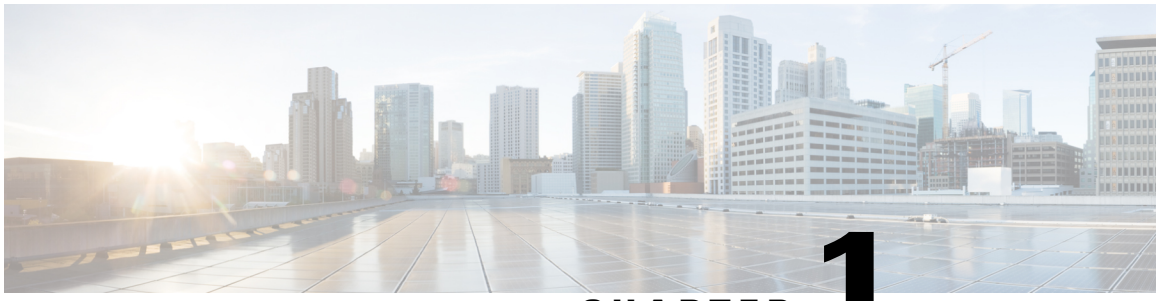
Cisco Finesse Web Services Developer Guide

First Published: June 18, 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

New in This Release

The following sections provide an overview of changes for Cisco Finesse Release 10.5(1).

Finesse and Multiline Support

In this release, Finesse supports the configuration of multiple lines on agent phones. If Unified CCE or Unified CCX is configured for multiline, you can configure one or more secondary lines on an agent phone. However, any events sent by the CTI server as a result of call activity on an agent's non-primary (non-ACD) lines are blocked by the Finesse server and are not published to the Finesse clients. No information about calls handled on non-ACD lines appears on the Finesse desktop.

Cisco Finesse Notification Service Upgrade

The Cisco Finesse Notification Service (OpenFire) has been upgraded from version 3.7.1 to version 3.8.2. Openfire was upgraded for the following reasons:

- To provide more robust eventing and session management
- To increase performance

With the previous version of Openfire (3.7.1), when a subscription using a full JID (username@domain/resource) existed with no active user session corresponding to that full JID, and an event was published to that node, Openfire fell back to the bare JID (username@domain) and looked for any active session for that user. This behavior is not aligned with the XMPP specification and has been corrected in Openfire version 3.8.2.

In version 3.8.2, Openfire sends a notification to a user if the subscription was made with either a bare JID for that user (user@domain) or if the subscription was made with the full JID (user@domain/resource) with which the user is currently logged in.

For more information, see section 10.5 "Node in the Same Domain" at http://xmpp.org/rfcs/rfc3920.html.

Schedule Callbacks

A new dialog API was added that allows a user to schedule, update, or cancel a callback for an Outbound Dialer call. For more information, see Dialog—Schedule or Cancel a Callback, on page 82

Direct Preview Outbound for Unified CCX

This release of Finesse supports Outbound with Unified CCX. A new dialog API was added to allow a user to accept, close, or reject a Direct Preview Outbound call. For more information, see Dialog—Accept, Close, or Reject a Direct Preview Outbound Reservation, on page 80.

Another dialog API was added to allow a user to reclassify a Direct Preview call. For more information, see Dialog—Reclassify a Direct Preview Call, on page 81.

Diagnostic Portal APIs

Diagnostic Portal APIs integrate Finesse with the Cisco Prime Contact Center Module and get information about the health of the Finesse system. For more information, see Diagnostic Portal APIs, on page 224.

Finesse Container Timer

The Finesse container provides a new service (the TimerTickEvent) for Release 10.5(1) that can be used by third-party gadget developers. This event is published every second and can be used to asynchronously update a gadget instead of using the JavaScript commands setInterval() or setTimeout(). For more information, see Finesse Container Timer, on page 257.

Multiple Column Support

This release supports adding multiple columns to the Finesse desktop. The layout XML includes tags for columns. The default desktop has one column. Additional columns can be added by placing gadgets within the column tags in the layout XML. The maximum number of columns is four. For more information, see LayoutConfig, on page 133.

Maximize Support for Third-Party Gadgets

This release adds support for maximizing third-party gadgets. You can use the gadgets.views parameter to specify a "canvas" view, which allows a gadget to support a maximized view. When maximized, a gadget expands to fill the height and width of the desktop container. For more information, see APIs Available to Gadget JavaScript, on page 248.

Reason Code Changes

In this release, Finesse enforces that reason code values are unique within a category (Not Ready or Sign Out). In previous releases, Finesse allowed you to create multiple reason codes with the same code value.

For Unified CCX deployments, Finesse now enforces that reason code values must be between 1 and 999.

If you try to create a new reason code with the same code value of an existing reason code, Finesse displays an error that states the reason code is already being used.

When you upgrade from a previous release, you may have reason codes in your database that violate these new rules. Finesse allows these reason codes to be migrated across the upgrade and allows you to continue to use them. However, they may cause unexpected behavior and incorrect reporting. Also, if you update an existing reason code that violates these new rules, Finesse displays an error when you attempt to save it.

Cisco Finesse REST APIs

This document is the official reference for the Cisco Finesse Application Programming Interface (API). The Finesse desktop APIs support the Finesse desktop, providing agent desktop functionality, such as call control and state changes.

The Finesse configuration APIs support the Finesse administration console, providing the ability to configure resources (such as reason codes, wrap-up reasons, and workflows).

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. The guide includes a section to assist developers with running and validating the APIs in a lab environment.

JavaScript Library and Sample Gadgets

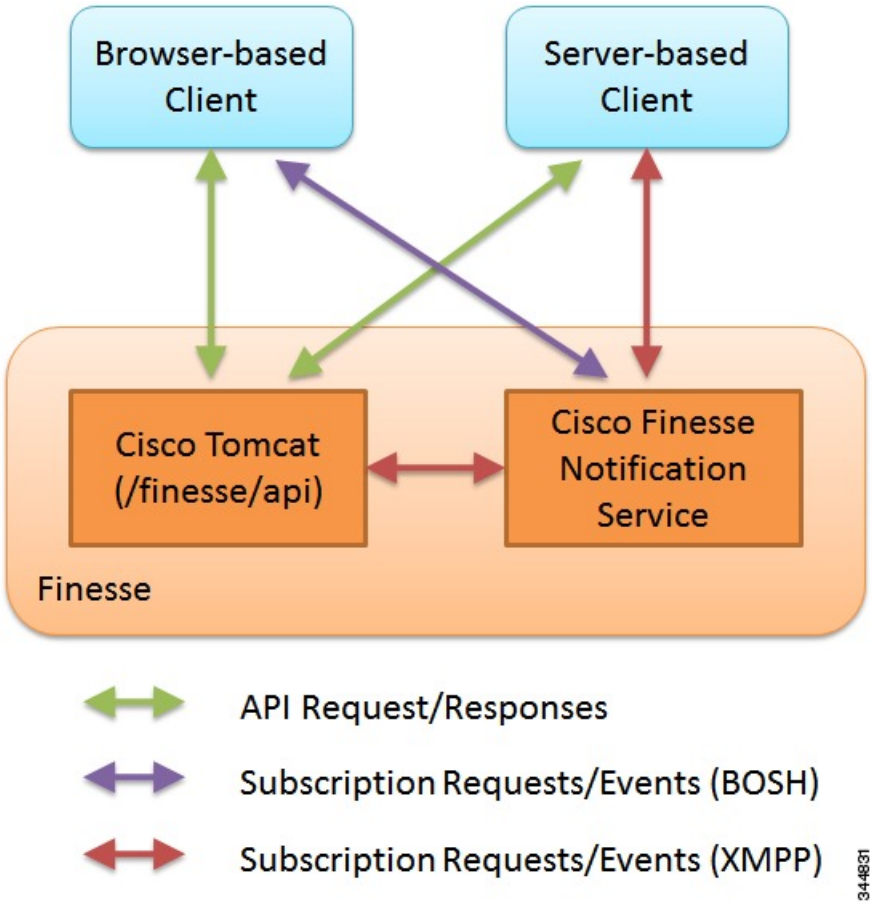
Finesse provides a JavaScript library (finesse.js) and a number of sample gadgets to help jumpstart your gadget development. The JavaScript library provides a substantial amount of fundamental code infrastructure that you would otherwise need to write yourself. The JavaScript library and sample gadgets 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.

Communication with the Cisco Finesse Web Service

The service names in the following 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.

Figure 1: Finesse API and Event Flow



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.

Note In a Unified CCE deployment, you do not need to include the port number in the URI for HTTP requests. In a Unified CCX deployment, you must include the port number.

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

```
http://<FQDN>:<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 in a Unified CCE deployment for all URIs and example URIs. If you want to make HTTP requests in a Unified CCX deployment, include the port number in the URIs:

If you want to use HTTPS requests (Unified CCE and Unified CCX), 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).

API Parameter Types

The following sections describe the parameter and data types for the Cisco Finesse APIs.

API Header Parameters

Name	Type	Description
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.
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.

Body Parameter

A body parameter (also known as a complex parameter) appears in the body of the message.

Path Parameter

A path parameter is included in the path of the URI. In the following example, *dialogId* is a path parameter.

```
http://<FQDN>/finesse/api/Dialog/<dialogId>
```

Query Parameter

A query parameter is passed in a query string on the end of the URI you are calling. The query parameter is preceded by a question mark. Multiple query parameters are connected by an ampersand (&). In the following example, *category* is a query parameter.

```
http://<FQDN>/finesse/api/User/<id>/ReasonCodes?category=NOT_READY
```

Data Types

The following table lists the data types used in API parameters and notification message fields.

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. If a maximum length exists, it is listed with the parameter description.

Cisco Finesse API Errors

Error codes for Cisco Finesse are categorized as follows:

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

Each error includes a failure response, error type, error message, and error data. The following is an example of a failure message format:

```
<ApiErrors>
   <ApiError>
      <ErrorType>Authentication Failure</ErrorType>
      <ErrorMessage>UNAUTHORIZED</ErrorMessage>
      <ErrorData>jsmith</ErrorData>
   </ApiError>
</ApiErrors>
```

In addition to Cisco Finesse API errors, a response may 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/.

HTTP Errors

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

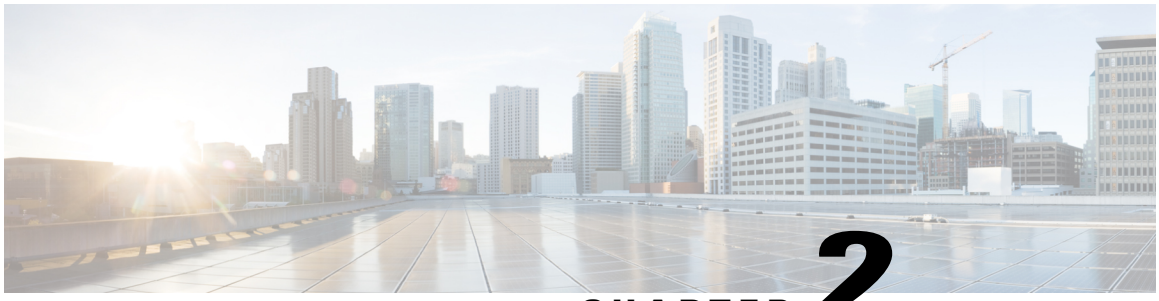
500 Internal Server Error

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.



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:

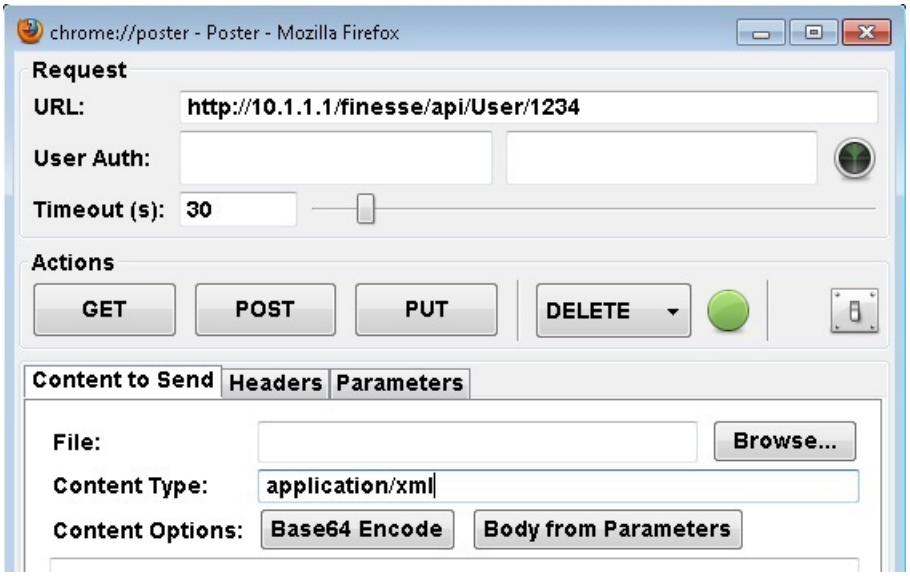
1. Click the **SSL** tab.
2. Click the **Etc** tab.
3. Ensure that the **Trust self-signed certificate** check box is checked.
4. Set the Hostname verifier to **Allow All**.

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 Sign In API 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:

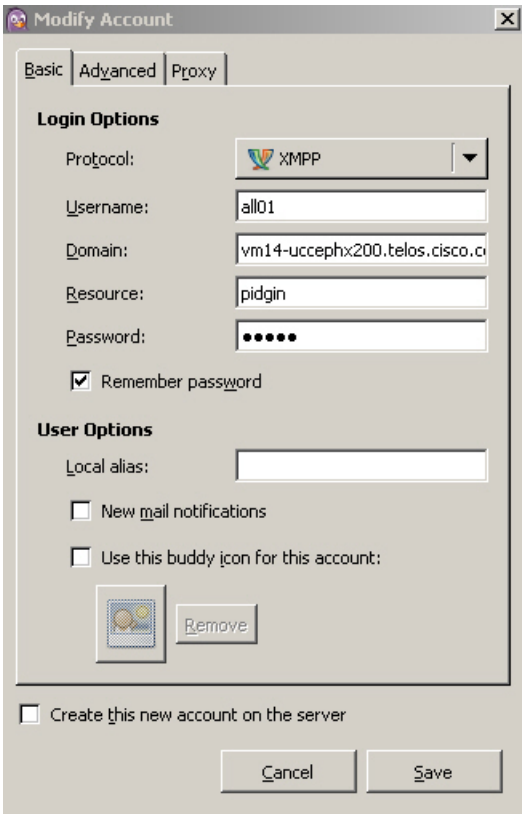
1. In Pidgin, go to **Tools > Plugins** to open the Plugins dialog box.
2. Check the **XMPP Console** and **XMPP Service Discovery** check boxes.

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

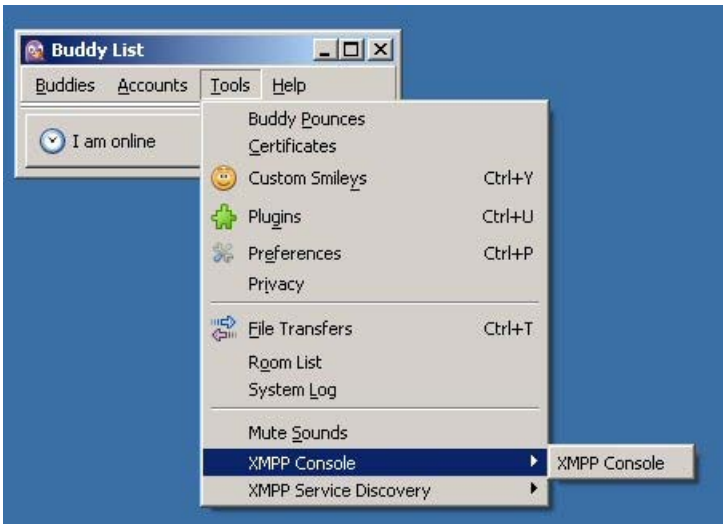


7 Click **Save**.
8 Click the **Advanced** tab.
9 Check the **Allow plaintext auth over unencrypted streams** check box.
10 For Connect Server, enter the IP address of the Finesse server.
11 If the Connection Security drop-down menu is present, choose **Use encryption if available**.
12 Click **Save**.

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.

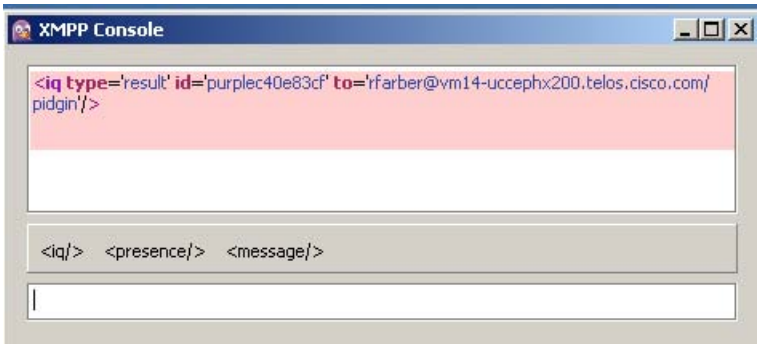
Figure 5: Open XMPP Console in Pidgin



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.

GET requests are synchronous. That is, the response body of a successful GET request contains all requested contents, which you can view in Poster or RESTClient. No event is published by XMPP and no event is received in Pidgin.

PUT and POST requests are asynchronous. A successful response is an HTTP return code of 200 or 202. The response body does not contain the updated object information.

If a PUT, POST, or DELETE request is on a User or Dialog object, the update is published by XMPP as a real-time event to Pidgin. If a PUT, POST, or DELETE request is on a configuration object (for example, a ReasonCode object), XMPP does not publish a real-time update. You must perform a GET request to get an updated copy of the object.

Failed GET, PUT, POST, and DELETE requests are synchronous. If a request fails, Poster or RESTClient display the error. No event is published by XMPP to Pidgin.

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

Note This example shows the URL field for a Unified CCE deployment. In a Unified CCX deployment, you must include the port number in the URL.

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/1234/Dialogs</dialogs>
      <extension>1001</extension>
      <firstName>John</firstName>
      <lastName>Smith</lastName>
      <loginId>1234</loginId>
      <loginName>jsmith</loginName>
      <roles>
        <role>Agent</role>
      </roles>
      <pendingState></pendingState>
      <reasonCodeId>-1</reasonCodeId>
      <settings>
        <wrapUpOnIncoming></wrapUpOnIncoming>
      <settings>
      <state>NOT_READY</state>
      <stateChangeTime>2014-05-27T00:33:44.836Z</stateChangeTime>
      <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.

Note This example shows the URL field for a Unified CCE deployment. In a Unified CCX deployment, you must include the port number in the URL.

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>
```

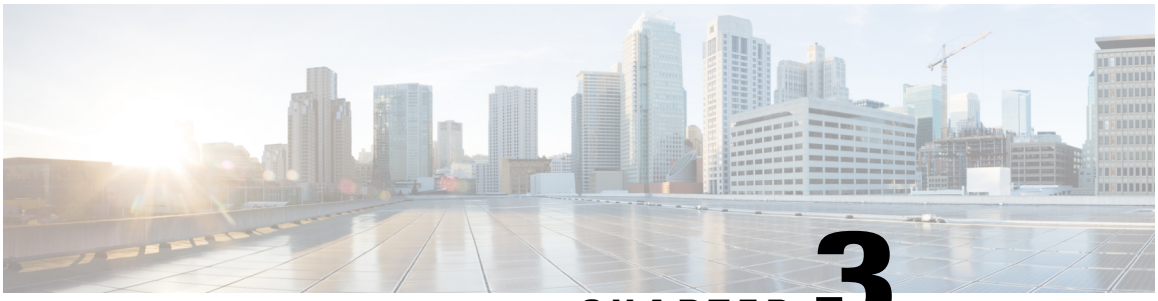
5. Click **PUT**.

