddress>1001002</toAddress>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted **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. 400-Parameter missing (for example, the fromAddress or toAddress is not provided) 400-Invalid input 400-Invalid destination 401-Authorization failure (for example, the user is not yet authenticated in the Web Session) 401-Invalid Authorization (for example, the authenticated user tried to use a fromAddress that is not their own) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>jsmith</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Invalid Destination • Authorization Failure • Invalid Authorization User Specified • Internal Server Error
Notifications Triggered	If a CTI error occurs, Finesse sends a Dialog CTI Error Notifications

Platform-Based API Differences

The following table describes API differences between a stand-alone Finesse deployment with Unified CCE or a coresident Finesse deployment with Unified CCX.

Scenario	Response
Make a call to a bad destination (the destination does not exist).	**Stand-alone Finesse with Unified CCE:** <apiError> <errorData>1</errorData> <errorMessage>CF_GENERIC_OPERATION</errorMessage> <errorType>Generic Error</errorType> </apiError> **Coresident Finesse with Unified CCX:** <apiError> <errorData>0</errorData> <errorMessage>CF_GENERIC_UNSPECIFIED</errorMessage> <errorType>Generic Error</errorType> </apiError>
Make a call to a busy destination.	**Stand-alone Finesse with Unified CCE:** <apiError> <errorData>1</errorData> <errorMessage>CF_GENERIC_OPERATION</errorMessage> <errorType>Generic Error</errorType> </apiError> **Coresident Finesse with Unified CCX:** <apiError> <errorData>0</errorData> <errorMessage>CF_GENERIC_UNSPECIFIED</errorMessage> <errorType>Generic Error</errorType> </apiError>
Agent or supervisor makes a call while in Ready state.	**Stand-alone Finesse with Unified CCE:** <apiError> <errorData>22</errorData> <errorMessage>CF_INVALID_OBJECT_STATE</errorMessage> <errorType>Invalid State</errorType> </apiError> **Coresident Finesse with Unified CCX:** Call is successful.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
id	String	Yes	The ID of the user.	—	Maximum length of 12 characters. The user is configured in Unified CCE or Unified CCX.
requestedAction	String	Yes	The way in which the dialog is created.	MAKE_CALL	—

Parameter	Type	Required	Description	Possible Values	Validation
fromAddress	String	Yes	Used to match the user media device. This value is the same as the extension with which the user is currently signed in.	—	—
toAddress	String	Yes	The destination for the call.	—	—

Dialog — Make a Consult Call Request

The Dialog - Make a consult call request API allows an agent to make a consult call request. After the consult call request succeeds, the agent can complete the call 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 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 does not support blind conference through the API or desktop.

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.

Finesse supports single-step transfer in Unified CCE deployments only. Finesse does not support single-step transfer when deployed with Unified CCX.

Note Only the conference controller (the user who initiates the conference) can add parties to that conference. For example Agent 1 is on a call with a customer. Agent 1 consults Agent 2 and then conferences Agent 2 into the call. Agent 2 then consults Agent 3. If Agent 2 attempts to add Agent 3 to the conference, the request fails.

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/Unified CCX 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/Unified CCX. Because an empty string is considered a valid value, when call variables are set to empty strings by Unified CCE/Unified CCX, 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://<FQDN>/finesse/api/Dialog/<dialogId>
Example URI:	`http://finesse1.xyz.com/finesse/api/Dialog/1234`
Security Constraints:	Role: Agent
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Dialog>`
	`  <requestedAction>CONSULT_CALL</requestedAction>`
	`  <toAddress>1001002</toAddress>`
	`  <targetMediaAddress>1001001<targetMediaAddress>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted 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. 400-Parameter missing (for example, the toAddress or requestedAction is not provided) 400-Invalid input (for example, the toAddress or requestedAction is invalid) 400-Invalid destination (for example, the toAddress is the same as the caller's extension) 401-Authorization failure 401-Invalid authorization (for example, a user tried to use a fromAddress that did not belong to that user) 500-Internal server error

Failure Response Example:	<pre><ApiErrors>
	<ApiError>
	<ErrorType>Authorization Failure</ErrorType>
	<ErrorMessage>UNAUTHORIZED</ErrorMessage>
	<ErrorData>jsmith</ErrorData>
	</ApiError>
	</ApiErrors></pre>

Error Codes:	• Parameter Missing • Invalid Input • Invalid Destination • Authorization Failure • Invalid Authorization User Specified • Internal Server Error
Notifications Triggered	• Dialog Notifications • Dialog CTI Error Notifications (if a CTI error occurs)

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
dialogId	Integer	Yes	The ID of the dialog.	-	-
requestedAction	String	Yes	The way in which the dialog is created.	CONSULT_CALL	-
targetMediaAddress	String	Yes	Used to locate the participant to target with the action request. This parameter is the same as the extension the user is signed in to.	The extension of the agent who is making the request.	-
toAddress	String	Yes	The destination for the call.	-	-

Dialog — Initiate a Single-Step Transfer

The Dialog - Initiate a single-step transfer API allows an agent to make a single-step transfer request. After an agent makes a successful single-step transfer request, that agent's active call is transferred to the destination used in the toAddress parameter.

The requestedAction for a single-step transfer call is TRANSFER_SST. This request is sent on the Dialog URL of an existing active call, from where the call is initiated. Therefore, the dialogId in the URL represents the Dialog ID of the active call.

Note The Dialog - Initiate a single-step transfer API applies to Finesse deployments with Unified CCE only. If you attempt to use this API on a Finesse deployment with Unified CCX, Finesse sends the following API error:

INVALID ACTION TRANSFER_SST

URI:	http://<FQDN>/finesse/api/Dialog/<dialogId>
Example URI:	`http://finesse1.xyz.com/finesse/api/Dialog/1234`
Security Constraints:	Role: Agent
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Dialog>`
	`  <requestedAction>TRANSFER_SST</requestedAction>`
	`  <toAddress>1001002</toAddress>`
	`  <targetMediaAddress>1001001</targetMediaAddress>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted **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. 400-Parameter missing (the toAddress, targetMediaAddress, or requestedAction is not provided) 400-Invalid input (the toAddress, targetMediaAddress, or requestedAction is not recognized or is invalid) 400-Invalid destination (the toAddress is the same as the extension of the user who is making the request) 401-Authorization failure (the user is not yet authenticated in the Web Session) 401-Invalid authorization user specified (the authenticated user tried to make a request for another participant) 500-Internal server error (any runtime exception, such as the connection is broken with the CTI server or another component)

Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Parameter Missing • Invalid Input • Invalid Destination • Authorization Failure • Invalid Authorization User Specified • Internal Server Error
Notifications Triggered	Dialog Notifications

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
requestedAction	String	Yes	The way in which to create the dialog.	TRANSFER_SST	-
targetMediaAddress	String	Yes	Used to locate the participant to target with the requested action. This parameter is the same as the extension to which the user is logged in.	The extension of the agent who is making the request	-
toAddress	String	Yes	The destination to which to transfer the call.	-	Validity determined by Unified CCE

Dialog — Make a Silent Monitoring Call

The Dialog - Make a silent monitoring call API allows a supervisor to silently monitor an agent who is on an active call and 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://<FQDN>/finesse/api/User/<id>/Dialogs
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/Dialogs`
Security Constraints:	Role: Supervisor, administrator 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. 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>`
	`<requestedAction>SILENT_MONITOR</requestedAction>`
	`<fromAddress>1001001</fromAddress>`
	`<toAddress>2001002</toAddress>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted 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. 400-Parameter missing (the fromAddress, toAddress, or requestedAction is not provided) 400-Invalid input (the fromAddress, toAddress, or requestedAction is invalid) 400-Invalid destination (for example, the fromAddress and toAddress are the same) 400-Invalid state (the supervisor is already silent monitoring or in an active call) 401-Authorization failure (the user is not authenticated in the web session yet or is not a supervisor or administrator) 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)

	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) 500-Internal server error
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Parameter Missing • Invalid Input • Invalid Destination • Invalid State • Authorization Failure • Invalid Authorization User Specified • Internal Server Error
Notifications Triggered	Dialog Notifications

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, supervisors can silently monitor agents who are on ICD calls. The supervisor must be in Not Ready state to start a silent monitoring session and the agent must be in Talking state. After the supervisor starts the silent monitoring session, the supervisor transitions to Talking state.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, supervisors can silently monitor agents who are on ICD calls or non-ICD calls (for example, calls to another agent). The supervisor must be in Not Ready state to start a silent monitoring state. The agent can be in Talking state (on an ICD call) or Not Ready state (on a non-ICD call). After the supervisor starts the silent monitoring call, the supervisor remains in Not Ready state.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
id	String	Yes	The ID of the user.	-	Maximum length of 12 characters. The user is configured in Unified CCE or Unified CCX.
requestedAction	String	Yes	The action to take on the targeted participant.	SILENT_MONITOR	-
fromAddress	String	Yes	Used to match the user media device. This value is the same as the extension the user is currently signed in to.	The extension of the supervisor who initiated the silent monitoring call.	-
toAddress	String	Yes	The destination for the call.	The extension of the agent that the supervisor wants to silently monitor.	-

Dialog — End a Silent Monitoring Call

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://<FQDN>/finesse/api/Dialog/<dialogid>
Example URI:	`http://finesse1.xyz.com/finesse/api/Dialog/32458`
Security Constraints:	Role: Supervisor, administrator 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>`
	`<requestedAction>DROP</requestedAction>`
	`<targetMediaAddress>1001002</targetMediaAddress>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted 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. 400-Parameter missing (the targetMediaAddress or requestedAction is not provided) 400-Invalid input (the targetMediaAddress or requestedAction is invalid) 401-Authorization failure (the user is not authenticated in the web session yet or is not an administrator or supervisor) 401-Invalid authorization user specified (for example, a supervisor tried to use a targetMediaAddress that did not belong to that supervisor) 404-Not found (the dialog specified by the dialogid does not exist) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorType>Authorization Failure</ErrorType>`
	`<ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`<ErrorData>jsmith</ErrorData>`
	`</ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Authorization Failure • Invalid Authorization User Specified • Not Found • Internal Server Error
Notifications Triggered	Dialog Notifications

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
dialogId	Integer	Yes	The ID of the dialog.	-	-

Parameter	Type	Required	Description	Possible Values	Validation
requestedAction	String	Yes	The action to take on the targeted participant.	DROP	-
targetMediaAddress	String	Yes	Used to locate the participant to target with the action request. This parameter is the same as the extension the user is signed in to.	The extension of the supervisor who initiated the silent monitoring call.	-

Dialog — Make a Barge Call

The Dialog - Make a barge call API allows a supervisor to barge in to an agent call that the supervisor is silently monitoring. A new Dialog is created, specifying the fromAddress (supervisor's extension), the toAddress (the agent's extension), and the associatedDialog (the relative URI of the silent monitor dialog that the supervisor initiated), and is posted to the user's Dialogs collection. The supervisor's silent monitor call is dropped. After the barge-in succeeds, the original silent monitor call becomes a conference call with the agent, caller, and supervisor as participants.

When this API is used to barge in to a conference call, the call must meet certain conditions for the barge request to succeed:

- Unified Communications Manager may limit the number of phone devices that can join a conference call (a configurable parameter). When a supervisor makes a barge call, the supervisor is added as a new party to the conference. If the resource limit has already been reached, the supervisor's barge request fails.
- Both Unified CCE and Unified CCX allow a barge request only through the conference controller (the agent who initiates the conference call). If the agent is not the conference controller, the supervisor's barge request fails.

URI:	http://<FQDN>/finesse/api/User/<id>/Dialogs
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/Dialogs`
Security Constraints:	Role: Supervisor Limitations: Supervisors can only make barge call requests using the fromAddress that they are currently logged in to and can only barge in to calls they are already silent monitoring. Administrators cannot barge in to any calls because they are not associated with a phone device.
HTTP Method:	POST
Content Type:	Application/XML

Input/Output Format:	XML

HTTP Request:	`<Dialog>`
	`<requestedAction>BARGE_CALL</requestedAction>`
	`<fromAddress>1001051</fromAddress>`
	`<toAddress>1081002</toAddress>`
	`<associatedDialogUri>/finesse/api/Dialog/68731222</associatedDialogUri>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted 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. 400-Parameter missing (the fromAddress, toAddress, associatedDialogUri, or requestedAction is not provided) 400-Invalid input (the fromAddress, toAddress, associatedDialogUri, or requestedAction is invalid) 400-Invalid destination (the associatedDialogUri is not pointing to the active call of the agent whose extension is provided in the toAddress) 400-Invalid state (the supervisor is not in TALKING state, is in TALKING state but on a call that is not a silent monitoring call, the agent call is not in ACTIVE state, or the agent call is not being silently monitored by the supervisor specified) 400-20999 (Barge via a non-conference-controller or the agent already has an outstanding consult call) 400-20700 (Conference resource limit violation) 401-Authorization failure (the user is not authenticated in the web session yet or is not a supervisor) 401-Invalid authorization user specified (a user tried to use the ID of another user in the request URL, is not the owner of the associated silent monitor call, or is an administrator) 401-Unauthorized (a supervisor tried to use a fromAddress that did not belong to that supervisor to request a barge call) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorType>Authorization Failure</ErrorType>`
	`<ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`<ErrorData>jsmith</ErrorData>`
	`</ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Invalid Destination • Invalid State • 400-20999 (Barge via a non-conference-controller)

	• 400-20700 (Conference resource limit violation) • Authorization Failure • Invalid Authorization User Specified • Internal Server Error
Notifications Triggered	Dialog Notifications

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

A supervisor must be silently monitoring a call before making a request to barge into that call. In a Finesse deployment with Unified CCE, the supervisor's state during the silent monitoring session is Talking. When the supervisor barges in to the call, the supervisor's state remains Talking. The agent's state is Talking before the silent monitoring request, during the silent monitoring session, and after the barge request succeeds.

Coresident Finesse with Unified CCX:

A supervisor must be silently monitoring a call before making a request to barge into that call. In a coresident Finesse deployment with Unified CCX, the supervisor is in Not Ready state during the silent monitoring session. If the agent is on an ICD call, the supervisor's state transitions to Talking after barging in to the call. The agent's state is Talking before the silent monitoring request, during the silent monitoring session, and after the barge request succeeds.

If the agent is on a non-ICD call (for example, a call to another agent), both the supervisor and the agent remain in Not Ready state during the silent monitoring session and after the barge request succeeds.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
requestedAction	String	Yes	The way in which to create the dialog.	BARGE_CALL	-
fromAddress	String	Yes	Used to match the supervisor's media device. The extension that the supervisor is currently logged in to.	Extension of the supervisor who is making the request.	Validity determined by Unified CCE/Unified CCX.
toAddress	String	Yes	The extension of the agent whose call the supervisor wants to barge in on.	Agent's extension.	Validity determined by Unified CCE/Unified CCX.

Parameter	Type	Required	Description	Possible Values	Validation
associatedDialogUri	String	Yes	The relative URI of the silent monitor dialog that is monitoring the call that the supervisor wants to barge in on.	-	Validity determined by Unified CCE/Unified CCX.

Dialog — End a Barge Call

The Dialog - End a barge call API allows a supervisor to leave a barge call that was initiated by that supervisor. The Dialog is updated, specifying a requestedAction of DROP and a targetMediaAddress to the extension of the supervisor how made the barge call.

The agent can still remain on the call unless the total participant becomes less than two when the supervisor leaves (like the drop operation of a conference call).

URI:	http://<FQDN>/finesse/api/Dialog/<dialogid>
Example URI:	`http://finesse1.xyz.com/finesse/api/Dialog/32458`
Security Constraints:	Role: Supervisor, agent Limitations: An agent or supervisor can only drop a barge call if that agent or supervisor is a participant in the call.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Dialog>`
	`  <requestedAction>DROP</requestedAction>`
	`  <targetMediaAddress>1001002</targetMediaAddress>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted 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. 400-Parameter missing (the targetMediaAddress or requestedAction is not provided) 400-Invalid input (the targetMediaAddress or requestedAction is invalid) 401-Authorization failure (the user is not authenticated in the web session yet or is not a supervisor)

	401-Invalid authorization user specified (the call is not initiated by the supervisor or the supervisor tried to use a targetMediaAddress that did not belong to that supervisor) 404-Not found (the dialog specified by the dialogid does not exist) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorType>Authorization Failure</ErrorType>`
	`<ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`<ErrorData>jsmith</ErrorData>`
	`</ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Authorization Failure • Invalid Authorization User Specified • Not Found • Internal Server Error
Notifications Triggered	Dialog Notifications

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
requestedAction	String	Yes	The way in which to create the dialog.	DROP	Validity determined by Unified CCE/Unified CCX.
targetMediaAddress	String	Yes	Used to locate the participant to target with the requestedAction. The extension that the user is currently logged in to.	The extension of the supervisor who initiated the barge call.	Validity determined by Unified CCE/Unified CCX.

Dialog — Drop Participant from Conference Call

The Dialog- Drop participant from conference call API allows a supervisor to make a request to drop a participant from a conference in which that supervisor is one of the call parties. For example, a supervisor can barge in to a call between an agent and a customer. The supervisor can then make a request to drop the agent from the call, leaving the supervisor on the call with the customer.

The request specifies the targetMediaAddress (agent's extension) of the participant to drop. The PUT request applies to the dialog object specified by the dialogId in the URL.

Note You can only drop a mediaAddress that corresponds to a logged-in agent. You cannot drop a CTI Route Point, IVR Port, a device to which no agent is logged in, or a caller device.

After the participant is dropped from the conference call, the call may become a two-party call or remain a conference call (if more than two parties remain on the call after the participant is dropped).

Note If wrap-up is enabled for an agent who is dropped from a call, that agent can still perform wrap-up after being dropped.

URI:	http://<FQDN>/finesse/api/Dialog/<dialogId>
Example URI:	`http://finesse1.xyz.com/finesse/api/Dialog/1234`
Security Constraints:	Role: Supervisor, administrator Limitations: A supervisor can only make a PARTICIPANT_DROP request for a conference call if the supervisor is one of the parties on the call.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Dialog>`
	`  <requestedAction>PARTICIPANT_DROP</requestedAction>`
	`  <targetMediaAddress>1001006</targetMediaAddress>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted 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. 400-Parameter missing (the targetMediaAddress or requestedAction is not provided) 400-Invalid input (the targetMediaAddress or requestedAction is invalid or not recognized)

	400-Invalid destination (the targetMediaAddress is not one of the parties in the dialog or not an agent's extension) 400-Invalid state (the dialog is not a conference call) 401-Authorization failure 401-Invalid authorization (for example, a user tried to use a fromAddress that did not belong to that user) 404-Not found (the dialog specified by the dialogId does not exist) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>jsmith</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Invalid Destination • Invalid State • Authorization Failure • Invalid Authorization User Specified • Not Found • Internal Server Error
Notifications Triggered	Dialog Notifications

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
requestedAction	String	Yes	The way in which to update the dialog.	PARTICIPANT_DROP	-
targetMediaAddress	String	Yes	Used to locate the participant to target with the action request. This parameter is the extension of the agent to remove from the conference.	The extension of the agent to remove from the conference call	-

Dialog — Start Recording

The Dialog-Start recording API allows a user to start recording an active call.

Note The Dialog-Start recording API applies to Finesse deployments with Unified CCX only. If you attempt to use this API on a Finesse deployment with Unified CCE, Finesse returns a "Not Implemented" error.

URI:	http://<FQDN>/finesse/api/Dialog/<dialogid>
Example URI:	`http://finesse1.xyz.com/finesse/api/Dialog/4321`
Security Constraints:	Role: Agent, supervisor Limitations: An user must be a participant in the dialog to perform this action. An agent cannot record another agent's call. A supervisor cannot record an agent's call if the supervisor is not a participant in the call. If a supervisor wants to record an agent's call, the supervisor must first start a silent monitoring session on the call. Supervisors can only silently monitor (and therefore record) agents who are assigned to that supervisor's teams.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Dialog>`
	`  <requestedAction>START_RECORDING</requestedAction>`
	`  <targetMediaAddress>1001001</targetMediaAddress>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted
	Note This response only indicates a successful completion of the request. The request is processed and the actual response of the action is sent as part of a Dialog notification.
	`<Update>`
	`  <event>PUT</event>`
	`  <requestId>12345</requestId>`
	`  <source>/finesse/api/User/1001001/Dialogs</source>`
	`  <data><dialog/></data>`
	`</Update>`
	400-Parameter missing (the targetMediaAddress or requestedAction is not provided)
	400-Invalid input (the targetMediaAddress or requestedAction is not recognized or is invalid)
	401-Authorization failure (the user is not authenticated in the web session yet or is not a participant in the dialog)
	401-Invalid authorization user specified (the targetMediaAddress is different from the authenticated user's extension)

	401-Invalid state (if the participant whose extension is the targetMediaAddress is in HELD state) 500-Internal server error 501-Not implemented (if a recording attempt is made in a Unified CCE deployment)

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>jsmith</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Authorization Failure • Invalid Authorization User Specified • Invalid State • Internal Server Error • Cisco Finesse API Errors
Notifications Triggered	Dialog Notifications

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
requestedAction	String	Yes	The way in which to create the dialog.	START_RECORDING	-
targetMediaAddress	String	Yes	The extension that the user is currently signed in to, which is used to locate the participant to target with the action request.	-	-

Dialog — Send DTMF String

The Dialog - Send DTMF string API allows a user to send a dual-tone multifrequency (DTMF) string during a call.

URI:	http://<FQDN>/finesse/api/Dialog/<dialogid>

Example URI:	`http://finesse1.xyz.com/finesse/api/Dialog/32458`
Security Constraints:	Role: Agent Limitations: An agent must be a participant in the dialog to perform this action.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Dialog>`
	`<requestedAction>SEND_DTMF</requestedAction>`
	`<targetMediaAddress>1001001</targetMediaAddress>`
	`<actionParams>`
	`<ActionParam>`
	`<name>dtmfString</name>`
	`<value>777</value>`
	`</ActionParam>`
	`</actionParams>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted **Note** This response only indicates a successful completion of the request. Although a successful response does not change anything in the Dialog object, Finesse publishes a notification that contains the requestID and an empty Dialog object within the data tag. 400-Parameter missing (the targetMediaAddress, requestedAction, or ActionParam with a name of dtmfString is not provided) 400-Invalid input (the targetMediaAddress or requestedAction is invalid or the value within the actionParam tag contains something other than 0-9, *, #, or A-D for Unified CCE and other than 0-9, *, or # for Unified CCX.) 401-Authorization failure (the user is not authenticated in the web session yet or is not a participant in the dialog) 401-Invalid authorization user specified (for example, a user tried to use a targetMediaAddress that did not belong to that user) 401-Invalid state (if the participant whose extension is the targetMediaAddress is in HELD state) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorType>Authorization Failure</ErrorType>`
	`<ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`<ErrorData>jsmith</ErrorData>`
	`</ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Invalid Destination • Authorization Failure

	• Invalid Authorization User Specified • Invalid State • Internal Server Error
Notifications Triggered	Dialog Notifications

Platform-Based API Differences

The following table describes API differences between a stand-alone Finesse deployment with Unified CCE or a coresident Finesse deployment with Unified CCX.

Scenario	Response

Send a DTMF request with an alphanumeric dtmfString.	**Stand-alone Finesse with Unified CCE:**
	Unified CCE accepts the alphanumeric dtmfString.
	Coresident Finesse with Unified CCX:
	Unified CCX allows only 0-9, *, or # in the dtmfString. Using any other values results in the following error:
	`<apiError>`
	`<errorData>3</errorData>`
	`<errorMessage>CF_VALUE_OUT_OF_RANGE</errorMessage>`
	`<errorType>Generic Error</errorType>`
	`</apiError>`

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
dialogId	Integer	Yes	The ID of the dialog.	-	-
requestedAction	String	Yes	The way in which the dialog is created.	SEND_DTMF	-
targetMediaAddress	String	Yes	The extension that the user is currently signed in to, which is used to locate the participant to target with the action request.	Agent's extension	-

Parameter	Type	Required	Description	Possible Values	Validation
actionParams	Collection	Yes	A collection of generic objects, called ActionParam, which contain name/value pairs.	The name must be *dtmfString*. The value is the DTMF string that you want to submit and can only contain 0-9, *, #, or A-D for Unified CCE. For Unified CCX, this value can only contain 0-9, *, or #.	-

Dialog — Accept, Close, or Reject an Outbound Option Preview Reservation

The Dialog - Accept, close, or reject an Outbound Option Preview reservation API allows a user to accept, close, or reject a reservation in an Outbound Option Preview campaign. Finesse signals an Outbound Option Preview reservation by posting a dialog event of type OUTBOUND_PREVIEW to the reserved user.