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>
      <pendingState></pendingState>
      <settings>
        <wrapUpOnIncoming></wrapUpOnIncoming>
      </settings>
      <stateChangeTime>2014-05-27T00:35:24.123Z</stateChangeTime>
      <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>
```



CHAPTER 3

Cisco Finesse Desktop APIs

Agents and supervisors use the Cisco Finesse Desktop APIs to communicate between the Finesse desktop and Finesse server, and Unified Contact Center Enterprise (Unified CCE) or Unified Contact Center Express (Unified CCX) to send and receive information about the following:

- Agents and agent states
- Calls and call states
- Teams
- Queues
- Client logs

The Finesse desktop APIs must provide BASIC authentication credentials, as described in Client Requests, on page 4.

User

The User object represents an agent or supervisor and includes information about the user, such as roles, state, and teams. The User object is structured as follows:

```
<User>
   <uri>/finesse/api/User/1001001</uri>
   <roles>
      <role>Agent</role>
      <role>Supervisor</role>
   </roles>
   <loginId>1001001</loginId>
   <loginName>csmith</loginName>
   <state>NOT_READY</state>
   <stateChangeTime>2012-03-01T17:58:21.234Z</stateChangeTime>
```

```
<pendingState></pendingState>
<reasonCodeId>16</reasonCodeId>
<ReasonCode>
   <category>NOT_READY</category
   <uri>/finesse/api/ReasonCode/16</uri>
   <code>10</code>
   <label>Team Meeting</label>
   <forAll>true</forAll/
   <id>16</id>
</ReasonCode>
<settings>
   <wrapUpOnIncoming>OPTIONAL</wrapUpOnIncoming>
</settings>
<extension>1001001</extension>
<mobileAgent>
   <mode>CALL_BY_CALL</mode>
   <dialNumber>4085551234</dialNumber>
</mobileAgent>
<firstName>Chris</firstName>
<lastName>Smith</lastName>
<teamId>500</teamId>
<teamName>Sales</teamName>
<dialogs>/finesse/api/User/1001001/Dialogs</dialogs>
<teams>
   <Team>
      <uri>/finesse/api/Team/2001</uri>
      <id>2001</id>
      <name>First Line Support</name>
   </Team>
   <Team>
      <uri>/finesse/api/Team/2002</uri>
      <id>2002</id>
      <name>Second Line Support</name>
    </Team>
   <Team>
      <uri>/finesse/api/Team/2003</uri>
      <id>2003</id>
      <name>Third Line Support</name>
   </Team>
   ... other teams ...
</teams>
</User>
```

User APIs

User—Sign In to Finesse

The User—Sign in to Finesse API allows a user to sign in to the CTI server. If the response is successful, the user is signed in to Finesse and is automatically placed in NOT_READY state.

This API forces a sign-in. That is, if the user is already signed in, that user is authenticated via the sign-in process. If the user's credentials are correct, the user is signed in again but the user keeps the current state. For example, if a user signs in, changes state to Ready, and then signs in again, the user remains in Ready state.

Note To sign in as a mobile agent, see User—Sign In as a Mobile Agent, on page 22.

URI:	http://<FQDN>/finesse/api/User/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234

Security Constraints:	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>`

Request Parameters:	id (required): The ID of the user state (required): The new state that the user wants to be in (LOGIN) extension (required): The extension with which the user wants to sign in
HTTP Response:	202: Success 400: Bad Request (for example, malformed or incomplete request, invalid extension) 400: Parameter Missing 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)

Example Failure Response:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>User Not Found</ErrorType>`
	`  <ErrorMessage>UNKNOWN_USER</ErrorMessage>`
	`  <ErrorData>4023</ErrorData>`
	`  </ApiError>`
	`</ApiErrors>`

Notifications Triggered:	User notification

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 supports 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).

User—Sign In as a Mobile Agent

The User—Sign in as a mobile agent API allows a user to sign in to the CTI server as a mobile agent. This API uses the existing User object with a LOGIN state only. The user must be authenticated to use this API successfully.

Note Additional configuration is required on Unified CCE and Unified Communications Manager 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 with Unified CCE. Unified CCX does not support mobile agents.

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

Request Parameters:	id (required): The ID of the user state (required): The new state that the user wants to be in (for this API, the state must be set to LOGIN) extension (required): The extension with which to sign in the user mobileAgent (required): Indicates that the user is a mobile agent mode (required): The connection mode for the call dialNumber (required): The phone number that the system calls to connect with the mobile agent

HTTP Response:	202: Success This response only indicates the 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 was not provided) 400: Generic Error 401: Unauthorized (for example, the user is not authenticated in the Web Session) 401: Invalid User Authorization Specified (an authenticated user tried to make a request for another user) 404: User Not Found (for example, the agent is not recognized)

Example Failure Response:	`<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>`

Notifications Triggered:	User notification

User—Sign Out of Finesse

This API allows a user to sign out of Finesse.

URI:	http://<FQDN>/finesse/api/User/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234
Security Constraints:	Agents and supervisors can use this API. 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>`

Request Parameters:	id (required): The ID of the user state (required): The new state that the user wants to be in (LOGOUT)

HTTP Response:	202: Success 400: Bad Request (for example, malformed or incomplete request, 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)

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

Notifications Triggered:	User notification

User—Get User

The User—Get user API allows a user to get a copy of the User object. For a mobile agent, this operation returns the full User object, including the mobile agent node.

Note Mobile agent information is available to the Finesse node on which the mobile agent is signed in. However, the other Finesse node in the cluster does not have the mobile agent information. If the mobile agent signs in to the other node (for example, during a client failover), the mobile agent information is lost and the User object does not return any mobile agent data fields. As a result, the Finesse desktop inaccurately represents the mobile agent as a regular agent (including all related features). Any other type of CTI failover also results in Finesse losing the current mobile agent information. However, the Unified Mobile Agent feature behaves as normal whether Finesse knows the agent is a mobile agent or not.

As a workaround, the mobile agent can sign out and sign back in as a mobile agent.

URI:	http://<FQDN>/finesse/api/User/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234
Security Constraints:	Agents can only get 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: Authorization Failure 401: Invalid Authorization User Specified 404: User Not Found 500: Internal Server Error 503: Service Unavailable

Example Response:

```
<User>
   <uri>/finesse/api/User/1234</uri>
   <roles>
      <role>Agent</role>
      <role>Supervisor</role>
   </roles>
   <loginId>1234</loginId>
   <loginName>csmith</loginName>
   <state>NOT_READY</state>
   <stateChangeTime>2012-03-01T17:58:21.234Z</stateChangeTime>
   <pendingState></pendingState>
   <reasonCodeId>16</reasonCodeId>
   <ReasonCode>
      <category>NOT_READY</category
      <uri>/finesse/api/ReasonCode/16</uri>
      <code>10</code>
      <label>Team Meeting</label>
      <forAll>true</forAll/
      <id16</id>
   </ReasonCode>
   <settings>
      <wrapUpOnIncoming>OPTIONAL</wrapUpOnIncoming>
   </settings>
   <extension>1001001</extension>
   <mobileAgent>
      <mode>CALL_BY_CALL</mode>
      <dialNumber>4085551234</dialNumber>
   </mobileAgent>
   <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/2001</uri>
         <id>2001</id>
         <name>First Line Support</name>
      </Team>
      <Team>
         <uri>/finesse/api/Team/2002</uri>
         <id>2002</id>
         <name>Second Line Support</name>
       </Team>
      <Team>
         <uri>/finesse/api/Team/2003</uri>
         <id>2003</id>
         <name>Third Line Support</name>
      </Team>
      ... other teams ...
   </teams>
 </User>
```

Example Response (Mobile Agent):

Note Mobile agent only applies to Unified CCE deployments).

```
 <User>
   ... Full User Object ...
   <mobileAgent>
      <mode>CALL_BY_CALL</mode>
      <dialNumber>4085551234</dialNumber>
   </mobileAgent>
</User>
```

Example Failure Response:

```
<ApiErrors>
     <ApiError>
          <ErrorType>User Not Found</ErrorType>
          <ErrorMessage>UNKNOWN_USER</ErrorMessage>
          <ErrorData>4023</ErrorData>
   </ApiError>
</ApiErrors>
```

User—Get List

This 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:	Only administrators can get a list of users.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 500: Internal Server Error 503: Service Unavailable

Example 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>`

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

User—Get List of Dialogs

This API allows an agent or administrator to get a list of dialogs associated with a particular user.

For more information about the Dialog object, see Dialog, on page 54.

URI:	http://<FQDN>/finesse/api/User/<id>/Dialogs
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234/Dialogs
Security Constraints:	Agents can only get a list of their own dialogs. Administrators can get a list of dialogs associated with any user.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 500: Internal Server Error

Example 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>`

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

User—Change Agent State

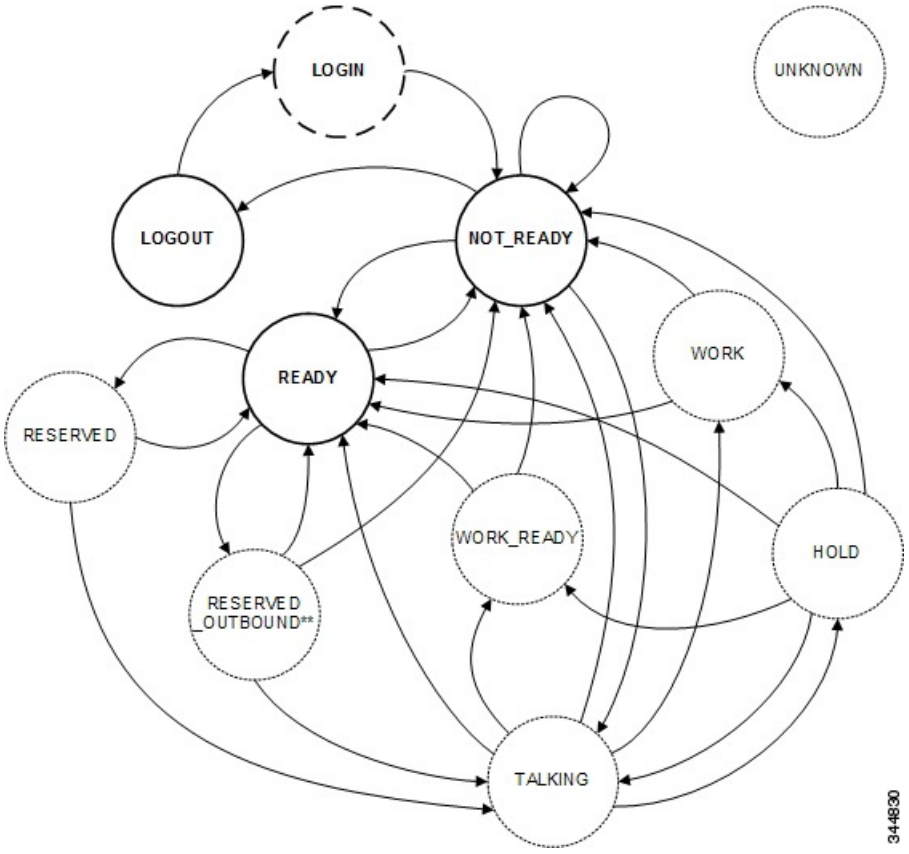
This API allows a user to change the state of an agent on the CTI server. Agents can change their own states

If the request to change an agent's state is successful, the response is sent as part of a User notification.

The following figure illustrates the supported state transitions by Unified CCE agents.

Note The following 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 failover conditions. The 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.

The following table describes supported agent state transitions for Unified CCE.

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.

From	To	Description
WORK	READY	To leave WORK state, agents can set their state to READY.
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.

The following table describes supported agent state transitions 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.
READY	RESERVED_ OUTBOUND_ PREVIEW	An outbound agent becomes reserved to handle an Outbound Option Direct Preview call.
RESERVED_ OUTBOUND_ PREVIEW	TALKING	An outbound agent accepts a direct preview call and the call is active.

Users can set the following states with 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.

Users can be in the following states while on a call. However, users cannot place themselves in these states. For example, agents cannot change their state to TALKING. Agents enter TALKING state when they answer a call.

- RESERVED
- RESERVED_OUTBOUND
- RESERVED_OUTBOUND_PREVIEW
- TALKING
- HOLD
- WORK
- WORK_READY

RESERVED_OUTBOUND user state:

Users who belong to Outbound Option skill groups transition from READY state to RESERVED_OUTBOUND state when those users are reserved for Progressive or Predictive Outbound Option calls.

In a Unified CCE deployment, users can change their state to READY or NOT_READY to exit this state. If not ready reason codes are configured, users must specify a reason code to transition to NOT_READY state. If the user does nothing and then the call is transferred to the user, the user transitions to TALKING state. If the call is not transferred to the user, the user transitions back to READY state.

In a Unified CCX deployment, users cannot change their state to exit RESERVED_OUTBOUND state. If auto-answer for the predictive or progressive call is not enabled and the agent does not answer the call, the agent transitions to NOT_READY state. If the call does not reach a voice contact or if the reservation timer on Unified CCX expires, the agent transitions to READY state.

RESERVED_OUTBOUND_PREVIEW user state:

Users who belong to Outbound Option skill groups transition from READY state to RESERVED_OUTBOUND_PREVIEW state when they are reserved for Outbound Option Preview or Direct Preview calls. Users cannot set their state to RESERVED_OUTBOUND_PREVIEW.

In a Unified CCE deployment, users can click Close or Reject on the Outbound Option dialog. Changing the user's state to READY or NOT_READY does not generate a state change notification but does affect the user state when the call is complete. For example, if the user selects NOT_READY state while in RESERVED_OUTBOUND_PREVIEW state, the user transitions to NOT_READY state after clicking Close or Reject.

In a Unified CCX deployment, users cannot change their state directly when in RESERVED_OUTBOUND_PREVIEW state. The state can only be changed by issuing a Dialog Accept, Close, or Reject request or when the reservation call times out.

WORK and WORK_READY user states:

A user is in WORK or WORK_READY state during wrap-up. A user is placed in WORK state when the user is set to transition to NOT_READY state when wrap-up ends. A user is in WORK_READY state when the user is set to transition to READY state when wrap-up ends.

A user transitions to WORK state for 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.

When the wrap-up timer expires, the user transitions to NOT_READY state.

WORK_READY state applies only to Unified CCE deployments. A user transitions to WORK_READY state for the following reasons:

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

When the wrap-up timer expires, the user transitions to READY state.

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

- A supervisor can only change the state of a user who is assigned to that supervisor's team.
- A supervisor can only set the state of another user to NOT_READY, READY, or LOGOUT.
- A supervisor can set the state of another user to READY only if that user is in RESERVED, TALKING, or HOLD state.
- A supervisor can set the state of a user to LOGOUT only if that user is in READY, NOT_READY, RESERVED, RESERVED_OUTBOUND, RESERVED_OUTBOUND_PREVIEW, TALKING, HOLD, WORK, or WORK_READY state.
- A supervisor can set the state of a user to NOT_READY only if that user is in READY, WORK, or WORK_READY state.
- When a supervisor uses this API to set the state of a user to NOT_READY, a reason code must not be used. If a reason code is provided, Finesse rejects it and returns a 400 Invalid Input error. Finesse sends a hard-coded reason code to indicate that the state change was performed by the supervisor.

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

Request Parameters:	id (required): The ID of the user state (required): The new state the user wants to be in (for example, LOGOUT, READY, NOT_READY)
HTTP Response:	200: Success 400: Bad Request 401: Invalid Supervisor 401: Unauthorized 404: Not Found 500: Internal Server Error 503: Service Unavailable

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

Notifications Triggered:	User notification

Platform-Based API Differences

The following table describes API differences between a stand-alone Finesse deployment with Unified CCE and 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>`

Scenario	Response
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>`
While talking on a non-ICD call (agent state can be TALKING in Unified CCE or NOT_READY in Unified CCX), 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>`

Scenario	Response
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 NOT_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>
A supervisor changes the state of an agent on that supervisor's team to NOT_READY.	**Stand-alone Finesse with Unified CCE:** Finesse sends a hard-coded reason code of 999 to indicate the forced state change. **Coresident Finesse with Unified CCX:** Finesse sends a hard-coded reason code of 33 to indicate the forced state change.

User—Change Agent State With Reason Code

This API allows a user to change the agent state in the CTI server and pass along the code value of a corresponding reason code. Users can use this API only when changing state to NOT_READY or LOGOUT.

URI:	http://<FQDN>/finesse/api/User/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234
Security Constraints:	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>
Request Parameters:	id (required): The ID of the user reasonCodeID (required if reason codes are configured for the given state): The database ID for the reason code state (required): The new state the user wants to be in (NOT_READY, LOGOUT)

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 Specified (for example, the authenticated user tried to make a request for another user)
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Parameter Missing</ErrorType> <ErrorData>state</ErrorData> <ErrorMessage>State Parameter missing</ErrorMessage> </ApiError> </ApiErrors>
Notifications Triggered:	User notification

User—Get Reason Code

This API allows an agent or supervisor to get an individual Not Ready or Sign Out reason code, which is already defined and stored in the Finesse database (and that is applicable to the agent or supervisor).

Users can select the reason code to display on their desktops when they change their state to NOT_READY or LOGOUT.

For more information about the ReasonCode object, see ReasonCode, on page 138.

URI:	http://<FQDN>/finesse/api/User/<id>/ReasonCode/<reasonCodeId>
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234/ReasonCode/12
Security Constraints:	Administrators, agents, and supervisors can use this API. A user must be signed in to get a reason code. The reason code must be global (forAll parameter set to true) or be assigned to a team to which the user belongs. Only an administrator can get another user's reason codes.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—

HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error (for example, the object does not exist, the object is stale, or violation of DB constraint) 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

Example Response:	`<ReasonCode>`
	`  <uri>finesse/api/ReasonCode/1</uri>`
	`  <category>NOT_READY</category>`
	`  <code>12</code>`
	`  <label>Lunch</label>`
	`  <forAll>true</forAll>`
	`</ReasonCode>`

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

User—Get Reason Code List

This API allows an agent or supervisor to get a list of Not Ready or Sign Out reason codes (that are applicable to that agent or supervisor), which are defined and stored in the Finesse database. Users can assign one of the reason codes on the desktop when they change their state to NOT_READY or LOGOUT.

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

Reason codes that have the forAll parameter set to true apply to any user.

The category parameter is required when making a request to get a list of reason codes.

For information about the ReasonCode object, see ReasonCode, on page 138.

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:	Administrators, agents and supervisors can use this API. A user must be signed in to get a list of reason codes. Only an administrator can get another user's list of reason codes.
HTTP Method:	GET

Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error (for example, the object does not exist, the object is stale, or violation of DB constraint) 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

Example Response:	`<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>`

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

User—Get Wrap-Up Reason

This API allows a user to get a WrapUpReason object.

For more information about the WrapUpReason object, see WrapUpReason, on page 146.

URI:	http://<FQDN>/finesse/api/User/<id>/WrapUpReason/<wrapUpReasonId>
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234/WrapUpReason/1001
Security Constraints:	Administrators, agents, and supervisors can use this API. A user must be signed in to get a wrap-up reason. Only an administrator can get another user's wrap-up reasons.

HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
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, or violation of DB constraint) 401: Authorization Failure 401: Invalid Authorization User Specified 404: Not Found (for example, the wrap-up reason does not exist or has been deleted) 500: Internal Server Error

Example Response:	`<WrapUpReason>`
	`  <uri>finesse/api/User/1234/WrapUpReason/205</uri>`
	`  <label>Product Question</label>`
	`  <forAll>true</forAll>`
	`</WrapUpReason>`

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

User—Get Wrap-Up Reason List

This API allows a user to get a list of all wrap-up reasons applicable for that user.

For more information about the WrapUpReason object, see WrapUpReason, on page 146.

URI:	http://<FQDN>/finesse/api/User/<id>/WrapUpReasons
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234/WrapUpReasons
Security Constraints:	Administrators, agents, and supervisors can use this API. A user must be signed in to get a list of wrap-up reason. Only an administrator can get another user's list of wrap-up reasons.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	—
HTTP Response:	200: Success 400: Finesse API Error (for example, the object does not exist, the object is stale, or violation of DB constraint) 401: Authorization Failure 401: Invalid Authorization User Specified 404: User Not Found 500: Internal Server Error

Example Response:	`<WrapUpReasons>`
	`  <WrapUpReason>`
	`  <label>Successful tech support call</label>`
	`  <forAll>true</forAll>`
	`  <uri>/finesse/api/User/1234/WrapUpReason/12</uri>`
	`  </WrapUpReason>`
	`  ... more wrap-up reasons ...`
	`</WrapUpReasons>`

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

User—Get Media Properties Layout

This API allows a user to get a copy of the MediaProperties object. The MediaPropertiesLayout object determines how call variables and ECC variables appear on the Finesse desktop.

Note Finesse supports a single default instance of the MediaPropertiesLayout object.

For more information about the MediaPropertiesLayout object, see MediaPropertiesLayout, on page 152.

URI:	http://<FQDN>/finesse/api/User/<id>/MediaPropertiesLayout
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234/MediaPropertiesLayout
Security Constraints:	Agents and supervisors can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—

HTTP Response:	200: Success 401: Authorization Failure 500: Internal Server Error
Example 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>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>1234</ErrorData> </ApiError> </ApiErrors>

User—Get List of Phone Books

This API allows a user to get a list of phone books and associated contacts for that user.

For more information about the PhoneBook object, see PhoneBook, on page 158.

URI:	http://<FQDN>/finesse/api/User/<id>/PhoneBooks
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234/PhoneBooks
Security Constraints:	Agents and supervisors can use this API. Any signed-in user can get a list of phone books for that user.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request (the request body is invalid) 400: Finesse API Error (for example, the object does not exist or the object is stale) 401: Authorization Failure 404: User Not Found 500: Internal Server Error
Example Response:	<PhoneBooks> <PhoneBook> <name>PhoneBook1</name> <type>GLOBAL</type> <Contacts> <Contact> ...Full Contact Object... </Contact> ...Full Contact Object... </Contact> </Contacts> </PhoneBook> <PhoneBook> <name>PhoneBook2</name> <type>TEAM</type> <Contacts> <Contact> ...Full Contact Object... </Contact> <Contact> ...Full Contact Object... </Contact> </Contacts> </PhoneBook> </PhoneBooks>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> </ErrorData>1234</ErrorData> </ApiError> </ApiErrors>

User—Get List of Workflows

This API allows a user to get a list of workflows and workflow actions assigned to that user.

For more information about the Workflow object, see Workflow, on page 173.

URI:	http://<FQDN>/finesse/api/User/<id>/Workflows
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234/Workflows
Security Constraints:	Any user who is signed in to Finesse can access their own workflows.
HTTP Method:	GET

Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request (the request body is invalid) 400: Finesse API Error (for example, the object is stale or there is a violation of database constraints) 401: Authorization Failure 404: Not Found (the resource is not found) 500: Internal Server Error

Example 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>ACT_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>
```

Example Failure Response:

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

User API Parameters

Parameter	Type	Description	Possible Values	Notes
id	String	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, the size is determined by Unified Communications Manager.
uri	String	The URI to get a new copy of the object.	—	
roles	Collection	List of roles for this user.	Agent, Supervisor	
-->role	String	One of the roles assigned to this user.	Agent, Supervisor	
loginId	String	The login ID for this user.	—	
loginName	String	The login name for this user.	—	
state	String	The state for this user.	LOGOUT, NOT_READY, READY, RESERVED, RESERVED_OUTBOUND, RESERVED_OUTBOUND_PREVIEW, TALKING, HOLD, WORK, WORK_READY, UNKNOWN	
stateChangeTime	String	The time at which the state of the user changed to the current state. The format for this parameter is YYYY-MM-DDThh:MM:ss.SSSZ.	—	This parameter is empty if the time of the state change is not available (if no agent state change notification was received yet).