Note The Dialog - Accept, close, or reject an Outbound Option Preview reservation API applies to Finesse deployments with Unified CCE only. If you attempt to use this API on a Finesse deployment with Unified CCX, Finesse returns a "Not Implemented" error.

URI:	http://<FQDN>/finesse/api/Dialog/<dialogid>
Example URI:	`http://finesse1.xyz.com/finesse/api/Dialog/32458`
Security Constraints:	Role: Agent Limitations: An agent must be a participant in the dialog to perform this action.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Dialog>`
	`  <requestedAction>{ACCEPT|CLOSE|REJECT}</requestedAction>`
	`  <targetMediaAddress>1001001</targetMediaAddress>`
	`</Dialog>`

HTTP Response:	202-Successfully accepted **Note** This response only indicates a successful completion of the request. The request is processed and the actual response to the action is sent as part of a Dialog notification.

	400-Parameter missing (the requestedAction or targetMediaAddress is not provided) 400-Invalid input (the requestedAction or targetMediaAddress is invalid or not recognized) 401-Authorization failure (the user is not authenticated in the web session yet) 401-Invalid authorization user specified (the authenticated user tried to make a request on behalf of another user) 404-Dialog not found (the dialogId provided is invalid and no such dialog exists) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>jsmith</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Invalid Destination • Authorization Failure • Invalid Authorization User Specified • Dialog Not Found • Internal Server Error
Notifications Triggered	Dialog Notifications

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
requestedAction	String	Yes	The action to take on the Outbound Option Preview reservation.	ACCEPT, CLOSE, or REJECT For more information about these values, see Outbound Option Preview Actions.	-
targetMediaAddress	String	Yes	The extension of the user who is handling the request.	The user's extension	Validated by Unified CCE.

Queue APIs

The Queue object represents a queue (skill group in Unified CCE) and contains the URI, name, and statistics for that queue. Queue statistics include the number of calls in queue, the start time of the longest call in queue, and the number of agents in each state.

Queue — Get Queue

The Queue - Get queue API allows a user to retrieve a Queue object.

Note Any user can use this API to retrieve information about a particular queue. The user does not need to belong to that queue.

Use this API to access statistics for a queue that is assigned to agents or supervisors. If you use this API on a queue that is not assigned to any agents or supervisors, the response contains a value of -1 for numeric statistics and is empty for string statistics.

URI:	http://<FQDN>/finesse/api/Queue/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/Queue/10`
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-
HTTP Response:	200-Success 401-Unauthorized (for example, the user is not authenticated in the Web Session) 404-Not found (for example, the queue or skill group does not exist in Unified CCE) 500-Internal server error

Successful Response:	<Queue> <uri>/finesse/api/Queue/{id}</uri> <name>Sales</name> <statistics> <callsInQueue>3</callsInQueue> <startTimeOfLongestCallInQueue>2012-02-15T17:58:21Z </startTimeOfLongestCallInQueue> <agentsReady>1</agentsReady> <agentsNotReady>2</agentsNotReady> <agentsTalkingInbound>3</agentsTalkingInbound> <agentsTalkingOutbound>4</agentsTalkingOutbound> <agentsTalkingInternal>5</agentsTalkingInternal> <agentsWrapUpNotReady>6</agentsWrapUpNotReady> <agentsWrapUpReady>7</agentsWrapUpReady> </statistics> </Queue>
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Authorization Failure • Not Found • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Platform-Based API Differences

The following statistics fields are updated only for a stand-alone Finesse deployment with Unified CCE:

- callsInQueue
- startTimeOfLongestCallInQueue
- agentsReady
- agentsNotReady
- agentsTalkingInbound
- agentsTalkingOutbound
- agentsTalkingInternal
- agentsWrapUpNotReady
- agentsWrapUpReady

In a coresident Finesse deployment with Unified CCX, these fields are not updated and the value for each field is -1.

Response Parameters

Parameter	Type	Description
uri	String	The URI to get a new copy of the Queue object.
id	String	The unique identifier for the queue.
name	String	The name of the queue.
statistics	XML	The statistics for a queue.
callsInQueue	Integer	The number of calls currently queued to this queue. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
startTimeOfLongestCallInQueue	String	The start time of the longest call in the queue. Format: YYYY-MM-DDTHH:MM:SSZ **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsReady	Integer	The number of agents assigned to the queue who are in Ready state. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsNotReady	Integer	The number of agents assigned to the queue who are in Not Ready state. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsTalkingInbound	Integer	The number of agents assigned to the queue who are in Talking state on inbound calls. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsTalkingOutbound	Integer	The number of agents assigned to the queue who are in Talking state on outbound calls. Outbound calls include non-routed calls placed to external devices that are not monitored by Unified CM and to devices in a different Unified CM cluster. Outbound Dialer calls are not included. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.

Parameter	Type	Description
agentsTalkingInternal	Integer	The number of agents assigned to the queue who are in Talking state on internal calls. Internal calls are consult calls. When an agent on a routed call initiates an internal consult call, this statistic is incremented for the queue associated with the original call. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsWrapUpNotReady	Integer	The number of agents assigned to the queue who are in Work Not Ready state. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsWrapUpReady	Integer	The number of agents assigned to the queue who are in Work Ready state. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.

Queue — Get Queue List for User

The Queue - Get queue list for user API allows a user to get a list of all queues associated with a user. A user can use this API to get a list of queues associated with any user.

Note The list of queues does not include the system-defined queue (skill group) present on Unified CCE to which all agents belong.

URI:	http://<FQDN>/finesse/api/User/<id>/Queues
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/Queues`
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-
HTTP Response:	200-Success 401-Unauthorized (for example, the user is not authenticated in the Web Session) 404-User not found (the agent ID is invalid and no such agent exists in CTI)

Successful Response:	<Queues> <Queue> <uri>/finesse/api/Queue/1234</uri> <name>Sales</name> <statistics> <callsInQueue>3</callsInQueue> <startTimeOfLongestCallInQueue>2012-02-15T17:58:21Z </startTimeOfLongestCallInQueue> <agentsReady>1</agentsReady> <agentsNotReady>2</agentsNotReady> <agentsTalkingInbound>3</agentsTalkingInbound> <agentsTalkingOutbound>4</agentsTalkingOutbound> <agentsTalkingInternal>5</agentsTalkingInternal> <agentsWrapUpNotReady>6</agentsWrapUpNotReady> <agentsWrapUpReady>7</agentsWrapUpReady> </statistics> </Queue> ... more queues ... </Queues
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Authorization Failure • User Not Found • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Platform-Based API Differences

The following statistics fields are updated only for a stand-alone Finesse deployment with Unified CCE:

- callsInQueue
- startTimeOfLongestCallInQueue
- agentsReady
- agentsNotReady
- agentsTalkingInbound
- agentsTalkingOutbound
- agentsTalkingInternal
- agentsWrapUpNotReady
- agentsWrapUpReady

In a coresident Finesse deployment with Unified CCX, these fields are not updated and the value for each field is -1.

Response Parameters

Parameter	Type	Description
uri	String	The URI to get a new copy of the Queue object.
id	String	The unique identifier for the queue.
name	String	The name of the queue.
statistics	XML	The statistics for a queue.
callsInQueue	Integer	The number of calls currently queued to this queue. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
startTimeOfLongestCallInQueue	String	The start time of the longest call in the queue. Format: YYYY-MM-DDTHH:MM:SSZ **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsReady	Integer	The number of agents assigned to the queue who are in Ready state. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsNotReady	Integer	The number of agents assigned to the queue who are in Not Ready state. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsTalkingInbound	Integer	The number of agents assigned to the queue who are in Talking state on inbound calls. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsTalkingOutbound	Integer	The number of agents assigned to the queue who are in Talking state on outbound calls. Outbound calls include non-routed calls placed to external devices that are not monitored by Unified CM and to devices in a different Unified CM cluster. Outbound Dialer calls are not included. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.

Parameter	Type	Description
agentsTalkingInternal	Integer	The number of agents assigned to the queue who are in Talking state on internal calls. Internal calls are consult calls. When an agent on a routed call initiates an internal consult call, this statistic is incremented for the queue associated with the original call. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsWrapUpNotReady	Integer	The number of agents assigned to the queue who are in Work Not Ready state. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.
agentsWrapUpReady	Integer	The number of agents assigned to the queue who are in Work Ready state. **Note** If the queue is not assigned to an agent or supervisor, this value is -1.

Team APIs

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

Team — Get Team

The Team - Get team API allows a user to get the configuration information for a specific team, which includes the Team ID, a list of agents that are a member of that team, and the time in state for each agent.

URI:	http://<FQDN>/finesse/api/Team/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/Team/5004`
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-
HTTP Response:	200-Success 401-Unauthorized (for example, the user is not authenticated in the Web Session) 404-Not found (for example, the team id is invalid)

Successful Response:	<Team> <uri>/finesse/api/Team/5004</uri> <id>5004</id> <name>My Team</team> <users> <User> <uri>/finesse/api/User/1234</uri> <loginId>100101</loginId> <firstName>Charles</firstName> <lastName>Smith</lastName> <extension>10011</extension> <state>LOGOUT</state> <stateChangeTime>2012-03-01T17:58:21Z</stateChangeTime> </User> <User> <uri>/finesse/api/User/9876</uri> <loginId>100102</loginId> <firstName>Jack</firstName> <lastName>Rrown</lastName> <extension>10012</extension> <state>NOT_READY</state> <stateChangeTime>2012-03-01T17:58:21Z</stateChangeTime> </User> ... other users ... </users> </Team>
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Authorization Failure • Not Found • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
uri	String	The URI to get a new copy of the Team object.
id	String	The unique identifier for the team.
name	String	The name of the team.
users	XML	The list of users that belong to the team.
User	XML	Information about one specific user on a team.
loginId	String	The login ID of the user.

Parameter	Type	Description
firstName	String	The first name of the user.
lastName	String	The last name of the user.
extension	String	The extension that the user is currently using.
state	String	The state of the user (for example, LOGOUT, NOT_READY, READY, RESERVED, RESERVED_OUTBOUND, TALKING, HOLD, WORK, WORK_READY, or UNKNOWN).
stateChangeTime	String	The time that the state of the user changed to the current state. Format: YYYY-MM-DDTHH:MM:SSZ **Note** This parameter is empty when the time of the state change is not available (if no agent state change event was received yet).

System APIs

The SystemInfo object represents the Finesse system and includes the deployment type (whether Finesse is deployed with Unified CCE or Unified CCX), 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.

SystemInfo - Get SystemInfo

The SystemInfo - Get SystemInfo API allows a user to get information about the deployment type, 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://<FQDN>/finesse/api/SystemInfo
Example URI:	`http://finesse1.xyz.com/finesse/api/SystemInfo`
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-

HTTP Response:	200-Success 500-Internal server error
Successful Response:	<SystemInfo> <currentTimeStamp>2013-12-04T04:26:44Z</currentTimeStamp> <deploymentType>UCCE</deploymentType> <status>OUT_OF_SERVICE</status> <xmppDomain>xmppserver.cisco.com</xmppDomain> <xmppPubSubDomain>pubsub.xmppserver.cisco.com </xmppPubSubDomain> <primaryNode> <host>172.16.204.25</host> </primaryNode> <secondaryNode> <host>172.16.204.26</host> </secondaryNode> </SystemInfo>
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Internal Server Error</ErrorType> <ErrorMessage>Runtime Exception</ErrorMessage> <ErrorData></ErrorData> </ApiError> </ApiErrors>
Error Codes:	Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
status	String	The state of the system. Possible values: • IN_SERVICE: The system is in service and normal operations are accepted. • OUT_OF_SERVICE: The system is out of service and normal operations result in a 503 Service Unavailable response.
currentTimeStamp	String	The current time (GMT time) in the format YYYY-MM-DDThh:MM:SSZ.
deploymentType	String	The type of deployment for Finesse. Possible values: • UCCE • UCCX
xmppDomain	String	The XMPP server domain.

Parameter	Type	Description
xmppPubSubDomain	String	The XMPP server pubSub domain.
primaryNode - host	String	The hostname or IP address of the primary Finesse node.
secondaryNode - host	String	The hostname or IP address of the secondary Finesse node.

Client Log APIs

The ClientLog object is a container element that holds client log data to be posted to the Finesse server.

Note This object supports a POST operation only.

ClientLog - Post to Finesse

The ClientLog - Post to Finesse API allows a user to submit client-side logs to the Finesse server. Finesse creates a log file from the data and stores it on disk.

URI:	http://<FQDN>/finesse/api/User/<id>/ClientLog
Example URI:	`http://finesse1.xyz.com/finesse/api/User/1234/ClientLog`
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<ClientLog>`
	`  <logData>`
	`  xxxxxxxxxxxxxxx\n`
	`  xxxxxxxxxxxxxxx\n`
	`  </logData>`
	`</ClientLog>`

HTTP Response:	202 - Successfully accepted Note: The 202 Successfully accepted response only indicates successful completion of the request. To notify the client of a successful operation, Finesse publishes a notification that contains the requestID and an empty ClientLog object within the data element and 400-Parameter missing (the logData field is not present) 400-Invalid input (the size of logData is greater than 1048576 characters) 400-Operation failure (the POST client log operation failed)

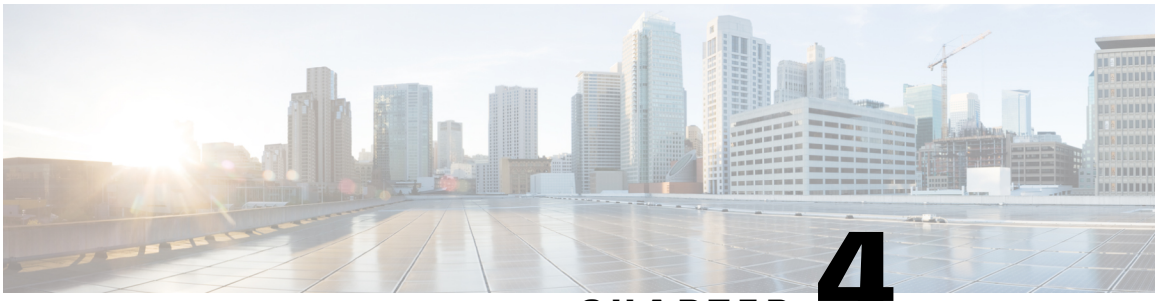
	401-Authorization failure (for example, the user is not yet authenticated in the Web Session) 401-Invalid Authorization (for example, the authenticated user tried to make a request for another user) 405-Method not allowed (GET or PUT operation is not allowed for this API. Only POST is allowed.)

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>jsmith</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Parameter Missing • Invalid Input • Operation Failure • Authorization Failure • Invalid Authorization User Specified • Method Not Allowed

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
id	String	Yes	The ID of the user. The ClientLog - Post to Finesse API uses the id in the name of the log file created on the server.	-	Maximum of 12 characters. The user is configured in Unified CCE or Unified CCX.
logData	String	Yes	The log data that the client sends to the server to be stored as a log file.	-	Must not be more than 1,048,576 characters. The user must be authorized to perform the POST operation.



CHAPTER 4

Cisco Finesse Configuration APIs

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

The configuration APIs require administrator credentials (the application user ID and password) to be passed into the basic authorization header.

System Configuration APIs

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

Note The system configuration APIs are supported only for Finesse deployments with Unified CCE. Because you need not configure CTI server settings for Unified CCX, these APIs are not supported for Finesse deployments with Unified CCX.

SystemConfig - Get

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

URI:	http://<FQDN>/finesse/api/SystemConfig
Example URI:	`http://finesse1.xyz.com/finesse/api/SystemConfig`
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-

Successful Response:	`<SystemConfig>`
	`<uri>/finesse/api/SystemConfig</uri>`
	`<cti>`
	`<host>10.1.1.1</host>`
	`<port>42027</port>`
	`<backupHost>10.1.1.2</backupHost>`
	`<backupPort>42027</backupPort>`
	`<peripheralId>5000</peripheralId>`
	`</cti>`
	`</SystemConfig>`

HTTP Response:	200-Success 401-Unauthorized (for example, the user is not authenticated in the Web Session) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorType>Authorization Failure</ErrorType>`
	`<ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`<ErrorData>jsmith</ErrorData>`
	`</ApiError>`
	`</ApiErrors>`

Error Codes:	• Authorization Failure • Forbidden • Internal Server Error

	For descriptions and other possible error codes, see Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
cti - host	String	The hostname or IP address of the primary (A Side) CTI server.
cti - port	Integer	The port number of the primary (A Side) CTI server.
cti - peripheralId	Integer	The ID of the CTI server peripheral.
cti - backupHost	String	The hostname or IP address of the backup (B Side) CTI server.
cti - backupPort	Integer	The port number of the backup (B Side) CTI server.

SystemConfig - Set

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://<FQDN>/finesse/api/SystemConfig
Example URI:	`http://finesse1.xyz.com/finesse/api/SystemConfig`
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<SystemConfig>`
	`  <cti>`
	`  <host>10.1.1.1</host>`
	`  <port>42027</port>`
	`  <backupHost>10.1.1.2</backupHost>`
	`  <backupPort>42027</backupPort>`
	`  <peripheralId>5000</peripheralId>`
	`  </cti>`
	`</SystemConfig>`

HTTP Response:	200-Success (the new settings were successfully written to the database)

	400-Bad request 401-Unauthorized (for example, the user is not authenticated in the Web Session) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Invalid Input</ErrorType>`
	`  <ErrorMessage>port</ErrorMessage>`
	`  <ErrorData>65536</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes	• Parameter Missing • Invalid Input • Authorization Failure • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
cti - host	String	Yes	The hostname or IP address of the primary (A Side) CTI server.	-	No special characters allowed except "." and "-".
cti - port	Integer	Yes	The port number of the primary (A Side) CTI server.	1-65535 Default value: 42027	Must be between 1 and 65535.
cti - peripheralId	Integer	Yes	The ID of the CTI server peripheral.	1-32767 Default value: 5000	Must be between 1 and 32767.

Parameter	Type	Required	Description	Possible Values	Validation
cti - backupHost	String	Required if backupPort is present in the request	The hostname or IP address of the backup (B Side) CTI server.	-	Must not be the same as the hostname or IP address of the primary (A Side) CTI server. No special characters allowed except "." and "-".
cti - backupPort	Integer	Required if backupHost is present in the request	The port number of the backup (B Side) CTI server.	1-65535	Must be between 1 and 65535.

Cluster Configuration APIs

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.

Note The cluster configuration APIs are supported only for Finesse deployments with Unified CCE. Because you need not set cluster settings for Unified CCX, these APIs are not supported for Finesse deployments with Unified CCX.

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

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

URI:	http://<FQDN>/finesse/api/ClusterConfig
Example URI:	`http://finesse1.xyz.com/finesse/api/ClusterConfig`
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	-
HTTP Response:	200-Success 401-Unauthorized (for example, the user is not authenticated in the Web Session) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error

Successful Response Example:	`<ClusterConfig>`
	`<uri>/finesse/api/ClusterConfig</uri>`
	`<secondaryNode>`
	`<host>10.1.1.1</host>`
	`</secondaryNode>`
	`</ClusterConfig>`

Failure Response Example:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorType>Authorization Failure</ErrorType>`
	`<ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`<ErrorData>jsmith</ErrorData>`
	`</ApiError>`
	`</ApiErrors>`

Error Codes	• Authorization Failure • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
secondaryNode - host	String	The hostname or IP address of the secondary Finesse node.

ClusterConfig - Set

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://<FQDN>/finesse/api/ClusterConfig

Example URI:	`http://finesse1.xyz.com/finesse/api/ClusterConfig`
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<ClusterConfig>`
	`  <secondaryNode>`
	`  <host>10.1.1.1</host>`
	`  </secondaryNode>`
	`</ClusterConfig>`

HTTP Response:	200-Success (the new settings were successfully written to the database) 400-Bad request 401-Unauthorized (for example, the user is not authenticated in the Web Session) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Datastore Error</ErrorType>`
	`  <ErrorData>5527</ErrorData>`
	`  <ErrorMessage>Error reading/writing ClusterConfig from the database</ErrorMessage>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes	• Parameter Missing • Invalid Input • Authorization Failure • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
secondaryNode - host	String	Yes	The hostname or IP address of the secondary Finesse node.	-	No special characters allowed except "." and "-".

Database Configuration APIs

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.

Note The database configuration APIs are supported only for Finesse deployments with Unified CCE. Because you need not configure database settings for Unified CCX, these APIs are not supported for Finesse deployments with Unified CCX.

EnterpriseDatabaseConfig - Get

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

URI:	http://<FQDN>/finesse/api/EnterpriseDatabaseConfig
Example URI:	`http://finesse1.xyz.com/finesse/api/EnterpriseDatabaseConfig`
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-

Successful Response:	`<EnterpriseDatabaseConfig>`
	`<uri>/finesse/api/EnterpriseDatabaseConfig</uri>`
	`<host>10.1.1.1</host>`
	`<backupHost></backupHost>`
	`<port></port>`
	`<databaseName>ucce8x_awdb</databaseName>`
	`<domain>example.com</domain>`
	`<username>Admin</username>`
	`<password>password</password>`
	`</EnterpriseDatabaseConfig>`

HTTP Response:	200-Success 401-Unauthorized (for example, the user is not authenticated in the Web Session) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error

Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Authorization Failure • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
host	String	The hostname or IP address of the AWDB server.
backupHost	String	The hostname or IP address of the backup AWDB server.
port	Integer	The port of the AWDB server.
databaseName	String	The name of the AWDB.
domain	String	The domain of the AWDB.
username	String	The username required to sign in to the AWDB.
password	String	The password required to sign in to the AWDB.

EnterpriseDatabaseConfig - Set

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

The URI for this API contains the query parameter override. This parameter is optional and can be set to true or false.

Certain errors returned by this API can be overridden. If an error can be overridden, it contains an override XML element within the body with a value of "true". If Finesse cannot connect to the Enterprise database with the supplied parameters, the following error is returned.

```
<ApiErrors>
 <ApiError>
  <ErrorType>Invalid Input</ErrorType>
  <ErrorMessage>Enterprise Database Connection Validation Failed</ErrorMessage>
  <ErrorData>Unable to authenticate against the primary enterprise database</ErrorData>
  <Overrideable>true</Overrideable>
```

```
 </ApiError>
</ApiErrors>
```

If this API is called with the query parameter override set to "true", the validation is skipped, the error is overridden, and the API continues to execute.

URI:	http://<FQDN>/finesse/api/EnterpriseDatabaseConfig?override='<true|false>'
Example URI:	`http://finesse1.xyz.com/finesse/api/EnterpriseDatabaseConfig?override='true'`
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<EnterpriseDatabaseConfig>`
	`<uri>/finesse/api/EnterpriseDatabaseConfig</uri>`
	`<host>10.1.1.1</host>`
	`<backupHost>10.1.1.2</backupHost>`
	`<port>1433</port>`
	`<databaseName>ucce8.x_awdb</databaseName>`
	`<domain>example.com</domain>`
	`<username>Admin</username>`
	`<password>password</password>`
	`</EnterpriseDatabaseConfig>`

HTTP Response:	200-Success (the new settings were successfully written to the database) 400-Invalid Input 401-Unauthorized (for example, the user is not authenticated in the Web Session) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorType>Invalid Input</ErrorType>`
	`<ErrorMessage>host</ErrorMessage>`
	`<ErrorData>10.1.1</ErrorData>`
	`</ApiError>`
	`</ApiErrors>`

Error Codes	• Parameter Missing • Invalid Input • Authorization Failure • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
host	String	Yes	The hostname or IP address of the AWDB server.	-	No special characters allowed except "." and "-".
backupHost	String	No	The hostname or IP address of the backup AWDB server. **Note** If you do not specify a 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.	-	No special characters allowed except "." and "-".
port	Integer	Yes	The port of the AWDB server.	1-65535	Must be between 1 and 65535.
databaseName	String	Yes	The name of the AWDB.	-	-
domain	String	Yes	The domain of the AWDB.	-	-
username	String	Yes	The username required to sign in to the AWDB.	-	-
password	String	Yes	The password required to sign in to the AWDB.	-	-

Layout Configuration APIs

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>/finesse/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>/desktop/gadgets/CallControl.jsp</gadget>
         </page>
         <tabs>
           <tab>
             <id>home</id>
             <label>finesse.container.tabs.agent.homeLabel</label>
           </tab>
           <tab>
             <id>manageCall</id>
             <label>finesse.container.tabs.agent.manageCallLabel</label>
           </tab>
         </tabs>
       </layout>
       <layout>
         <role>Supervisor</role>
         <page>
           <gadget>/desktop/gadgets/CallControl.jsp</gadget>
         </page>
         <tabs>
           <tab>
             <id>home</id>
             <label>finesse.container.tabs.supervisor.homeLabel</label>
             <gadgets>
               <gadget>/desktop/gadgets/TeamPerformance.jsp</gadget>
               <gadget>/desktop/gadgets/QueueStatistics.jsp</gadget>
             </gadgets>
           </tab>
           <tab>
             <id>manageCall</id>
             <label>finesse.container.tabs.supervisor.manageCallLabel</label>
           </tab>
         </tabs>
       </layout>
     </finesseLayout>
     </layoutxml>
</LayoutConfig>
```

Note Gadgets that reside on the Finesse server can be specified by an absolute path, as shown in the following example:

/desktop/gadgets/TeamPerformance.xml

Gadgets that are hosted on a server other than the Finesse server must be specified with a fully-qualified URL, as in the following example:

http://server.com/<path to gadget>/<gadget name>.xml

When the desktop loads, Finesse reads the label for each tab and attempts to find it in the resource bundle (as a key). If Finesse finds the key, it displays the value in the tab. If Finesse does not find the key, it displays the key as the default value for the tab.

The following example shows how the key mappings appear in the resource bundle for the Home and Manage Call tabs:

```
finesse.container.tabs.agent.homeLabel=Home
finesse.container.tabs.agent.manageCallLabel=Manage Call
finesse.container.tabs.supervisor.homeLabel=Home
finesse.container.tabs.supervisor.manageCallLabel=Manage Call
```

LayoutConfig — Get

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

URI:	http://<FQDN>/finesse/api/LayoutConfig/default
Example URI:	`http://finesse1.xyz.com/finesse/api/LayoutConfig/default`
Security Constraints:	Role: Administrator 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>`
	`<uri>/finesse/api/LayoutConfig/default</uri>`
	`<layoutxml>`
	`...`
	`</layoutxml>`
	`</LayoutConfig>`

HTTP Response:	200-Success 401-Unauthorized (for example, the user is not authenticated in the Web Session) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error

Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Authorization Failure • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
layoutxml	String	The XML data that determines the layout of the Finesse desktop.

LayoutConfig - Set

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://<FQDN>/finesse/api/LayoutConfig/default
Example URI:	`http://finesse1.xyz.com/finesse/api/LayoutConfig/default`
Security Constraints:	Role: Administrator 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> <layoutxml><?xml version="1.0" encoding="UTF-8"?> ... </layoutxml> </LayoutConfig>
HTTP Response:	200-Success (the new settings were successfully written to the database) 400-Parameter missing (the XML file was not provided) 400-Invalid input (the submitted XML is invalid or does not conform to the Finesse layout schema) 401-Authorization failure (for example, the user is not yet authenticated in the web session) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Invalid Input</ErrorType> <ErrorMessage>layoutxml</ErrorMessage> </ApiError> </ApiErrors>
Error Codes	• Parameter Missing • Invalid Input • Authorization Failure • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
layoutxml	String	Yes	The XML data that determines the layout of the Finesse desktop.	-	Must be valid XML and must conform with the Finesse schema.

Reason Code APIs

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 sign out reason codes. The ReasonCode APIs are for administrator operations.

The structure of a ReasonCode object is as follows:

```
<ReasonCode>
    <uri>/finesse/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.

To prevent reporting problems, you must define reason codes consistently on both Finesse and the platform (Unified CCE or Unified CCX) side. For example, if you configure a not ready reason code on Finesse with a code of 413 and a label of "Meeting", but configure a reason code on Unified CCE with a code of 413 and a description of "Lunch break", the Unified CCE report shows "Lunch break" for the agent who selected that code.

Note Certain predefined reason codes are available for Unified CCE and Unified CCX.

For more information about predefined reason codes for Unified CCE, see the *Reporting Guide for Cisco Unified Intelligent Contact Management and Unified Contact Center Enterprise & Hosted* (http://www.cisco.com/en/US/products/sw/custcosw/ps1844/products_user_guide_list.html).

For more information about predefined reason codes for Unified CCX, see the *Cisco Unified Contact Center Express CTI Protocol Developer Guide* (http://developer.cisco.com/web/uccxcti/documentation)

ReasonCode — Get

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

URI:	http://<FQDN>/finesse/api/ReasonCode/<id>

Example URI:	`http://finesse1.xyz.com/finesse/api/ReasonCode/476`
Security Constraints:	Role: Administrator 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>`
	`<uri>/finesse/api/ReasonCode/{id}</uri>`
	`<category>NOT_READY</category>`
	`<code>10</code>`
	`<label>Team Meeting</label>`
	`<forAll>true</forAll>`
	`</ReasonCode>`

HTTP Response:	200-Success 401-Authorization failure 401-Invalid Authorization User Specified 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 404-Not Found (the resource cannot be found, for example, it might have been deleted) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorType>Authorization Failure</ErrorType>`
	`<ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`<ErrorData>jsmith</ErrorData>`
	`</ApiError>`
	`</ApiErrors>`

Error Codes:	• Authorization Failure • Invalid Authorization User Specified • Forbidden • Not Found • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
category	String	The category of the reason code (NOT_READY or LOGOUT).
code	Integer	The value of the reason code.
label	String	The UI label for the reason code (for example, Lunch, End of Shift).
forAll	String	Whether the reason code applies to all agents (possible values are true or false).

ReasonCode — Get List

The ReasonCode - Get list API allows an administrator to get a list of not ready or sign out reason codes. The required URL parameter *category* specifies whether to retrieve sign out reason codes or not ready reason codes. If this URL parameter is missing, the API returns an error.

URI:	http://<FQDN>/finesse/api/ReasonCodes?category=NOT_READY|LOGOUT
Example URI:	`http://finesse1.xyz.com/finesse/api/ReasonCodes?category=NOT_READY`
Security Constraints:	Role: Administrator 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:	<ReasonCodes> <category>NOT_READY</category> <ReasonCode> ... Full ReasonCode Object ... </ReasonCode> <ReasonCode> ... Full ReasonCode Object ... </ReasonCode> <ReasonCode> ... Full ReasonCode Object ... </ReasonCode> </ReasonCodes>
HTTP Response:	200-Success 401-Invalid input 401-Authorization failure 401-Invalid Authorization User Specified 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Invalid Input • Authorization Failure • Invalid Authorization User Specified • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

URL Request Parameter

Parameter	Type	Required	Description	Possible Values	Validation
category	String	Yes	The category of reason code to retrieve.	NOT_READY, LOGOUT	-

Response Parameters

Parameter	Type	Description
ReasonCodes	XML	Represents a list of ReasonCode objects.
category	String	The category of the reason code (NOT_READY or LOGOUT).
ReasonCode	XML	A ReasonCode object. This object can have a category of NOT_READY or LOGOUT, a descriptive label, and a numeric code value.

ReasonCode - Create

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.

Finesse supports a maximum of 100 global reason codes and 100 non-global reason codes for each category. You can create up to 100 global and 100 non-global reason codes for the LOGOUT category, and 100 global and 100 non-global reason codes for the NOT_READY category.

Note The forAll parameter determines whether a reason code is global (true) or non-global (false).

URI:	http://<FQDN>/finesse/api/ReasonCode/
Example URI:	`http://finesse1.xyz.com/finesse/api/ReasonCode/`
Security Constraints:	Role: Administrator 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> <category>NOT_READY</category> <code>24</code> <label>Lunch Break</label> <forAll>true</forAll> </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 400-Bad request 400-Finesse API error (for example, the reason code already exists) 400-Maximum exceeded 401-Authorization failure 401-Invalid Authorization User Specified 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Maximum Exceeded • Authorization Failure • Invalid Authorization User Specified • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
category	String	Yes	The category of the reason code.	NOT_READY, LOGOUT	The combination of category, code, and label for a reason code must be unique.

Parameter	Type	Required	Description	Possible Values	Validation
code	Integer	Yes	The value of the reason code.	Unified CCE: 1-65535 Unified CCX: 1-999	The combination of category, code, and label for a reason code must be unique.
label	String	Yes	The UI label for the reason code.	-	The combination of category, code, and label for a reason code must be unique. The label cannot exceed 40 characters.
forAll	Boolean	Yes	The access range of the reason code.	true, false	Must be true or false.

ReasonCode - Update

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 At least one of the following parameters must be present in the HTTP request to update a reason code: code, label, or forAll. If none of these parameters are present, Finesse returns an Invalid Input error.

Finesse supports a maximum of 100 global reason codes and 100 non-global reason codes for each category. You can create up to 100 global and 100 non-global reason codes for the LOGOUT category, and 100 global and 100 non-global reason codes for the NOT_READY category.

Note The forAll parameter determines whether a reason code is global (true) or non-global (false).

URI:	http://<FQDN>/finesse/api/ReasonCode/<id>

Example URI:	`http://finesse1.xyz.com/finesse/api/ReasonCode/476`
Security Restraints:	Role: Administrator 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>`
	`  <code>1001</code>`
	`  <label>Lunch break</label>`
	`  <forAll>true</forAll>`
	`</ReasonCode>`

HTTP Response:	200-Success (The Finesse server successfully updated the reason code) 400-Finesse API error 401-Authorization failure 401-Invalid Authorization User Specified 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 404-Not found 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Invalid Input</ErrorType>`
	`  <ErrorMessage>The reason code entered is already being used</ErrorMessage>`
	`<ErrorData>finesse.api.reasoncode.duplicated_code</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Maximum Exceeded • Authorization Failure • Invalid Authorization User Specified • Forbidden • Not Found • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
code	Integer	No	The value of the reason code.	Unified CCE: 1-65535 Unified CCX: 1-999	The combination of category, code, and label for a reason code must be unique.
label	String	No	The UI label for the reason code.	-	The combination of category, code, and label for a reason code must be unique.
forAll	Boolean	No	The access range of the reason code.	true, false	Must be true or false.

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>
```

ReasonCode - Delete

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

URI:	http://<FQDN>/finesse/api/ReasonCode/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/ReasonCode/4235`
Security Constraints:	Role: Administrator 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) 401-Authorization failure 401-Invalid Authorization User Specified 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>jsmith</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	• Authorization Failure • Invalid Authorization User Specified • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Wrap-Up Reason APIs

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>/finesse/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.

WrapUpReason — Get

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

URI:	http://<FQDN>/finesse/api/WrapUpReason/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/WrapUpReason/31`
Security Restraints:	Role: Administrator 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>`
	`  <uri>/finesse/api/WrapUpReason/31</uri>`
	`  <label>Product Question</label>`
	`  <forAll>true</forAll>`
	`</WrapUpReason>`

HTTP Response:	200-Success 401-Authorization failure 401-Invalid Authorization User Specified (for example, an authenticated user tried to use the identity of another user) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 404-Not found (for example, the wrap-up reason was deleted) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>jsmith</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	For possible error codes and their descriptions, see Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	String	The URI to get a new copy of the WrapUpReason object.
label	String	The UI label for the wrap-up reason (for example, Sales Call, Complaint).
forAll	Boolean	Whether the wrap-up reason applies globally (true) or non-globally (false).

WrapUpReason — Get List

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

URI:	http://<FQDN>/finesse/api/WrapUpReasons
Example URI:	`http://finesse1.xyz.com/finesse/api/WrapUpReasons`
Security Constraints:	Role: Administrator 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>`
	`  <WrapUpReason>`
	`  ... Full WrapUpReason Object ...`
	`  </WrapUpReason>`
	`  <WrapUpReason>`
	`  ... Full WrapUpReason Object ...`
	`  </WrapUpReason>`
	`  <WrapUpReason>`
	`  ... Full WrapUpReason Object ...`
	`  </WrapUpReason>`
	`</WrapUpReasons>`

HTTP Response:	200-Success

	401-Authorization failure 401-Invalid Authorization User Specified (for example, an authenticated user tried to use the identity of another user) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 404-Not found (for example, the wrap-up reason was deleted) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>jsmith</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	For possible error codes and their descriptions see Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	String	The URI to get a new copy of the WrapUpReason object.
label	String	The UI label for the wrap-up reason (for example, Sales Call, Complaint).
forAll	Boolean	Whether the wrap-up reason applies globally (true) or non-globally (false).

WrapUpReason - Create

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.

Finesse supports a maximum of 100 global wrap-up reasons and 100 non-global wrap-up reasons.

Note The forAll parameter determines whether a wrap-up reason is global (true) or non-global (false).

URI:	http://<FQDN>/finesse/api/WrapUpReason/
Example URI:	`http://finesse1.xyz.com/finesse/api/WrapUpReason/`
Security Constraints:	Role: Administrator 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>`
	`  <label>Recommendation</label>`
	`  <forAll>true</forAll>`
	`</WrapUpReason>`

HTTP Response:	200-Success 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. 400-Maximum exceeded. 401-Authorization failure 401-Invalid Authorization User Specified 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>jsmith</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	For possible error codes and their descriptions see Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
label	String	Yes	The UI label for the wrap-up reason (for example, Sales Call, Complaint).	-	The label must be unique. The label cannot be longer than 39 bytes (which is equal to 39 US English characters).
forAll	Boolean	Yes	Whether the wrap-up reason applies globally (true) or non-globally (false).	true, false	Must be true or false.

WrapUpReason - Update

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 At least one of the following parameters must be present in the HTTP request to update a wrap-up reason: label or forAll. If neither of these parameters is present, Finesse returns an Invalid Input error.

Finesse supports a maximum of 100 global wrap-up reasons and 100 non-global wrap-up reasons.

URI:	http://<FQDN>/finesse/api/WrapUpReason/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/WrapUpReason/23`
Security Restraints:	Role: Administrator 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> <label>Sales call</label> <forAll>true</forAll> </WrapUpReason>
HTTP Response:	200-Success (The Finesse server successfully updated the wrap-up reason) 400-Finesse API error 400-Maximum exceeded 401-Authorization failure 401-Invalid Authorization User Specified 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 404-Not found (for example, the wrap-up reason was deleted) 500-Internal server error
Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions see Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
label	String	Yes	The UI label for the wrap-up reason (for example, Sales Call, Complaint).	-	The label must be unique. The label cannot be longer than 39 bytes (which is equal to 39 US English characters).
forAll	Boolean	Yes	Whether the wrap-up reason applies globally (true) or non-globally (false).	true, false	Must be true or false.

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>
```

WrapUpReason - Delete

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://<FQDN>/finesse/api/WrapUpReason/<id>
Example URI:	`http://finesse.xyz.com/finesse/api/WrapUpReason/475`
Security Constraints:	Role: Administrator 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) 401-Authorization failure 401-Invalid Authorization User Specified 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Authorization Failure</ErrorType>`
	`  <ErrorMessage>UNAUTHORIZED</ErrorMessage>`
	`  <ErrorData>jsmith</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	For possible error codes and their descriptions see Cisco Finesse API Errors.

Phone Book APIs

The PhoneBook object represents a phone book that contains contacts. A phone book can be assigned globally or to specific teams. There is a system-wide maximum of 10 GLOBAL type phone books and 50 TEAM type phone books.

A phone book name can be a maximum of 64 characters.

Whenever you get a PhoneBook object, you also get a Contacts summary object with it.

The PhoneBook object is structured as follows:

```
<PhoneBook>
    <uri>/finesse/api/PhoneBook/{id}</uri>
    <name>PhoneBook 1</name>
    <type>GLOBAL</type>
    <contacts>/finesse/api/PhoneBook/{id}/Contacts</contacts>
</PhoneBook>
```

The PhoneBook object uses the following fields.

Field	Description	Type	Size	Required?
uri	In the uri, the id maps to the primary key of the phone book entry.	String		
type	The type of phone book: GLOBAL or TEAM.	String		
name	The name of the phone book.	String		

PhoneBook — Get Phone Book

The Get Phone Book API allows an administrator to retrieve a specific phone book.

URI:	http://<FQDN>/finesse/api/PhoneBook/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/1234
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML

Successful Response:	<PhoneBook> ... Full PhoneBook Object ... </PhoneBook>
HTTP Response:	200-Success 400-Bad request. The request body is invalid. 400-Finesse API error. There is an API error, for example, the object does not exist or the object is stale. 401-Authorization failure. The user is not yet authenticated in the web session. 401-Invalid authorization user specified. The authenticated user tried to use an identity that is not their own. 404-Not found. The phone book cannot be found. 500-Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	-
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	String	In the uri, the id maps to the primary key of the phone book entry.
type	String	The type of phone book (GLOBAL or TEAM).
name	String	The name of the phone book.

PhoneBook — Get List of Phone Books

The Get List of Phone Books API allows an administrator to get a list of all global and team phone books. It does not return agents' personal phone books.

URI:	http://<FQDN>/finesse/api/PhoneBooks
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBooks
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--
Successful Response:	<PhoneBooks> <PhoneBook> ... Full PhoneBook Object ... </PhoneBook> <PhoneBook> ... Full PhoneBook Object ... </PhoneBook> <PhoneBook> ... Full PhoneBook Object ... </PhoneBook> </PhoneBooks>
HTTP Response:	200--Success 400--Bad request. The request parameter is invalid. 400--Finesse API error. There is an API error, such as the object is stale or there is a violation of database constraints. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 401--Invalid authorization user specified. The authenticated user tried to use someone else's identity. 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	string	The ID in the URI maps to the primary key of the phone book entry
type	string	The type of phone book, either GLOBAL or TEAM
name	string	The name of the phone book

Phone Book — Add New Phone Book

The Add new phone book API allows an administrator to create a new phone book.

URI:	http://<FQDN>/finesse/api/PhoneBook/
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/
Security Constraints:	User must be logged in as an administrator
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<PhoneBook>`
	`  <name>PhoneBook 1</name>`
	`  <type>GLOBAL</type>`
	`</PhoneBook>`

Successful Response:	When the API returns a successful response code (200), the Finesse server has successfully created the new phone book The server response contains an empty response body and a location header denoting the absolute URL of the new phone book.
HTTP Response:	200--Success 400--Missing parameter. Failed to add the phone book because a parameter is missing. 400--Invalid input. Failed to add the phone book because one of the input parameters exceeded constraints. 401--Unauthorized. The user is not authenticated in the web session yet, or does not have the required role. 500--Runtime exception. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Description
type	string	The type of phone book, either GLOBAL or TEAM
name	string	The name of the phone book

Phone Book — Edit Phone Book

The Edit phone book API allows an administrator to update an existing phone book.

URI:	http://<FQDN>/finesse/api/PhoneBook/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/1234
Security Constraints:	User must be logged in as an administrator
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<PhoneBook>`
	`  <name>{string}</name>`
	`  <type>GLOBAL|TEAM</type>`
	`</PhoneBook>`

Successful Response:	When the API returns a successful response code (200), the Finesse server has successfully updated the specified phone book
HTTP Response:	200--Success 400--In use. Cannot change a work flow group phone book to global when it is in use. 400--Missing parameter. Failed to add the phone book because a parameter is missing. 400--Invalid input. Failed to add the phone book because one of the input parameters exceeded constraints. 401--Unauthorized. The user is not authenticated in the web session yet, or does not have the required role. 500--Runtime exception. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Description
type	string	The phone book type, either GLOBAL or TEAM
name	string	The name of the phone book (must be unique)

Phone Book — Delete Phone Book

The Delete Phone Book API allows an administrator to delete an existing phone book.

URI:	http://<FQDN>/finesse/api/PhoneBook/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/PhoneBook/1234`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	DELETE
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--
Successful Response:	When the API returns a successful response code (200), the Finesse server has successfully deleted the specified phone book
HTTP Response:	200--Success 400--In use. The phone book is assigned to the team named in the error message. 401--Unauthorized. The user is not authenticated in the web session, or does not have the required role. 404--Not found. Could not find a phone book with the specified ID. 500--Runtime exception. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Phone Book — Import List of Contacts (CSV File)

This API enables administrators to replace all the contacts in a specific phone book by importing contacts in a CSV file. The CSV file can contain a maximum of 1500 records. All existing records in the phone book are deleted before the new records are inserted. Records that contain errors are not loaded. Records that are free of errors, or records with missing or empty fields, are loaded. In general, the import is very fault-tolerant. The CSV file should be sent using standard web form syntax. It is delivered to the Finesse server as multipart/form data. This format is very particular about formatting---lines in the CSV file must be separated by carriage returns and newlines (\r\n).

URI:	http://<FQDN>/finesse/api/PhoneBook/<id>/Contacts/csvFileContent
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/1234/Contacts/csvFileContent
Security Constraints:	Role: Administrator Limitations: A user must be signed in as an administrator to import contacts.
HTTP Method:	PUT
Content Type:	text/CSV
Input/Output Format:	text/plain, text/CSV

Sample HTML Form:	`<form action="/finesse/api/PhoneBook/1/import" enctype="multipart/form-data" method="post">`
	`  <p>`
	`  File(s):`
	`  <input type="file" name="datafile" size="40">`
	`  </p>`
	`  <div>`
	`  <input type="submit" value="Import">`
	`  </div>`
	`</form>`

Sample HTTP Request:	`----------13290916118636`
	`Content-Disposition: form-data; name="phonebook"`
	`----------13290916118636`
	`Content-Disposition: form-data; name="datafile"; filename="pb.csv"`
	`Content-Type: application/vnd.ms-excel`
	
	`"First Name","Last Name","Phone Number","Notes"`
	`"Amanda","Cohen","6511234",""`
	`"Nicholas","Knight","6125551228","Sales"`
	`"Natalie","Lambert","9525559876","Benefits"`
	`"Joseph","Stonetree","6515557612","Manager"`

Successful Response:	The HTTP response code 202 (successfully accepted) only indicates successful completion of the request. The request will be processed and the actual response of the state change will be sent as part of and updated to the PhoneBook object.

HTTP Response:	202--Successfully accepted 400--Maximum exceeded. The maximum number of 1500 records was exceeded. 400--Invalid input. Some of the data could not be imported because it was invalid. The ErrorData field contains a list of lines that were not imported. Note that this represents a partial failure only, as some data might have been uploaded. If the ErrorData field contains no entries, it is an indication that there was no data to import. This could be because the file was empty or did not contain any valid lines. It could also be because the multipart mime message was improperly formatted or did not contain a file. In this case, the existing records are overwritten. 401--Unauthorized. The user is not authenticated in the web session yet, or does not have the proper role. 404--Not found. Could not find the phone book with the specified ID. 500--Runtime exception. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Phone Book — Import List of Contacts (XML Import)

This API allows administrators to replace all the contacts for a specific phone book by importing a contacts collection. This API can be used to import a maximum of 1500 records.

All existing records for the phone book are deleted before the new records are inserted.

URI:	http://<FQDN>/finesse/api/PhoneBook/<id>/Contacts
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/1234/Contacts
Security Constraints:	Role: Administrator Limitations: A user must be signed in as an administrator to import contacts.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

Sample HTTP Request:	<Contacts> <Contact> ...Full Contact Object... </Contact> <Contact> ...Full Contact Object... </Contact> </Contacts>
Successful Response:	The HTTP response code 202 (successfully accepted) only indicates successful completion of the request. The request is processed and the actual response of the state change is sent as part of, and updated to the PhoneBook object.
HTTP Response:	200--Invalid input. Some of the data could not be imported because it was invalid. The ErrorData field contains a list of lines that were not imported. Note that this represents a partial success only, because some data was uploaded. 400--Maximum exceeded. The maximum number of 1500 records was exceeded. 400--Invalid input. None of the data could be imported because it was invalid. The ErrorData field contains a list of lines that were not imported. If the ErrorData field contains no entries, it is an indication that there was no data to import. This could be because the file was empty or did not contain any valid lines. It could also be because the multipart mime message was improperly formatted or did not contain a file. In this case, the existing records are overwritten. 401--Unauthorized. The user is not authenticated in the web session yet, or does not have the proper role. 404--Not found. Could not find the phone book with the specified ID. 500--Runtime exception. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Phone Book — Export List of Contacts

This API enables administrators to export a list of contacts belonging in a specific phone book. The list is exported in CSV format.