Parameter	Type	Description	Possible Values	Notes
pendingState	String	The state to which the user will transition next.	LOGOUT	For Unified CCE deployments, this parameter is empty. For Unified CCX deployments, when an agent is in TALKING state and a Finesse failover/reconnect occurs, this parameter is set to LOGOUT. The pendingState indicates that the agent transitions to LOGOUT when the call ends.
reasonCodeId	Integer	The database ID for the reason code that indicates why the user is in the current state.	If the user has not selected the reason code, this parameter is empty. Otherwise, the value of this parameter is the database ID for the selected reason code.	The value of the reasonCodeId may be -1 in the following cases: • No reason codes are configured for the category. • The agent has just signed in (transitioned from LOGIN to NOT_READY) • A failover occurred. The agent is in NOT_READY state but Finesse could not recover the reasonCode used before failover.

Parameter	Type	Description	Possible Values	Notes
ReasonCode	Collection	Information about the reason code currently associated with this user.	—	
-->category	String	The category of the reason code.	NOT_READY, LOGOUT	
-->uri	String	The full URI for the reason code.	—	
-->code	Integer	CTI code associated with this reason code.	—	
-->forAll	Boolean	Whether the reason code is global (true) or non-global (false).	true, false	
-->id	Integer	The ID of the reason code.	—	
-->label	String	The label associated with this reason code.	—	
settings	Collection	The settings for this user.	—	The settings parameter is only present for Unified CCE deployments.
-->wrapUpOn Incoming	String	Indicates whether this user required or allowed to enter wrap-up data on an incoming call.	REQUIRED, OPTIONAL, NOT_ALLOWED, REQUIRED_WITH_ WRAP_UP_DATA	This parameter applies only to Unified CCE deployments.

Parameter	Type	Description	Possible Values	Notes
extension	String	The extension that this user is currently using.	—	The extension must exist 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, the size is determined by Unified CCX.
mobileAgent	Collection	Indicates that the user is a mobile agent.	—	This parameter is returned for mobile agents only. Finesse supports mobile agents only in Unified CCE deployments.
-->mode	String	The work mode for the mobile agent	CALL_BY_CALL, NAILED_CONNECTION	This parameter is returned for mobile agents only. Finesse supports mobile agents only in Unified CCE deployments.
-->dialNumber	String	The external number that the system calls to connect to the mobile agent.	—	This parameter is returned for mobile agents only. Finesse supports mobile agents only in Unified CCE deployments. Validated by the Unified Communications Manager dial plan.
firstName	String	The first name of this user.	—	
lastName	String	The last name of this user.	—	

Parameter	Type	Description	Possible Values	Notes
teamId	String	The ID of the team to which this user belongs.	—	
teamName	String	The name of the team to which this user belongs.	—	
dialogs	String	URI to the collection of dialogs that the user is a part of.	—	
teams	Collection	If the user has a role of Supervisor, a list of teams that the user supervises.	—	
-->Team	Collection	Set of information for a team.	—	
--->uri	String	The URI to get a new copy of the Team object.	—	
--->id	String	The ID for the team.	—	
--->name	String	The name of the team.	—	

User API Errors

Status	Error Type	Description
400	Bad Request	The request is malformed or incomplete or the extension provided is invalid.
400	Generic Error	An unaccounted for error occurred. The root cause could not be determined.
400	Invalid Input	One of the parameters provided as part of the user input is invalid or not recognized (for example, the mode for a mobile agent or the state for a user)
400	Invalid State	The requested state change is not allowed (for example, a user in LOGOUT state requests a state change to LOGOUT or a supervisor tries to change an agent's state to something other than READY or LOGOUT).

Status	Error Type	Description
400	Parameter Missing	The extension, state, or requestedAction is not provided. If signing in a mobile agent, the mode or dialNumber is not provided.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (for example, an agent tries to use an API that only a supervisor or administrator is authorized to use).
401	Invalid Authorization User Specified	The authenticated user tried to make a request for another user.
401	Invalid State	A user tried to change to a state that is not supported in the scenario.
401	Invalid Supervisor	A supervisor tried to change the state of an agent who does not belong to that supervisor's team.
404	Not Found	The resource specified is invalid or does not exist.
404	User Not Found	The user ID provided is invalid or is not recongnized. No such user exists in CTI.
500	Internal Server Error	Any runtime exception is caught and responded with this error.
503	Service Unavailable	A dependent service is down (for example, the Cisco Finesse Notification Service or Cisco Finesse Database). Finesse is OUT_OF_SERVICE.

Dialog

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.

The Dialog object is structured as follows:

```
<Dialog>
   <uri>/finesse/api/Dialog/12345678</uri>
   <associatedDialogUri>/finesse/api/Dialog/321654</associatedDialogUri>
   <mediaType>Voice</mediaType>
   <state>ACTIVE</state>
   <fromAddress>2002</fromAddress>
   <toAddress>2000</toAddress>
   <mediaProperties>
      <dialedNumber>2000</dialedNumber>
      <callType>AGENT_INSIDE</callType>
      <DNIS>2000</DNIS>
```

```
      <wrapUpReason>Sales Call</wrapUpReason>
      <callvariables>
         <CallVariable>
            <name>callVariable1</name>
            <value>Chuck Smith</value>
         </CallVariable>
         </CallVariable>
            <name>callVariable2</name>
            <value>Cisco Systems, Inc.</value>
         ...Other CallVariables ...
      </callvariables>
   </mediaProperties>
   <participants>
      <Participant>
         <actions>
            <action>HOLD</action>
            <action>DROP</action>
         </actions>
         <mediaAddress>2002</mediaAddress>
         <mediaAddressType>AGENT_DEVICE</mediaAddressType>
         <startTime>2014-02-11T16:10:23.121Z</startTime>
         <state>ACTIVE</state>
         <stateCause></stateCause>
         <stateChangeTime>2014-02-11T16:10:23.121Z</stateChangeTime>
      </Participant>
      <Participant>
         <actions>
            <action>RETRIEVE</action>
            <action>DROP</action>
         </actions>
         <mediaAddress>2000</mediaAddress>
         <mediaAddressType>AGENT_DEVICE</mediaAddressType>
         <startTime>2014-02-11T16:10:23.121Z</startTime>
         <state>HELD</state>
         <stateCause></stateCause>
         <stateChangeTime>2014-02-11T16:10:36.543Z</stateChangeTime>
      </Participant>
   </participants>
   <scheduledCallbackInfo>
      <callbackTime>2014-03-07T14:30</callbackTime>
      <callbackNumber>9785551212</callbackNumber>
   </scheduledCallbackTime>
</Dialog>
```

Dialog APIs

Dialog—Get Dialog

This API allows a user to get a copy of a Dialog object.

URI:	http://<FQDN>/finesse/api/Dialog/<dialogId>
Example URI:	http://finesse1.xyz.com/finesse/api/Dialog/12345678
Security Constraints:	Agents and administrators can use this API. Agents can only get 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 401: Invalid Authorization 404: Not Found 500: Internal Server Error

Example 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 Call Variables (up to 10)
         <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>
   </mediaProperties>
   <participants>
      <Participant>
         <actions>
            <action>HOLD</action>
            <action>DROP</action>
         </actions>
         <mediaAddress>1081001</mediaAddress>
         <mediaAddressType>AGENT_DEVICE<mediaAddressType>
         <startTime>2014-02-04T15:33:16.653Z</startTime>
         <state>ACTIVE</state>
         <stateCause></stateCause>
         <stateChangeTime>2014-02-04T15:33:26.653Z</stateChangeTime>
      </Participant>
      <Participant>
         <actions>
            <action>RETRIEVE</action>
            <action>DROP</action>
         </actions>
         <mediaAddress>1081002</mediaAddress>
         <mediaAddressType>AGENT_DEVICE<mediaAddressType>
         <startTime>2014-02-04T15:33:16.653Z</startTime>
         <state>HELD</state>
         <stateCause></stateCause>
         <stateChangeTime>2014-02-04T15:33:27.584Z</stateChangeTime>
      </Participant>
   </participants>
</Dialog
```

Example Failure Response:

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

Dialog—Create a New Dialog (Make a Call)

This API allows a user to make a call. To make a call, a new Dialog object is created that specifies the fromAddress (the caller's extension) and the toAddress (the destination target). The new Dialog object is posted to the Dialog collection for that user.

In a Unified CCE deployment, you can also use this API to pass call variables with the MAKE_CALL request. The API supports call variable 1 through call variable 10 and ECC variables. You cannot pass BA variables or wrap-up reasons with the request.

Note In a Unified CCX deployment, you cannot use this API to pass call variables. If you supply the mediaProperties parameter with a MAKE_CALL request in a Unified CCX deployment, Finesse returns a 400 Invalid Input error.

URI:	http://<FQDN>/finesse/api/User/<id>/Dialogs
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234/Dialogs
Security Constraints:	All users can use this API. Users can only create dialogs using 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>1002002</toAddress> </Dialog>
HTTP Request with Call Variables (Unified CCE only):	<Dialog> <requestedAction>MAKE_CALL</requestedAction> <fromAddress>1001001</fromAddress> <toAddress>1002002</toAddress> <mediaProperties> <callvariables> <CallVariable> <name>callVariable1</name> <value>testcallvar1</value> </CallVariable> <CallVariable> <name>user.Finesse_ecc2</name> <value>A</value> </CallVariable> <CallVariable> <name>user.finesse_array[0]</name> <value>array_val_0</value> </CallVariable> <CallVariable> <name>user.finesse_array[1]</name> <value>array_val_1</value> </CallVariable> <CallVariable> <name>user.finesse_array[2]</name> <value>array_val_2</value> </CallVariable> <CallVariable> <name>user.finesse_array[3]</name> <value>array_val_3</value> </CallVariable> <CallVariable> <name>user.finesse_array[4]</name> <value>array_val_4</value> </CallVariable> </callvariables> </mediaProperties> </Dialog>
Request Parameters:	id (required): The ID of the user requested Action (required): The way in which the dialog is created (MAKE_CALL) fromAddress (required): The extension with which the user is currently signed in toAddress (required): The destination for the call mediaProperties (optional): Collection of media-specific properties related to the dialog callvariables (optional): Collection of call variables to include as part of the initial call CallVariable (optional): Name and value pair for a call variable

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 dialog notification. 400: Bad Request (the request body is invalid) 400: Parameter Missing 400: Invalid Input (a request in a Unified CCX deployment includes mediaProperties) 400: Invalid Destination (the toAddress and fromAddress are the same) 401: Authorization Failure 401: Invalid Authorization 500: Internal Server Error

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

Notifications Triggered:	Dialog notification

Dialog—Take Action on Participant

This API allows a user to take 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:	Agents can use this API. 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>
Request Parameters:	dialogId (required): The ID of the dialog targetMediaAddress (required): The extension with which the user is currently signed in (used to locate the participant to target with the action request) requestedAction (required): The action to take on the targeted participant
HTTP Response:	202: Successfully Accepted 400: Parameter Missing (the targetMediaAddress or requestedAction is not provided) 400: Invalid Input 401: Authorization Failure 401: Invalid Authorization User Specified 404: Dialog Not Found 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Invalid Input</ErrorType> <ErrorData>requestedAction</ErrorData> <ErrorMessage>Invalid 'requestedAction' specified for dialog</ErrorMessage> </ApiError> </ApiErrors>
Notifications Triggered:	Dialog notification Dialog CTI error notification (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>

Dialog—Update Call Variable Data

This API allows a user to set or change call variables (including named variables or ECC variables) on a dialog. If the user is an agent, the user must be the participant that they are targeting with the action. A corresponding notification is published if there is an update to any of the values of the call variables or named variables.

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:	Agents can use this API. 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>
Request Parameters:	dialogId (required): The ID of the dialog mediaProperties (required): Collection of media-specific properties related to the dialog to be modified wrapUpReason (optional): A description of the call callvariables (optional): A list of call variables to modify (either wrapUpReason or callvariables must be present in the request) CallVariable (required if the callvariables parameter is present): Contains the name and value of a call variable belonging to this dialog. The name must be present and cannot be empty. Duplicate names cannot exist. The value tag must be specified but can be empty.
HTTP Response:	202: Successfully Accepted 400: Parameter Missing 400: Invalid Input 401: Authorization Failure 401: Invalid Authorization User Specified 404: Dialog Not Found 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Notifications Triggered:	Dialog notification Dialog CTI error notification (if a CTI error occurs)

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. 2 The request payload contained no valid ECC or call variables.	The CTI server sends an error to Finesse.	Finesse forwards the error to the client.
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—Send DTMF String

This 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/54321
Security Constraints:	Agents can use this API. 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>`

Request Parameters:	dialogId (required): The ID of the dialog requestedAction (required): The way in which the dialog is created (SEND_DTMF) targetMediaAddress (required): The extension of the agent actionParams (required): A collection of objects called ActionParam, which contain name/value pairs. The name must be dtmfString. The value is the DTMF string to submit and can contain 0-9, *, #, or A-D for Unified CCE. For Unified CCX, the value can only contain 0-9, *, or #.
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 400: Invalid Input 401: Authorization Failure 401: Invalid Authorization User Specified 401: Invalid State (the targetMediaAddress specifies an extension of a participant in HELD state) 500: Internal Server Error

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

Notifications Triggered:	Dialog notification

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>

Dialog—Make a Consult Call Request

This API allows an agent to make a consult call request. After the request succeeds, the agent can complete the call as a conference or transfer. The requestedAction for a consult call is CONSULT_CALL. The request is sent to the Dialog URL of an existing active call, from where the call is initiated.

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 call. Finesse does not support blind conference through the API or the desktop.

Blind conference is defined as follows:

An agent has an active call and initiates a consult call to a destination. The agent starts a conference while the call is ringing at the destination.

Finesse does allow single-step transfer in Unified CCE deployments only. Finesse does not support single-step transfer in Unified CCX deployments.

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

Finesse maintains a copy of the call variables (including call peripheral variables and ECC variables) for each call in the system. When Unified CCE or Unified CCX sets the call variables to values that are not NULL (through CTI events, such as CALL_DATA_UPDATE_EVENT), the call variables maintained by Finesse are updated with these 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 values are set to empty strings, Finesse updates its version of the same call variables to empty strings and then updates the clients.

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/54321
Security Constraints:	Agents can use this API. An agent must be a participant in the dialog and the agent's extension must match the targetMediaAddress.
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>`

Request Parameters:	dialogId (required): The ID of the dialog requestedAction (required): The way in which the dialog is created (CONSULT_CALL) toAddress (required): The destination for the call targetMediaAddress (required): The extension of the agent, used to locate the participant to target with the requestedAction

HTTP Response:	202: Successfully Accepted 400: Parameter Missing 400: Invalid Input 401: Authorization Failure 401: Invalid Authorization User Specified 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Notifications Triggered:	Dialog notification Dialog CTI error notification (if a CTI error occurs)

Dialog—Initiate a Single Step Transfer

This API allows a user to make a single-step transfer request. After a user makes a successful request, that user's active call is transferred to the destination provided in the toAddress parameter.

The requestedAction for a single-step transfer 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 dialogId of the active call.

Note This API applies only to Unified CCE deployments. If you try 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/54321
Security Constraints:	Agents can use this API. An agent must be a participant in the dialog and the agent's extension must match the targetMediaAddress.
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>
Request Parameters:	dialogId (required): The ID of the dialog requestedAction (required): The way in which the dialog is created (TRANSFER_SST) toAddress (required): The destination to which to transfer the call targetMediaAddress (required): The extension of the agent who is making the request
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 400: Invalid Input 400: Invalid Destination 401: Authorization Failure 401: Invalid Authorization User Specified 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Notifications Triggered:	Dialog notification

Dialog—Make a Silent Monitor Call

This 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 (the supervisor's extension) and the toAddress (the agent's extension). The dialog is posted to the supervisor's dialog collection.

Note Agent phones 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:	Supervisors can use this API. A supervisor must be signed in to the fromAddress (extension) being used to create the silent monitor call. Agent to be monitored must be assigned to a team that the supervisor is responsible for. A supervisor can silently monitor any call except a silent monitor call. If an agent drops from or 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>1001002</fromAddress>`
	`  <toAddress>1001001</toAddress>`
	`</Dialog>`

Request Parameters:	id (required): The ID of the user requestedAction (required): The way in which the dialog is created (SILENT_MONITOR) fromAddress (required): The extension of the supervisor who initiated the silent monitor request toAddress (required): The extension of the agent that the supervisor wants to monitor

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 400: Invalid Input 400: Invalid Destination 400: Invalid State 401: Authorization Failure 401: Invalid Authorization User Specified 500: Internal Server Error

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

Notifications Triggered:	Dialog notification

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 or non-ICD calls (for example a call to another agent). 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.

Dialog—End a Silent Monitor Call

This API allows a supervisor to drop a silent monitor call that was initiated by that supervisor. The Dialog object is updated by specifying a requestedAction of DROP and the targetMediaAddress of the extension of the supervisor who initiated the silent monitor call.

URI:	http://<FQDN>/finesse/api/Dialog/<dialogId>
Example URI:	http://finesse1.xyz.com/finesse/api/Dialog/32458

Security Constraints:	Supervisors and administrators can use this API. A supervisor can only end a silent monitor call that was initiated by that supervisor. An administrator can end any silent monitor call.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

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

Request Parameters:	dialogId (required): The ID of the dialog requestedAction (required): The action to take on the targeted participant (DROP) targetMediaAddress (required): The extension of the supervisor who initiated the silent monitor call
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 400: Invalid Input 401: Authorization Failure 401: Invalid Authorization User Specified 404: Not Found (the dialog specified by the dialogId does not exist) 500: Internal Server Error

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

Notifications Triggered:	Dialog notification

Dialog—Make a Barge Call

This API allows a supervisor to barge in to an agent call that the supervisor is silently monitoring. The request specifies the fromAddress (supervisor's extension), the toAddress (agent's extension), and the associatedDialog (the URI of the silent monitor dialog that the supervisor initiated). When the barge request succeeds, the agent's original Dialog object is updated and is posted to the supervisor's dialog collection. The supervisor's

silent monitor call is dropped. After the barge request succeeds, the original silent monitor call becomes a conference call with the supervisor, agent, and caller as participants.

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 original call is not a conference call, after the barge request succeeds, the call becomes a conference call and the agent is the conference controller. If the original call is a conference call and 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:	Supervisors can use this API. Supervisors can only make barge call requests using the fromAddress that they are currently signed 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>1001002</fromAddress>`
	`<toAddress>1001001</toAddress>`
	`<associatedDialogUri>/finesse/api/Dialog/`
	`6873122</associatedDialogUri>`
	`</Dialog>`

Request Parameters:	requestedAction (required): The way in which to create the dialog (BARGE_CALL) fromAddress (required): The extension of the supervisor who initiated the barge request toAddress (required): The extension of the agent whose call the supervisor wants to barge in on associatedDialogUri (required): The relative URI of the silent monitor dialog on which the supervisor wants to barge in

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 400: Invalid Input 400: Invalid Destination 400: Invalid State 400: 20700 (Conference resource limit violation) 400: 20999 (Barge via non-conference-controller or the agent already has an outstanding consult call) 401: Authorization Failure 401: Invalid Authorization User Specified 500: Internal Server Error

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

Notifications Triggered:	Dialog notification

Platform-Based API Differences

Stand-alone Finesse with Unified CCE:

A supervisor must be silently monitoring a call before making a request to barge in to 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.

Dialog—End a Barge Call

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

The agent can remain on the call unless the total number of participants 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:	Supervisors can use this API. A supervisor can only drop barge call if that 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>`

Request Parameters:	requestedAction (required): The way in which to create the dialog (DROP) targetMediaAddress (required): The extension of the supervisor who initiated the barge call
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 400: Invalid Input 401: Authorization Failure 401: Invalid Authorization User Specified 404: Not Found (the dialog specified by the dialogId does not exist) 500: Internal Server Error

Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Notifications Triggered:	Dialog notification

Dialog—Drop Participant from Conference

This API allows a supervisor to make a request to drop a participant from a conference in which that supervisor is a participant. 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 specified by the dialogId in the URI.

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

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

If wrap-up is enabled for the agent who is dropped, that agent can perform wrap-up after being dropped.

URI:	http://<FQDN>/finesse/api/Dialog/<dialogId>
Example URI:	http://finesse1.xyz.com/finesse/api/Dialog/54321
Security Constraints:	Supervisors and administrators can use this API. A supervisor can only make a drop request for a conference call if the supervisor is a participant in the call.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	<Dialog> <requestedAction>PARTICIPANT_DROP</requestedAction> <targetMediaAddress>1001001</targetMediaAddress> </Dialog>
Request Parameters:	requestedAction (required): The way in which to create the dialog (PARTICIPANT_DROP) targetMediaAddress (required): The extension of the agent to drop from the conference call

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 400: Invalid Input 400: Invalid Destination (the targetMediaAddress is not one of the parties in the dialog or is not an agent extension) 400: Invalid State (the dialog is not a conference call) 401: Authorization Failure 401: Invalid Authorization User Specified 500: Internal Server Error

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

Notifications Triggered:	Dialog notification

Dialog—Start Recording

This API allows a user to start recording an active call.

Note This API applies to Unified CCX deployments 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/54321
Security Constraints:	Agents and supervisors can use this API. A user must be a participant in the call to perform this action. An agent cannot record the call of another agent. 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. A supervisor can only silently monitor (and therefore record) agents who belong to teams assigned to that supervisor.
HTTP Method:	PUT

Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	<Dialog> <requestedAction>START_RECORDING</requestedAction> <targetMediaAddress>1001001</targetMediaAddress> </Dialog>
Request Parameters:	requestedAction (required): The way in which to create the dialog (START_RECORDING) targetMediaAddress (required): The extension of the agent whose call to record
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 400: Invalid Input 401: Authorization Failure 401: Invalid Authorization User Specified 401: Invalid State (the targetMediaAddress specifies an extension of a participant in HELD state) 500: Internal Server Error 501: Not Implemented (a recording attempt was made in a Unified CCE deployment)
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Notifications Triggered:	Dialog notification

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

This 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 notification of type OUTBOUND_PREVIEW to the reserved user.

Note This API applies to 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/54321
Security Constraints:	Agents can use this API. 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>`

Request Parameters:	dialogId (required): The ID of the dialog requestedAction (required): The action to take on the Outbound Option Preview reservation (ACCEPT, CLOSE, or REJECT) targetMediaAddress (required): The extension of the agent
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 400: Invalid Input 401: Authorization Failure 401: Invalid Authorization User Specified 404: Dialog Not Found 500: Internal Server Error 501: Not Implemented

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

Notifications Triggered:	Dialog notification

Dialog—Accept, Close, or Reject a Direct Preview Outbound Reservation

This API allows a user to accept, close, or reject an Direct Preview Outbound reservation . Finesse signals a Direct Preview reservation by posting a dialog notification of type OUTBOUND_PREVIEW to the reserved user.

Note This API applies to 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/54321
Security Constraints:	Agents can use this API. 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>`

Request Parameters:	dialogId (required): The ID of the dialog requestedAction (required): The action to take on the Direct Preview reservation (ACCEPT, CLOSE, or REJECT) targetMediaAddress (required): The extension of the agent
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 400: Invalid Input 401: Authorization Failure 401: Invalid Authorization User Specified 404: Dialog Not Found 500: Internal Server Error 501: Not Implemented

Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Notifications Triggered:	Dialog notification

Dialog—Reclassify a Direct Preview Call

This API allows a user to reclassify an Outbound Option Direct Preview call. A call can be reclassified as VOICE, FAX, ANS_MACHINE, INVALID, DO_NOT_CALL, or BUSY. The call type is then sent back to Unified CCX for processing.

Note This API applies to Unified CCX only. You cannot use this API in a Finesse deployment with Unified CCE.

URI:	http://<FQDN>/finesse/api/Dialog/<dialogId>
Example URI:	http://finesse1.xyz.com/finesse/api/Dialog/54321
Security Constraints:	Agents can use this API. Agents can only act on their own Dialog object.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<Dialog> <requestedAction>RECLASSIFY</requestedAction> <targetMediaAddress>1001001</targetMediaAddress> <actionParams> <ActionParam> <name>outboundClassification</name> <value>FAX</value> </ActionParam> </actionParams> </Dialog>
Request Parameters:	dialogId (required): The ID of the dialog requestedAction (required): The action to perform (RECLASSIFY) targetMediaAddress (required): The extension of the agent who is making the request actionParams (required): A collection of objects called ActionParam, which contain name/value pairs. The name must be outboundClassification. The value can be VOICE, FAX, ANS_MACHINE, INVALID, DO_NOT_CALL, or BUSY. A single parameter must be specified for the value. Any additional parameters are ignored.
HTTP Response:	202: Successfully Accepted **Note** This response only indicates a successful completion of the request. The request is processed asynchronously and the state change is sent as part of and updated to the Dialog object. The response is in the BAResponse call variable, which contains the value sent to the CTI server for the callback action. No confirmation is returned, other than the value in the BAResponse. 400: Bad Request 400: Finesse API Error (for example, the object does not exist or is stale) 400: Parameter Missing 401: Authorization Failure 401: Invalid Authorization User Specified 404: Dialog Not Found 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Notifications Triggered:	Dialog notification

Dialog—Schedule or Cancel a Callback

This API allows a user to schedule or cancel a callback. The dialog action UPDATE_SCHEDULED_CALLBACK is used to schedule or update a callback. The dialog action CANCEL_SCHEDULED_CALLBACK is used to cancel a previously scheduled callback.

URI:	http://<FQDN>/finesse/api/Dialog/<dialogId>
Example URI:	http://finesse1.xyz.com/finesse/api/Dialog/54321
Security Constraints:	Agents can use this API. Agents can only act on their own Dialog object.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request (Update Scheduled Callback):	`<Dialog>`
	`<requestedAction>UPDATE_SCHEDULED_CALLBACK</requestedAction>`
	`<targetMediaAddress>1001001</targetMediaAddress>`
	`<actionParams>`
	`<ActionParam>`
	`<name>callbackTime</name>`
	`<value>2013-12-07T14:30</value>`
	`</ActionParam>`
	`</actionParams>`
	`</Dialog>`

HTTP Request (Cancel Scheduled Callback):	`<Dialog>`
	`<requestedAction>CANCEL_SCHEDULED_CALLBACK</requestedAction>`
	`<targetMediaAddress>100100</targetMediaAddress>`
	`</Dialog>`

Request Parameters:	dialogId (required): The ID of the dialog requestedAction (required): The action to perform (UPDATE_SCHEDULED_CALLBACK, CANCEL_SCHEDULED_CALLBACK) targetMediaAddress (required): The extension of the agent who is making the request actionParams (required): A collection of objects called ActionParam, which contain name/value pairs. The name must be UPDATE_SCHEDULED_CALLBACK. The value can be callbackTime or callbackNumber. A single parameter must be specified for the value. Any additional parameters are ignored.
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 400: Invalid Input 401: Authorization Failure 401: Invalid Authorization User Specified 404: Dialog Not Found 500: Internal Server Error 501: Not Implemented

Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>
Notifications Triggered:	Dialog notification

Dialog API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the object.	—	
associatedDialogUri	String	The URI to a Dialog object that is associated with this Dialog object	/finesse/api/Dialog/dialogId	
mediaType	String	The type of media under which this dialog is classified.	Voice, Email, Chat	
state	String	The last state of this dialog.	For a list of possible values, see State (Dialog) Parameter Values, on page 91.	
fromAddress	String	The calling line ID of the caller.	—	
toAddress	String	The destination for the call.	—	
mediaProperties	Collection	A collection of media-specific properties for the dialog.	—	
-->dialedNumber	String	The number dialed.	—	

Parameter	Type	Description	Possible Values	Notes
-->callType	String	The type of call.	ACD_IN, PREROUTE_ACD_IN, PREROUTE_DIRECT_AGENT, TRANSFER, OTHER_IN, OUT, AGENT_INSIDE, CONSULT, CONFERENCE, SUPERVISOR_MONITOR, OUTBOUND, OUTBOUND_PREVIEW, OUTBOUND_CALLBACK, OUTBOUND_CALLBACK_PREVIEW, OUTBOUND_PERSONAL_CALLBACK, OUTBOUND_PERSONAL_CALLBACK_PREVIEW, OUTBOUND_DIRECT_PREVIEW	
-->DNIS	String	The DNIS provided with the call. For routed calls, the DNIS is the route point.	—	
-->wrapUpReason	String	A description of the call.	—	The maximum size of this parameter is 39 bytes (which equals 39 US English characters).
-->callVariables	Collection	A list of call variables associated with this dialog.	—	

Parameter	Type	Description	Possible Values	Notes
--->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 (for example, user.myarray[2]). Outbound Option call variables provide additional details about an Outbound Option call.	callvariable1 through callvariable10 ECC variables The following Outbound variables: • BACampaign • BAAccountNumber • BAResponse • BAStatus • BADialedListID • BATimeZone • BABuddyName • BACustomerNumber (Unified CCX only) For information about possible values for BAStatus, see Outbound Call Types and BAStatus, on page 104.	Size: • Call variable: 40 bytes • ECC/named variable: Sum of all names, values, and index (if array) must be less than or equal to 2000 bytes. Each ECC variable value cannot exceed the length defined in the CTI server administration user interface.
participants	Collection	A list of all participants (both internal and external) involved in the dialog.	—	
-->Participant	Collection	Information about one participant in the dialog.	—	
--->actions	Collection	A list of actions that are allowed for a participant.	For a list of possible values, see Actions Parameter Values, on page 92.	
--->mediaAddress	String	Point of contact for the participant.	Possible values include the extension of an agent or ANI for a caller who are participants in the call.	
--->mediaAddress Type	Collection	The device type specified by the mediaAddress.	AGENT_DEVICE or empty string	

Parameter	Type	Description	Possible Values	Notes
--->startTime	String	The UTC time when the participant initiated the call or the first time the participant call state becomes active. Finesse uses the Finesse server timestamp (not the CTI even timestamp) to determine the startTime. A time difference may exist between the Finesse server on side A and side B. Although they are synchronized using an NTP server, a few milliseconds of drift may exist. Therefore, the startTime may be different for a participant if Finesse fails over from side A to side B.	The start time in the format YYYY-MM-DDThh:MM:ss.SSSZ or an empty string	

Parameter	Type	Description	Possible Values	Notes
				When an agent signs in with an extension that has an active call, Finesse does not have a call object tracking the call and sets the startTime for this participant as an empty string. If the call does have a participant who is an agent, Finesse can reuse the call object for the extension and the startTime is available For example, if an agent is on a call with a customer and then signs in, Finesse does not have the call object. If the agent is on a call with another agent and then signs in, Finesse can reuse the call object for the extension. In a Unified CCE deployment, Finesse on side B is in standby and keeps track of agent states and calls. When failover occurs, Finesse can recover the startTime for the agent. In a Unified CCX deployment, Finesse on side B does not have the

Parameter	Type	Description	Possible Values	Notes
				agent state or call information. After failover occurs, Finesse sets the startTime parameter as an empty string.
--->state	String	The last participant state in a dialog.	For a list of possible values, see State (Participant) Parameter Values, on page 95.	
--->stateCause	String	The cause for the last participant state in a dialog.	BUSY, BAD_DESTINATION, OTHER	This parameter is normally associated with a FAILED participant state.

Parameter	Type	Description	Possible Values	Notes
--->stateChange Time	String	The UTC time when the participant changed to the current state. Finesse uses the Finesse server timestamp (not the CTI even timestamp) to determine the stateChangeTime. A time difference may exist between the Finesse server on side A and side B. Although they are synchronized using an NTP server, a few milliseconds of drift may exist. Therefore, the stateChangeTime may be different for a participant if Finesse fails over from side A to side B.	The state change time in the format YYYY-MM-DDThh:MM:ss.SSSZ or an empty string	When Finesse cannot determine the stateChangeTime, this parameter is an empty string. For example, if a participant is in HELD state and a failover occurs, after failover, Finesse can determine that the participant is in HELD state but cannot determine when the call was put on hold. Therefore, Finesse sets the stateChangeTime parameter to an empty string. In a Unified CCE deployment, Finesse on side B is in standby and keeps track of agent states and calls. When failover occurs, Finesse can recover the stateChangeTime for the agent. In a Unified CCX deployment, Finesse on side B does not have the agent state or call information. After failover occurs, Finesse sets the stateChangeTime parameter as an empty string.

Parameter	Type	Description	Possible Values	Notes
scheduled CallbackInfo	Collection	For Outbound Option campaigns, provides information about scheduled callbacks.	—	This parameters is provided only if a callback is scheduled for this dialog.
-->callbackTime	String	The callback time in the format YYYY-MM-DDThh:MM (for example, 2013-12-15T11:45). The time is in the customer's timezone. Optionally, a full ISO-8601 format time string (ex. 2013-12-25T23:59:59 .9999999+03:00) can be sent, but everything beyond the minutes, including the time zone, is ignored.	—	This parameter is provided only if a callback time has been set. Value returned in the BAReponse: Callback MMDDYYYY HH:MM (for example, Callback 12072013 14:30)
-->callbackNumber	String	The phone number to call for the callback.	—	This parameter is provided only if a callback number has been set. Value returned in the BAResponse: P#<callbackNumber> (for example, P#9780001)

State (Dialog) Parameter Values

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

Dialog State	Description
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	Allows an agent to make an outgoing call
ANSWER	Answer	Allows an agent to answer an incoming call
HOLD	Hold	Allows an agent to hold a call that is currently active
RETRIEVE	Retrieve	Allows an agent to retrieve a call that was on hold
DROP	End	Allows an agent to drop the participant of a call
UPDATE_CALL_DATA	—	Allows an 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	—	Allows an agent to send DTMF digits for the call
CONSULT_CALL	Consult	Allows an agent to make a consult call for transfer or conference
CONFERENCE	Conference	Allows an agent to start a conference between the selected held call and the existing active call on the desktop

Participant Allowable Action	Enabled Button on Desktop	Description
TRANSFER	Transfer	Allows an agent to complete a transfer between the selected held call and the existing active call on the desktop
TRANSFER_SST	Direct Transfer	Allows an agent to initiate a single-step transfer (Unified CCE only)
SILENT_MONITOR	Start Monitoring	Allows a supervisor to silent monitor an agent who is in TALKING state on an active call
BARGE_CALL	Barge In	Allows a supervisor to barge in on an agent call that the supervisor is silently monitoring
PARTICIPANT_DROP	Drop	Allows a supervisor to drop a participant from a conference call
START_RECORDING	Start Recording	Allows an agent to start a recording (Unified CCX only, requires integration with MediaSense)
UPDATE_SCHEDULED_CALLBACK	Callback, Schedule	Allows an agent to update the details for a scheduled callback
CANCEL_SCHEDULED_CALLBACK	Callback, Cancel	Allows an agent to cancel a scheduled callback

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.

Outbound Option Direct Preview Actions

The following table describes the actions available to an agent who is reserved in an Outbound Option Direct 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 CCX.

For more information about the BAResponse variable, see the section "Outbound Option Extended Call Variables" in the *Cisco Unified Contact Center Express CTI Protocol Developer Guide*.

Action	BAResponse Value	Description
ACCEPT	Accept	Performing the ACCEPT action while reserved in an Outbound Option Direct Preview campaign instructs Unified CCX to establish a call with the customer.
CLOSE	Reject-Close	Performing the CLOSE action while reserved in an Outbound Option Direct Preview campaign rejects the current preview call and prevents the number from being called again in the campaign.

Action	BAResponse Value	Description
REJECT	Reject	Performing the REJECT action while reserved in an Outbound Option Direct Preview campaign instructs Unified CCX to retry the previewed number at a later time.
RECLASSIFY	Reclassify	Performing the RECLASSIFY action while reserved in an Outbound Option Direct Preview campaign instructs Unified CCX to reclassify the previewed number as voice (successful case), a modem/fax, answering machine, an invalid number, do not call, or busy.

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)
FAILED	DROP	Error	Indicates that the call failed (BAD_DESINATION)
FAILED	DROP	Error	Indicates that the call failed (OTHER)

Participant State	Allowable Actions for the Participant State	Call State on Finesse Desktop	Description
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

Scenario	CTI Event	Event Method	Dialog State	Participant State (Caller)	Participant State (Recipient)
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 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
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

Scenario	CTI Event (Original Call)	CTI Event (Consult Call)	Event Method	Dialog State	Participant State
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
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	-	-	-	-	-	-	-
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)

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)
-	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)

Scenario	CTI Event	Event Method	Dialog State	Participant State (Caller)	Participant State (Agent A)	Participant State (Supervisor)
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
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

Scenario	CTI Event	Event Method	Dialog State	Participant State (Caller)	Participant State (Agent A)	Participant State (Supervisor)
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.

Outbound Call Types and BAStatus

The following tables list the call types for outbound calls and the associated values for BAStatus for Unified CCE deployments and Unified CCX deployments.

Note When a user transfers or conferences an outbound call, the callType changes to TRANSFER or CONFERENCE.

In Unified CCE deployments, the BAStatus of the call remains unchanged. In Unified CCX deployments, the BAStatus changes to TRANSFERRED or CONFERENCED for Progressive and Predictive outbound calls and remains OUTBOUND for Direct Preview outbound calls.

When failover occurs in a Unified CCE deployment, the callType and BAStatus remain unchanged. In Unified CCX deployments, the callType parameter is null or empty after failover for all outbound dialing modes. The BAStatus parameter is removed as the call no longer functions as an outbound call.

Table 7: Outbound Call Types and BAStatus for Finesse with Unified CCE

	Progressive	Predictive	Preview
Reservation Call	—	—	callType: OUTBOUND_PREVIEW BAStatus: PREVIEW_OUTBOUND_RESERVATION
Customer Call	callType: OUTBOUND BAStatus: PROGRESSIVE_OUTBOUND	callType: OUTBOUND BAStatus: PREDICTIVE_OUTBOUND	callType: OUTBOUND BAStatus: PREVIEW_OUTBOUND
Callback Reservation Call	—	—	callType: OUTBOUND_CALLBACK_PREVIEW BAStatus: PREVIEW_OUTBOUND_RESERVATION
Callback Customer Call	callType: OUTBOUND_CALLBACK BAStatus: PROGRESSIVE_OUTBOUND	callType: OUTBOUND_CALLBACK BAStatus: PREDICTIVE_OUTBOUND	callType: OUTBOUND_CALLBACK BAStatus: PREVIEW_OUTBOUND
Personal Callback Reservation Call	callType: OUTBOUND_PERSONAL_CALLBACK_PREVIEW BAStatus: PERSONAL_CALLBACK_OUTBOUND_RESERVATION	callType: OUTBOUND_PERSONAL_CALLBACK_PREVIEW BAStatus: PERSONAL_CALLBACK_OUTBOUND_RESERVATION	callType: OUTBOUND_PERSONAL_CALLBACK_PREVIEW BAStatus: PERSONAL_CALLBACK_OUTBOUND_RESERVATION
Personal Callback Customer Call	callType: OUTBOUND_PERSONAL_CALLBACK BAStatus: PERSONAL_CALLBACK_OUTBOUND	callType: OUTBOUND_PERSONAL_CALLBACK BAStatus: PERSONAL_CALLBACK_OUTBOUND	callType: OUTBOUND_PERSONAL_CALLBACK BAStatus: PERSONAL_CALLBACK_OUTBOUND

Table 8: Outbound Call Types and BAStatus for Finesse with Unified CCX

	Progressive	Predictive	Direct Preview
Reservation Call	—	—	callType: OUTBOUND_DIRECT_PREVIEW BAStatus: DIRECT_PREVIEW_OUTBOUND_RESERVATION
Customer Call	callType: OUTBOUND BAStatus: OUTBOUND	callType: OUTBOUND BAStatus: OUTBOUND	callType: OUTBOUND BAStatus: DIRECT_PREVIEW_OUTBOUND
Callback Reservation Call	—	—	callType: OUTBOUND_DIRECT_PREVIEW BAStatus: DIRECT_PREVIEW_OUTBOUND_RESERVATION
Callback Customer Call	callType: OUTBOUND_CALLBACK BAStatus: OUTBOUND	callType: OUTBOUND_CALLBACK BAStatus: OUTBOUND	callType: OUTBOUND BAStatus: DIRECT_PREVIEW_OUTBOUND
Personal Callback Reservation Call	—	—	—
Personal Callback Customer Call	—	—	—

Dialog API 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.

Status	Error Type	Description
400	20999 (Barge via a non-conference-controller)	The agent specified in the toAddress is not the controller of the conference call or the agent already has an outstanding conference 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 (if users attempt to call their own extension). For the Dialog—Drop Participant from Conference API, this error occurs if the targetMediaAddress is not one of the parties on the call or is not an agent extension. For the Dialog—Make a Barge Call API, this error occurs if the supervisor tries to barge in on an agent call when the agent's extension is in HELD state.
400	Invalid Input	One of the parameters provided as part of the user input is invalid or not recognized (for example, the fromAddress, toAddress, targetMediaAddress, requestedAction). For the Dialog—Update Call Variable Data 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 Option variables: BACampaign, BAAccountNumber, BAResponse, BAStatus, BADialedListID, BATimeZone, BABuddyName, BACustomerNumber (Unified CCX only).
400	Invalid State	A supervisor who is already on an active call (in TALKING or HOLD state) makes a silent monitor request.
400	Parameter Missing	A required parameter was not provided in the request. For example, if creating a dialog, the fromAddess or toAddress was not provided.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (for example, an agent tries to use an API that only a supervisor or administrator is authorized to use).

Status	Error Type	Description
401	Invalid Authorization User Specified	The authenticated user tried to make a request for another user. The authenticated user tried to use a fromAddress that does not belong to that user.
401	Invalid State	The targetMediaAddress in a Dialog—Start Recording request specifies an extension of a participant in HELD state.
401	Invalid Supervisor	A supervisor tried to change the state of an agent who does not belong to that supervisor's team.
404	Not Found	The resource specified is invalid or does not exist.
404	Dialog Not Found	The dialogId provided is invalid or no such dialog exists.
500	Internal Server Error	Any runtime exception is caught and responded with this error.
501	Not Implemented	A user attempted to use the API in a deployment where it is not supported. For example, a recording attempt was made in a Unified CCE deployment.

Queue

The Queue object represents a queue (or 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.

The Queue object is structured as follows:

```
<Queue>
   <uri>/finesse/api/Queue/10</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>2</agentsTalkingOutbound>
      <agentsTalkingInternal>1</agentsTalkingInternal>
      <agentsWrapUpNotReady>2</agentsWrapUpNotReady>
      <agentsWrapUpReady>3</agentsWrapUpReady>
   </statistics>
</Queue>
```

Queue APIs

Queue—Get Queue

This API allows a user to get a Queue object. Use this API to access statistics for a queue that is assigned to agents or supervisors.

If you use this API to get a queue that is not assigned to any users, 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
Security Constraints:	Any user can use this API to retrieve information about a specific queue. The user does not need to belong to that queue.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 404: Not Found 500: Internal Server Error
Example Response:	<Queue> <uri>/finesse/api/Queue/10</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>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

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.

Queue—Get List of Queues for User

This API allows a user to get a list of all queues associated with that user.

Note The list of queues does not include the system-defined queue (skill group) present in 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
Security Constraints:	All users can use this API to retrieve a list of queues for any user.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 404: User Not Found 500: Internal Server Error

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

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.

Queue API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the Queue object.	—	

Parameter	Type	Description	Possible Values	Notes
id	String	The unique identifier for the queue.	—	
name	String	The name of the queue.	—	
statistics	Collection	A list of statistics for the queue.	—	
-->callsInQueue	Integer	The number of calls currently queued to this queue.	—	If the queue is not assigned to an agent or supervisor, this value is -1.
-->startTimeOf LongestCallInQueue	String	The start time of the longest call in the queue. The format for this parameter is YYYY-MM-DDThh:MM:ssZ.	—	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.	—	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.	—	If the queue is not assigned to an agent or supervisor, this value is -1.
-->agentsTalking Inbound	Integer	The number of agents assigned to the queue who are in TALKING state on inbound calls.	—	If the queue is not assigned to an agent or supervisor, this value is -1.

Parameter	Type	Description	Possible Values	Notes
-->agentsTalkingOutbound	Integer	The number of agents assigned to the queue who are in TALKING state on outbound calls.	—	If the queue is not assigned to an agent or supervisor, this value is -1. Outbound calls include non-routed calls placed to external devices that are not monitored by Unified Communications Manager or to devices in a different Unified Communications Manager cluster. Outbound Dialer calls are not included.
-->agentsTalkingInternal	Integer	The full URI for the reason code.	—	If the queue is not assigned to an agent or supervisor, this value is -1.
-->agentsWrapUpNotReady	Integer	CTI code associated with this reason code.	—	If the queue is not assigned to an agent or supervisor, this value is -1.
-->agentsWrapUpReady	Integer	The label associated with this reason code.	—	If the queue is not assigned to an agent or supervisor, this value is -1.

Queue API Errors

Status	Error Type	Description
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session).
404	Not Found	The resource specified is invalid or does not exist.
404	User Not Found	The user ID provided is invalid or is not recongnized. No such user exists in CTI.

Status	Error Type	Description
500	Internal Server Error	Any runtime exception is caught and responded with this error.

Team

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

The Team object does not contain a full User object for each of the team's users, but a summary object that contains the User uri, loginId, firstName, lastName, ReasonCode, and extension parameters. For more information about these parameters, see User API Parameters, on page 49.

The Team object is structured as follows:

```
<Team>
   <uri>/finesse/api/Team/34</uri>
   <id>34</id>
   <name>My Team</name>
   <users>
      <User>
         <uri>/finesse/api/User/1234/</uri>
         <loginId>1234</loginId>
         <firstName>Charles</firstName>
         <lastName>Brown</lastName>
         <dialogs>/finesse/api/User/1234/Dialogs</dialogs>
         <extension>1001001</extension>
         <pendingState></pendingState>
         <state>LOGOUT</state>
         <stateChangeTime>2012-03-01T17:58:21.345Z</stateChangeTime>
      </User>
      <User>
         <uri>/finesse/api/User/1235/</uri>
         <loginId>1235</loginId>
         <firstName>Jack</firstName>
         <lastName>Brawn</lastName>
         <dialogs>/finesse/api/User/1235/Dialogs</dialogs>
         <extension>1001002</extension>
         <pendingState></pendingState>
         <state>NOT_READY</state>
         <reasonCode>
            <category>NOT_READY</category>
            <code>12</code>
            <label>Lunch Break</label>
            <id>1</id>
            <uri>/finesse/api/ReasonCode/1</uri>
         </reasonCode>
         <stateChangeTime>2012-03-01T18:22:25.123Z</stateChangeTime>
      </User>
      ...Other Users...
   </users>
</Team>
```

Team APIs

Team—Get Team

This API allows a user to get a copy of the Team object. The Team object contains the configuration information for a specific team, which includes the URI, the team ID, the team name, and a list of agents who are members of that team.

URI:	http://<FQDN>/finesse/api/Team/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/Team/10
Security Constraints:	Supervisors can use this API to get a list of users assigned to their team.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 404: Not Found 500: Internal Server Error

Example Response:	<Team> <uri>/finesse/api/Team/34</uri> <id>34</id> <name>My Team</name> <users> <User> <uri>/finesse/api/User/1234/</uri> <loginId>1234</loginId> <firstName>Charles</firstName> <lastName>Brown</lastName> <dialogs>/finesse/api/User/1234/Dialogs</dialogs> <extension>1001001</extension> <pendingState></pendingState> <state>LOGOUT</state> <stateChangeTime>2012-03-01T17:58:21.345Z</stateChangeTime> </User> <User> <uri>/finesse/api/User/1235/</uri> <loginId>1235</loginId> <firstName>Jack</firstName> <lastName>Brawn</lastName> <dialogs>/finesse/api/User/1235/Dialogs</dialogs> <extension>1001002</extension> <pendingState></pendingState> <state>NOT_READY</state> <reasonCode> <category>NOT_READY</category> <code>12</code> <label>Lunch Break</label> <id>1</id> <uri>/finesse/api/ReasonCode/1</uri> </reasonCode> <stateChangeTime>2012-03-01T18:22:25.123Z</stateChangeTime> </User> ...Other Users... </users> </Team>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

Team API Parameters

Parameter	Type	Description	Possible Values	Notes
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.	—	

Parameter	Type	Description	Possible Values	Notes
users	Collection	The list of users that belong to this team.	—	
-->User	Collection	Information about one specific user on the team.	—	The Team object contains a subset of the User parameters. These parameters include the uri, loginId, firstName, lastName, dialogs, pendingState, state, stateChangeTime, extension, and ReasonCode. For information about these parameters, see User API Parameters, on page 49.

Team API Errors

Status	Error Type	Description
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session).
404	Not Found	The team id is invalid. No such team exists.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

ClientLog

The ClientLog object is a container element that holds client log data to post to the Finesse server. This object supports a POST operation only.

The ClientLog object is structured as follows:

```
<ClientLog>
   <logData>
      ...client logs...
   </logData>
</ClientLog>
```

ClientLog—Post to Finesse

This 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.

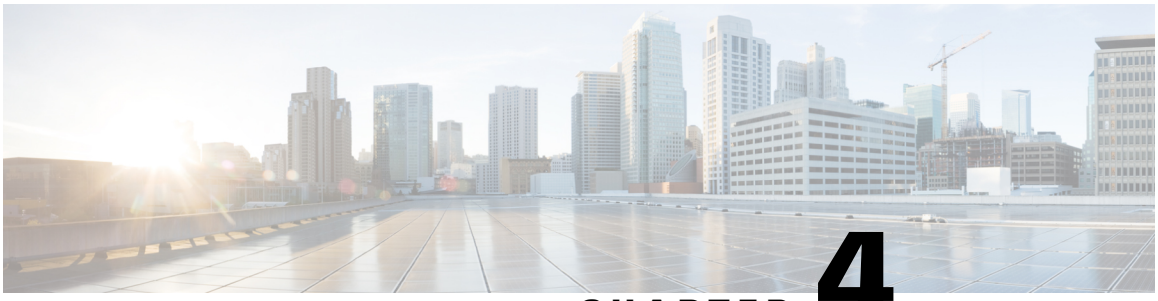
URI:	http://<FQDN>/finesse/api/User/<id>/ClientLog
Example URI:	http://finesse1.xyz.com/finesse/api/User/1234/ClientLog
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	<ClientLog> <logData> xxxxxxxxxxxxxxx\n xxxxxxxxxxxxxxx\n </logData> </ClientLog>
Request Parameters:	id (required): The ID of the user logData (required): The log data that the client sends to the server
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 CLIENT_LOG_EVENT that contains empty data elements and a matching requestId. 400: Parameter Missing 400: Invalid Input 400: Operation Failure 401: Authorization Failure 401: Invalid Authorization User Specified 405: Method Not Available
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>User Not Found</ErrorType> <ErrorMessage>UNKNOWN_USER</ErrorMessage> <ErrorData>4023</ErrorData> </ApiError> </ApiErrors>

ClientLog API Parameters

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

ClientLog API Errors

Status	Error Type	Description
400	Parameter Missing	The logData parameter is not present.
400	Invalid Input	The size of the logData exceeds 1,048,576 characters.
400	Operation Failure	The POST client log operation failed.
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 user.
405	Method Not Allowed	GET or PUT HTTP method not allowed for client-side log collection.



CHAPTER 4

Cisco Finesse Configuration APIs

Administrators use the Cisco Finesse configuration APIs to configure the following:

- System, cluster, and database settings
- Finesse desktop and call variable layout
- Reason codes and wrap-up reasons
- Phonebooks and contacts
- Team resources
- Workflows and workflow actions

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

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

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.

SystemConfig

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

Note SystemConfig APIs apply only to Finesse deployments with Unified CCE. Because you need not configure these settings for Finesse with Unified CCX, these APIs are not supported for deployments with Unified CCX.

The SystemConfig object is structured as follows:

```
<SystemConfig>
   <uri>/finesse/api/SystemConfig</uri>
   <cti>
      <host></host>
      <port></port>
      <backupHost></backupHost>
      <backupPort></backupPort>
      <peripheralId></peripheralId>
   </cti>
</SystemConfig>
```

SystemConfig APIs

SystemConfig—Get

This API allows an administrator to get a copy of the SystemConfig object.

URI:	http://<FQDN>/finesse/api/SystemConfig
Example URI:	http://finesse1.xyz.com/finesse/api/SystemConfig
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	—
HTTP Response:	200: Success 401: Unauthorized 403: Forbidden 500: Internal Server Error
Example 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>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

SystemConfig—Set

This API allows an administrator to configure the CTI server settings.

Note If you do not specify the backupHost and backupPort during a PUT operation but 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
Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<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>
Request Parameters:	host (required): Hostname or IP address of the primary (A Side) CTI server Port (required): Port number of the primary (A Side) CTI server backupHost (required if backupPort is present): Hostname or IP address of the backup (B Side) CTI server backupPort (required if backupHost is present): Port number of the backup (B Side) CTI server peripheralId (required): ID of the CTI server peripheral
HTTP Response:	200: Success 400: Invalid Input 400: Parameter Missing 401: Authorization Failure 403: Forbidden 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Invalid Input</ErrorType> <ErrorMessage>port</ErrorMessage> <ErrorData>65536</ErrorData> </ApiError> </ApiErrors>

SystemConfig API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the SystemConfig object.	—	
cti	Collection	Information about the CTI server settings.	—	
-->host	String	The hostname or IP address of the primary (A Side) CTI server.	—	No special characters allowed except "." and "-".

Parameter	Type	Description	Possible Values	Notes
-->port	Integer	The port of the primary (A Side) CTI server.	1–65535 Default value: 42027	
-->peripheralId	Integer	The ID of the CTI server peripheral.	1–32767 Default value: 5000	
-->backupHost	String	The hostname or IP address of the (B Side) backup 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 "-".
-->backupPort	Integer	The port of the backup (B Side) CTI server.	1–65535	

SystemConfig API Errors

Status	Error Type	Description
400	Invalid Input	One of the parameters provided as part of the user input is invalid or not recognized.
400	Parameter Missing	A required parameter was not provided in the request. For example, if the backupPort is provided but the backupHost is missing.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

ClusterConfig

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 ClusterConfig APIs apply only to Finesse deployments with Unified CCE. Because you need not configure cluster settings for Unified CCX deployments, these APIs are not supported for Finesse 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.

The ClusterConfig object is structured as follows:

```
<ClusterConfig>
   <uri>/finesse/api/ClusterConfig</uri>
   <secondaryNode>
      <host></host>
   </secondaryNode>
</ClusterConfig>
```

ClusterConfig APIs

ClusterConfig—Get

This API allows an administrator to get a copy of the ClusterConfig object.

URI:	http://<FQDN>/finesse/api/ClusterConfig
Example URI:	http://finesse1.xyz.com/finesse/api/ClusterConfig
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—

HTTP Response:	200: Success 401: Unauthorized 403: Forbidden 500: Internal Server Error
Example Response:	<ClusterConfig> <uri>/finesse/api/ClusterConfig</uri> <secondaryNode> <host>10.1.1.1</host> </secondaryNode> </ClusterConfig>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

ClusterConfig—Set

This API allows an administrator to configure cluster settings for Finesse.

URI:	http://<FQDN>/finesse/api/ClusterConfig
Example URI:	http://finesse1.xyz.com/finesse/api/ClusterConfig
Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	<ClusterConfig> <uri>/finesse/api/ClusterConfig</uri> <secondaryNode> <host>10.1.1.1</host> </secondaryNode> </ClusterConfig>
Request Parameters:	host (required): Hostname or IP address of the secondary Finesse server
HTTP Response:	200: Success 400: Invalid Input 400: Parameter Missing 401: Authorization Failure 403: Forbidden 500: Internal Server Error

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

ClusterConfig API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the ClusterConfig object.	—	
secondaryNode	Collection	Information about secondary Finesse node.	—	
-->host	String	The hostname or IP address of the secondary Finesse node.	—	No special characters allowed except "." and "-".

ClusterConfig API Errors

Status	Error Type	Description
400	Invalid Input	One of the parameters provided as part of the user input is invalid or not recognized.
400	Parameter Missing	A required parameter was not provided in the request.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

EnterpriseDatabaseConfig

The EnterpriseDatabaseConfig object is a container element that holds the properties required for Finesse to connect to the Administration & Data Server database (AWDB) for user authentication.

Note The EnterpriseDatabaseConfig APIs apply only to Finesse deployments with Unified CCE. Because these settings do not apply to Finesse deployments with Unified CCX, these APIs are not supported with Unified CCX.

The EnterpriseDatabaseConfig object is structured as follows:

```
<EnterpriseDatabaseConfig>
   <uri>/finesse/api/EnterpriseDatabaseConfig</uri>
   <host></host>
   <backupHost></backupHost>
   <port></port>
   <databaseName></databaseName>
   <domain></domain>
   <username></username>
   <password></password>
</EnterpriseDatabaseConfig>
```

EnterpriseDatabaseConfig APIs

EnterpriseDatabaseConfig—Get

This API allows an administrator to get a copy of the EnterpriseDatabaseConfig object.

URI:	http://<FQDN>/finesse/api/EnterpriseDatabaseConfig
Example URI:	http://finesse1.xyz.com/finesse/api/EnterpriseDatabaseConfig
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Unauthorized 403: Forbidden 500: Internal Server Error

Example Response:	<EnterpriseDatabaseConfig> <uri>/finesse/api/EnterpriseDatabaseConfig</uri> <host>10.1.1.1</host> <backupHost>10.1.1.2</backupHost> <port>1433</port> <databaseName>ucce8x_awdb</databaseName> <domain>xyz.com</domain> <username>Administrator</username> <password>admin_password</password> </EnterpriseDatabaseConfig>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

EnterpriseDatabaseConfig—Set

This API allows an administrator to configure the enterprise database settings.

Note If you do not specify the backupHost during a PUT operation but it was configured at an earlier time, the PUT operation resets the value for this parameter to blank.

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 run.

URI:	http://<FQDN>/finesse/api/EnterpriseDatabaseConfig?override='<true|false>'
Example URI:	http://finesse1.xyz.com/finesse/api/EnterpriseDatabaseConfig?override='true'
Security Constraints:	Only administrators can use this API.
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>
Request Parameters:	host (required): Hostname or IP address of the AWDB server backupHost (optional): Hostname or IP address of the backup AWDB server Port (required): Port number of the AWDB server databaseName (required): Name of the AWDB domain (required): Domain of the AWDB username (required): Username to sign in to the AWDB password (required): Password to sign in to the AWDB
HTTP Response:	200: Success 400: Invalid Input 400: Parameter Missing 401: Authorization Failure 403: Forbidden 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Invalid Input</ErrorType> <ErrorMessage>port</ErrorMessage> <ErrorData>65536</ErrorData> </ApiError> </ApiErrors>

EnterpriseDatabaseConfig API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the EnterpriseDatabaseConfig object.	—	
host	String	The hostname or IP address of the AWDB server.	—	No special characters allowed except “.” and “-”.

Parameter	Type	Description	Possible Values	Notes
backupHost	String	The hostname or IP address of the backup AWDB server.	—	No special characters allowed except "." and "-".
port	Integer	The port of the AWDB server.	1–65535	
databaseName	String	The name of the AWDB (for example, *ucceinstance*_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 API Errors

Status	Error Type	Description
400	Invalid Input	One of the parameters provided as part of the user input is invalid or not recognized.
400	Parameter Missing	A required parameter was not provided in the request. For example, if the backupPort is provided but the backupHost is missing.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

LayoutConfig

The LayoutConfig object is a container element that holds the layout XML for the Finesse desktop. The layout XML defines how tabs, labels, columns, and gadgets appear on the Finesse agent and supervisor desktops.

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 in 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
```

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

```
/desktop/gadgets/<gadgetname>.xml
```

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

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

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>
                   <columns>
                       <column>
                           <gadgets>
   <!--
            The following Gadget (Agent Queue Statistics) is *not* supported in Packaged
            CCE deployment.
            If you are using Packaged CCE you must comment out or remove this gadget.
   -->
                            <gadget>/desktop/gadgets/QueueStatistics.jsp</gadget>

   <!--
           The following Gadgets are for LiveData.  They are *ONLY* supported in a Packaged
 CCE
           Deployment.
           If you are using Packaged CCE and wish to show LiveData Reports, then do the
following:
           1) Uncomment out each Gadget you wish to show.
           2) Replace all instances of "my-cuic-server" with the Fully Qualified Domain
              Name of your Intelligence Center Server.
           3) [OPTIONAL] Adjust the height of the gadget by changing the "gadgetHeight"
              parameter.