URI:	http://<FQDN>/finesse/api/PhoneBook/<id>/export
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/1234/export
Security Constraints:	User must be logged in as an administrator

HTTP Method:	GET
Content Type:	text/CSV
Input/Output Format:	Multipart/form-data type=file

Sample Exported CSV File:	`"First Name","Last Name","Phone Number","Notes"`
	`"Amanda","Cohen","6511234",""`
	`"Nicholas","Knight","6125551228","Sales"`
	`"Natalie","Lambert","9525559876","Benefits"`
	`"Joseph","Stonetree","6515557612","Manager"`

Successful Response:	The HTTP response code 202 (successfully accepted) only indicates successful completion of the request. The request will be processed and the actual response of the state change will be sent as part of and updated to the PhoneBook object.
HTTP Response:	202--Successfully accepted 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error (for example, an object is stale or an object does not exist) 400--Parameter missing. The state value is not provided. I400--Invalid input. The state as part of user input is not recognized or is invalid. 401--Authorization failure. The user is not authenticated in the web session yet, or does not have the proper role. 401--Invalid authorization user specified. The authenticated user attempted to make a request as another user. 404--Not found. Could not find the phone book with the specified ID. 500--Invalid server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Contact APIs

The Contact object represents a contact that can be assigned to a phone book. There is a system-wide maximum of 1500 contacts and a per-phone book maximum of 1500 contacts.

The Contact object is structured as follows:

```
<Contact>
     <firstName>John</firstName>
     <lastName>Doe</lastName>
     <phoneNumber>5559120</phoneNumber>
     <description>true</description>
     <uri>/finesse/api/PhoneBook/{phoneBookId}/Contact/{id}</uri>
</Contact>
```

The Contact object uses the following fields.

Field	Description	Type	Size	Required
uri	in the URI, the phoneBookId maps to the primary key of the phone book that the contact belongs to. The id maps to the primary key of the contact entry.	string		
firstName	The contact's first name	string	128	no
lastName	The contact's last name	string	128	no
description	A description of the contact	string	128	no
phoneNumber	the contact's phone number	string	128	yes

Contact — Get Contact

The Get contact API allows an administrator to retrieve a specific phone book contact.

URI:	http://<FQDN>/finesse/api/PhoneBook/<phoneBookId>/Contact/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/1234/Contact/4567
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--

Successful Response:	`<Contact>`
	`<firstName>John</firstName>`
	`<lastName>Doe</lastName>`
	`<phoneNumber>5559120</phoneNumber>`
	`<description>true</description>`
	`<uri>/finesse/api/PhoneBook/{phoneBookId}/Contact/{id}</uri>`
	`</Contact>`

HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The contact cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	string	The URI of the contact. The phoneBookId maps to the primary key of the phone book that the contact belongs to. The id maps to the primary key of the contact entry.
firstName	string	The contact's first name
lastName	string	The contact's last name
description	string	The description of the contact
phoneNumber	string	The contact's phone number

Contact — Get Contact List

The Get contact list API allows an administrator to retrieve a list of contacts in a specific phone book.

URI:	http://<FQDN>/finesse/api/PhoneBook/<phoneBookId>/Contacts
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/1234/Contacts
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--

Successful Response:	<Contacts> <Contact> ... Full Contact Object ... </Contact> <Contact> ... Full Contact Object ... </Contact> <Contact> ... Full Contact Object ... </Contact> </Contacts>
HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The contact cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	string	The URI of the contact. The phoneBookId maps to the primary key of the phone book that the contact belongs to. The id maps to the primary key of the contact entry.
firstName	string	The contact's first name
lastName	string	The contact's last name

description	string	The description of the contact
phoneNumber	string	The contact's phone number

Contact — Get Contact List for Agent

The Get contact list for agent API allows a user to retrieve a list of contacts available to a specific user (agent) with the private key <userId>.

URI:	http://<FQDN>/finesse/api/User/<userId>/Contacts
Example URI:	http://finesse1.xyz.com/finesse/api/User/7894/Contacts
Security Constraints:	The user can be logged in with any role
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--
Successful Response:	<Contacts> <Contact> ... Full Contact Object ... </Contact> <Contact> ... Full Contact Object ... </Contact> <Contact> ... Full Contact Object ... </Contact> </Contacts>
HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The contact cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	string	The URI of the contact. The phoneBookId maps to the primary key of the phone book that the contact belongs to. The id maps to the primary key of the contact entry.
firstName	string	The contact's first name
lastName	string	The contact's last name
description	string	The description of the contact
phoneNumber	string	The contact's phone number

Contact — Add Contact

The Add contact API allows an administrator to add a new phone book contact.

URI:	http://<FQDN>/finesse/api/PhoneBook/<phoneBookId>/Contact/
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/1234/Contact/
Security Constraints:	User must be logged in as an administrator
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Contact>`
	`  ... Full Contact Object ...`
	`</Contact>`

Successful Response:	When the API returns a success response code (200), the server has successfully created the new contact, and its response will contain an empty response body, and a location header denoting the absolute URL of the new contact.

HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The contact cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Description
firstName	string	The contact's first name
lastName	string	The contact's last name
description	string	The description of the contact
phoneNumber	string	The contact's phone number

Contact — Edit Contact

The Edit contact API allows an administrator to modify a specific phone book contact.

URI:	http://<FQDN>/finesse/api/PhoneBook/<phoneBookId>/Contact/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/1234/Contact/4567
Security Constraints:	User must be logged in as an administrator
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Contact>`
	`... Full Contact Object ...`
	`</Contact>`

Successful Response:	When the API returns a success response code (200), the server has successfully updated the phone book contact.

HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The contact cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Description
firstName	string	The contact's first name
lastName	string	The contact's last name
description	string	The description of the contact
phoneNumber	string	The contact's phone number

Contact — Delete Contact

The Delete contact API allows an administrator to delete an existing phone book contact.

URI:	http://<FQDN>/finesse/api/PhoneBook/<phoneBookId>/Contact/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/1234/Contact/4567
Security Constraints:	User must be logged in as an administrator
HTTP Method:	DELETE
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--
Successful Response:	When the API returns a success response code (200), the server has successfully deleted the specified contact.

HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The contact cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	--
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

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 object 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 The MediaPropertiesLayout API supports callVariable1 through callVariable10, ECC variables, and the following blended agent (outbound) variables:

- BACampaign
- BAAccountNumber
- BAResponse
- BAStatus
- BADialedListID
- BATimeZone
- BABuddyName

MediaPropertiesLayout — Get

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

Finesse supports only a single default instance.

URI:	http://<FQDN>/finesse/api/MediaPropertiesLayout/default
Example URI:	`http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/default`
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-

Successful Response:	`<MediaPropertiesLayout>`
	`... Full MediaPropertiesLayout Object ...`
	`</MediaPropertiesLayout>`

HTTP Response:	200-Success 401-Unauthorized (for example, the user is not authenticated in the Web Session) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error

Failure Response Example:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Error Codes:	• Authorization Failure • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
header	Entry object	A single entry (combination of displayName and mediaProperty) that appears in the call header on the desktop for each call.
column	List of entry objects	Grouping of media properties for agent and supervisor desktops. Finesse 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.
entry	Entry object containing a name and media property	A displayName and mediaProperty combination. The displayName can be empty.

Parameter	Type	Description
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).
mediaProperty	String	The name of the variable that is displayed to the agent or supervisor (maximum length of 32 characters). Each entry must have exactly one mediaProperty. Allowed strings include callVariable1 through callVariable10, any valid ECC variable (user.*), and any of the following Outbound Option variables: • BACampaign • BAAccountNumber • BAResponse • BAStatus • BADialedListID • BATimeZone • BABuddyName

MediaPropertiesLayout - Set

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

URI:	http://<FQDN>/finesse/api/MediaPropertiesLayout/default
Example URI:	`http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/default`
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<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>
HTTP Response:	200-Success (the new settings were successfully written to the database) 400-Parameter missing (at least one of the required parameters was not provided) 400-Invalid input (at least one of the parameters provided is not valid) 401-Authorization failure (for example, the user is not yet authenticated in the Web Session) 403-Forbidden (configuration APIs cannot be run against the secondary Finesse server) 500-Internal server error
Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>The entry contained an invalid media property: callVariable11</ErrorData> <ErrorType>Invalid Input</ErrorType> <ErrorMessage>HTTP Status code: 400 (Bad Request) Api Error Type: Invalid Input Error Message: Invalid media property name 'callVariable11' </ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	• Parameter Missing • Invalid Input • Authorization Failure • Forbidden • Internal Server Error For descriptions and other possible error codes, see Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Required	Description	Possible Values	Validation
header	Entry object	No	A single entry (combination of displayName and mediaProperty) that appears in the call header on the desktop for each call.	-	-
column	List of entry objects	No	Grouping of media properties for agent and supervisor desktops. Finesse 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 specified in the PUT request appears on the left of the call control gadget on the desktop. The second column appears on the right.	-	Valid entry tags
entry	Entry object containing a name and media property	No	A displayName and mediaProperty combination.	-	Each entry must have exactly one displayName and one mediaProperty. The displayName can be empty.

Parameter	Type	Required	Description	Possible Values	Validation
displayName	String	Yes	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.	Any valid string (including an empty string).	Each entry must have exactly one displayName. HTML tags are rendered as plain text and do not retain formatting. The maximum length is 50 characters.
mediaProperty	String	Yes	Part of an entry. The name of the variable that is displayed to the agent or supervisor (maximum length of 32 characters).	Allowed strings include callVariable1 through callVariable10, any valid ECC variable (user.*), and any of the following Outbound Option variables: • BACampaign • BAAccountNumber • BAResponse • BAStatus • BADialedListID • BATimeZone • BABuddyName	Each entry must have exactly one mediaProperty. The maximum length is 32 characters.

Team APIs

The Team object represents a team and the resources associated with that team.

Team — Get List of Teams

The Get List of Teams API allows an administrator to retrieve a list of teams. The team must have agents and/or supervisors assigned to it in order for the team to appear in the retrieved list.

URI:	http://<FQDN>/finesse/api/Teams

Example URI:	`http://finesse1.xyz.com/finesse/api/Teams` http://host/finesse/api/Teams
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--

Successful Response:	`<Teams>`
	`<Team>`
	`... Summary Team Object ...`
	`</Team>`
	`<Team>`
	`... Summary Team Object ...`
	`</Team>`
	`<Team>`
	`... Summary Team Object ...`
	`</Team>`
	`<Team>`
	`... Summary Team Object ...`
	`</Team>`
	`<Team>`
	`... Summary Team Object ...`
	`</Team>`
	`... Additional Teams...`
	`</Teams>`

HTTP Response:	200--Success 401--Authorization failed. The user is unauthorized, for example is not authenticated in the web session. 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorType>Unauthorized</ErrorType>`
	`<ErrorMessage>The user is not authorized to perform this operation</ErrorMessage>`
	`</ApiError>`
	`</ApiErrors>`

Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	string	The URI to get a new copy of the Team object
id	string	The unique identifier of the team.
name	string	The name of the team.

Team — Get List of Reason Codes for Team

The Get List of Reason Codes for a Team API allows an administrator to retrieve a list of reason codes for the specified category assigned to a specific team. The list is in the same format as defined in the Reason Code APIs, on page 124

URI:	http://<FQDN>/finesse/api/Team/<Id>/ReasonCodes?category=<category>
Example URI:	`http://finesse1.xyz.com/finesse/api/Team/1234/ReasonCodes?category=NOT_READY`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-

Successful Response:	<ReasonCodes category="NOT_READY"> <ReasonCode> ... Full Reason Code Object ... </ReasonCode> <ReasonCode> ... Full Reason Code Object ... </ReasonCode> <ReasonCode> ... Full Reason Code Object ... </ReasonCode> .... </ReasonCodes>
HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The reason code cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>500</ErrorData> <ErrorType>finesse.api.team.team_assignment_invalid_ team&</ErrorType> <ErrorMessage>HTTP Status code: 404 (Not Found) Api Error Type:finesse.api.team.team_assignment_invalid_team Error Message: This is not a valid team</ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Request Parameters

Parameter	Type	Description

category	string	This property specifies whether to assign Not Ready or Sign Out (Logout) reason codes to the team. Possible values are NOT_READY and LOGOUT.

Team — Update List of Reason Codes for Team

The Update list of reason codes for team API allows an administrator to assign/unassign a list of reason codes of the specified category to a team.

If multiple users try to update the same team's reason codes at the same time, the changes made by the last user to update will overwrite changes made by the other users.

This list includes all reason codes of the specified category that are assigned to a team. Any reason codes you assign or unassign will overwrite the current reason code list.

Note The category attribute of the ReasonCodes tag is not required for the update. If it is included in the request, it will be ignored. However, all the reason codes in the list must have a category specified in the category query parameter. Inclusion of a reason code whose category does not match, or is global, will result in a Finesse API error (Status 400).

URI:	http://<FQDN>/finesse/api/Team/<Id>/ReasonCodes?category=<category>
Example URI:	`http://finesse1.xyz.com/finesse/api/Team/1234/ReasonCodes?category=NOT_READY`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<ReasonCodes> <ReasonCode> <uri>/finesse/api/ReasonCode/123</uri> </ReasonCode> <ReasonCode> <uri>/finesse/api/ReasonCode/456</uri> </ReasonCode> <ReasonCode> <uri>/finesse/api/ReasonCode/789</uri> </ReasonCode> .... </ReasonCodes>
HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The reason code cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>category NOT_READ is invalid</ErrorData> <ErrorType>Invalid Input</ErrorType> <ErrorMessage>HTTP Status code:400 (Bad Request) Api Error Type:Invalid Input Error Message:Category must be NOT_READY or LOGOUT</ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Description
category	string	This property specifies whether to assign Not Ready or Sign Out (Logout) Reason Codes to the team. Possible values are NOT_READY and LOGOUT.

Team — Get List of Wrap-Up Reasons for Team

The Get list of wrap-up reasons for team API allows an administrator to retrieve a list of wrap-up reasons assigned to a specific team. The list is in the same format as defined in the Wrap-Up Reason APIs, on page 133

URI:	http://<FQDN>/finesse/api/Team/<TeamId>/WrapUpReasons
Example URI:	`http://finesse1.xyz.com/finesse/api/Team/1234/WrapUpReasons`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-

Successful Response:	`<WrapUpReasons>`
	`  <WrapUpReason>`
	`  ... Full WrapUpReason Object ...`
	`  </WrapUpReason>`
	`  <WrapUpReason>`
	`  ... Full WrapUpReason Object ...`
	`  </WrapUpReason>`
	`  <WrapUpReason>`
	`  ... Full WrapUpReason Object ...`
	`  </WrapUpReason>`
	`</WrapUpReasons>`

HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The wrap-up reason cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorData>6000</ErrorData>`
	`  <ErrorType>finesse.api.team.team_assignment_invalid_`
	`  team</ErrorType>`
	`  <ErrorMessage>HTTP Status code:404 (Not Found)`
	`  Api Error Type:finesse.api.team.team_assignment_`
	`  invalid_team`
	`  Error Message:This is not a valid team</ErrorMessage>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description

uri	String	The URI to get a new copy of the WrapUpReason object.
label	String	The UI label for the wrap-up reason (for example, Sales Call, Complaint)
forAll	Boolean	Indicates if the wrap-up reason applies to all agents. Value is always "false".

Team — Update List of Wrap-Up Reasons for Team

The Update list of wrap-up reasons for team API allows an administrator to assign or unassign a list of wrap-up reasons to a team.

If multiple users try to update the same team's wrap-up reasons at the same time, the changes made by the last user to update will overwrite changes made by the other users.

This list includes all wrap-up reasons that are assigned to a team. Any wrap-up reasons you assign or unassign will overwrite the current wrap-up reason list.

URI:	http://<FQDN>/finesse/api/Team/<TeamId>/WrapUpReasons
Example URI:	`http://finesse1.xyz.com/finesse/api/Team/1234/WrapUpReasons`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<WrapUpReasons>`
	`  <WrapUpReason>`
	`  <uri>/finesse/api/WrapUpReason/12345</uri>`
	`  </WrapUpReason>`
	`  <WrapUpReason>`
	`  <uri>/finesse/api/WrapUpReason/98765</uri>`
	`  </WrapUpReason>`
	`  <WrapUpReason>`
	`  <uri>/finesse/api/WrapUpReason/45678</uri>`
	`  </WrapUpReason>`
	`...`
	`</WrapUpReasons>`

HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The wrap-up reason cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>The Wrap-Up Reason is invalid</ErrorData> <ErrorType>Invalid Input</ErrorType> <ErrorMessage>HTTP Status code:400 (Bad Request) Api Error Type:Invalid Input Error Message:Wrap-up reason does not exist, id=530</ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Team — Get List of Phone Books for Team

The Get list of phone books for team API allows an administrator to retrieve a list of phone books assigned to a specific team. This list is in the same format as defined in the Phone Book APIs, on page 141.

URI:	http://<FQDN>/finesse/api/Team/<TeamId>/PhoneBooks
Example URI:	http://finesse1.xyz.com/finesse/api/Team/1234/PhoneBooks
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-
Successful Response:	<PhoneBook> <PhoneBook> ... Full PhoneBook Object ... </PhoneBook> <PhoneBook> ... Full PhoneBook Object ... </PhoneBook> <PhoneBook> ... Full PhoneBook Object ... </PhoneBook> </PhoneBook>
HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The phone book cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	<ApiErrors>
	  <ApiError>
	  <ErrorData>6000</ErrorData>
	  <ErrorType>finesse.api.team.team_assignment_
	  invalid_team</ErrorType>
	  <ErrorMessage>HTTP Status code: 404 (Not Found)
	  Api Error Type:finesse.api.team.team_assignment_
	  invalid_team Error Message:
	  This is not a valid team</ErrorMessage>
	  </ApiError>
	</ApiErrors>

Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	String	The URI to get a new copy of the PhoneBook object.
type	String	The type of phone book, GLOBAL or TEAM
name	String	The name of the phone book

Team — Update List of Phone Books for Team

The Update list of phone books for team API allows an administrator to assign or unassign a list of phone books to a team.

If multiple users try to update the same team's phone books at the same time, the changes made by the last user to update will overwrite changes made by the other users.

This list includes all phone books that are assigned to a team. Any phone books you assign or unassign will overwrite the current phone books list.

URI:	http://<FQDN>/finesse/api/Team/<TeamId>/PhoneBooks
Example URI:	`http://finesse1.xyz.com/finesse/api/Team/1234/PhoneBooks`
Security Constraints:	User must be logged in as an administrator

HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	<PhoneBooks> <PhoneBook> <uri>/finesse/api/PhoneBook/12345</uri> </PhoneBook> <PhoneBook> <uri>/finesse/api/PhoneBook/98765</uri> </PhoneBook> <PhoneBook> <uri>/finesse/api/PhoneBook/45678</uri> </PhoneBook> ... </PhoneBooks>
HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 404--Not found. The phone book cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, the connection might have been broken with the CTI server or any other component.
Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>Type must be team</ErrorData> <ErrorType>Invalid Input</ErrorType> <ErrorMessage>HTTP Status code:400 (Bad Request) Api Error Type:Invalid Input Error Message: Type needs to be TEAM for Team Assignments</ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see theCisco Finesse API Errors.

Team — Get Layout Configuration Assigned to Team

The Get layout configuration assigned to team API allows an administrator to retrieve the layout configuration assigned to a specific team.

URI:	http://<FQDN>/finesse/api/Team/<TeamID>/LayoutConfig
Example URI:	http://finesse1.xyz.com/finesse/api/Team/1234/LayoutConfig
Security Constraints:	User must be logged in as an administrator

HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
Successful Response:	<TeamLayoutConfig> <useDefault>false</useDefault> <layoutxml> <finesseLayout xmlns="http://www.cisco.com/vtg/finesse"> <layout> <role>Agent</role> ... </layout> <layout> <role>Supervisor</role> ... </layout> </finesseLayout> </layoutxml> </TeamLayoutConfig>
HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, for example, the object does not exist or the object is stale. 401--Authorization failure. The user is not yet authenticated in the web session. 404--Not found. The team cannot be found. 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>50000</ErrorData> <ErrorType>finesse.api.team.team_assignment_ invalid_team</ErrorType> <ErrorMessage>HTTP Status code:404 (Not Found) Api Error Type:finesse.api.team.team_assignment_ invalid_team Error Message: This is not a valid team</ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
useDefault	Boolean	Determines if the default layout is being used for this team. A value of true indicates that layoutxml is the default layout. A value of false indicates that layoutxml is a team-specific layout.
layoutxml	String	The xml data that determines the layout. It contains a team-specific layout if one is defined. Otherwise it contains the xml for the default layout. It must be valid XML that conforms to the Finesse schema. The layoutxml is in the same format as defined in Layout Configuration APIs.

Team — Update Layout Configuration Assigned to Team

The Update layout configuration assigned to team API allows an administrator to assign or unassign a layout configuration to a team.

If multiple users try to update the same team's layout configuration at the same time, the changes made by the last user to update will overwrite changes made by the other users.

URI:	http://<FQDN>/finesse/api/Team/<TeamID>/LayoutConfig
Example URI:	`http://finesse1.xyz.com/finesse/api/Team/1234/LayoutConfig`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	Example of assigning a team-specific layout:
	`<TeamLayoutConfig>`
	`  <useDefault>false</useDefault>`
	`  <layoutxml>`
	`  <finesseLayout xmlns="http://www.cisco.com/vtg/finesse">`
	`  <layout>`
	`  <role>Agent</role>`
	`  ...`
	`  </layout>`
	`  <layout>`
	`  <role>Supervisor</role>`
	`  ...`
	`  </layout>`
	`  </finesseLayout>`
	`  </layoutxml>`
	`</TeamLayoutConfig>`
	Example of assigning the team to use the default layout:
	`<TeamLayoutConfig>`
	`  <useDefault>true</useDefault>`
	`</TeamLayoutConfig>`

HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, for example, the object does not exist or the object is stale. 401--Authorization failure. The user is not yet authenticated in the web session. 404--Not found. The team cannot be found. 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>50000</ErrorData> <ErrorType>finesse.api.team.team_assignment_invalid_team</ErrorType> <ErrorMessage>HTTP Status code:404 (Not Found) Api Error Type: finesse.api.team.team_assignment_invalid_team Error Message: This is not a valid team</ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Description
useDefault	Boolean	Determines if the default layout should be used for this team. A value of true sets the team to use the default layout. A value of false sets the team to use the layout provided in layoutxml.
layoutxml	String	Required if useDefault is set to false. Not required if useDefault is set to true. The xml data that determines the layout. If useDefault is true and layoutxml is provided, layoutxml is ignored. The layoutxml is in the same format as defined in Layout Configuration APIs.

Team — Get List of Workflows

The Team - Get list of workflows API allows an administrator to retrieve a list of workflows assigned to a specific team. The list is in the same format as defined in Workflow APIs, on page 180.

URI:	https://<FQDN>/finesse/api/Team/<TeamId>/Workflows
Example URI:	`http://finesse1.xyz.com/finesse/api/Team/1234/Workflows`
Security Constraints:	User must be logged in as an administrator

HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	-
Successful Response:	<Workflows> <Workflow> ... Summary Workflow Object ... </Workflow> <Workflow> ... Summary Workflow Object ... </Workflow> <Workflow> ... Summary Workflow Object ... </Workflow> ... </Workflows>
HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 403--Forbidden. Configuration APIs cannot be run against the secondary Finesse server. 404--Not found. The workflow cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>500</ErrorData> <ErrorType>finesse.api.team.team_assignment_invalid_team</ErrorType> <ErrorMessage>HTTP Status code:404 (Not Found) Api Error Type:finesse.api.team.team_assignment_invalid_team Error Message:This is not a valid team</ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	String	The ID in the URI maps to the primary key of the workflow entry.
name	String	The name of the workflow.
description	String	The description of the workflow.

Team — Update List of Workflows

The Team - Update list of workflows API allows an administrator to assign/unassign a list of workflows to a team.

If multiple users try to update the same team's workflows at the same time, the changes made by the last user to update will overwrite changes made by the other users.

This list includes all workflows that are assigned to a team. Any workflows you assign or unassign will overwrite the current workflow list. Since the order in which workflows are evaluated is important, the order of the workflows in the list are preserved in the GET method (see Team — Get List of Workflows, on page 177).

URI:	http://<FQDN>/finesse/api/Team/<TeamId>/Workflows
Example URI:	`http://finesse1.xyz.com/finesse/api/Team/1234/Workflows`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<Workflows> <Workflow> <uri>/finesse/api/Workflow/4321</uri> </Workflow> <Workflow> <uri>/finesse/api/Workflow/9876</uri> </Workflow> <Workflow> <uri>/finesse/api/Workflow/6543</uri> </Workflow> ... </Workflows>
HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 403--Forbidden. Configuration APIs cannot be run against the secondary Finesse server. 404--Not found. The workflow cannot be found (for example, it was deleted) 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>The Workflow is invalid</ErrorData> <ErrorType>Invalid Input</ErrorType> <ErrorMessage>HTTP Status code:400 (Bad Request) Api Error Type:Invalid Input Error Message: Workflow does not exist, id=530</ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Workflow APIs

The Workflow object represents a workflow that can be assigned to a team. Workflows manage agent activity based on call events. Workflows have triggers and conditions used to determine whether the associated actions are executed.