       IMPORTANT NOTES:
```

```
            - In order for these Gadgets to work, you must have performed all documented
              pre-requisite steps.
            - The use of HTTP/HTTPS *must* match what your Users use for the Finesse Desktop

              (HTTP or HTTPS).
            - If you wish to use HTTP, then HTTP must be enabled on both Finesse and
              Intelligence Center.
            - Do *NOT* change the viewId (unless you have built a custom report and know
              what you are doing).
            - The "teamName" will be automatically replaced with the Team Name of the User
              logged into Finesse.
    -->

                        <!-- HTTPS Version of LiveData Gadgets -->
                            <!-- "Agent" Report -->
                           <!-- <gadget>https://my-cuic-server:8444/cuic/gadget/LiveData/

                                  LiveDataGadget.jsp?gadgetHeight=310&
                                  viewId=99E6C8E2100001410000000D80A0006C4&
                                  filterId=agent.id=CL%20teamName</gadget> -->
                            <!-- "Agent Skill Group" Report -->
                           <!-- <gadget>https://my-cuic-server:8444/cuic/gadget/LiveData/

                                  LiveDataGadget.jsp?gadgetHeight=310&
                                  viewId=9AB7848B100001410000001C50A0006C4&
                                  filterId=agent.id=CL%20teamName</gadget> -->
                            <!-- "Agent All Fields" Report -->
                           <!-- <gadget>https://my-cuic-server:8444/cuic/gadget/LiveData/

                                  LiveDataGadget.jsp?gadgetHeight=310&
                                  viewId=9A08E2351000014100000001230A0006C4&
                                  filterId=agent.id=CL%20teamName</gadget> -->
                            <!-- "Agent Skill Group All Fields" Report -->
                           <!-- <gadget>https://my-cuic-server:8444/cuic/gadget/LiveData/

                                  LiveDataGadget.jsp?gadgetHeight=310&
                                  viewId=A30EC25810000141000003A60A0006C4&
                                  filterId=agent.id=CL%20teamName</gadget> -->

                        <!-- HTTP Version of LiveData Gadgets -->
                            <!-- "Agent" Report -->
                            <!-- <gadget>http://my-cuic-server:8081/cuic/gadget/LiveData/
                                  LiveDataGadget.jsp?gadgetHeight=310&
                                  viewId=99E6C8E2100001410000000D80A0006C4&
                                  filterId=agent.id=CL%20teamName</gadget> -->
                            <!-- "Agent Skill Group" Report -->
                            <!-- <gadget>http://my-cuic-server:8081/cuic/gadget/LiveData/
                                  LiveDataGadget.jsp?gadgetHeight=310&
                                  viewId=9AB7848B100001410000001C50A0006C4&
                                  filterId=agent.id=CL%20teamName</gadget> -->
                            <!-- "Agent All Fields" Report -->
                            <!-- <gadget>http://my-cuic-server:8081/cuic/gadget/LiveData/
                                  LiveDataGadget.jsp?gadgetHeight=310&
                                  viewId=9A08E2351000014100000001230A0006C4&
                                  filterId=agent.id=CL%20teamName</gadget> -->
                            <!-- "Agent Skill Group All Fields" Report -->
                            <!-- <gadget>http://my-cuic-server:8081/cuic/gadget/LiveData/
                                  LiveDataGadget.jsp?gadgetHeight=310&
                                  viewId=A30EC25810000141000003A60A0006C4&
                                  filterId=agent.id=CL%20teamName</gadget> -->
                        </gadgets>
                      </column>
                    </columns>
                </tab>
                <tab>
                    <id>manageCall</id>
                    <label>finesse.container.tabs.agent.manageCallLabel</label>
                </tab>

    <!--
            The following Tab and Gadgets are for LiveData.  They are *ONLY* supported in a
Packaged CCE
            Deployment.
```

```
            If you are using Packaged CCE and wish to show LiveData Reports, then do the
following:
            1) Remove these comments leaving the tab and gadgets you wish to show.
            2) Uncomment out each Gadget you wish to show.
            3) Replace all instances of "my-cuic-server" with the Fully Qualified Domain
               Name of your Intelligence Center Server.
            4) [OPTIONAL] Adjust the height of the gadget by changing the "gadgetHeight"
               parameter.
         IMPORTANT NOTES:
            - In order for these Gadgets to work, you must have performed all documented
              pre-requisite steps.
            - The use of HTTP/HTTPS *must* match what your Users use for the Finesse Desktop

              (HTTP or HTTPS).
            - If you wish to use HTTP, then HTTP must be enabled on both Finesse and
              Intelligence Center.
            - Do *NOT* change the viewId (unless you have built a custom report and
              know what you are doing).
            - The "teamName" will be automatically replaced with the Team Name of the User
              logged into Finesse.
   -->
   <!--
            If you are showing the tab, then also uncomment this section.
                <tab>
                    <id>moreReports</id>
                    <label>finesse.container.tabs.agent.moreReportsLabel</label>
                    <gadgets>-->

                    <!-- HTTPS Version of LiveData Gadgets -->
                        <!-- "Agent Skill Group" Report -->
                        <!-- <gadget>https://my-cuic-server:8444/cuic/gadget/LiveData/
                              LiveDataGadget.jsp?gadgetHeight=310&
                              viewId=9AB7848B10000141000001C50A0006C4&
                              filterId=agent.id=CL</gadget> -->
                    <!-- HTTP Version of LiveData Gadgets -->
                        <!-- "Agent Skill Group" Report -->
                        <!-- <gadget>http://my-cuic-server:8081/cuic/gadget/LiveData/
                              LiveDataGadget.jsp?gadgetHeight=310&
                              viewId=9AB7848B10000141000001C50A0006C4&
                              filterId=agent.id=CL</gadget> -->
   <!--
            If you are showing the tab, then also uncomment this section as well.
                    </gadgets>
                </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>
                    <columns>
                        <column>
                            <gadgets>
                                <gadget>/desktop/gadgets/TeamPerformance.jsp</gadget>
                                <gadget>/desktop/gadgets/QueueStatistics.jsp</gadget>
                            </gadgets>
                        </column>
                    </columns>
                </tab>
                <tab>
                    <id>manageCall</id>
                    <label>finesse.container.tabs.supervisor.manageCallLabel</label>
                </tab>
            </tabs>
        </layout>
   </finesseLayout>
   </layoutxml>
</LayoutConfig>
```

LayoutConfig APIs

LayoutConfig—Get

This API allows an administrator to get a copy of the LayoutConfig object.

URI:	http://<FQDN>/finesse/api/LayoutConfig/default
Example URI:	http://finesse1.xyz.com/finesse/api/LayoutConfig/default
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Unauthorized 403: Forbidden 500: Internal Server Error
Example Response:	<LayoutConfig> <uri>/finesse/api/LayoutConfig/default</uri> <layoutxml> ... </layoutxml> </LayoutConfig>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

LayoutConfig—Set

This API allows an administrator to update the default layout settings for the Finesse desktop.

Note The XML data is verified to ensure that it is valid XML 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:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<LayoutConfig>`
	`  <layoutxml><?xml version="1.0" encoding="UTF-8">`
	`  ...`
	`  </layoutxml>`
	`</LayoutConfig>`

Request Parameters:	layoutxml (required): The XML data that determines the layout of the Finesse desktop
HTTP Response:	200: Success 400: Invalid Input 400: Parameter Missing 401: Authorization Failure 403: Forbidden 500: Internal Server Error

Example Failure Response:	`<ApiErrors>`
	`  <ApiError>`
	`  <ErrorType>Invalid Input</ErrorType>`
	`  <ErrorMessage>layoutxml</ErrorMessage>`
	`  </ApiError>`
	`</ApiErrors>`

LayoutConfig API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the LayoutConfig object.	—	

Parameter	Type	Description	Possible Values	Notes
layoutxml	String	The XML data that determines the layout of the Finesse desktop.	—	Must be valid XML and must conform to the Finesse schema.

LayoutConfig API Errors

Status	Error Type	Description
400	Invalid Input	The submitted XML is invalid or does not conform to the Finesse schema.
400	Parameter Missing	The layout XML file was not provided.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

ReasonCode

The ReasonCode object represents a reason code that can be applied when an agent changes state. There are two categories of reason codes: not ready reason codes and sign out reason codes.

Administrators can use either the ReasonCode APIs or the Finesse administration console to configure not ready and sign out reason codes. When using the APIs to configure reason codes, the administrator specifies the category of reason code in the request (NOT_READY or LOGOUT).

To prevent reporting problems, define your reason codes consistently on both Finesse and the platform (Unified CCE or Unified CCX). For example, if you create a not ready reason code in Finesse with a code of 413 and a label of “Meeting”, but create a not ready reason code in Unified CCE with a code of 413 and a description of “Lunch Break”, the Unified CCE report shows “Lunch Break” for any agent who selects 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)

The ReasonCode object is structured as follows:

```
<ReasonCode>
   <uri>/finesse/api/ReasonCode/{id}</uri>
   <category>NOT_READY|LOGOUT</category>
   <code></code>
   <label></label>
   <forAll>true|false</forAll>
</ReasonCode>
```

ReasonCode APIs

ReasonCode—Get

This API allows an administrator to get a copy of the ReasonCode object.

URI:	http://<FQDN>/finesse/api/ReasonCode/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/ReasonCode/45
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error

Example Response:	<ReasonCode> <uri>/finesse/api/ReasonCode/45</uri> <category>NOT_READY</category> <code>10</code> <label>Team Meeting</label> <forAll>true</forAll> </ReasonCode>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

ReasonCode—Get List

This API allows an administrator to get a list of not ready or sign out reason codes. The required URI parameter *category* specifies whether to retrieve not ready reason codes or sign out reason codes. If the category 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:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Invalid Input 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error

Example Response:	<ReasonCodes category="NOT_READY">
	  <ReasonCode>
	  ... Full ReasonCode Object ...
	  </ReasonCode>
	  <ReasonCode>
	  ... Full ReasonCode Object ...
	  </ReasonCode>
	  <ReasonCode>
	  ... Full ReasonCode Object ...
	  </ReasonCode>
	</ReasonCodes>

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

ReasonCode—Create

This 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 with a category of NOT_READY, and 100 global and 100 non-global reason codes with a category of LOGOUT.

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

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

URI:	http://<FQDN>/finesse/api/ReasonCode/
Example URI:	http://finesse1.xyz.com/finesse/api/ReasonCode/
Security Constraints:	Only administrators can use this API.
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<ReasonCode> <category>NOT_READY</category> <code>24</code> <label>Lunch</label> <forAll>true</forAll> </ReasonCode>
Request Parameters:	category (required): The category of reason code (NOT_READY or LOGOUT) code (required):The code for the reason code label (required): The UI label for the reason code forAll (required): Whether the reason code is global (true) or non-global (false)
HTTP Response:	200: Success **Note** Finesse 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 400: Maximum Exceeded 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

ReasonCode—Update

This API allows an administrator to modify an existing reason code. The administrator specifies an existing reason code via the uri, which includes its id, along with the value of the field to update.

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.

You do not need to include the attributes (code, label, or forAll) that you do not want 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>
```

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

URI:	http://<FQDN>/finesse/api/ReasonCode/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/ReasonCode/456
Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	`<ReasonCode>`
	`  <code>101</code>`
	`  <label>Lunch Break</label>`
	`  <forAll>true</forAll>`
	`</ReasonCode>`

Request Parameters:	id (required): The database ID for the reason code code:The code for the reason code label: The UI label for the reason code forAll: Whether the reason code is global (true) or non-global (false) **Note** Your request must include at least one of the following parameters: code, label, or forAll.
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error

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

ReasonCode—Delete

This 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/ 423
Security Constraints:	Only administrators can use this API.
HTTP Method:	DELETE
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error

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

ReasonCode API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the ReasonCode object.	—	
category	String	The category of the reason code.	NOT_READY, LOGOUT	
code	Integer	The code for the reason code	Unified CCE: 1–65535 Unified CCX: 1–999	The combination of code and category must be unique.

Parameter	Type	Description	Possible Values	Notes
label	String	The UI label for the reason code.	—	Maximum of 40 characters. The combination of label and category must be unique.
forAll	Boolean	Whether a reason code is global (true) or non-global (false).	true, false	

ReasonCode API Errors

Status	Error Type	Description
400	Bad Request	One of the required parameters was not provided or is invalid
400	Finesse API Error	API error such as duplicated reason code or the reason code does not exist.
400	Maximum Exceeded	The maximum number of items has been exceeded.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
401	Invalid Authorization User Specified	The authenticated user tried to use the identity of another user.
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
404	Not Found	The specified resource cannot be found.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

WrapUpReason

The WrapUpReason object represents a 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></label>
   <forAll>true|false</forAll>
</WrapUpReason>
```

WrapUpReason APIs

WrapUpReason—Get

This API allows an administrator to get a copy of the WrapUpReason object.

URI:	http://<FQDN>/finesse/api/WrapUpReason/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/WrapUpReason/31
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error

Example Response:	<WrapUpReason> <uri>/finesse/api/WrapUpReason/31</uri> <label>Product Question</label> <forAll>false</forAll> </WrapUpReason>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

WrapUpReason—Get List

This API allows an administrator to get a list of wrap-up reasons.

URI:	http://<FQDN>/finesse/api/WrapUpReasons
Example URI:	http://finesse1.xyz.com/finesse/api/WrapUpReasons
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error

Example Response:	<WrapUpReasons> <WrapUpReason> ... Full WrapUpReason Object ... </WrapUpReason> <WrapUpReason> ... Full WrapUpReason Object ... </WrapUpReason> <WrapUpReason> ... Full WrapUpReason Object ... </WrapUpReason> </WrapUpReasons>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

WrapUpReason—Create

This 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. for each category.

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

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

URI:	http://<FQDN>/finesse/api/WrapUpReason/
Example URI:	http://finesse1.xyz.com/finesse/api/WrapUpReason/
Security Constraints:	Only administrators can use this API.
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	<WrapUpReason> <label>Recommendation</label> <forAll>true</forAll> </WrapUpReason>
Request Parameters:	label (required): The UI label for the wrap-up reason forAll (required): Whether the wrap-up reason is global (true) or non-global (false)

HTTP Response:	200: Success **Note** Finesse successfully created the new WrapUpReason. 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 500: Internal Server Error

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

WrapUpReason—Update

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

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.

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 reason code from "Wrong Number" to "Wrong Department", you can send the following request:

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

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

URI:	http://<FQDN>/finesse/api/WrapUpReason/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/WrapUpReason/43
Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<WrapUpReason> <label>Sales Call</label> <forAll>true</forAll> </WrapUpReason>
Request Parameters:	id (required): The database ID for the wrap-up reason label (required): The UI label for the reason code forAll (required): Whether the reason code is global (true) or non-global (false)
HTTP Response:	200: Success 400: Finesse API Error 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

WrapUpReason—Delete

This API allows an administrator to delete an existing wrap-up reason.

URI:	http://<FQDN>/finesse/api/WrapUpReason/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/WrapUpReason/23
Security Constraints:	Only administrators can use this API.
HTTP Method:	DELETE
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—

HTTP Response:	200: Success 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

WrapUpReason API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the WrapUpReason object.	—	
label	String	The UI label for the wrap-up reason.	—	Maximum of 39 bytes (which is equal to 39 US English characters). The label must be unique.
forAll	Boolean	Whether a wrap-up reason is global (true) or non-global (false).	true, false	

WrapUpReason API Errors

Status	Error Type	Description
400	Bad Request	The request body is invalid
400	Finesse API Error	API error such as duplicated wrap-up reason or the wrap-up reason does not exist.
400	Maximum Exceeded	The maximum number of items has been exceeded.

Status	Error Type	Description
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
401	Invalid Authorization User Specified	The authenticated user tried to use the identity of another user.
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
404	Not Found	The specified resource cannot be found.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

MediaPropertiesLayout

The MediaPropertiesLayout object represents the appearance of media properties in the call control gadget on the agent or supervisor desktop. Media properties are carried in Dialog objects. Administrators can customize the layout of media properties.

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

- BACampaign
- BAAccountNumber
- BAResponse
- BAStatus
- BADialedListID
- BATimeZone
- BABuddyName
- BACustomerNumber (Unified CCX only)

The MediaPropertiesLayout object is structured as follows:

```
<MediaPropertiesLayout>
   <header>
      <entry>
         <displayName></displayName>
         <mediaProperty></mediaProperty>
      </entry>
   </header>
```

```
   <column>
      <entry>
         <displayName></displayName>
         <mediaProperty></mediaProperty>
      </entry>
      <entry>
         <displayName></displayName>
         <mediaProperty></mediaProperty>
      </entry>
   </column>
   <column>
      <entry>
         <displayName></displayName>
         <mediaProperty></mediaProperty>
      </entry>
      <entry>
         <displayName></displayName>
         <mediaProperty></mediaProperty>
      </entry>
   </column>
</MediaPropertiesLayout>
```

MediaPropertiesLayout APIs

MediaPropertiesLayout—Get

This API allows an administrator to get a copy of the default MediaPropertiesLayout object.

Finesse supports a single default instance.

URI:	http://<FQDN>/finesse/api/MediaPropertiesLayout/default
Example URI:	http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/default
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 403: Forbidden 500: Internal Server Error

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

MediaProperties Layout—Set

This API allows an administrator set the default call variable layout for the Finesse desktop.

URI:	http://<FQDN>/finesse/api/MediaPropertiesLayout/default
Example URI:	http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/default
Security Constraints:	Only administrators can use this API.
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>
Request Parameters:	header (optional): Contains displayName and mediaProperty that appears in the call header on the desktop column (optional): Grouping of media properties for the Finesse desktop (can contain a maximum of 10 entries) entry (optional): Contains a displayName and mediaProperty displayName (required): A label that describes the mediaProperty for that entry mediaProperty (required): The name of the variable for that entry
HTTP Response:	200: Success 400: Invalid Input 400: Parameter Missing 401: Authorization Failure 403: Forbidden 500: Internal Server Error
Example Failure Response:	<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>

MediaPropertiesLayout API Parameters

Parameter	Type	Description	Possible Values	Notes
header	Object	Contains a single entry (combination of displayName and mediaProperty) that appears in the call header on the desktop for each call.	—	
column	Object	Grouping of media properties for agent and supervisor desktops. Contains a list of entry objects	—	Finesse supports up to two columns in the MediaProperties Layout object. Columns can contain up to 10 entries and can be empty. The first column supplied in a PUT is always the left column. The second column (if any) is always the right column.
-->entry	Object	A displayName and mediaProperty combination.	—	Each entry must contain one displayName and one mediaProperty. The displayName can be empty.
--->displayName	String	Part of an entry. A label that describes the mediaProperty for that entry (for example, Customer Name). The label appears on the Finesse desktop.	—	Maximum of 50 characters.

Parameter	Type	Description	Possible Values	Notes
--->mediaProperty	String	The name of the variable that is displayed on the Finesse desktop. Each entry contains exactly one mediaProperty.	Allowed strings include callVariable1 through callVariable10, any valid ECC variable (user.*), and the following Outbound Option variables: • BACampaign • BAAccountNumber • BAResponse • BAStatus • BADialedListID • BATimeZone • BABuddyName • BACustomerNumber (Unified CCX only)	Maximum of 32 characters.

MediaPropertiesLayout API Errors

Status	Error Type	Description
400	Invalid Input	At least one of the parameters provided is not valid.
400	Parameter Missing	At least one of the required parameters was not provided.
400	Maximum Exceeded	The maximum number of items has been exceeded.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

PhoneBook

The PhoneBook object represents a phone book that contains contacts. Each PhoneBook object contains a Contacts summary object.

Phone books can be assigned globally (to all agents) or to specific teams. Finesse supports a maximum of 10 global phone books and 50 team phone books.

The PhoneBook object is structured as follows:

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

PhoneBook APIs

PhoneBook—Get

This API allows an administrator to get a specific phone book.

URI:	http://<FQDN>/finesse/api/PhoneBook/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/34
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Finesse API Error 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error

Example Response:	<PhoneBook> <uri>/finesse/api/PhoneBook/34</uri> <name>Phonebook 1</name> <type>GLOBAL</type> <contacts>/finesse/api/PhoneBook/34/Contacts</contacts> </PhoneBook>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

PhoneBook—Get List

This API allows an administrator to get a list of all global and team phone books. Agents' personal phone books are not returned.

URI:	http://<FQDN>/finesse/api/PhoneBooks
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBooks
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 500: Internal Server Error

Example Response:	<PhoneBooks> <PhoneBook> ...Full PhoneBook Object... </PhoneBook> <PhoneBook> ...Full PhoneBook Object... </PhoneBook> <PhoneBook> ...Full PhoneBook Object... </PhoneBook> </PhoneBooks>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

PhoneBook—Create

This API allows an administrator to create a new phone book. The administrator specifies the name and type for the phone book.

URI:	http://<FQDN>/finesse/api/PhoneBook/
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/
Security Constraints:	Only administrators can use this API.
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	<PhoneBook> <name>PhoneBook1</name> <type>GLOBAL</type> </PhoneBook>
Request Parameters:	name (required): The name of the phone book type (required): The type of phone book (GLOBAL or TEAM)
HTTP Response:	200: Success **Note** Finesse successfully created the new phone book. The server response contains an empty response body and a location header that denotes the absolute URL of the new phone book. 400: Invalid Input 400: Parameter Missing 401: Authorization Failure 403: Forbidden 500: Internal Server Error

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

PhoneBook—Update

This API allows an administrator to modify an existing phone book.

URI:	http://<FQDN>/finesse/api/PhoneBook/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/45
Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	<PhoneBook> <name>PhoneBook2</name> <type>TEAM</type> </PhoneBook>
Request Parameters:	id (required): The database ID for the phone book name (required): The name of the phone book type (required): The type of phone book (GLOBAL or TEAM)
HTTP Response:	202: Successfully Accepted 400: In Use 400: Invalid Input 400: Parameter Missing 401: Authorization Failure 403: Forbidden 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

PhoneBook—Delete

This 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/43
Security Constraints:	Only administrators can use this API.
HTTP Method:	DELETE
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: In Use 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

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

PhoneBook—Import Contact List (CSV)

This API allows an administrator to replace all the contacts in a specific phone book by importing a list of contacts in a comma-separated values (CSV) file. The CSV file can contain up to 1500 contacts.

All existing contacts in the phone book are deleted before the new contacts are inserted. Contacts that contain errors are not inserted. Contacts that are error-free or contacts that contain missing or empty fields are inserted.

In general, the import is fault-tolerant. The CSV file is sent using standard web form syntax and 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/34/Contacts/csvFileContent

Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	text/CSV
Input/Output Format:	text/plain, text/CSV

Example 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>`

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"`

Request Parameters:	id (required): The database ID for the phone book
HTTP Response:	202: Successfully Accepted **Note** This response indicates a successful completion of the request. The request is processed and the actual response is sent as part of and updated to the PhoneBook object. 400: Invalid Input 400: Maximum Exceeded 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

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

PhoneBook—Import Contact List (XML)

This API allows an administrator to replace all the contacts in a specific phone book by importing a collection of contacts. The collection can contain up to 1500 contacts.

All existing contacts in the phone book are deleted before the new contacts are inserted. Contacts that contain errors are not inserted.

URI:	http://<FQDN>/finesse/api/PhoneBook/<id>/Contacts
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contacts
Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	<Contacts> <Contact> ...Full Contact Object... </Contact> <Contact> ...Full Contact Object... </Contact> <Contact> ...Full Contact Object </Contact>
Request Parameters:	id (required): The database ID for the phone book
HTTP Response:	202: Successfully Accepted **Note** This response indicates a successful completion of the request. The request is processed and the actual response is sent as part of and updated to the PhoneBook object. **Note** Some of the data could not be imported because it was invalid. The ErrorData field contains a list of lines that were not imported. This response indicates partial success because some data was uploaded. 400: Invalid Input 400: Maximum Exceeded 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

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

PhoneBook—Export Contact List

This API allows an administrator to export a list of contacts that belong to a specific phone book. The list is exported in CSV format.

URI:	http://<FQDN>/finesse/api/PhoneBook/<id>/Contacts/csvFileContent
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contacts/csvFileContent
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	text/CSV
Input/Output Format:	Multipart/form-data type=file
Example 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"
HTTP Response:	200: Success **Note** This response indicates a successful completion of the request. After a successful request, browser clients are prompted to save the returned content as a CSV file. 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

PhoneBook API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the PhoneBook object.	—	The id in the URI maps to the primary key of the phone book entry.
name	String	The name of the phone book.	—	
type	String	The type of phone book.	GLOBAL, TEAM	

PhoneBook API Errors

Status	Error Type	Description
400	Finesse API Error	API error such as the object is stale or does not exist.
400	Invalid Input	One of the input parameters exceeded constraints. Contacts could not be imported because the data was invalid. The file may be empty or may not contain any valid lines. If the ErrorData field contains no lines, there may not be data to import. The multipart mime message may have been improperly formatted or did not contain a file. The multipart mime message may have been improperly formatted or did not contain a file. In this case, the existing records are overwritten.
400	In Use	The phone book is assiged to a team. You cannot change a team phone book to a global phone book if it is use. You cannot delete a phone book if it is use.
400	Maximum Exceeded	The maximum number of phone books or contacts has been exceeded.
400	Parameter Missing	A required parameter was not present in the request.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).

Status	Error Type	Description
401	Invalid Authorization User Specified	The authenticated user tried to use the identity of another user.
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
404	Not Found	The specified resource cannot be found.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

Contact

The Contact object represents a contact that can be assigned to a phone book. A phone book can contain up to 1500 contacts. Finesse supports a system-wide total of 1500 contacts.

The Contact object is structured as follows:

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

Contact APIs

Contact—Get

This API allows an administrator to get a specific phone book contact.

URI:	http://<FQDN>/finesse/api/PhoneBook/<phoneBookId>/Contact/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contact/785
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML

Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

Example Response:	`<Contact>`
	`<firstName>John</firstName>`
	`<lastName>Doe</lastName>`
	`<phoneNumber>5551234</phoneNumber>`
	`<description>Accounts Manager</description>`
	`<uri>/finesse/api/PhoneBook/34/Contact/785</uri>`
	`</Contact>`

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

Contact—Get List

This API allows an administrator to get a list of contacts for a specific phone book.

URI:	http://<FQDN>/finesse/api/PhoneBook/<phoneBookId>/Contacts
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contacts
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—

HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error
Example Response:	<Contacts> <Contact> ...Full Contact Object... </Contact> <Contact> ...Full Contact Object... </Contact> <Contact> ...Full Contact Object... </Contact> </Contacts>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

Contact—Create

This API allows an administrator to create a new phone book contact.

URI:	http://<FQDN>/finesse/api/PhoneBook/<phoneBookId>/Contact/
Example URI:	http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contact/
Security Constraints:	Only administrators can use this API.
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<Contact> <firstName>Jerry</firstName> <lastName>Green</lastName> <phoneNumber>5554444</phoneNumber> <description>Product Expert</description> </Contact>
Request Parameters:	phoneBookId (required): Maps to the primary key of the phone book to which the contact belongs firstName (optional): The first name of the contact lastName (optional): The last name of the contact phoneNumber (required): The phone number of the contact description (optional): A description for the contact
HTTP Response:	200: Success **Note** Finesse successfully created the new contact. The server response contains an empty response body and a location header that denotes the absolute URL of the new contact. 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

Contact—Update

This 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/45 /Contact/787
Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<Contact> <firstName>Marie</firstName> <lastName>Brown</lastName> <phoneNumber>5554444</phoneNumber> <description>Product Expert</description> </Contact>
Request Parameters:	phoneBookId (required): Maps to the primary key of the phone book to which the contact belongs id (required): Maps to the primary key of the contact entry firstName (optional): The first name of the contact lastName (optional): The last name of the contact phoneNumber (required): The phone number of the contact description (optional): A description for the contact
HTTP Response:	202: Successfully Accepted 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

Contact—Delete

This 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/43 /Contact/1523
Security Constraints:	Only administrators can use this API.
HTTP Method:	DELETE
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—

HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Authorization Failure</ErrorType> <ErrorMessage>UNAUTHORIZED</ErrorMessage> <ErrorData>jsmith</ErrorData> </ApiError> </ApiErrors>

Contact API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the Contact object.	—	The phoneBookId in the URI maps to the primary key of the phone book to which the contact belongs. The id in the URI maps to the primary key of the contact entry.
firstName	String	The first name of the contact.	—	Maximum of 128 characters.
lastName	String	The last name of the contact.	—	Maximum of 128 characters.
phoneNumber	String	The phone number for the contact.	—	Maximum of 32 characters.
description	String	A description of the contact.	—	Maximum of 128 characters.

Contact API Errors

Status	Error Type	Description
400	Bad Request	The request body is invalid.
400	Finesse API Error	API error such as the object is stale or does not exist.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
404	Not Found	The specified resource cannot be found.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

Workflow

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, which are used to determine whether the associated actions are executed.

The Workflow object contains the following subobjects: TriggerSet, ConditionSet, and workflowActions.

The Workflow object is structured as follows:

```
<Workflow>
   <uri>/finesse/api/Workflow/{id}</uri>
   <name></name>
   <description></description>
   <TriggerSet>
      <type></type>
      <name></name>
      <allowOverlappingCallWorkflow></allowOverlappingCallWorkflow>
      <triggers>
         <Trigger>
            <Variable>
               <name></name>
               <node></node>
               <type></type>
            </Variable>
            <comparator></comparator>
            <value></value>
         </Trigger>
         <Trigger>
```

```
            <Variable>
               <name></name>
               <node></node>
               <type></type>
            </Variable>
            <comparator></comparator>
            <value></value>
         </Trigger>
      </triggers>
   </TriggerSet>
   <ConditionSet>
      <applyMethod></applyMethod>
      <conditions>
         <Condition>
            <Variable>
               <name></name>
               <type></type>
            </Variable>
            <comparator></comparator>
            <value></value>
         </Condition>
         <Condition>
            <Variable>
               <name></name>
               <type></type>
            </Variable>
            <comparator></comparator>
            <value></value>
         </Condition>
      </conditions>
   </ConditionSet>
   <workflowActions>
      <WorkflowAction>
         <name></name>
         <type></type>
         <uri>/finesse/api/WorkflowAction/{id}</uri>
      </WorkflowAction>
      <WorkflowAction>
         <name></name>
         <type></type>
         <uri>/finesse/api/WorkflowAction/{id}</uri>
      </WorkflowAction>
   </workflowActions>
</Workflow>
```

The following SYSTEM TriggerSets are defined by the Finesse system. When you create a workflow, you need only specify the name and type of SYSTEM. The TriggerSets are automatically expanded when retrieved by the User—Get list of workflows API.

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>
```

Workflow APIs

Workflow—Get

This API allows an administrator to get a specific Workflow object.

URI:	http://<FQDN>/finesse/api/Workflow/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/Workflow/195
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML

Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

Example Response:	

```
<Workflow>
   <uri>/finesse/api/Workflow/195</uri>
   <name>Workflow A</name>
   <description>Workflow description</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,OUT,OUTBOUND,OUTBOUND_
             CALLBACK,OUTBOUND_PERSONAL_CALLBACK,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[.=${userExtension}]/../state</node>
               <type>CUSTOM</type>
            </Variable>
            <comparator>IS_IN_LIST</comparator>
            <value>ALERTING,ACTIVE,HELD</value>
         </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>Company Web Page</name> <type>BROWSER_POP</type> <uri>/finesse/api/WorkflowAction/9876</uri> </WorkflowAction> </workflowActions> </Workflow>
Example Failure Response:	<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>

Workflow—Get List

This 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:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 500: Internal Server Error

Example Response:	<Workflows> <Workflow> ...Full Workflow Object... </Workflow> <Workflow> ...Full Workflow Object... </Workflow> <Workflow> ...Full Workflow Object... </Workflow> </Workflows>
Example Failure Response:	<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>

Workflow—Create

This API allows an administrator to create a new workflow. Finesse supports a maximum of 100 workflows.

Note If you provide two or more duplicate tags during a POST, the value of the last duplicate tag is processed and all other duplicate tags are ignored.

URI:	http://<FQDN>/finesse/api/Workflow/
Example URI:	http://finesse1.xyz.com/finesse/api/Workflow/
Security Constraints:	Only administrators can use this API.
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<Workflow> ...Full Workflow Object... </Workflow>
Request Parameters:	id (required): Maps to the primary key of the workflow entry name (required): The name of the workflow description (optional): A description of the workflow TriggerSet (required): A set of events that cause the conditions to be evaluated ConditionSet (optional): A set of conditions that determine if the workflow is executed workflowActions (optional): A list of workflow actions to execute if the trigger and conditions are satisfied
HTTP Response:	200: Success **Note** Finesse successfully created the new workflow. The server response contains an empty response body and a location header that denotes the absolute URL of the new phone book. 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 500: Internal Server Error
Example Failure Response:	<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>

Workflow—Update

This API allows an administrator to update an existing workflow.

If the attributes (name, description, TriggerSet, ConditionSet, workflowActions) for the specified workflow do not change, the request does not need to include those attributes. If an attribute is not specified, the current value is retained. However, you must specify at least one attribute in the request.

If you only want to change the description of the workflow, you can make the following request:

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

Note If you provide two or more duplicate tags during a PUT, the value of the last duplicate tag is processed and all other duplicate tags are ignored.

URI:	http://<FQDN>/finesse/api/Workflow/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/Workflow/769
Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

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

Request Parameters:	id (required): Maps to the primary key of the workflow entry name (optional): The name of the workflow description (optional): A description of the workflow TriggerSet (optional): A set of events that cause the conditions to be evaluated ConditionSet (optional): A set of conditions that determine if the workflow is executed workflowActions (optional): A list of workflow actions to execute if the trigger and conditions are satisfied
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

Example Failure Response:	<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>

Workflow—Delete

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

URI:	http://<FQDN>/finesse/api/Workflow/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/Workflow/768
Security Constraints:	Only administrators can use this API.
HTTP Method:	DELETE
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error
Example Failure Response:	<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>

Workflow API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the Workflow object.	—	The id in the URI maps to the primary key of the workflow.
name	String	The name of the workflow.	—	Must be unique. Maximum of 40 characters.
description	String	A description of the workflow.	—	Maximum of 128 characters.
TriggerSet	Object	A set of events that cause the conditions to be evaluated.	—	
ConditionSet	Object	A set of conditions that determine whether the workflow executes.	—	You can assign up to five conditions to a workflow.
workflowActions	Object	A list of workflow actions to execute if the trigger and its conditions are met. Actions execute in the order in which they appear in this list.	—	You can assign up to five workflow actions to a workflow. When getting a workflow or list of workflows, this list contains summary workflow actions (name, type, and URL). When creating or updating a workflow, only the URI is required in each workflow action. For more information, see WorkflowAction, on page 190.

ConditionSet Parameters

Parameter	Type	Description	Possible Values	Notes
applyMethod	String	Determines whether any or all of the conditions must be met for the workflow to execute.	ANY, ALL	
conditions	Object	A list of conditions for the workflow.	—	Maximum of five conditions for a workflow. A workflow with no conditions is specified by a conditions parameter with no Condition elements.
Condition	Object	Information about a workflow condition.	—	
Variable	Object	A piece of data from the Trigger event used to filter the event.	—	Leading and trailing spaces are removed from the variable during evaluation. Comma-separated values in a list also have leading and trailing spaces removed. If the value contains only spaces, it is treated as an empty value.
comparator	String	The operator used to compare the event variable to the desired value.	IS_EQUAL, IS_NOT_EQUAL, BEGINS_WITH, ENDS_WITH, CONTAINS, IS_EMPTY, IS_NOT_EMPTY, IS_IN_LIST, IS_NOT_IN_LIST	

Parameter	Type	Description	Possible Values	Notes
value	String	The value to compare the event variable with.	When type is SYSTEM, valid values are CALL_ARRIVES, CALL_ANSWERED, CALL_ENDS, CALL_IS_MADE, and CALL_IS_PREVIEWED.	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).

TriggerSet Parameters

Parameter	Type	Description	Possible Values	Notes
type	String	The type of TriggerSet.	SYSTEM	
name	String	The name of the TriggerSet	When type is SYSTEM, valid values are CALL_ARRIVES, CALL_ANSWERED, CALL_ENDS, CALL_IS_MADE, and CALL_IS_PREVIEWED.	
allow Overlapping CallWorkflow	Boolean	Indicates whether workflow for a second simultaneous call can fir while the call for this trigger is in process.	TRUE, FALSE	Default for this parameter is FALSE.
triggers	Object	List of Trigger subobjects.	—	For workflow admin, this field is not returned and is ignored if the type is SYSTEM.

Trigger Parameters

Parameter	Type	Description	Possible Values	Notes
Variable	Object	A piece of data from the trigger event to be used to filter the event. Contains a name, node, and type.	—	
name	String	A unique name for the variable. Used as a readable, unique key for the variable.	—	
node	String	The XPath to use to extract the value of the variable from an XMPP event that might contain it.	—	
type	String	Indicates whether this is a system or custom variable.	SYSTEM, 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.

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

Variable Name	Value	Data Type
${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 API Errors

Status	Error Type	Description
400	Bad Request	The request body is invalid.
400	Finesse API Error	API error such as the object is stale or does not exist.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
404	Not Found	The specified resource cannot be found.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

WorkflowAction

The WorkflowAction object represents a workflow action that can be assigned to a workflow. Finesse supports a system-wide maximum of 100 workflow actions.

The WorkflowAction object is structured as follows:

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

```
   <handledBy></handledBy>
   <params>
     <Param>
        <name><name>
        <value></value>
     </Param>
     <Param>
        <name></name>
        <value></value>
     </Param>
   </params>
   <actionVariables>
     <ActionVariable>
       <name></name>
       <type></type>
     </ActionVariable>
   </actionVariables>
</WorkflowAction>
```

There are two types of workflow actions: BROWSER_POP and HTTP_REQUEST.

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>
```