Note A Summary Workflow Object contains the uri, name, and description attributes only.

TheWorkflow object is structured as follows:

```
<Workflow>
  <uri>/finesse/api/Workflow/{id}</uri>
  <name>Name of Workflow</name>
  <description>Description for Workflow</description>
```

```
<TriggerSet>
  <type>SYSTEM</type>
  <name>CALL_ARRIVES</name>
  <triggers>
    <Trigger>
      <Variable>
        <name>mediaType</name>
        <node>//Dialog/mediaType</node>
        <type>CUSTOM</type>
      </Variable>
      <comparator>IS_EQUAL</comparator>
      <value>Voice</value>
    </Trigger>
    <Trigger>
      <Variable>
        <name>callType</name>
        <node>//Dialog/mediaProperties/callType</node>
        <type>CUSTOM</type>
      </Variable>
      <comparator>IS_IN_LIST</comparator>
      <value>ACD_IN,PREROUTE_ACD_IN,PREROUTE_DIRECT_AGENT,TRANSFER,OVERFLOW_IN,
       OTHER_IN,AGENT_OUT,AGENT_INSIDE,OFFERED,CONSULT,CONSULT_OFFERED,
       CONSULT_CONFERENCE,CONFERENCE,TASK_ROUTED_BY_ICM,
       TASK_ROUTED_BY_APPLICATION</value>
    </Trigger>
    <Trigger>
      <Variable>
        <name>state</name>
        <node>//Dialog/participants/Participant/mediaAddress
         [.=${userExtension}]/../state</node>
        <type>CUSTOM</type>
      </Variable>
      <comparator>IS_IN_LIST</comparator>
      <value>ALERTING,ACTIVE,HELD</value>
    </Trigger>
    <Trigger>
      <Variable>
        <name>fromAddress</name>
        <node>//Dialog/fromAddress</node>
        <type>CUSTOM</type>
      </Variable>
      <comparator>IS_NOT_EQUAL</comparator>
      <Variable>
        <name>userExtension</name>
        <type>SYSTEM</type>
      </Variable>
    </Trigger>
  </triggers>
</TriggerSet>
<ConditionSet>
  <applyMethod>ALL</applyMethod>
  <conditions>
    <Condition>
      <Variable>
        <name>callVariable1</name>
        <type>SYSTEM</type>
      </Variable>
      <comparator>CONTAINS</comparator>
      <value>1234</value>
    </Condition>
    <Condition>
      <Variable>
        <name>user.foo.bar[1]</name>
        <node>/dialogs/Dialog/mediaProperties/callvariables/CallVariable/
         name[.="user.foo.bar[1]"]/../value</node>
        <type>CUSTOM</type>
      </Variable>
      <comparator>IS_NOT_EMPTY</comparator>
    </Condition>
  </conditions>
</ConditionSet>
<workflowActions>
  <WorkflowAction>
```

```
      <name>Google</name>
      <type>BROWSER_POP</type>
      <uri>/finesse/api/WorkflowAction/1234</uri>
    </WorkflowAction>
    <WorkflowAction>
      <name>AltaVista</name>
      <type>BROWSER_POP</type>
      <uri>/finesse/api/WorkflowAction/9876</uri>
    </WorkflowAction>
  </workflowActions>
</Workflow>
```

The Workflow object uses the following fields.

Field	Description	Type	Size	Required?
uri	In the uri, the id maps to the primary key of the Workflow entry	String		No
name	The name of the workflow. It must be unique.	String	40	Create only
description	A description of the workflow.	String	128	No
TriggerSet	A set of events that cause the conditions to be evaluated (see below).	Collection		Yes
ConditionSet	A set of conditions that determine if the Workflow will be executed. There can be at most 5 conditions assigned to a workflow.	Collection		No
workflowActions	A list of workflow actions (WorkflowAction objects) that are to be executed if the trigger and its conditions are satisfied by the data in the event. The actions are executed in the order they appear in this list. There can be at most 5 workflow actions assigned to a workflow. When getting a workflow or list of workflows, this list will contain summary workflow actions that contain the name, type, and URL of each workflow action. When creating or updating a workflow, only the URL is required in each workflow action. For more on the WorkflowAction object, see WorkflowAction APIs, on page 197.	Collection		No

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.

The ConditionSet subobject is structured as follows:

```
<ConditionSet>
```

```
    <applyMethod>ALL</applyMethod>
    <conditions>
        <Condition>
            <Variable>
                <name>callVariable1</name>
                <type>SYSTEM</type>
            </Variable>
            <comparator>CONTAINS</comparator>
            <value>1234</value>
        </Condition>
        <Condition>
            <Variable>
                <name>user.foo.bar[1]</name>

<node>/dialogs/Dialog/mediaProperties/callvariables/CallVariable/name[.="user.foo.bar[1]"]/../value</node>

                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_NOT_EMPTY</comparator>
        </Condition>
    </conditions>
</ConditionSet>
```

The ConditionSet subobject uses the following fields:

Field	Description	Type	Size	Required?
applyMethod	Determines whether ANY or ALL of the conditions must be met for the Workflow to execute.	String		Yes
conditions	List of Condition subobjects. There can be no more than 5 conditions for the workflow. A workflow with no Conditions is specified by a conditions element with no Condition elements.	Collection		Yes

The Condition subobject is structured as follows:

```
<Condition>
    <Variable>
        <name>callVariable1</name>
        <type>SYSTEM</type>
    </Variable>
    <comparator>CONTAINS</comparator>
    <value>1234</value>
</Condition>
```

The Condition subobject uses the following fields:

Field	Description	Type	Size	Required?
Variable	A piece of data from the Trigger event that is to be used to filter the event. The variable value will have leading and trailing white spaces removed during evaluations. Comma-separated values in a list will also have their leading and trailing white spaces removed during evaluations. If the value contains only white spaces, it is treated as an empty value			Yes

comparator	The operator that will be used to compare the event variable to the desired value. Possible values are IS_EQUAL, IS_NOT_EQUAL, BEGINS_WITH, ENDS_WITH, CONTAINS, IS_EMPTY, IS_NOT_EMPTY, IS_IN_LIST, IS_NOT_IN_LIST	String	32	Yes
value	The value to compare the event value to. Possible values are: If the comparator is IS_IN_LIST or IS_NOT_IN_LIST, the value is one of a comma separated list of values. If an explicit comma is needed it must be escaped with a backslash (\,). If a backslash is needed it must be escaped with a backslash (for example. apple,slash\\here,comma\,here,ball). The value will have leading and trailing white spaces removed during evaluations. Comma-separated values in a list will also have their leading and trailing white spaces removed during evaluations. If the value contains only white spaces, it is treated as an empty value	String	500	Yes, if the comparator is IS_EQUAL, IS_NOT_EQUAL, BEGINS_WITH, ENDS_WITH, CONTAINS, IS_IN_LIST, or IS_NOT_IN_LIST

The TriggerSet subobject is structured as follows:

```
<TriggerSet>
   <type>SYSTEM</type>
   <name>CALL_ARRIVES</name>
   <triggers>
      <Trigger>
         <Variable>
            <name>state</name>
            <node>//dialogs/Dialog/state</node>
            <type>CUSTOM</type>
         </Variable>
         <comparator>IS_EQUAL</comparator>
         <value>ALERTING</value>
      </Trigger>
      <Trigger>
         <Variable>
            <name>mediaType</name>
            <node>//dialogs/Dialog/mediaType</node>
            <type>CUSTOM</type>
         </Variable>
         <comparator>IS_EQUAL</comparator>
         <value>Voice</value>
      </Trigger>
      <Trigger>
         <Variable>
            <name>callType</name>
            <node>//dialogs/Dialog/mediaProperties/callType</node>
            <type>CUSTOM</type>
         </Variable>
         <comparator>CONTAINS</comparator>
         <value>_IN</value>
      </Trigger>
   </triggers>
</TriggerSet>
```

The TriggerSet subobject uses the following fields:

Field	Description	Type	Size	Required?
type	The type of the TriggerSet. Possible value is SYSTEM.	String	32	Yes
name	The name of the TriggerSet. When type is SYSTEM, the valid values are CALL_ARRIVES, CALL_ANSWERED, CALL_ENDS, CALL_IS_MADE, and CALL_IS_PREVIEWED.	String	32	Yes
triggers	List of Trigger subobjects. For workflow admin, this field is not returned and is ignored if type is SYSTEM.	Collection		No

The following SYSTEM Triggersets are defined by the finesses system. When creating a Workflow only the name and type of SYSTEM must be specified. They will automatically be expanded when retrieved by the API "User - Get List of Workflows for a User" as follows:

CALL_ARRIVES

```
<TriggerSet>
    <type>SYSTEM</type>
    <name>CALL_ARRIVES</name>
    <triggers>
        <Trigger>
            <Variable>
                <name>mediaType</name>
                <node>//Dialog/mediaType</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_EQUAL</comparator>
            <value>Voice</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>callType</name>
                <node>//Dialog/mediaProperties/callType</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_IN_LIST</comparator>
          <value>ACD_IN,PREROUTE_ACD_IN,PREROUTE_DIRECT_AGENT,TRANSFER,OVERFLOW_IN,OTHER_IN,


AGENT_OUT,OUT,OUTBOUND,AGENT_INSIDE,OFFERED,CONSULT,CONSULT_OFFERED,CONSULT_CONFERENCE,
          CONFERENCE,TASK_ROUTED_BY_ICM,TASK_ROUTED_BY_APPLICATION,VOICE_CALL_BACK,NON_ACD,

            SUPERVISOR_BARGE_IN,NULL</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>state</name>

<node>//Dialog/participants/Participant/mediaAddress[.=${extension}]/../state</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_IN_LIST</comparator>
            <value>ALERTING,ACTIVE,HELD</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>fromAddress</name>
                <node>//Dialog/fromAddress</node>
                <type>CUSTOM</type>
            </Variable>
```

```
            <comparator>IS_NOT_EQUAL</comparator>
            <value>${extension}</value>
        </Trigger>
    </triggers>
</TriggerSet>
```

CALL_ANSWERED

```
<TriggerSet>
    <type>SYSTEM</type>
    <name>CALL_ANSWERED</name>
    <triggers>
        <Trigger>
            <Variable>
                <name>mediaType</name>
                <node>//Dialog/mediaType</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_EQUAL</comparator>
            <value>Voice</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>callType</name>
                <node>//Dialog/mediaProperties/callType</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_IN_LIST</comparator>
            <value>ACD_IN,PREROUTE_ACD_IN,PREROUTE_DIRECT_AGENT,TRANSFER,OVERFLOW_IN,
            OTHER_IN,AGENT_OUT,OUT,OUTBOUND,AGENT_INSIDE,OFFERED,CONSULT,CONSULT_OFFERED,
            CONSULT_CONFERENCE,CONFERENCE,TASK_ROUTED_BY_ICM,TASK_ROUTED_BY_APPLICATION,
            VOICE_CALL_BACK,NON_ACD,SUPERVISOR_BARGE_IN,NULL</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>state</name>

<node>//Dialog/participants/Participant/mediaAddress[.=${extension}]/../state</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_EQUAL</comparator>
            <value>ACTIVE</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>fromAddress</name>
                <node>//Dialog/fromAddress</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_NOT_EQUAL</comparator>
            <value>${extension}</value>
        </Trigger>
    </triggers>
</TriggerSet>
```

CALL_ENDS

```
<TriggerSet>
    <type>SYSTEM</type>
    <name>CALL_ENDS</name>
    <triggers>
        <Trigger>
            <Variable>
                <name>mediaType</name>
                <node>//Dialog/mediaType</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_EQUAL</comparator>
            <value>Voice</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>callType</name>
                <node>//Dialog/mediaProperties/callType</node>
                <type>CUSTOM</type>
```

```
            </Variable>
            <comparator>IS_IN_LIST</comparator>
            <value>ACD_IN,PREROUTE_ACD_IN,PREROUTE_DIRECT_AGENT,TRANSFER,OVERFLOW_IN,
            OTHER_IN,AGENT_OUT,OUT,OUTBOUND,AGENT_INSIDE,OFFERED,CONSULT,CONSULT_OFFERED,
            CONSULT_CONFERENCE,CONFERENCE,TASK_ROUTED_BY_ICM,TASK_ROUTED_BY_APPLICATION,
            VOICE_CALL_BACK,NON_ACD,SUPERVISOR_BARGE_IN,NULL</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>state</name>

<node>//Dialog/participants/Participant/mediaAddress[.=${extension}]/../state</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_IN_LIST</comparator>
            <value>DROPPED,WRAP_UP</value>
        </Trigger>
    </triggers>
</TriggerSet>
```

CALL_IS_MADE

```
<TriggerSet>
    <type>SYSTEM</type>
    <name>CALL_IS_MADE</name>
    <triggers>
        <Trigger>
            <Variable>
                <name>mediaType</name>
                <node>//Dialog/mediaType</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_EQUAL</comparator>
            <value>Voice</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>callType</name>
                <node>//Dialog/mediaProperties/callType</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_IN_LIST</comparator>
            <value>ACD_IN,PREROUTE_ACD_IN,PREROUTE_DIRECT_AGENT,TRANSFER,OVERFLOW_IN,
            OTHER_IN,AGENT_OUT,OUT,OUTBOUND,AGENT_INSIDE,OFFERED,CONSULT,CONSULT_OFFERED,
            CONSULT_CONFERENCE,CONFERENCE,TASK_ROUTED_BY_ICM,TASK_ROUTED_BY_APPLICATION,
            VOICE_CALL_BACK,NON_ACD,SUPERVISOR_BARGE_IN,NULL</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>state</name>

<node>//Dialog/participants/Participant/mediaAddress[.=${extension}]/../state</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_IN_LIST</comparator>
            <value>INITIATED,FAILED,ACTIVE,HELD</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>fromAddress</name>
                <node>//Dialog/fromAddress</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_EQUAL</comparator>
            <value>${extension}</value>
        </Trigger>
    </triggers>
</TriggerSet>
```

CALL_IS_PREVIEWED

```
<TriggerSet>
    <type>SYSTEM</type>
    <name>CALL_IS_PREVIEWED</name>
    <triggers>
```

```
        <Trigger>
            <Variable>
                <name>mediaType</name>
                <node>//Dialog/mediaType</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_EQUAL</comparator>
            <value>Voice</value>
        </Trigger>
        <Trigger>
            <Variable>
                <name>callType</name>
                <node>//Dialog/mediaProperties/callType</node>
                <type>CUSTOM</type>
            </Variable>
            <comparator>IS_EQUAL</comparator>
            <value>OUTBOUND_PREVIEW</value>
        </Trigger>
    </triggers>
</TriggerSet>
```

The Trigger subobject is structured as follows:

```
<Trigger>
   <Variable>
      <name>state</name>
      <node>//dialogs/Dialog/state</node>
      <type>CUSTOM</type>
   </Variable>
   <comparator>IS_EQUAL</comparator>
   <value>ALERTING</value>
</Trigger>
```

The Trigger subobject uses the following fields:

Field	Description	Type	Size	Required?
Variable	A piece of data from the trigger event that is to be used to filter the event.			Yes (Create/Update)
comparator	The operator that will be used to compare the event variable to the desired value. Possible values are: • IS_EQUAL • IS_NOT_EQUAL • BEGINS_WITH • ENDS_WITH • CONTAINS • IS_EMPTY • IS_NOT_EMPTY • IS_IN_LIST • IS_NOT_IN_LIST	String	32	Yes

value	The value to compare the event variable with. If the comparator is IS_IN_LIST or IS_NOT_IN_LIST, the value is one of a comma-separated list of values. If an explicit comma is needed, it must be escaped with a backslash (\,). If a backslash is needed, it must be escaped with a backslash (\\) (for example, apple,slash\\here,comma\,here,ball).	String	500	Yes, if comparator is IS_EQUAL, IS_NOT_EQUAL, BEGINS_WITH, ENDS_WITH, CONTAINS, IS_IN_LIST, or IS_NOT_IN_LIST

The Variable subobject is structured as follows:

```
<Variable>
   <name>state</name>
   <node>//dialogs/Dialog/state</node>
   <type>CUSTOM</type>
</Variable>
```

The Variable subobject uses the following fields:

Field	Description	Type	Size	Required?
name	A unique name for the variable. This is used as a readable unique key for the variable.	String	32	Yes
node	The XPath to use to extract the value of the variable from an XMPP event that might contain it. See XPath Variables below for a description of the predefined variables that can be a part of the XPath.	String	500	Yes, if the type is CUSTOM
type	Indicates whether it is a system or custom variable. Possible values are SYSTEM and CUSTOM. SYSTEM variables are name references to the values returned by SystemVariable and do not require a node value. CUSTOM variables are self defining and require a node and a unique name that does not conflict with any system variable.	String	32	Yes

XPath Variables

Nodes can contain the following predefined variables as part of their XPath. When the node is evaluated, the current value as received in the most recent User event will be substituted in place of the variable. Variables are surrounded by ${} when specified in XPath as shown in the table below.

Note These variables are a subset of those defined by the SystemVariable resource

SYSTEM variables are name references to the values returned by SystemVariable and do not require a node value. CUSTOM variables are self-defining and require a node and a unique name that does not conflict with any system variable.

Variable Name	Value	Data Type
${userExtension}	The extension this user is currently using.	String
${userLoginId}	The login ID of the user.	String
${userLoginName}	The user's login name.	String
${userTeamName}	The name of the team the user belongs to.	String
${userTeamId}	The ID of the team the user belongs to.	String
${userFirstName}	The first name of the user.	String
${userLastName}	The last name of the user.	String

Workflow — Get Workflow

The Workflow - Get workflow API allows an administrator to retrieve a specific workflow.

URI:	http://<FQDN>/finesse/api/Workflow/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/Workflow/1234`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML

Successful Response:	<Workflow> ... Full Workflow Object ... </Workflow>
HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, for example, the object does not exist or the object is stale. 401--Authorization failure. The user is not yet authenticated in the web session. 401--Invalid authorization user specified. The authenticated user tried to use an identity that is not their own. 403--Forbidden. Configuration APIs cannot be run against the secondary Finesse server. 404--Not found. The workflow cannot be found. 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>Workflow 10009 not found.</ErrorData> <ErrorType>Not Found</ErrorType> <ErrorMessage>HTTP Status code:404 (Not Found) Api Error Type: Not Found Error Message: Workflow not found with an id of 10009 </ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	String	The ID in the URI maps to the primary key of the workflow entry.

name	String	The name of the workflow.
description	String	The description of the workflow.
TriggerSet	Collection	A set of events that cause the conditions to be evaluated.
ConditionSet	Collection	A set of conditions that determine if the workflow will be executed.
workflowActions	Collection	A list of existing workflow actions (WorkflowAction objects) that are executed if the trigger and its conditions are satisfied by the data in the event.

Workflow — Get List of Workflows

The Workflow - Get list of workflows API allows an administrator to get a list of workflows.

URI:	http://<FQDN>/finesse/api/Workflows
Example URI:	`http://finesse1.xyz.com/finesse/api/Workflows`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--

Successful Response:	`<Workflows>`
	`  <Workflow>`
	`  ... Full Workflow Object ...`
	`  </Workflow>`
	`  <Workflow>`
	`  ... Full Workflow Object ...`
	`  </Workflow>`
	`  <Workflow>`
	`  ... Full Workflow Object ...`
	`  </Workflow>`
	`</Workflows>`

HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or there is a violation of database constraints. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 403--Forbiddden. Configuration APIs cannot be run against the secondary Finesse server. 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>Database read/write error</ErrorData> <ErrorType>Bad Request</ErrorType> <ErrorMessage> HTTP Status code: 400 (Bad Request) Api Error Type: Bad Request Error Message: Database read/write error </ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	String	The ID in the URI maps to the primary key of the workflow entry.
name	String	The name of the workflow.
description	String	The description of the workflow.
TriggerSet	Collection	A set of events that cause the conditions to be evaluated.
ConditionSet	Collection	A set of conditions that determine if the workflow will be executed.
workflowAction	Collection	A list of existing workflow actions (WorkflowAction objects) that are executed if the trigger and its conditions are satisfied by the data in the event.

Workflow - Create

The Workflow - Create API allows an administrator to create a new workflow. Finesse supports a maximum of 100 workflows.

URI:	http://<FQDN>/finesse/api/Workflow/

Example URI:	`http://finesse1.xyz.com/finesse/api/Workflow/`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Workflow>`
	`  ... Full Workflow Object ...`
	`</Workflow>`

HTTP Response:	200--Success. The new workflow has been successfully created. The server response contains an empty response body and a location header denoting the absolute URL of the new workflow. 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 403--Forbidden. Configuration APIs cannot be run against the secondary Finesse server. 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorData>Duplicate Workflow name.</ErrorData>`
	`  <ErrorType>Database constraint violation</ErrorType>`
	`  <ErrorMessage>`
	`  HTTP Status code: 400 (Bad Request)`
	`  Api Error Type:  Database constraint violation`
	`  Error Message: A workflow with the same name already exists`
	`  </ErrorMessage>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Description
uri	String	The ID in the URI maps to the primary key of the workflow entry.
name	String	The name of the workflow.
description	String	The description of the workflow.
TriggerSet	Collection	A set of events that cause the conditions to be evaluated.

ConditionSet	Collection	A set of conditions that determine if the workflow will be executed.
workflowActions	Collection	A list of existing workflow actions (WorkflowAction objects) that are executed if the trigger and its conditions are satisfied by the data in the event.

Workflow - Update

The Workflow - Update API allows an administrator to update an existing workflow.

URI:	http://<FQDN>/finesse/api/Workflow/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/Workflow/1234`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<Workflow>`
	`  ... Full Workflow Object ...`
	`</Workflow>`

HTTP Response:	200--Success. The finesse server has successfully updated the specified workflow. 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 403--Forbidden. Configuration APIs cannot be run against the secondary Finesse server. 404--Not found. The workflow cannot be found (for example, it was deleted). 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>For update, at least one field must be set.</ErrorData> <ErrorType>Invalid Input</ErrorType> <ErrorMessage> HTTP Status code: 400 (Bad Request) Api Error Type: Invalid Input Error Message: Updating a Workflow requires specifying at least one value to be changed. </ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Description
uri	String	The ID in the URI maps to the primary key of the workflow entry.
name	String	The name of the workflow.
description	String	The description of the workflow.
TriggerSet	Collection	A set of events that cause the conditions to be evaluated.
ConditionSet	Collection	A set of conditions that determine if the workflow will be executed.
workflowActions	Collection	A list of existing workflow actions (WorkflowAction objects) that are executed if the trigger and its conditions are satisfied by the data in the event.

Note If the attributes (name, description, TriggerSet, ConditionSet, workflowActions) for the specified workflow do not change, the request option does not need to include those attributes. If an attribute is not specified, the old value will be retained. At least one attribute must be included in the request.

For example, if you only want to change the description of the workflow, the request could be like the following:

```
<Workflow>
   <description>New Descripion</description>
</Workflow>
```

Workflow - Delete

The Workflow - Delete API allows an administrator to delete an existing workflow. The administrator references an existing Workflow object by its ID.

URI:	http://<FQDN>/finesse/api/Workflow/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/Workflow/1234`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	DELETE
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--
HTTP Response:	200--Success. The Finesse server has successfully deleted the specified workflow. 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 403--Forbidden. Configuration APIs cannot be run against the secondary Finesse server. 404--Not found. The workflow cannot be found (for example, it was deleted). 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorData>Workflow 1009 not found.</ErrorData>`
	`  <ErrorType>Not Found</ErrorType>`
	`  <ErrorMessage>`
	`  HTTP Status code: 404 (Not Found)`
	`  Api Error Type:  Not Found`
	`  Error Message: Workflow not found with an id of 1009`
	`  </ErrorMessage>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

WorkflowAction APIs

The WorkflowAction object represents a workflow action that can be assigned to a workflow. There can be a maximum of 100 actions in the system.

TheWorkflowAction object is structured as follows:

```
<WorkflowAction>
    <uri>/finesse/api/WorkflowAction/{id}</uri>
    <name>DuckDuckGo</name>
    <type>BROWSER_POP</type>
```

```
    <handledBy>FINESSE_DESKTOP</handledBy>
    <params>
      <Param>
         ... Parameter SubObject...
      </Param>
      <Param>
         ... Parameter SubObject...
       </Param>
   </params>
   <actionVariables>
      <ActionVariable>
         ... ActionVariable SubObject...
      </ActionVariable>
      <ActionVariable>
         ... ActionVariable SubObject...
      </ActionVariable>
   </actionVariables>
</WorkflowAction>
```

The WorkflowAction object uses the following fields.

Field	Description	Type	Size	Required?
uri	In the uri, the id maps to the primary key of the WorkflowAction entry.	String		No
name	The name of the workflow action.	String	64	Yes (create/update)
type	The type of workflow action. Possible values are BROWSER_POP, HTTP_REQUEST	String		Yes (create/update)
handledBy	Indicates who will handle the action when it is triggered by a workflow. FINESSE_DESKTOP: The Finesse workflow engine executes the action. For example, in the case of a BROWSER_POP, the Finesse desktop will open a window with the resultant URL. OTHER: The action event will be published on the OpenAJAX hub, but will not be executed by the Finesse desktop. This allows a third party gadget to execute the action.	String	64	Yes (create/update)
params	List of Param subobjects (see below).	Collection		No
actionVariables	List of ActionVariable subobjects (see below). You can assign a maximum of five Action Variable subobjects to a workflow.	Collection		No

The Param subobject is structured as follows:

```
<Param>
   <name>path</name>
   <value>http://www.google.com/</value>
</Param>
```

The Param subobject uses the following fields:

Field	Description	Type	Size	Required?
name	The name of the parameter.	String	40	Yes
value	The value of the parameter.	String	500	Yes

Params are flexible and can contain any value. Validation is based on the type of the WorkflowAction in which they are contained. These limits are summarized by the WorkflowAction type below.

BROWSER_POP

The BROWSER_POP type is structured as follows:

```
<WorkflowAction>
    <uri>/finesse/api/WorkflowAction/{id}</uri>
    <name>DuckDuckGo</name>
    <type>BROWSER_POP</type>
    <handledBy>FINESSE_DESKTOP</handledBy>
    <params>
 <Param>
  <name>path<name>
                <value>http://www.example.com?q=${callVariable1}</value>
 </Param>
 <Param>
  <name>windowName</name>
                <value>theWindow</value>
  </Param>
    </params>
    <actionVariables>
 <ActionVariable>
  <name>callVariable1</name>
                <type>SYSTEM</type>
 </ActionVariable>
    </actionVariables>
</WorkflowAction>
```

Parameter	Description	Possible Values	Size	Required?
path	The path to use in the BROWSER_POP action	The URL path is validated only to make sure its length is at least 1 and no longer than the maximum length. It is up to the user to provide a valid URL. Variables can be embedded into the URL by using a dollar sign and curly braces. For example: `http://www.example.com?q=${callVariable1}` causes the workflow engine to substitute the value of callVariable1 into the path. If a literal curly brace or dollar sign is needed in the URL, it must be escaped with a backslash (for example, \{). A literal backslash must be escaped with another backslash (\\).	500	Yes

windowName	The window name to pop open	The window name is passed to the browser Window Open method by the work flow engine. The value can be any string other than _parent, _self, or _top. It can also be an empty string or missing entirely, in which case the workflow engine passes _blank to the Window Open method.	40	No

HTTP_REQUEST

The HTTP_REQUEST type is structured as follows:

```
<WorkflowAction>
    <name>Test with Content Type</name>
    <type>HTTP_REQUEST</type>
    <handledBy>FINESSE_DESKTOP</handledBy>
        <Param>
            <name>path</name>
            <value>http://www.example.com?q=${callVariable1}</value>
        </Param>
        <Param>
            <name>method</name>
            <value>PUT</value>
        </Param>
        <Param>
            <name>authenticationType</name>
            <value>BASIC</value>
        </Param>
        <Param>
            <name>location</name>
            <value>OTHER</value>
        </Param>
        <Param>
            <name>contentType</name>
            <value>application/xml</value>
        </Param>
        <Param>
            <name>body</name>
            <value>${callVariable1},${callVariable2}</value>
        </Param>
    </params>
    <actionVariables>
        <ActionVariable>
            <name>callVariable1</name>
            <type>SYSTEM</type>
        </ActionVariable>
        <ActionVariable>
            <name>callVariable2</name>
            <type>SYSTEM</type>
        </ActionVariable>
    </actionVariables>
</WorkflowAction>
```

Parameter	Description	Possible Values	Size	Required?

path	The path to use in the HTTP_REQUEST action	The URL path is validated only to make sure its length is at least 1 and no longer than the maximum length. It is up to the user to provide a valid URL. Variables can be embedded into the URL by using a dollar sign and curly braces. For example:
		`http://www.example.com?q=${callVariable1}`
		will cause the workflow engine to substitute the value of callVariable1 into the path. If a literal curly brace or dollar sign is needed in the URL, they must be escaped with a backslash (e.g. \{). A literal backslash must be escaped with another backslash (e.g. \\).
		When location is FINESSE, the protocol, host, and port should not be specified. These will be inferred automatically by Finesse when it executes the REST request. For example, to send a dialog request for dialog id 32458, the following URL should be entered:
		`/finesse/api/Dialog/32458`
			500	Yes

method	The method to use in the HTTP_REQUEST	PUT, POST		Yes
authenticationType	The authentication type to use in the HTTP_REQUEST	BASIC: A basic access authentication header is included in the REST request each time it is made. NONE: No authentication is used with the request, no authentication headers or other negotiation is done as part of the request.		No
location	Defines if the HTTP_REQUEST is to Finesse or to a third party application	FINESSE: The request is made to Finesse and passes the credentials of the currently logged-in user NONE: No credentials are included as part of the request.		No
contentType	The value of the content type header to send with the HTTP_REQUEST	The content type is only validated to ensure it does not exceed the maximum length. You must make sure you provide a valid content type. If the parameter is empty, no content type header is sent with the HTTP_REQUEST.	500	No

body	The body to send with the HTTP_REQUEST	A free form text string that is included in the body of the request. It may be JSON, XPATH or any other format. It is not validated. If xml is included in the value it must be well formed xml. Variables may be embedded into the body by using a dollar sign curly braces. For example:
		`<foo>${callVariable1}</foo>`
		causes the workflow engine to substitute the value of callVariable1 into the body. If a literal curly brace or dollar sign is needed in the body it must be escaped with a backslash:
		`\{`
		A literal backslash must be escaped with another backslash :
		`\\`
			2000	No

The ActionVariable subobject is structured as follows:

```
<ActionVariable>
    <name>user.foo.bar[1]</name>

<node>//dialogs/Dialog/mediaProperties/callvariables/CallVariable/name[.="user.foo.bar[1]"]/../value</node>

    <type>CUSTOM</type>
    <testValue>1234</testValue>
</ActionVariable>
```

The ActionVariable subobject uses the following fields:

Field	Description	Type	Size	Required?
name	The name of the variable.	String	32	Yes
node	The XPath to extract from the dialogs xml.	String	500	Yes (CUSTOM variables only)
type	Indicates if this is a CUSTOM or SYSTEM variable.	String	32	Yes
testValue	The value used to test the variable.	String	128	No

SYSTEM variables are name references to the values returned by SystemVariable and do not require a node value. CUSTOM variables are self-defining and require a node and a unique name that does not conflict with any system variable.

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.

WorkflowAction — Get Workflow Action

The WorkflowAction - Get workflow action API allows an administrator to retrieve a specific workflow action.

URI:	http://<FQDN>/finesse/api/WorkflowAction/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/WorkflowAction/1234`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML

Successful Response:	`<WorkflowAction>`
	`  ... Full WorkflowAction Object ...`
	`</WorkflowAction>`

HTTP Response:	200--Success 400-Bad request. The request body is invalid. 400-Finesse API error. There is an API error, for example, the object does not exist or the object is stale. 401-Authorization failure. The user is not yet authenticated in the web session. 401-Invalid authorization user specified. The authenticated user tried to use an identity that is not their own. 403--Forbidden. Configuration APIs cannot be run against the secondary Finesse server. 404-Not found. The workflow action cannot be found. 500-Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorData>Action 150 not found.</ErrorData>`
	`  <ErrorType>Not Found</ErrorType>`
	`  <ErrorMessage>`
	`  HTTP Status code: 404 (Not Found)`
	`  Api Error Type:  Not Found`
	`  Error Message: This is not a valid action`
	`  </ErrorMessage>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	String	The ID in the URI maps to the primary key of the workflowAction entry.
name	String	The name of the workflow action.
type	String	The type of workflow action.
params	Collection	List of Param subobjects.
actionVariables	Collection	List of ActionVariable subobjects.

WorkflowAction — Get List of Workflow Actions

The WorkflowAction - Get list of workflow actions API allows an administrator to get a list of workflow actions.

URI:	http://<FQDN>/finesse/api/WorkflowActions
Example URI:	`http://finesse1.xyz.com/finesse/api/WorkflowActions`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--

Successful Response:	<WorkflowActions> <WorkflowAction> <name>WorkflowAction 1</name> <type>HTTP</name> <uri>/finesse/api/WorkflowAction/{id}</uri> </WorkflowAction> <WorkflowAction> <name>WorkflowAction 2</name> <type>DELAY</name> <uri>/finesse/api/WorkflowAction/{id}</uri> </WorkflowAction> </WorkflowActions>
HTTP Response:	200--Success 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or there is a violation of database constraints. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 403--Forbidden. Configuration APIs cannot be run against the secondary Finesse server. 404--Not found. The resource is not found (for example, the workflow action has been deleted). 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>Database read/write error</ErrorData> <ErrorType>Bad Request</ErrorType> <ErrorMessage> HTTP Status code: 400 (Bad Request) Api Error Type: Bad Request Error Message: Database read/write error </ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

In a stand-alone Finesse deployment with Unified CCE, you cannot run configuration APIs against the secondary Finesse server. If you attempt to run this API against the secondary Finesse server, Finesse responds with a 403 “Forbidden” error.

Coresident Finesse with Unified CCX:

In a coresident Finesse deployment with Unified CCX, access to administration on the secondary node is read-only. You can run a GET request against the secondary node. However, other requests (PUT, POST, or DELETE) result in a 403 “Forbidden” error.

Response Parameters

Parameter	Type	Description
uri	String	The ID in the URI maps to the primary key of the workflowAction entry.
name	String	The name of the workflow action.
type	String	The type of workflow action.
params	Collection	List of Param subobjects.
actionVariables	Collection	List of ActionVariable subobjects.

WorkflowAction - Create

The WorkflowAction - Create API allows an administrator to create a new workflow action.

URI:	http://<FQDN>/finesse/api/WorkflowAction/
Example URI:	`http://finesse1.xyz.com/finesse/api/WorkflowAction/`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<WorkflowAction>`
	`  ... Full WorkflowAction Object ...`
	`</WorkflowAction>`

HTTP Response:	200--Success. The Finesse server has successfully created the new workflow action. The server response contains an empty response body and a location header for the absolute URL of the new workflow action. 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 403--Forbidden. Configuration APIs cannot be run against the secondary Finesse server. 404--Not found. The resource is not found (for example, the workflow action has been deleted). 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	<ApiErrors> <ApiError> <ErrorData>Action Type is invalid.</ErrorData> <ErrorType>Invalid Input</ErrorType> <ErrorMessage> HTTP Status code: 400 (Bad Request) Api Error Type: Invalid Input Error Message: type is invalid </ErrorMessage> </ApiError> </ApiErrors>
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Description
name	String	The name of the workflow action.
type	String	The type of workflow action.
params	Collection	List of Param subobjects.
actionVariables	Collection	List of ActionVariable subobjects.

WorkflowAction - Update

The WorkflowAction - Update API allows an administrator to update an existing workflow action.

URI:	http://<FQDN>/finesse/api/WorkflowAction/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/WorkflowAction/1234`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<WorkflowAction>`
	`... Full WorkflowAction Object ...`
	`</WorkflowAction>`

HTTP Response:	200--Success. The Finesse server has successfully updated the specified workflow action. 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 403--Forbidden. Configuration APIs cannot be run against the secondary Finesse server. 404--Not found. The workflow action cannot be found (for example, it was deleted). 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorData>Duplicate Action name.</ErrorData>`
	`<ErrorType>Database constraint violation</ErrorType>`
	`<ErrorMessage>`
	`HTTP Status code: 400 (Bad Request)`
	`Api Error Type: Database constraint violation`
	`Error Message: An action with the same name already exists`
	`</ErrorMessage>`
	`</ApiError>`
	`</ApiErrors>`

Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Request Parameters

Parameter	Type	Description
name	String	The name of the workflow action.
type	String	The type of workflow action.
params	Collection	List of Param subobjects.
actionVariables	Collection	List of ActionVariable subobjects.

WorkflowAction - Delete

The WorkflowAction - Delete API allows an administrator to delete an existing workflow action. The administrator references an existing workflow action by its ID.

URI:	http://<FQDN>/finesse/api/WorkflowAction/<id>
Example URI:	`http://finesse1.xyz.com/finesse/api/WorkflowAction/1234`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	DELETE
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--
HTTP Response:	200--Success. The Finesse server has successfully deleted the specified workflow action. 400--Bad request. The request body is invalid. 400--Finesse API error. There is an API error, such as the object is stale or the object does not exist. 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 403--Configuration APIs cannot be run against the secondary Finesse server. 404--Not found. The workflow action cannot be found (for example, it was deleted). 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.

Failure Response Example:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorData>Action 150 not found.</ErrorData>`
	`  <ErrorType>Not Found</ErrorType>`
	`  <ErrorMessage>`
	`  HTTP Status code: 404 (Not Found)`
	`  Api Error Type:  Not Found`
	`  Error Message: This is not a valid action`
	`  </ErrorMessage>`
	`  </ApiError>`
	`</ApiErrors>`

Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

SystemVariable APIs

The SystemVariable object represents a variable that can be extracted from a Finesse Event object and displayed on an agent's desktop or used in a workflow.

The SystemVariable object is structured as follows:

```
<SystemVariable>
   <name>callVariable1</name>
```

```
<node>//Dialog/mediaProperties/callvariables/CallVariable/name[.="callVariable1"]/../value</node>
</SystemVariable>
```

The SystemVariable object uses the following fields.

Field	Description	Type	Size	Required?
name	A unique name for the variable. This is used as a readable unique key for the variable.	String	32	Yes
node	The XPath to use to extract the value of the variable from an XMPP event, which might contain it. Must be a valid XPath expression.	String	500	Yes

SystemVariable - List

The SystemVariable - List API allows an administrator to get a list of system variables. The SystemVariables object represents a list of SystemVariable objects. It returns the XML shown below.

URI:	http://<FQDN>/finesse/api/SystemVariables
Example URI:	`http://finesse1.xyz.com/finesse/api/SystemVariables`
Security Constraints:	User must be logged in as an administrator
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	--

Successful Response:	

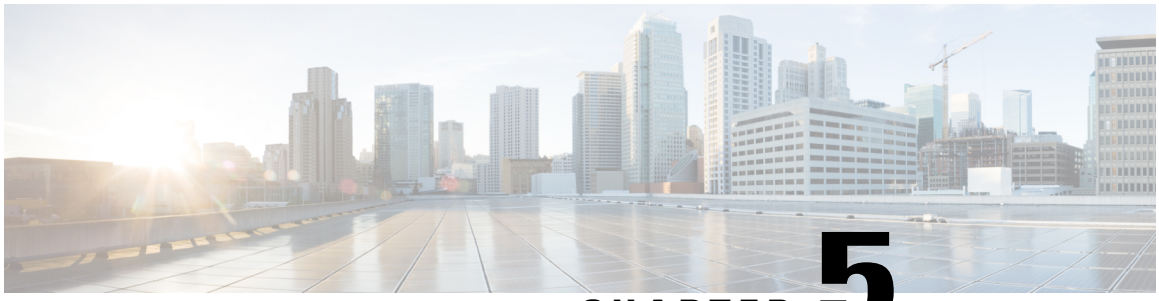
```
<SystemVariables>
  <SystemVariable>
    <name>callVariable1</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="callVariable1"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>callVariable2</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="callVariable2"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>callVariable3</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="callVariable3"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>callVariable4</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="callVariable4"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>callVariable5</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="callVariable5"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>callVariable6</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="callVariable6"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>callVariable7</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="callVariable7"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>callVariable8</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="callVariable8"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>callVariable9</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="callVariable9"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>callVariable10</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="callVariable10"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>BAAccountNumber</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="BAAccountNumber"]/../value</node>
  </SystemVariable>
    <name>BABuddyName</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="BABuddyName"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>BACampaign</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="BACampaign"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>BADialedListID</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="BADialedListID"]/../value</node>
  </SystemVariable>
  <SystemVariable>
```

```
    <name>BAResponse</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="BAResponse"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>BAStatus</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="BAStatus"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>BATimeZone</name>
    <node>//Dialog/mediaProperties/callvariables/CallVariable/
     name[.="BATimeZone"]/../value</node>
  </SystemVariable>
  <SystemVariable>
    <name>DNIS</name>
    <node>//Dialog/mediaProperties/DNIS</node>
  </SystemVariable>
  <SystemVariable>
    <name>fromAddress</name>
    <node>//Dialog/fromAddress</node>
  </SystemVariable>
  <SystemVariable>
    <name>Extension</name>
    <node>//User/extension</node>
  </SystemVariable>
  <SystemVariable>
    <name>loginId</name>
    <node>//User/loginId</node>
  </SystemVariable>
  <SystemVariable>
    <name>loginName</name>
    <node>//User/loginName</node>
  </SystemVariable>
  <SystemVariable>
    <name>teamName</name>
    <node>//User/teamName</node>
  </SystemVariable>
  <SystemVariable>
    <name>teamId</name>
    <node>//User/teamId</node>
  </SystemVariable>
  <SystemVariable>
    <name>firstName</name>
    <node>//User/firstName</node>
  </SystemVariable>
  <SystemVariable>
    <name>lastName</name>
    <node>//User/lastName</node>
  </SystemVariable>
</SystemVariables>
```

HTTP Response:	200--Success 401--Authorization failure. The user is unauthorized, for example is not authenticated in the web session. 500--Internal server error. Any runtime exception is caught and responded to with this error. For example, connection might have been broken with the CTI server or any other component.
Failure Response Example:	No API errors returned. Responses are 401/403/404 Errors.
Error Codes:	For possible error codes and their descriptions, see the Cisco Finesse Cisco Finesse API Errors.

Response Parameters

Parameter	Type	Description
name	String	The name of the variable.
node	String	The XPath to use to extract the value of the variable from an XMPP event, which might contain it.



CHAPTER 5

API Parameter Reference

Parameter Types and Data Types

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

Parameter Types

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
```

Data Types

The following table lists the data types used in API parameters and event message fields:

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 64-bit wide integer.
String	A variable-length string variable. If a maximum length exists, it is listed with the parameter description.

API Header Parameters

Name	Parameter Type / Data Type	Description	Used as a Request Parameter by
password	String	The password used in the request header to make any Finesse API request. Finesse supports a "Basic" authorization scheme only and authorization is required for each Finesse API request.	User — Sign In to Finesse

username	String	The username used in the request header to make any Finesse API request. Finesse supports a "Basic" authorization scheme only and authorization is required for each Finesse API request.	User — Sign In to Finesse

State (Dialog) Parameter Values

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

Dialog State	Description
INITIATING	Indicates that the phone is off the hook at a device
INITIATED	Indicates that the phone is dialing at the device
ALERTING	Indicates that the call is ringing at a device
ACTIVE	Indicates that the dialog has at least one active participant
FAILED	Indicates that the dialog has failed
DROPPED	Indicates that the dialog has no active participants
ACCEPTED	Indicates the user has accepted the OUTBOUND_PREVIEW dialog

Actions Parameter Values

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
RETRIEVE	Retrieve	Enables the agent to retrieve a call that was on hold
DROP	End	Enables the agent to drop the participant of a call

Participant Allowable Action	Enabled Button on Desktop	Description
UPDATE_CALL_DATA	-	Enables the agent to set call data for the call **Note** Finesse does not allow an agent to set call data from the desktop. A user can set call data through the API only.
SEND_DTMF	-	Enables the agent to send DTMF digits for the call
CONSULT_CALL	Consult	Enables the agent to make a consult call for transfer or conference
CONFERENCE	Conference	Enables the agent to start a conference between the selected held call and the existing active call on the desktop
TRANSFER	Transfer	Enables the agent to complete a transfer between the selected held call and the existing active call on the desktop
TRANSFER_SST	Direct Transfer	Enables the agent to perform a single-step transfer between the active call and the targetMediaAddress specified
SILENT_MONITOR	Start Monitoring	Enables the supervisor to silent monitor an agent who is in TALKING state on an active call
BARGE_CALL	Barge In	Enables the supervisor to barge in on an agent call that the supervisor is silently monitoring

Note 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.

Outbound Option Preview Actions

The following table describes the actions available to an agent who is reserved in an Outbound Option Preview campaign, the value to which Finesse sets the BAResponse variable, and the effect it has on the customer number in the campaign.

Note Performing the actions listed in this table causes Finesse to set the BAResponse variable to a corresponding value. Each value triggers a specific action in Unified CCE.

For more information about the BAResponse variable, see the section "Outbound Option Extended Call Variables" in the *Outbound Option Guide for Cisco Unified Contact Center Enterprise*.

Action	BAResponse Value	Description
ACCEPT	Accept	Performing the ACCEPT action while reserved in an Outbound Option Preview campaign instructs Unified CCE to establish a call with the customer.
CLOSE	Reject-Close	Performing the CLOSE action while reserved in an Outbound Option Preview campaign rejects the current preview call and prevents the number from being called again in the campaign.
REJECT	Reject	Performing the REJECT action while reserved in an Outbound Option Preview campaign instructs Unified CCE to retry the previewed number at a later time.

State (Participant) Parameter Values

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)

Participant State	Allowable Actions for the Participant State	Call State on Finesse Desktop	Description
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
ACCEPTED	-	-	Indicates that the participant has accepted the dialog. This state is applicable to OUTBOUND_PREVIEW dialogs.

Note In Finesse Release 9.0(1) and earlier, when a dialog participant wraps up, a dialog event is sent only to the participant who transitions to wrap-up state. In Finesse Release 9.1(1) and later, a dialog event is sent to each participant in the dialog.

CTI Event Mappings for Dialog and Participant States

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 toward the agent receiving an incoming call.

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

Table 1: Incoming Call

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_CONNECTION_CLEARED	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 toward the caller making an outgoing call.

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

Table 2: Outgoing Call

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

Note For the Finesse API, a silent monitor call request only specifies the agent's extension for the supervisor to silent monitor. Unified CCE/Unified CCX decides which of the agent's active calls to monitor. In most cases, an agent only has one active call to be monitored. This table describes the scenario where a call already exists between the caller and Agent A. The focus is on the silent monitor call only. In this scenario, the original agent call is not affected. The silent monitor call is created and the agent becomes a participant with no allowable action. The agent has two active calls: the original call and the silent monitor call. Finesse considers the silent monitor call to be a "passive" active call of the agent.

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.

Table 3: Holding a Call

Scenario	CTI Event	Event Method	Dialog State	Participant State (Agent)	Participant State (Caller)
Call arrives and is 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.

Table 4: Call Transfer

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 Agent A: HELD (original call) Agent B: Not yet a participant

Scenario	CTI Event (Original Call)	CTI Event (Consult Call)	Event Method	Dialog State	Participant State
Agent A takes phone off-hook (BEGIN_CALL_EVENT assumed)	-	CALL_SERVICE_INITIATED_EVENT	PUT (consult call only)	Original call: ACTIVE Consult call: INITIATING	Caller: ACTIVE Agent A: INITIATING (consult call) Agent B: Not yet a participant
Agent A dials number	-	CALL_ORIGINATED_EVENT	PUT (consult call only)	Original call: ACTIVE Consult call: INITIATED	Caller: ACTIVE Agent A: INITIATED (consult call) Agent B: Not yet a participant
Agent B receives the call	-	CALL_DELIVERED	PUT (consult call, on Agent A POST (consult call on Agent B	Original call: ACTIVE Consult call: ALERTING	Caller: ACTIVE Agent A: INITIATED (consult call) Agent B: ALERTING
Agent B answers the call	-	CALL_ESTABLISHED	PUT (consult call only)	Original call: ACTIVE Consult call: ACTIVE	Caller: ACTIVE Agent A: ACTIVE (consult call) Agent B: ACTIVE

Scenario	CTI Event (Original Call)	CTI Event (Consult Call)	Event Method	Dialog State	Participant State
Agent A completes the transfer of the caller to Agent B	CALL_TRANSFERRED_EVENT	-	DELETE (original call on Agent A) DELETE (consult call on Agent A) DELETE (consult call on Agent B) POST (original call on Agent B)	Original call: DROPPED (Agent A), ACTIVE (Agent B) Consult call: DROPPED (both Agent A and Agent B)	Caller: ACTIVE Agent A: DROPPED (original and consult call) 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.

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

Note For the Finesse API, a silent monitor call request only specifies the agent's extension for the supervisor to silent monitor. Unified CCE/Unified CCX decides which of the agent's active calls to monitor. In most cases, an agent only has one active call to be monitored. This table describes the scenario where a call already exists between the caller and Agent A. The focus is on the silent monitor call only. In this scenario, the original agent call is not affected. The silent monitor call is created and the agent becomes a participant with no allowable action. The agent has two active calls: the original call and the silent monitor call. Finesse considers the silent monitor call to be a "passive" active call of the agent.

Table 5: Silent Monitor Call

Scenario	CTI Event (Silent Monitor Call)	Event Method	Dialog State (Original Call)	Dialog State (Silent Monitor Call)	Participant State (Caller)	Participant State (Agent A)	Participant State (Supervisor)
Agent call arrives and is answered	-	-	-	-	-	-	-

Scenario	CTI Event (Silent Monitor Call)	Event Method	Dialog State (Original Call)	Dialog State (Silent Monitor Call)	Participant State (Caller)	Participant State (Agent A)	Participant State (Supervisor)
Supervisor starts the silent monitor call	BEGIN_CALL	POST (SILENT_MONITOR)	ACTIVE	INITIATING	ACTIVE (original call)	ACTIVE (original call)	INITIATING (silent monitor call)
-	CALL_SERVICE_INITIATED_EVENT CALL_DATA_UPDATE_EVENT	-	ACTIVE	INITIATING	ACTIVE (original call)	ACTIVE (original call)	INITIATING (silent monitor call)
-	CALL_ORIGINATED_EVENT CALL_DATA_UPDATE_EVENT	-	ACTIVE	INITIATED	ACTIVE (original call)	ACTIVE (original call)	INITIATED (silent monitor call)
-	CALL_DELIVERED_EVENT CALL_DELIVERED_EVENT	-	ACTIVE	ALERTING	ACTIVE (original call)	ACTIVE (original call)	INITIATED (silent monitor call)
-	CALL_ESTABLISHED_EVENT	-	ACTIVE	ACTIVE	ACTIVE (original call)	ACTIVE (original call) ACTIVE (passive - silent monitor call)	ACTIVE (silent monitor call)

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

Note This table describes a scenario where a call already exists between the caller and Agent A and the supervisor is silently monitoring that call. The focus is on the barge only. In this scenario, the agent call is temporarily put on hold, the silent monitor call is dropped, and a consult call is created. The agent call becomes a conference call with the caller, agent, and supervisor as participants.

Table 6: Barge Call

Scenario	CTI Event	Event Method	Dialog State	Participant State (Caller)	Participant State (Agent A)	Participant State (Supervisor)
Agent call arrives and is answered	-	-	-	-	-	-
Supervisor silent monitors the call	-	-	ACTIVE (original call) ACTIVE (silent monitor call)	ACTIVE	ACTIVE (original call) ACTIVE (passive, silent monitor call)	ACTIVE (silent monitor call)
Supervisor starts barge call	-	POST (BARGE)	ACTIVE (original call) ACTIVE (silent monitor call)	ACTIVE	ACTIVE (original call) ACTIVE (passive, silent monitor call)	ACTIVE (silent monitor call)
Finesse drops silent monitor call through Unified CCE	CALL_CONNECTION_CLEARED (silent monitor call) CALL_CLEARED (silent monitor call) END_CALL (silent monitor call)	-	ACTIVE (original call) DROPPED (silent monitor call)	ACTIVE (original call)	ACTIVE (original call) ACTIVE (silent monitor call)	DROPPED (silent monitor call)
Unified CCE puts original call on hold	CALL_HELD (original call)	-	ACTIVE (original call)	ACTIVE (original call)	HELD (original call)	Not a participant yet
Unified CCE generates consult call	BEGIN_CALL (consult call) CALL_SERVICE_INITIATED_EVENT (consult call)	-	ACTIVE (original call) INITIATING (consult call)	ACTIVE	HELD (original call) INITIATING (consult call)	Not a participant yet
Unified CCE dials supervisor's extension	CALL_ORIGINATED_EVENT (consult call)	-	ACTIVE (original call) INITIATED (consult call)	ACTIVE	HELD (original call) INITIATED (consult call)	Not a participant yet

Scenario	CTI Event	Event Method	Dialog State	Participant State (Caller)	Participant State (Agent A)	Participant State (Supervisor)
Agent receives the consult call	CALL_DELIVERED (consult call)	-	ACTIVE (original call) INITIATED (consult call)	ACTIVE	HELD (original call) INITIATED (consult call)	Not a participant yet
Supervisor receives the consult call	CALL_DELIVERED (consult call)	-	ACTIVE (original call) ALERTING (consult call)	ACTIVE	HELD (original call) INITIATED (consult call)	ALERTING
Unified CCE answers the consult call on behalf of the supervisor and changes the original agent call to a conference call	CALL_ CONFERENCED	-	ACTIVE (original call) ALERTING (consult call)	ACTIVE	HELD (original call) INITIATED (consult call)	ALERTING
Unified CCE ends the consult call	END_CALL (consult call)	-	ACTIVE (original call) DROPPED (consult call)	ACTIVE	HELD (original call) DROPPED (consult call)	-
Unified CCE changes the original call type to conference	CALL_DATA_ UPDATE (original call)	-	ACTIVE (original call)	ACTIVE	ACTIVE (original call, callType=15 =Conference)	-
Unified CCE answers call on behalf of supervisor	CALL_ESTABLISHED (original call)	-	ACTIVE (original call)	ACTIVE	ACTIVE (original call)	ACTIVE

If the caller is also an agent, the caller receives a dialog update (PUT) with an updated participant list on the conference.

RESERVED_OUTBOUND User State

An agent transitions to the RESERVED_OUTBOUND state when that agent is reserved for a progressive or predictive outbound call. Agents can exit this state by changing their status to READY or NOT_READY. If not ready reason codes are configured, an agent must specify a reason code to transition to NOT_READY.

Agents cannot set their state to RESERVED_OUTBOUND.

RESERVED_OUTBOUND_PREVIEW User State

In a Unified CCE system, an agent transitions to the RESERVED_OUTBOUND_PREVIEW state when that agent is reserved for an Outbound Option preview call. To exit this state, an agent can execute the CLOSE or REJECT action against the current dialog. Changing state to READY or NOT_READY does not generate any state change events but does affect the agent's state after the agent is done receiving calls. For example, if an agent changes to NOT_READY state while reserved for an Outbound Option preview call, the agent transitions to NOT_READY after executing the CLOSE action on the Outbound Option preview dialog.

In a Unified CCX system, an agent transitions to the RESERVED_OUTBOUND_PREVIEW state when that agent is reserved for a preview outbound call. Agents cannot change their state directly when in this state. The state can only be changed by issuing a Dialog - Accept, close, or reject or if Unified CCX times out the reservation call.

In both a Unified CCE and Unified CCX system, agents cannot set their state to RESERVED_OUTBOUND_PREVIEW.

WORK and WORK_READY User States

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.

Note WORK_READY state applies only to Unified CCE deployments.

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.

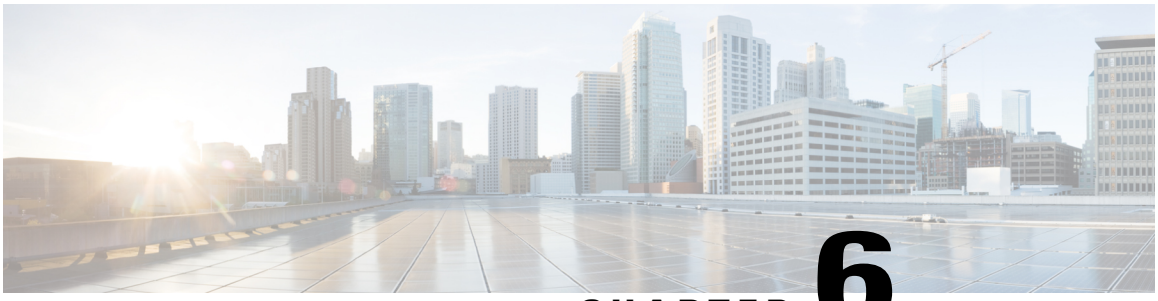
A user transitions to WORK_READY state for one of the following reasons:

- 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.



CHAPTER 6

Cisco Finesse Errors

HTTP Errors

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 - DB_RUNTIME_EXCEPTION (database error, but the database is thought to be operational)
- 500 - RUNTIME_EXCEPTION (a non-database error)
- 500 - 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.

Cisco Finesse API Errors

Error codes for Cisco Finesse are categorized as follows:

- 4xx - Client-related errors
- 5xx - Server-related errors

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

```
<ApiErrors>
    <ApiError>
        <ErrorType>Authorization Failure</ErrorType>
        <ErrorMessage>UNAUTHORIZED</ErrorMessage>
        <ErrorData>jsmith</ErrorData>
    </ApiError>
</ApiErrors>
```

In addition to Cisco Finesse API errors, a response might return a CTI error or an HTTP error.

Note This document contains information about error type and error message. You can find information about error data values for most User and Dialog errors in the following documents:

For Finesse deployments with Unified CCE, see the *CTI Server Message Reference Guide for Unified Contact Center Enterprise*, which you can find at https://developer.cisco.com/site/collaboration/contact-center/contact-center-ent/cti-protocol/documentation/.

For Finesse deployments with Unified CCX, see the *Cisco Unified Contact Center Express CTI Protocol Developer Guide*, which you can find at https://developer.cisco.com/site/collaboration/contact-center/express-cti/documentation/.

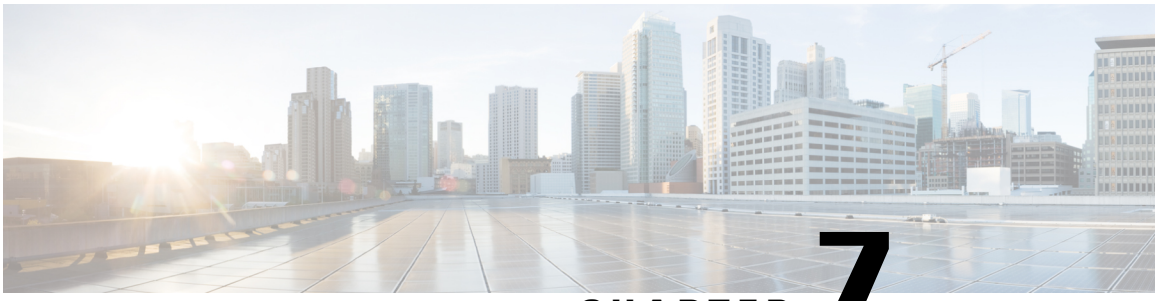
Table 7: HTTP Errors

Status	Error Type	Description
400	20700 (conference resource limit violation)	The barge call will cause the total number of parties on the conference call to exceed the allowed resource limit for the conference bridge.
400	20999 (Barge via a non-conference-controller)	The agent specified by the toAddress is not the controller of the conference call or the agent already has an outstanding consult call.
400	Generic Error	An unaccounted for error occurred. The root cause could not be determined.
400	Invalid Destination	The toAddress and fromAddress are the same. This error occurs if users attempt to call their own extensions. For the Dialog - Drop a participant from a conference call API, this error occurs if the targetMediaAddress is not one of the parties on the call or is not an agent's extension.
400	Invalid Device	The extension is invalid.

Status	Error Type	Description
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. The submitted XML is not valid (LayoutConfig APIs). For the User - Sign in as mobile agent API, the mode specified is invalid. For the Dialog - Update call variable API, the call variable name or action is invalid or not recognized, or there are duplicate call variable names. This error is also returned if a user attempts to set any of the following outbound variables: • BACampaign • BAAccountNumber • BAResponse • BAStatus • BADialedListID • BATimeZone • BABuddyName For the ClientLog - Post to Finesse API, the size of logData is greater than 1048576 characters.
400	Invalid State	The requested state change is not allowed (for example, a user already in LOGOUT state requests a state change to LOGOUT or a supervisor tries to change an agent's state to something other than LOGOUT or READY). A supervisor who is already in an active call (in TALKING or HOLD state) makes a silent monitoring request. For the Dialog - Drop a participant from a conference call API, the dialog is not a conference call.
400	Maximum Exceeded	The maximum number of items has been exceeded.
400	Operation Failure	The post client log operation failed.

Status	Error Type	Description
400	Parameter Missing	The extension, state value, or requestedAction is not provided. If signing in a mobile agent, the mode or dialNumber is not provided. If creating a dialog, the fromAddress or toAddress is not provided. If creating a layout, the XML file is not provided. If posting client logs to the server, the logData field is not present.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in Web Session). 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 User Specified	The authenticated user tried to make a request for another user. The authenticated user tried to use a fromAddress or targetMediaAddress that does not belong to that user. The primary and backup AWDB servers are down and Finesse cannot authenticate the user.
401	Invalid State	The participant whose extension is the targetMediaAddress is in HELD state in the dialog.
401	Invalid Supervisor	The supervisor who tried to change the agent's state does not supervise that agent's team.
403	Forbidden	The user attempted to run a configuration API against the secondary Finesse server. Finesse configuration APIs cannot be run against the secondary Finesse server.
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.
405	Method Not Allowed	The HTTP method used is not allowed for this API. The ClientLog API does not support GET or PUT.

Status	Error Type	Description
500	Internal Server Error	A runtime exception is caught (for example, a broken connection with the CTI server or another component).
501	Not Implemented	The API is not implemented on the platform (for example, attempting to use the Dialog - Start recording API in a Unified CCE deployment).
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).



CHAPTER 7

Cisco Finesse Notifications

About Cisco Finesse Notifications

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.

Notification Frequency

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

Subscription Management

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.

Each agent is automatically subscribed to the following notification feeds, where {id} represents the agent ID for that agent:

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

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.

Resources

User Notifications

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>`
	`<event>{put|delete}</event>`
	`<source>/finesse/api/User/{id}</source>`
	`<data>`
	`<user>`
	`<!-- full User object -->`
	`</user>`
	`</data>`
	`</Update>`

Notification Triggers:	• Addition of a user • Deletion of a user • State change • First or last name change • Role change

Notification Parameters

- event
- source
- data

Sample Notification Payload

```
<Update>
   <event>put</event>
   <source>/finesse/api/User/csmith</source>
   <data>
      <User>
        <dialogs>/finesse/api/User/1001001/Dialogs</dialogs>
        <extension></extension>
        <firstName>AGENT</firstName>
        <lastName>1001001</lastName>
        <loginId>1001001</loginId>
        <loginName>agent1</loginName>
        <pendingState></pendingState>
        <reasonCodeId>2</reasonCodeId>
        <roles>
          <role>Agent</role>
        </roles>
        <state>LOGOUT</state>
        <stateChangeTime></stateChangeTime>
        <teamId>5000</teamId>
        <teamName>FunctionalAgents</teamName>
        <uri>/finesse/api/User/1001001</uri>
      </User>
   </data>
</Update>
```

Dialog Notifications

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.

For the purpose of notifications, the fromAddress and toAddress parameters of the Dialog object are defined as follows:

- fromAddress: The extension of the caller who initiated the original call. If an unmonitored caller placed the call, the fromAddress is the unmonitored caller's extension. If an agent placed the call, the fromAddress is the agent's extension. For an Outbound Option Dialer call, the fromAddress is the extension of the agent on the outbound call. For a reservation call in Preview Outbound mode, the fromAddress is the dialer port. .
- toAddress: The dialed number of the original call. If the caller calls a route point, the toAddress is the route point. If the caller called an agent directly, the toAddress is the agent's extension. For an Outbound Option Dialer call, the toAddress is the customer phone number called by the dialer. For a reservation call in Outbound Option Preview mode, the toAddress is the extension of the agent who received the call.

Note When a call is transferred, the fromAddress and toAddress in subsequent dialog notifications are those of the surviving call. For example, if an agent who is on a call places a consult call and then transfers the original call, the fromAddress and toAddress in the subsequent dialog notifications are those of the original call because the original call is the surviving call. However, if the agent puts the consult call on hold, retrieves the original call, and then transfers the consult call, the fromAddress and toAddress in subsequent dialog notifications are those of the consult call. In this case, the consult call is the surviving call.

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) /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 • Modification of participant state (for example, when a participant answers or hangs up a call) • A new participant to the call • 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</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</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>
```

Dialog CTI Error Notifications

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> <data> <apiErrors> <apiError> <errorData>[CTI Error Code]</errorData> <errorMessage>[CTI Error Constant]</errorMessage> <errorType>Call Operation Failure</errorType> </apiError> </apiErrors> </data> <event>PUT</event> <requestId></requestId> <source>/finesse/api/Dialog/[ID]</source> </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

Team Notifications

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> <event>{put}</event> <source>/finesse/api/User/{id}</source> <requestId>xxxxxxxxx</requestId> <data> <user> <uri>/finesse/api/User/{id}</uri> <loginId>{id}</loginId> <firstName>Jack</firstName> <lastName>Brown</lastName> <state>NOT_READY</state> <stateChangeTime>2012-03-01T17:58:21Z</stateChangeTime> <ReasonCode> <uri>finesse/api/ReasonCode/{id}</uri> <code>10</code> <label>Team Meeting</label> </user> </data> </Update>
Notification Triggers:	• Agent name is changed for an agent belonging to the team • 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/1234</uri>
               <loginId>100101</loginId>
               <firstName>Charles</firstName>
               <lastName>Norrad</lastName>
               <state>LOGOUT</state>
               <stateChangeTime>2012-03-01T17:58:21Z</stateChangeTime>
            </User>
            <User>
               <uri>/finesse/api/User/9876</uri>
               <loginId>100102</loginId>
               <firstName>Jack</firstName>
               <lastName>Brown</lastName>
               <state>NOT_READY</state>
               <stateChangeTime>2012-03-01T17:58:21Z</stateChangeTime>
            </User>
            ... other users ...
         </users>
      </team>
   </data>
</Update>
```

Queue Notifications

Finesse sends a queue notification every 10 seconds (if queue statistics change).

Note Notifications for this node are sent only for a stand-alone Finesse deployment with Unified CCE. Notifications for this node are not sent for a coresident Finesse deployment with Unified CCX.

Format:	XML
Node:	/finesse/api/Queue/{id}
Source:	/finesse/api/Queue/{id}
Data:	Queue object

Payload (PUT):	<Update> <event>{put}</event> <source>/finesse/api/Queue/{id}</source> <requestId>xxxxxxxxx</requestId> <data> <Queue> <uri>/finesse/api/Queue/{id}</uri> <name>Sales</name> <statistics> <callsInQueue>3</callsInQueue> <startTimeOfLongestCallInQueue>2012-02-15 T17:58:21Z</startTimeOfLongestCallInQueue> <agentsReady>1</agentsReady> <agentsNotReady>2</agentsNotReady> <agentsTalkingInbound>3</agentsTalkingInbound> <agentsTalkingOutbound>4</agentsTalkingOutbound> <agentsTalkingInternal>5</agentsTalkingInternal> <agentsWrapUpNotReady>6</agentsWrapUpNotReady> <agentsWrapUpReady>7</agentsWrapUpReady> </statistics> </Queue> </data> </Update>
Payload (DELETE):	<Update> <event>{delete}</event> <source>/finesse/api/Queue/{id}</source> <requestId></requestId> <data> <Queue> <uri>/finesse/api/Queue/{id}</uri> </Queue> </data> </Update>
Notification Triggers:	Finesse publishes a notification • every 10 seconds, if queue statistics change • when a queue name changes • when a queue is deleted

Notification Parameters

- event
- source
- data
- requestId

Sample PUT Notification Payload

```
<Update>
     <event>put</event>
     <source>/finesse/api/Queue/1004</source>
     <requestId>xxxxxxxxx</requestId>
```

```
     <data>
         <Queue>
            <uri>/finesse/api/Queue/1004</uri>
            <name>Sales</name>
            <statistics>
                 <callsInQueue>3</callsInQueue>
                 <startTimeOfLongestCallInQueue>2012-02-15
                  T17:58:21Z</startTimeOfLongestCallInQueue>
                 <agentsReady>1</agentsReady>
                 <agentsNotReady>2</agentsNotReady>
                 <agentsTalkingInbound>3</agentsTalkingInbound>
                 <agentsTalkingOutbound>4</agentsTalkingOutbound>
                 <agentsTalkingInternal>5</agentsTalkingInternal>
                 <agentsWrapUpNotReady>6</agentsWrapUpNotReady>
                 <agentsWrapUpReady>7</agentsWrapUpReady>
            </statistics>
         </Queue>
      </data>
</Update>
```

Sample DELETE Notification Payload

```
<Update>
     <event>delete</event>
     <source>/finesse/api/Queue/1004</source>
     <requestId></requestId>
     <data>
         <Queue>
            <uri>/finesse/api/Queue/1004</uri>
         </Queue>
      </data>
</Update>
```

User/Queues Notifications

Finesse sends a User/Queues notification when a queue is added or removed from the current user's list of queues or if a queue assigned to that user is removed from the system.

Note Notifications for this node are sent only for a stand-alone Finesse deployment with Unified CCE. Notifications for this node are not sent for a coresident Finesse deployment with Unified CCX.

Format:	XML
Node:	/finesse/api/User/{id}/Queues
Source:	/finesse/api/User/{id}/Queues
Data:	User/Queues object

Payload (POST):	<Update>
	  <event>{post}</event>
	  <source>/finesse/api/User/{id}/Queues</source>
	  <requestId></requestId>
	  <data>
	  <Queues>
	  <Queue>
	  <uri>/finesse/api/Queue/{id}</uri>
	  <name>Sales</name>
	  <statistics>
	  <callsInQueue>3</callsInQueue>
	  <startTimeOfLongestCallInQueue>2012-02-15
	  T17:58:21Z</startTimeOfLongestCallInQueue>
	  <agentsReady>1</agentsReady>
	  <agentsNotReady>2</agentsNotReady>
	  <agentsTalkingInbound>3</agentsTalkingInbound>
	  <agentsTalkingOutbound>4</agentsTalkingOutbound>
	  <agentsTalkingInternal>5</agentsTalkingInternal>
	  <agentsWrapUpNotReady>6</agentsWrapUpNotReady>
	  <agentsWrapUpReady>7</agentsWrapUpReady>
	  </statistics>
	  </Queue>
	  ... more queues ...
	  </Queues>
	  </data>
	</Update>

Payload (DELETE):	<Update> <event>{delete}</event> <source>/finesse/api/User/{id}/Queues</source> <requestId></requestId> <data> <Queues> <Queue> <uri>/finesse/api/Queue/{id}</uri> </Queue> <Queue> <uri>/finesse/api/Queue/{id}</uri> </Queue> <Queue> <uri>/finesse/api/Queue/{id}</uri> </Queue> ... more queues ... </Queues> </data> </Update>
Notification Triggers:	• A queue is added or removed from the user's list of queues. • A queue assigned to the user is removed from the system.

Notification Parameters

- event
- source
- data
- requestId

Sample POST Notification Payload

```
Update>
    <event>post</event>
    <source>/finesse/api/User/1001001/Queues</source>
    <requestId></requestId>
    <data>
       <Queues>
          <Queue>
             <uri>/finesse/api/Queue/1215</uri>
             <name>Sales</name>
             <statistics>
                 <callsInQueue>3</callsInQueue>
                 <startTimeOfLongestCallInQueue>2012-02-15
                  T17:58:21Z</startTimeOfLongestCallInQueue>
                 <agentsReady>1</agentsReady>
                 <agentsNotReady>2</agentsNotReady>
                 <agentsTalkingInbound>3</agentsTalkingInbound>
                 <agentsTalkingOutbound>4</agentsTalkingOutbound>
                 <agentsTalkingInternal>5</agentsTalkingInternal>
                 <agentsWrapUpNotReady>6</agentsWrapUpNotReady>
                 <agentsWrapUpReady>7</agentsWrapUpReady>
             </statistics>
           </Queue>
          ... more queues ...
      </Queues>
    </data>
</Update>
```

Sample DELETE Notification Payload

```
<Update>
    <event>delete</event>
    <source>/finesse/api/User/1001001/Queues</source>
    <requestId></requestId>
    <data>
       <Queues>
          <Queue>
             <uri>/finesse/api/Queue/1326</uri>
          </Queue>
          <Queue>
             <uri>/finesse/api/Queue/1364</uri>
          </Queue>
          <Queue>
             <uri>/finesse/api/Queue/1389</uri>
          </Queue>
          ... more queues ...
      </Queues>
    </data>
</Update>
```

SystemInfo Notifications

Finesse sends a SystemInfo notification when the in-service status of the system changes.

Format:	XML
Node:	-
Source:	-

Data:	SystemInfo
Payload:	<Update> <data> <systemInfo> <!-- full SystemInfo object --> </systemInfo> </data> <event>PUT</event> <source>/finesse/api/SystemInfo</source> </Update>
Notification triggers:	The system transitions between IN_SERVICE and OUT_OF_SERVICE

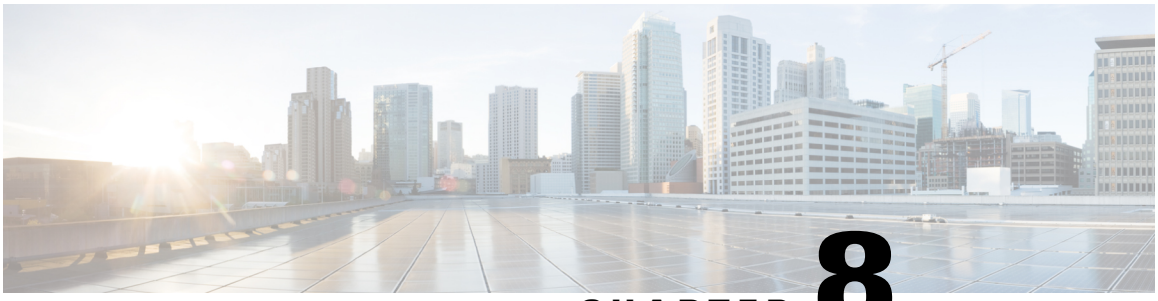
Sample Notification Payload

```
<Update>
    <data>
        <systemInfo>
            <currentTimeStamp>2013-11-27T18:57:00Z</currentTimeStamp>
            <deploymentType>UCCE</deploymentType>
            <primaryNode>
                <host>172.16.204.25</host>
            </primaryNode>
            <secondaryNode>
                <host>172.16.204.26</host>
            </secondaryNode>
            <status>IN_SERVICE</status>
            <uri>/finesse/api/SystemInfo</uri>
            <xmppDomain>xmppserver.cisco.com</xmppDomain>
            <xmppPubSubDomain>pubsub.xmppserver.cisco.com</xmppPubSubDomain>
        </systemInfo>
    </data>
    <event>PUT</event>
    <requestId/>
    <source>/finesse/api/SystemInfo</source>
</Update>
```

Notification Parameter Reference

Name	Data Type	Description	Possible Values	Used by These Notifications
Data	Object	Provides the new representation of the modified User, Team, Dialog, Queue, or User/Queues object. This information is not provided when a user is deleted. On an error notification, provides the list of ApiError objects representing the failure conditions detected by the server.	The entire User, Team, Dialog, or Queue object in its most current and updated form. The Team object includes all of its agents. A list of queues that were added to or removed from the User's list (User/Queues notification).	User Notifications Dialog Notifications Dialog CTI Error Notifications Team Notifications Queue Notifications User/Queues Notifications

Event	String	The type of modification that occurred to the User, Team, Dialog, Queue, or User/Queues object.	**PUT:** A property of the User, Team, Dialog, or Queue object has been modified. **DELETE:** The User, Team, Dialog, or Queue object has been deleted. Specifies the queues removed from the current user's list of queues (User/Queues notification). **POST:** A User, Team, Dialog, or Queue object was added. Specifies the queues that were added to the current user's list of queues (User/Queues notification).	User Notifications Dialog Notifications Dialog CTI Error Notifications Team Notifications Queue Notifications User/Queues Notifications
Source	String	The User, Dialog, Team, Queue, or User/Queues resource location that was modified.	/finesse/api/User/{id} /finesse/api/Dialog/{id} /finesse/api/Team/{id} /finesse/api/User/{id}/Dialogs /finesse/api/Queue/{id} /finesse/api/User/{id}/Queues	User Notifications Dialog Notifications Dialog CTI Error Notifications Team Notifications Queue Notifications User/Queues Notifications
RequestId	String	The requestId that was returned when the triggering REST API request was made. If the event was unsolicited, this tag is empty. This tag is empty for a User/Queues notification.	An opaque, unique string, used to correlate the originating request with the resulting event.	Dialog Notifications Dialog CTI Error Notifications Team Notifications Queue Notifications User/Queues Notifications



CHAPTER 8

Finesse High Availability

Availability of a Finesse server is determined by the following information (and in this order):

1. The status of the server as provided by the SystemInfo object:

   The status of the server indicates whether the server is in service and available to accept requests.

2. The status and availability of a BOSH connection to the Cisco Finesse Notification Service:

Note In a Unified CCX deployment, this service is called the Unified CCX Notification Service.

   An active BOSH connection to the Cisco Notification Service is required to receive notifications. Loss of this connection may mean that the server itself is unavailable or that the client cannot reach the server.

3. The presence of the 'finesse' BOSH user:

   Presence indicates whether Finesse has an active connection to the Cisco Finesse Notification Service (Unified CCE) or the Cisco Unified CCX Notification Service (Unified CCX) . An UNAVAILABLE presence for the 'finesse' BOSH user may mean that the connection is lost or that the Finesse web app crashed.

A Finesse server must meet the following criteria to be fully available for client use:

1. The status of the server must be IN_SERVICE.
2. A successful BOSH connection is made.
3. The presence of the 'finesse' BOSH user is AVAILABLE.

We highly recommend that the preceding conditions are checked in the order listed as failure of the criteria at the top of the list means the rest of the criteria will also fail or will not be relevant. For example the presence of the 'finesse' BOSH user cannot be checked without a BOSH connection. A BOSH connection is not useful if the server is OUT_OF_SERVICE.

- Failure Scenarios, page 254
- Desktop Presence and Forced Logout, page 255

Failure Scenarios

The following table lists possible failure scenarios and describes how a client can determine when a failure occurs.

Scenario	Notification mechanism
Cisco Finesse Notification Service goes down. **Note** In a Unified CCX deployment, this service is called the Cisco Unified CCX Notification Service.	Client loses BOSH connection to the Cisco Finesse Notification Service. **Note** This condition can occur while the Cisco Finesse Notification Services is running if the client loses network connectivity to the server (for example, a client experiences a complete loss of network connectivity).
Cisco Tomcat goes down. **Note** In a Unified CCX deployment, this is called Cisco Finesse Tomcat.	The 'finesse' user presence becomes UNAVAILABLE (if BOSH is still connected to the Cisco Finesse Notification Service).
Finesse web app goes down.	The 'finesse' user presence becomes UNAVAILABLE (if BOSH is still connected to the Cisco Finesse Notification Service).
Finesse loses connection to the CTI server.	Finesse sends a SystemInfo notification of status OUT_OF_SERVICE (if BOSH is still connected to the Cisco Finesse Notification Service).

Recovery

When any of the preceding failure scenarios are detected, the recommended course of action is to attempt or detect recovery of the server on which the scenario occurred, as well as to check for the availability of an alternate server using the following criteria (when applicable):

1. The BOSH connection is down.

   Periodically check the SystemInfo object for IN_SERVICE status. After the system is IN_SERVICE, attempt to re-establish the BOSH connection.

2. If BOSH is still connected and a SystemInfo OUT_OF_SERVICE notification is received:

   As long as the BOSH connection remains available, wait for a SystemInfo notification that the system is IN_SERVICE.

3. A 'finesse' user UNAVAILABLE presence is received.

   As long as the BOSH connection remains available, wait for an AVAILABLE presence notification for the 'finesse' user. Then wait for the SystemInfo IN_SERVICE notification.

Desktop Presence and Forced Logout

The Finesse server subscribes to the presence of the XMPP users of the Finesse desktop to monitor the health of the connection between the server and desktop.

Under certain conditions, Finesse sends a forced logout with a reason code of 255 to the CTI server.

In a Unified CCE deployment, the actual behavior of the desktop under these conditions depends on the setting for Logout on Agent Disconnect (LOAD).

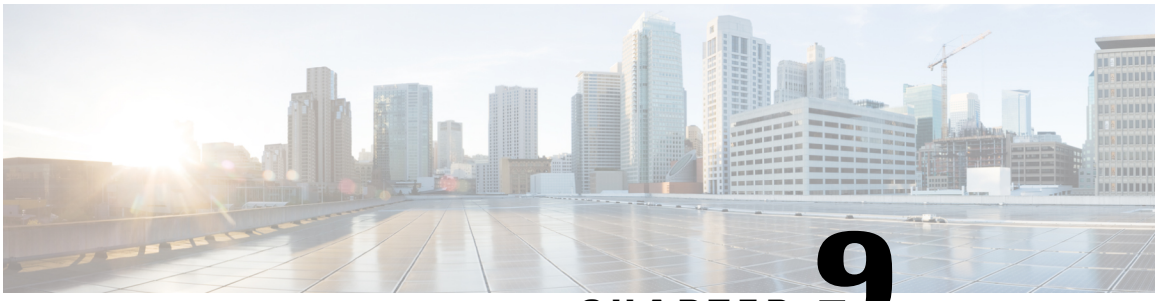
In a Unified CCX deployment, the agent is logged out.

Note Finesse takes up to 120 seconds to detect when an agent closes the browser or the browser crashes and Finesse waits 60 seconds before sending a forced logout request to the CTI server. Under these conditions, Finesse can take up to 180 seconds to sign out the agent.

The following table lists the conditions under which Finesse sends a forced logout to the CTI server:

Scenario	Desktop Behavior	Server Action	Race Conditions
The client closes, the browser crashes, or the agent clicks the Back button on the browser.	When you close the browser or navigate away from the Finesse desktop, the Finesse desktop makes a best-effort attempt to notify the server.	Finesse receives a presence notification of *Unavailable* from the client. Finesse waits 60 seconds, and then sends a forced logout request to the CTI server.	**1** The agent closes the browser window. Finesse receives a presence notification of *Unavailable* for the user. Finesse tries to sign the agent out; however, that agent is already signed out. **2** If the browser crashes, it can take the Finesse server up to 120 seconds to detect that the client is gone and send a presence notification to Finesse. A situation can occur where the client signs in to the secondary Finesse server before the primary Finesse server receives the presence notification caused by the browser crash. In this case, the agent may be signed out or put into Not Ready state on the secondary Finesse server. **3** If the Finesse desktop is running over a slower network connection, Finesse may not always receive an *Unavailable* presence notification from the client browser. In this situation, the behavior mimics a browser crash, as described in the preceding condition.
The client refreshes the browser	—	Finesse receives a presence notification of *Unavailable* from the client. Finesse	—

		waits 60 seconds before sending a forced logout request to the CTI server to allow the browser to reconnect after the refresh.	
The client encounters a network glitch (Finesse is in service)	Because the connection to the Finesse server temporarily goes down, the client fails over to the secondary Finesse server.	The primary Finesse server receives a presence notification of *Unavailable* from the client. Because Finesse is in service, it sends a forced logout request to the CTI server for the agent.	A situation can occur where the forced logout does not happen before the client signs in to the secondary Finesse server. If the agent is on a call, the primary Finesse server sends the forced logout request after the call ends. In a Unified CCE deployment, the agent is signed out or put into Not Ready state when the call ends, even though the client is already signed in to the secondary Finesse server. In a Unified CCX deployment, the agent is signed out.



CHAPTER 9

Finesse Desktop Gadget Development

Supported OpenSocial Features

The Finesse Desktop supports OpenSocial Core Gadget Specification 1.1.

Gadget Specification XML Features

The following table lists supported features that can be specified in the Gadget Specification XML or are available as an API for use in the JavaScript code of a gadget.

Name	Description
Locale	The <Locale> element specifies the locales that the gadget supports. The Finesse Desktop Gadget Container takes the locale provided by the browser and renders the gadget with the specific message bundle when available.
ModulePrefs: Scrolling	The Scrolling attribute of the ModulePrefs tag renders the gadget frame with a value of auto for scrolling. When the content exceeds the viewport, the browser renders a vertical or horizontal scrollbar. For a better user experience, we recommend that you use the gadgets.window.adjustHeight API to dynamically resize the gadget as needed instead of using this feature.

Name	Description
ModulePrefs: Title	The string provided is used for the title of the gadget shown in the title bar. You can also use the gadgets.window.setTitle API to set the title at runtime, which may offer more flexibility.

Required Module pref Features

Finesse requires that all gadgets use the following module pref features:

- <Require feature="pubsub-2" />: This feature is required for the gadget to load in the OpenAjax Hub.
- <Require feature="setprefs"/>: This feature is required for the Finesse JavaScript library to set the agent authorization string in the gadget prefs.

Note Before you can access the authorization string through the gadget prefs, you must first import the Finesse JavaScript library.

APIs Available to Gadget JavaScript

The following table lists the available APIs and methods.

Name	Parameters	Description
<static> gadgets.window.adjustHeight(opt_height)	opt_height (integer)-Preferred height in pixels. This parameter is optional. If the opt_height is not specified, the API attempts to fit the gadget to its content.	Adjusts the height of the gadget.
<static> gadgets.window.setTitle(title)	title (string)-Preferred title of the gadget.	Sets the title of the gadget.
<static> gadgets.io.makeRequest (url, callback, opt_params)	url (string)-Address from which content is fetched. callback (function)-Executed after content from the url is fetched. opt_params (Map<String, String>)-Additional optional parameters to pass to the request.	Fetches content from the provided URL and feeds that content into the callback function.

Gadget Preferences

The gadgets.Prefs class provides access to user preferences, module dimensions, and messages. Clients can access their preferences by constructing an instance of gadgets.Prefs (and optionally, passing in their module ID). Gadget preferences can then be set using the standard OpenSocial gadget APIs.

```
var myPrefs = new gadgets.Prefs();
myPrefs.set(“counter”, count +1);
```

In the Finesse Desktop, gadget preferences persist in the browser. After a gadget sets its preferences, anytime that gadget is constructed in the same browser, these preferences continue to be available through the APIs.

```
var myPrefs = new gadgets.Prefs();
helloValue = myPrefs.getString(“hello”);
```

Note Do not use preferences to persist critical application data. This data is stored in 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 for complex application data. Additionally, when the browser runs out of the allocated storage space, this data is purged.

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

Caveats

Although OpenSocial is a web standard, gadgets may exhibit different behaviors in various OpenSocial containers. You should always thoroughly test gadgets in Finesse to ensure that functionality is in accordance with customer requirements. The Finesse team will document known issues and best practices as they are discovered to help customers and partners build gadgets for the Finesse Desktop.

Gadget Caching

Gadget caching is enabled on the Finesse container. If you add a gadget, delete a gadget, or change the layout of the gadget on the desktop, you must restart Cisco Tomcat to clear the cache.

Note In a Unified CCX deployment, the service is called Cisco Finesse Tomcat.

If you make changes to the code of an existing gadget, you can restart Cisco Tomcat or you can pass a “nocache” parameter in the URL to clear the cache. You can pass the nocache parameter at the root level or at the desktop web app.

Example:

- http://server?nocache
- http://server/desktop?nocache
- http://server/desktop/container?nocache

Notifications on Finesse Desktop

The Finesse desktop contains support for OpenSocial Core Gadget Specification 1.1 (for more information, see http://opensocial-resources.googlecode.com/svn/spec/1.1/Core-Gadget.xml). OpenSocial Core Gadget Specification 1.1 supports an intergadget notification system that is based on the OpenAjax Hub 2.0 Specification (for more information, see http://www.openajax.org/member/wiki/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 recover based on the recovery strategy described inFinesse High Availability, on page 253 . If the BOSH connection cannot be re-established, the Finesse Desktop triggers a failover to the alternate Finesse server.

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

Finesse Notifications in Third-Party Containers

Strict requirements 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.

Finesse Topics

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.

Connection Information

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.

Finesse Notifications

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 Finesse Desktop), the Cisco Finesse 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 Cisco Finesse Notification Service to create a subscription for the notification feed (see Subscription Management on Finesse Desktop).

When connecting to the Cisco Finesse Notification Service, you must always specify a resource to identify your connection. Issues occur if the resource is omitted when the connection is created.

The resource “desktop” is reserved for the Finesse Desktop. Do not use this resource for other connections as it causes a conflict with 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}"
        }
    }
}
```

Finesse Requests

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.

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.

Finesse Responses

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);
```

Workflow Action Event

Topic Name	finesse.containerservices.workflowActionEvent
Topic Type	Gadgets subscribe to this topic.

Gadgets subscribe to the finesse.containerservices.workflowActionEvent topic to receive workflow action events to execute as a result of workflow evaluations.

Note Third-party gadgets subscribing directly to the OpenAjax Hub for the Workflow Action Event topic might cause the Finesse Workflow Engine to lose its subscription and no longer be able to execute workflow actions. Third party gadgets should instead implement something like the following:

```
var _containerServices = finesse.containerservices.ContainerServices.init();
    _containerServices.addHandler("finesse.containerservices.workflowActionEvent",
function(data) {
         // Perform logic on "data", which is a WorkflowActionEvent object
        });
```

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

```
{
  uri: string,
  name: string,
  type: string,
  params: [
    {
      name: string,
      value: string,
      expandedValue: string
    }
  ],
  actionVariables: [
    {
      name: string,
      node: string,
      type: string,
      testValue: string,
      actualValue: string
    }
  ]
}
```

Field	Description
uri	In the uri, the id maps to the primary key of the WorkflowAction entry.
name	The name of the workflow action.
type	The type of workflow action. Possible value is BROWSER_POP.
params	List of Param subobjects (see below).
actionVariables	List of ActionVariable subobjects (see below). There can be at most 5 Action Variable subobjects assigned to a workflow action.

The Param subobject uses the following fields:

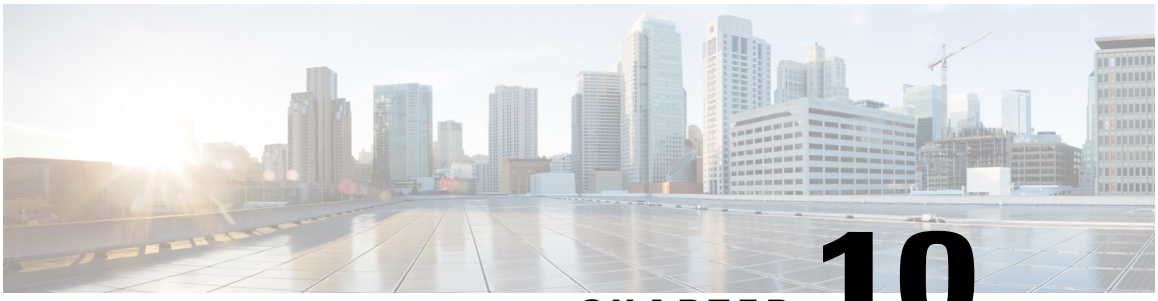
Field	Description
name	The name of the parameter.
value	The value of the parameter.
expandedValue	The value of the parameter with variables substituted with their values.

The ActionVariable subobject uses the following fields:

Field	Description
name	The name of the variable.
node	The XPath to extract from the dialogs XML.
type	Indicates if this is a SYSTEM or CUSTOM variable.
testValue	The value used to test the variable.
actualValue	The actual value of the variable in context of the events used by the workflow evaluation.

Subscription Management on Finesse Desktop

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 Finesse Requests.



CHAPTER 10

Third-Party Gadgets

Cisco Finesse provides a mechanism for you to upload third-party gadgets to the Finesse server. This mechanism allows one user in the Finesse system to upload gadgets to one directory using secure FTP (SFTP).

The account used to upload gadgets is named 3rdpartygadget.The directory structure to where third-party gadgets are deployed is:

```
/files
```

The 3rdpartygadget account only has permission to this directory (and any directories created under it).

Password for 3rdpartygadget Account

Use the following CLI command to set (or reset) the password for the 3rdpartygadget account:

utils reset_3rdpartygadget_password
You are prompted to enter a password. After you enter a password, you are prompted to confirm the password.

You must set the password before you can upload gadgets using SFTP.

Note The password for the 3rdpartygadget account must be between 5 and 32 characters long and must not contain spaces or double quotes (").

Upload Third-Party Gadgets

After you set the password for the 3rdpartygadget account, you can use SFTP to upload third-party gadgets to the Finesse server, as illustrated in the following example.

```
my_workstation:gadgets user$ sftp 3rdpartygadget@<finesse>
3rdpartygadget@<finesse>'s password:
Connected to <finesse>.
sftp> cd /files
sftp> put HelloWorld.xml
Uploading HelloWorld.xml to /files/HelloWorld.xml
HelloWorld.xml                                                                    100%
 2751     2.7KB/s   00:00
sftp> exit
```

After you upload a gadget, it is available under the following URL:

http://<finesse>/3rdpartygadget/files/

To access the gadget uploaded in the previous example, use the following URL:

http://<finesse>/3rdpartygadget/files/HelloWorld.xml

When you add a gadget to the desktop layout, that gadget can be referenced using a relative path. To include the gadget that was uploaded in the previous example in the desktop layout, add the following XML (highlighted) to the layout:

```
<finesseLayout xmlns="http://www.cisco.com/vtg/finesse">
      <layout>
        <role>Agent</role>
        <page>
          <gadget>/desktop/gadgets/CallControl.jsp</gadget>
          <gadget>/3rdpartygadget/files/HelloWorld.xml</gadget>
        </page>
        ...
      </layout>
      <layout>
        <role>Supervisor</role>
        <page>
          <gadget>/desktop/gadgets/CallControl.jsp</gadget>
          <gadget>/3rdpartygadget/files/HelloWorld.xml</gadget>
        </page>
        ...
      </layout>
    </finesseLayout>
```

Note Because of browser caching and caching in the Finesse web server, you may need to clear the browser cache or restart the Cisco Tomcat service before gadget changes take effect. If you make a change to a gadget and the change is not reflected on the Finesse desktop, clear your browser cache.

If you do not see the changes after you clear the browser cache, use the following CLI command to restart the Cisco Tomcat service:

admin:utils service restart Cisco Tomcat

Replication

You must set the password for the 3rdpartygadget account on both the primary and secondary Finesse servers.

Gadgets must be manually uploaded to both the primary and secondary Finesse servers.

Migration

When you perform an upgrade, third-party gadgets are migrated to the new version.

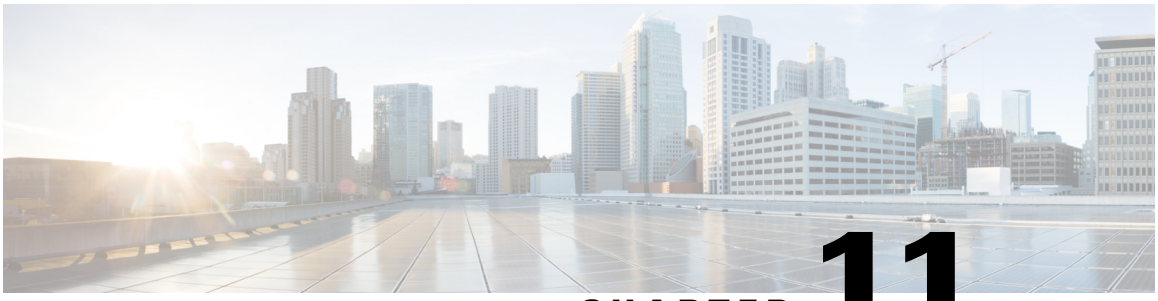
The 3rdpartygadget account password is not migrated across upgrades. After an upgrade, you must reset the password for the 3rdpartygadget account before you can make changes to third-party gadgets. You must reset the password on both the primary and secondary Finesse servers.

Backup and Restore

Third-party gadgets are preserved when you perform a DRS backup and restore.

Restrictions

Any attempt to GET JavaServer Pages (jsp) using the URL http://<finesse>/3rdpartygadget/files is blocked. You will receive a 403 (Access Denied) error code when attempting to retrieve a jsp.



CHAPTER 11

Documents and Documentation Feedback

Documents

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

1. Point your browser to http://developer.cisco.com/web/finesse/overview.
2. Click the link for Cisco Finesse.
3. Sign in with your Cisco credentials.
4. 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 Upgrade Guide*
- *Cisco Finesse Administration Guide*
- *Release Notes for Cisco Finesse*

JavaScript Library and Sample Gadgets

The Finesse JavaScript library and sample gadgets are available on the CDN at the following link: http://developer.cisco.com/web/finesse/documentation

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.

Glossary

Terms and definitions are provided in this section.

AWDB

Acronym for Administrative Workstation Database. This database resides on the Administration & Data Server.

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 Cisco Finesse.