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>
```

WorkflowAction APIs

WorkflowAction—Get

This API allows an administrator to get a specific WorkflowAction object.

URI:	http://<FQDN>/finesse/api/WorkflowAction/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/WorkflowAction/674
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

Example Response:	<WorkflowAction> ...Full WorkflowAction Object... </WorkflowAction>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorData>Action 674 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 674 </ErrorMessage> </ApiError> </ApiErrors>

WorkflowAction—Get List

This 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:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 500: Internal Server Error

Example 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>
Example Failure Response:	<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>

WorkflowAction—Create

This API allows an administrator to create a new workflow action.

Note If you provide two or more duplicate tags during a POST, the value of the last duplicate tag is processed and all other duplicate tags are ignored.

URI:	http://<FQDN>/finesse/api/WorkflowAction/
Example URI:	http://finesse1.xyz.com/finesse/api/WorkflowAction/
Security Constraints:	Only administrators can use this API.
HTTP Method:	POST
Content Type:	Application/XML
Input/Output Format:	XML

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

Request Parameters (Browser Pop):	name (required): The name of the workflow action type (required): The type of workflow action handledBy (required): Indicates what handles the action params (required): List of Params for the workflow action actionVariables (required): list of actionVariables for the workflow path (required): The path to use in the action windowName (optional): The window name to pop open
Request Parameters (HTTP Request):	name (required): The name of the workflow action type (required): The type of workflow action handledBy (required): Indicates what handles the action params (required): List of Params for the workflow action actionVariables (required): list of actionVariables for the workflow path (required): The path to use in the action method (required): The method to use in the request authenticationType (optional): The authentication type to use in the request location (required): Whether the request is to Finesse or a third party contentType (optional): The value of the content type header to send with the request body (optional): The body to send with the request
HTTP Response:	200: Success **Note** Finesse successfully created the new workflow action. The server response contains an empty response body and a location header that denotes the absolute URL of the new workflow action. 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 500: Internal Server Error

Example Failure Response:	<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>

WorkflowAction—Update

This API allows an administrator to update an existing workflow action.

If the attributes (name, description, TriggerSet, ConditionSet, workflowActions) for the specified workflow do not change, the request does not need to include those attributes. If an attribute is not specified, the current value is retained. However, you must specify at least one attribute in the request.

If you only want to change the description of the workflow, you can make the following request:

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

Note If you provide two or more duplicate tags during a PUT, the value of the last duplicate tag is processed and all other duplicate tags are ignored.

URI:	http://<FQDN>/finesse/api/WorkflowAction/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/WorkflowAction/769
Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	<WorkflowAction> ...WorkflowAction Object... </WorkflowAction>
Request Parameters:	id (required): Maps to the primary key of the workflowAction entry name (required): The name of the workflow action type (required): The type of workflow action handledBy (required): Indicates what handles the action params (required): List of Params for the workflow action actionVariables (required): list of actionVariables for the workflow

HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error
Example Failure Response:	<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>

WorkflowAction—Delete

This API allows an administrator to delete an existing workflow action. The administrator references the existing WorkflowAction object by its ID.

URI:	http://<FQDN>/finesse/api/WorkflowAction/<id>
Example URI:	http://finesse1.xyz.com/finesse/api/WorkflowAction/768
Security Constraints:	Only administrators can use this API.
HTTP Method:	DELETE
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

Example Failure Response:	<ApiErrors>
	  <ApiError>
	  <ErrorData>Action 768 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>

WorkflowAction API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the WorkflowAction object.	—	The id in the URI maps to the primary key of the WorkflowAction.
name	String	The name of the workflow action.	—	Must be unique. Maximum of 64characters.
type	String	The type of workflow action	BROWSER_POP, HTTP_REQUEST	
handledBy	String	Indicates what handles the action when it is triggered by a workflow.	FINESSE_DESKTOP, OTHER	For FINESSE_DESKTOP, the Finesse workflow engine executes the action. For OTHER, the action event is published on the OpenAJAX hub but is not executed by the Finesse desktop. This allows a third-party gadget to execute the action.
params	Object	A list of Param subobjects.	—	

Parameter	Type	Description	Possible Values	Notes
-->Param	Object	Includes a name and value pair.	—	Params are flexible and can contain any value. Validation is based on the type of the WorkflowAction in which they are contained. See the following tables for more information.
--->name	String	The name of the parameter.	—	
--->value	String	The value of the parameter.	—	
actionVariables	Object	List of ActionVariable subobjects.	—	
-->ActionVariable	Object	Set of information about one ActionVariable.	—	You can assign up to five ActionVariable parameters to a workflow.
--->name	String	The name of the variable.	—	Maximum of 32 characters.
--->node	String	The XPath to extract from the dialog XML.	—	Maximum of 500 characters. 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.
--->type	String	Indicates the type of variable	CUSTOM, SYSTEM	
--->testValue	String	The value used to test the variable.	—	Maximum of 128 characters.

Param Values (BROWSER_POP)

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

Param (HTTP_REQUEST)

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

WorkflowAction API Errors

Status	Error Type	Description
400	Bad Request	The request body is invalid.
400	Finesse API Error	API error such as the object is stale or does not exist.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
404	Not Found	The specified resource cannot be found.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

Team

The Team object represents a team and the resources associated with that team. For more information, see Team, on page 114.

The administrator uses the Team configuration APIs to assign or unassign resources (such as reason codes, wrap-up reasons, phonebooks, layout configuration, and workflows) to a specific team.

Team APIs

Team—Get List

This API allows an administrator to get a list of teams. The team must have agents or supervisors assigned to it for the team to appear in the retrieved list.

URI:	http://<FQDN>/finesse/api/Teams
Example URI:	http://finesse1.xyz.com/finesse/api/Teams
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 403: Forbidden 500: Internal Server Error
Example Response:	<Teams> <Team> ...Summary Team Object... </Team> <Team> ...Summary Team Object... </Team> <Team> ...Summary Team Object... </Team> </Teams>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Unauthorized</ErrorType> <ErrorMessage>The user is not authorized to perform this operation.</ErrorMessage> </ApiError> </ApiErrors>

Team—Get List of Reason Codes

This API allows an administrator to get 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 section ReasonCode, on page 138.

URI:	http://<FQDN>/finesse/api/Team/<id>/ReasonCodes?category=<category>
Example URI:	http://finesse1.xyz.com/finesse/api/Team/574/ReasonCodes?category=NOT_READY
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

Example Response:	`<ReasonCodes category="NOT_READY">`
	`  <ReasonCode>`
	`  ... Full Reason Code Object ...`
	`  </ReasonCode>`
	`  <ReasonCode>`
	`  ... Full Reason Code Object ...`
	`  </ReasonCode>`
	`  <ReasonCode>`
	`  ... Full Reason Code Object ...`
	`  </ReasonCode>`
	`....`
	`</ReasonCodes>`

Example Failure Response:	`<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>`

Team—Update List of Reason Codes

This API allows an administrator to assign or unassign a list of reason codes of the specified category to a team.

If multiple users try to update the reason code for the same team at the same time, the changes made by the last user to update overwrite the 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 that you assign or unassign 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 is 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 results 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/574/ReasonCodes?category=NOT_READY
Security Constraints:	Only administrators can use this API.
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>
Request Parameters:	id (required): The database ID for the team category (required): The category of reason code (NOT_READY or LOGOUT)

HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error
Example Failure Response:	<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>

Team—Get List of Wrap-Up Reasons

This API allows an administrator to get a list of wrap-up reasons assigned to a specific team. The list is in the same format as defined in the section WrapUpReason, on page 146.

URI:	http://<FQDN>/finesse/api/Team/<id>/WrapUpReasons
Example URI:	http://finesse1.xyz.com/finesse/api/Team/574/WrapUpReasons
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—

HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

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

Example Failure Response:	`<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>`

Team—Update List of Wrap-Up Reasons

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

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

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

URI:	http://<FQDN>/finesse/api/Team/<Id>/WrapUpReasons
Example URI:	http://finesse1.xyz.com/finesse/api/Team/574/WrapUpReasons
Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<WrapUpReasons> <WrapUpReason> <uri>/finesse/api/WrapUpReason/123</uri> </WrapUpReason> <WrapUpReason> <uri>/finesse/api/WrapUpReason/456</uri> </WrapUpReason> <WrapUpReason> <uri>/finesse/api/WrapUpReason/789</uri> </WrapUpReason> .... </WrapUpReasons>
Request Parameters:	id (required): The database ID for the team
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorData>574</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>

Team—Get List of Phone Books

This API allows an administrator to get a list of phone books assigned to a specific team. The list is in the same format as defined in the section PhoneBook, on page 158.

URI:	http://<FQDN>/finesse/api/Team/<id>/PhoneBooks
Example URI:	http://finesse1.xyz.com/finesse/api/Team/574/PhoneBooks
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML

Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

Example Response:	`<PhoneBooks>`
	`<PhoneBook>`
	`... Full PhoneBook Object ...`
	`</PhoneBook>`
	`<PhoneBook>`
	`... Full PhoneBook Object ...`
	`</PhoneBook>`
	`<PhoneBook>`
	`... Full PhoneBook Object ...`
	`</PhoneBook>`
	`....`
	`</PhoneBooks>`

Example Failure Response:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorData>574</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>`

Team—Update List of Phone Books

This API allows an administrator to assign or unassign a list of phone books to a team.

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

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

URI:	http://<FQDN>/finesse/api/Team/<Id>/PhoneBooks
Example URI:	http://finesse1.xyz.com/finesse/api/Team/574/PhoneBooks
Security Constraints:	Only administrators can use this API.

HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	<PhoneBooks> <PhoneBook> <uri>/finesse/api/PhoneBook/123</uri> </PhoneBook> <PhoneBook> <uri>/finesse/api/PhoneBook/456</uri> </PhoneBook> <PhoneBook> <uri>/finesse/api/PhoneBook/789</uri> </PhoneBook> .... </PhoneBooks>
Request Parameters:	id (required): The database ID for the team
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorData>574</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>

Team—Get Layout Configuration

This API allows an administrator to get the layout configuration assigned to a specific team.

URI:	http://<FQDN>/finesse/api/Team/<id>/LayoutConfig
Example URI:	http://finesse1.xyz.com/finesse/api/Team/574/LayoutConfig
Security Constraints:	Only administrators can use this API.

HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

Example 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>`

Example Failure Response:	`<ApiErrors>`
	`<ApiError>`
	`<ErrorData>574</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>`

Team—Update Layout Configuration

This API allows an administrator to assign or unassign a layout configuration to a team.

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

URI:	http://<FQDN>/finesse/api/Team/<Id>/LayoutConfig

Example URI:	http://finesse1.xyz.com/finesse/api/Team/574/LayoutConfig
Security Constraints:	Only administrators can use this API.
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 default layout to a team:
	`<TeamLayoutConfig>`
	`<useDefault>true</useDefault>`
	`</TeamLayoutConfig>`

Request Parameters:	id (required): The database ID for the team useDefault (required): Whether to use the default desktop layout for this team layoutxml (required if useDefault is false): The XML data that determines the layout of the Finesse desktop
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error

Example Failure Response:	<ApiErrors> <ApiError> <ErrorData>574</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>

Team—Get List of Workflows

This API allows an administrator to get a list of workflows assigned to a specific team. The list is in the same format as defined in the section Workflow, on page 173.

URI:	http://<FQDN>/finesse/api/Team/<id>/Workflows
Example URI:	http://finesse1.xyz.com/finesse/api/Team/574/Workflows
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error

Example Response:	<Workflows> <Workflow> ... Summary Workflow Object ... </Workflow> <Workflow> ... Summary Workflow Object ... </Workflow> <Workflow> ... Summary Workflow Object ... </Workflow> .... </Workflows>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorData>574</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>

Team—Update List of Workflows

This API allows an administrator to assign or unassign a list of workflows to a team.

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

This list includes all workflows that are assigned to a team. Any workflows that you assign or unassign overwrite the current workflow list.

Note Because the order in which workflows are evaluated is important, the order of the workflows in the list is preserved in the GET method (see Team—Get List of Workflows, on page 213).

URI:	http://<FQDN>/finesse/api/Team/<Id>/workflows
Example URI:	http://finesse1.xyz.com/finesse/api/Team/574/Workflows
Security Constraints:	Only administrators can use this API.
HTTP Method:	PUT
Content Type:	Application/XML
Input/Output Format:	XML

HTTP Request:	<Workflows> <Workflow> <uri>/finesse/api/Workflow/123</uri> </Workflow> <Workflow> <uri>/finesse/api/Workflow/456</uri> </Workflow> <Workflow> <uri>/finesse/api/Workflow/789</uri> </Workflow> .... </Workflows>
Request Parameters:	id (required): The database ID for the team
HTTP Response:	200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 401: Invalid Authorization User Specified 403: Forbidden 404: Not Found 500: Internal Server Error
Example Failure Response:	<ApiErrors> <ApiError> <ErrorData>574</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>

Team API Parameters

Parameter	Type	Description	Possible Values	Notes
uri	String	The URI to get a new copy of the Team, ReasonCode, WrapUpReason, LayoutConfig, or Workflow object.	—	
id	String	The unique identifier for the team.		
name	String	The name of the team.	—	

Parameter	Type	Description	Possible Values	Notes
category	String	Specifies the type of reason code.	NOT_READY, LOGOUT	
useDefault	Boolean	Determines whether to use the default desktop layout for this team.	true, false	
layoutxml	String	The XML data that determines the desktop layout.	—	If useDefault is set to true and the layoutxml is provided in a request, the layoutxml is ignored.

Team API Errors

Status	Error Type	Description
400	Bad Request	The request body is invalid.
400	Finesse API Error	API error such as the object is stale or does not exist.
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
404	Not Found	The specified resource cannot be found.
500	Internal Server Error	Any runtime exception is caught and responded with this error.

SystemVariable

The SystemVariable object represents a variable that can be extracted from a Finesse event object and displayed on the Finesse desktop or used in a workflow.

The SystemVariable object is structured as follows:

```
<SystemVariable>
   <name></name>
   <node></node>
</SystemVariable>
```

SystemVariable APIs

SystemVariable—List

This API allows an administrator to get a list of all system variables.

Note The Outbound variable BACustomerNumber only appears in the response when Finesse is deployed with Unified CCX.

URI:	http://<FQDN>/finesse/api/SystemVariables
Example URI:	http://finesse1.xyz.com/finesse/api/SystemVariables
Security Constraints:	Only administrators can use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success 401: Authorization Failure 403: Forbidden 500: Internal Server Error

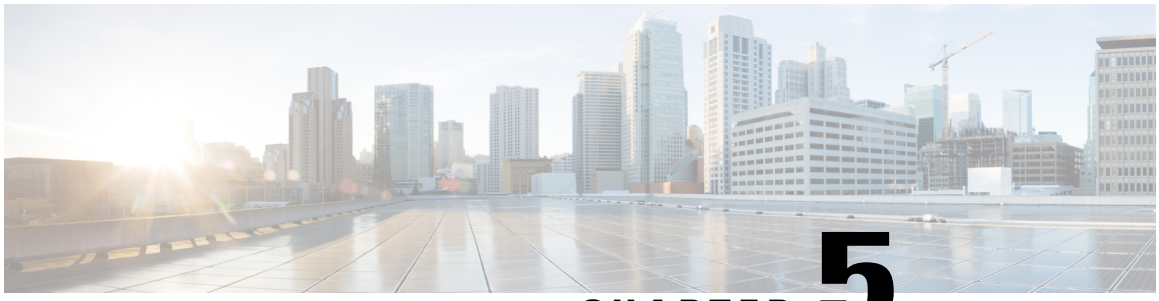
Example Response:	<SystemVariables> <SystemVariable> <name>callVariable1</name> <node>//Dialog/mediaProperties/callvariables/CallVariable/ name[.="callVariable1"]/../value</node> </SystemVariable> <SystemVariable> <name>callVariable2</name> <node>//Dialog/mediaProperties/callvariables/CallVariable/ name[.="callVariable2"]/../value</node> </SystemVariable> <SystemVariable> <name>callVariable3</name> <node>//Dialog/mediaProperties/callvariables/CallVariable/ name[.="callVariable3"]/../value</node> </SystemVariable> ...Other callVariables (4 through 10)... <SystemVariable> <name>BAAccountNumber</name> <node>//Dialog/mediaProperties/callvariables/CallVariable/ name[.="callVariable3"]/../value</node> </SystemVariable> <SystemVariable> <name>callVariable5</name> <node>//Dialog/mediaProperties/callvariables/CallVariable/ name[.="BAAccountNumber"]/../value</node> </SystemVariable> <SystemVariable> <name>BABuddyName</name> <node>//Dialog/mediaProperties/callvariables/CallVariable/ name[.="BABuddyName"]/../value</node> </SystemVariable> ...Other Outbound Variables... <SystemVariable> <name>DNIS</name> <node>//Dialog/mediaProperties/DNIS</node> <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>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>
Example Failure Response:	No API errors are returned. Responses are 401/403/404 Errors.

SystemVariable API Parameters

Parameter	Type	Description	Possible Values	Notes
name	String	A unique name for the variable.	—	The name is used as a readable, unique key for the variable. Maximum of 32 characters.
node	String	The XPath to use to extract the value of this variable from an XMPP event that may contain the variable.	—	Maximum of 500 characters.

SystemVariable API Errors

Status	Error Type	Description
401	Authorization Failure	Unauthorized (for example, the user is not yet authenticated in the Web Session). The user is not authorized to use the API (the user is not an administrator).
403	Forbidden	The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.
500	Internal Server Error	Any runtime exception is caught and responded with this error.



CHAPTER 5

Cisco Finesse Serviceability APIs

SystemInfo

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, and the hostnames of the primary and secondary (if configured) Finesse nodes.

The SystemInfo object is structured as follows:

```
<SystemInfo>
   <deploymentType><deploymentType>
   <currentTimeStamp></currentTimeStamp>
   <status></status>
   <timezoneOffset></timezoneOffset>
   <xmppDomain></xmppDomain>
   <xmppPubSubDomain></xmppPubSubDomain>
   <primaryNode>
      <host></host>
   </primaryNode>
   <secondaryNode>
      <host></host>
   </secondaryNode>
</SystemInfo>
```

SystemInfo—Get

This API allows a user to get information about the Finesse system.

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
Example Response:	<SystemInfo> <deploymentType>UCCE<deploymentType> <currentTimeStamp>2014-01-27T13:07:08.687Z</currentTimeStamp> <status>IN_SERVICE</status> <timezoneOffset>300</timezoneOffset> <xmppDomain>xmppserver.xyz.com</xmppDomain> <xmppPubSubDomain>pubsub.xmppserver.xyz.com</xmppPubSubDomain> <primaryNode> <host>10.1.1.1</host> </primaryNode> <secondaryNode> <host>10.1.1.2</host> </secondaryNode> </SystemInfo>
Example Failure Response:	<ApiErrors> <ApiError> <ErrorType>Internal Server Error</ErrorType> <ErrorMessage>Runtime Exception</ErrorMessage> <ErrorData></ErrorData> </ApiError> </ApiErrors>

SystemInfo API Parameters

Parameter	Type	Description	Possible Values	Notes
currentTimeStamp	String	The current time (GMT time) in the following format: YYYY-MM-DDThh:MM:ss.SSZ	—	
deploymentType	String	The type of deployment for Finesse	UCCE, UCCX	

Parameter	Type	Description	Possible Values	Notes
status	String	The state of the Finesse system	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.	
timezoneOffset	Integer	The difference (in minutes) between the server time and GMT time.	—	For example, a value of 300 means the server time is GMT + 5 hours. A value of -300 means the server time is GMT - 5 hours.
xmppDomain	String	The XMPP server domain.		
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.		

SystemInfo API Errors

Status	Error Type	Description
500	Internal Server Error	Any runtime exception is caught and responded with this error.

Diagnostic Portal APIs

Diagnostic Portal APIs are primarily to integrate Finesse with the Cisco Prime Contact Center Module and get information about the health of the Finesse system. You can access these APIs only through HTTPS.

Diagnostic Portal—Get Performance Information

The Diagnostic Portal——Get Performance Information API allows an administrator to get performance information to a Diagnostic Portal object.

URI:	https://FQDN/finesse-dp/rest/DiagnosticPortal/GetPerformanceInformation
Example URI:	https://finesse1.xyz.com/finesse-dp/rest/DiagnosticPortal/GetPerformanceInformation
Security Constraints:	A user must be signed in as an administrator to use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—
HTTP Response:	200: Success **Note** All requests that reach the Finesse Diagnostic Portal web application return a 200 response. However, requests that are not successfully handled return XML that includes an error code and optionally, an error string. 401: Authorization Failure 404: Not Found 500: Internal Server Error

Successful Response:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <dp:GetPerformanceInformationReply ReturnCode="0" xmlns:dp="http://www.cisco.com/vtg/diagnosticportal"> <dp:Schema Version="1.0" /> <dp:PerformanceInformation> <dp:PropertyList> <dp:Property Value="109441280" Name="Tomcat/Heap Memory Utilized"/> <dp:Property Value="50921904" Name="Tomcat/Non Heap Memory Utilized"/> <dp:Property Value="0" Name="CTI Statistics/Incoming Responses Queue"/> <dp:Property Value="0" Name="CTI Statistics/Outgoing Responses Queue"/> <dp:Property Value="0" Name="Tomcat/Average Request Process Time"/> <dp:Property Value="0" Name="Tomcat/Longest Request Process Time"/> <dp:Property Value="1.47" Name="Average System Load"/> <dp:Property Value="183" Name="Tomcat/Thread Count"/> <dp:Property Value="183" Name="Tomcat/Peak Thread Count"/> <dp:Property Value="0" Name="CTI Statistics/Events In Queue"/> <dp:Property Value="0" Name="CTI Statistics/Decoding Responses Queue"/> <dp:Property Value="0" Name="Active Totals/Logged In Agents"/> <dp:Property Value="0" Name="Active Totals/Current Calls"/> <dp:Property Value="0" Name="Running Totals/Calls Received or Initiated"/> <dp:Property Value="0" Name="Running Totals/Calls Failed"/> </dp:PropertyList> </dp:PerformanceInformation> </dp:GetPerformanceInformationReply>
Example Failure Response:	<?xml version="1.0" encoding="UTF-8" ?> <dp:GetProductLicenseReply ReturnCode="1" ErrorString="License file license.txt could not be read" xmlns:dp="http://www.cisco.com/vtg/diagnosticportal"> <dp:Schema Version="1.0"/> </dp:GetProductLicenseReply>

Diagnostic Portal—Get Product Version

This API allows an administrator to get product version information for Finesse.

URI:	https://FQDN/finesse-dp/rest/DiagnosticPortal/GetProductVersion
Example URI:	https://finesse1.xyz.com/finesse-dp/rest/DiagnosticPortal/GetProductVersion
Security Constraints:	A user must be signed in as an administrator to use this API.
HTTP Method:	GET
Content Type:	Application/XML
Input/Output Format:	XML
HTTP Request:	—

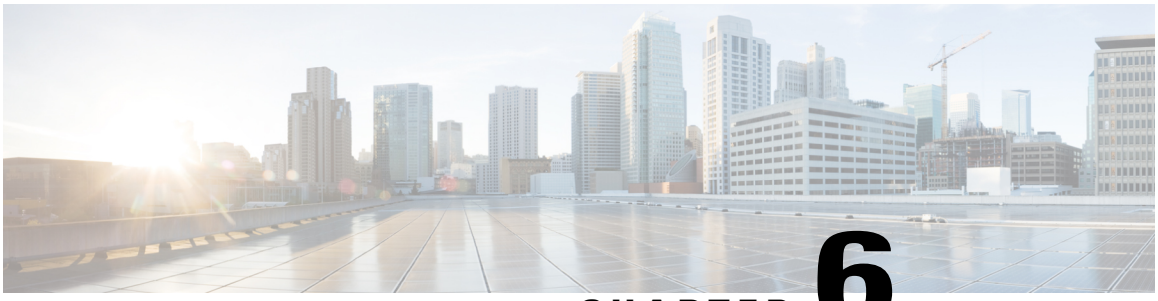
HTTP Response:	200: Success **Note** All requests that reach the Finesse Diagnostic Portal web application return a 200 response. However, requests that are not successfully handled return XML that includes an error code and optionally, an error string. 401: Authorization Failure 404: Not Found 500: Internal Server Error

Successful Response:	`<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`
	`<dp:GetProductVersionReply xmlns:dp="http://www.cisco.com/vtg/ diagnosticportal" ReturnCode="0">`
	`<dp:Schema Version="1.0"/>`
	`<dp:ProductVersion VersionString="10.5(1)" Maintenance="1" Minor="5" Major="10" Name="Cisco Finesse"/>`
	`<dp:ComponentVersionList/>`
	`</dp:GetProductVersionReply>`

Example Failure Response:	`<?xml version="1.0" encoding="UTF-8" ?>`
	`<dp:GetProductLicenseReply ReturnCode="1" ErrorString="License file license.txt could not be read" xmlns:dp="http://www.cisco.com/vtg/ diagnosticportal">`
	`<dp:Schema Version="1.0"/>`
	`</dp:GetProductLicenseReply>`

Diagnostic Portal API Errors

Status	Error Type	Description
401	Authorization Error	The user is not authorized to access this API.
404	Not Found	The resource is not found (for example, the DiagnosticPortal has been deleted).
500	Internal Server Error	Any runtime exception is caught and responded with this error.



CHAPTER 6

Cisco Finesse Notifications

About Cisco Finesse Notifications

The Cisco Finesse Web Service sends notifications to clients that subscribe to that class of resource.

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

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

Note 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

Finesse publishes notifications when a change occurs in the resource characteristics.

Subscription Management

Finesse clients can interface directly with the Cisco Finesse Notification Service to send subscribe and unsubscribe requests. Clients subscribe to 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.

To avoid increasing notification traffic for other users, use a full JID (username@domain/resource) when making explicit subscriptions.

Make sure to unsubscribe to any explicit subscriptions before disconnecting the XMPP session. Any subscriptions that are left behind persist on that node in the Cisco Finesse Notification Service.

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>
```

Perform a GET using the SystemInfo API (http://<server>/finesse/api/SystemInfo) to obtain connection details. The returned payload provides 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.
- The Cisco Finesse 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 Notification

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>
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> <ReasonCode> <uri>/finesse/api/ReasonCode/{id}</uri> <code>10</code> <label>Team Meeting</label> </ReasonCode> <settings> <wrapUpOnIncoming>OPTIONAL</wrapUpOnIncoming> </settings> <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>
Notification Triggers:	• Addition of a user • Deletion of a user • State change • First or last name change • Role change

Dialog Notification

Finesse sends a Dialog notification when information (or an action) changes for 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. .

  For a reservation call in Direct 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 calls 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.

  For a reservation call in Direct Preview Outbound mode, the toAddress is the extension of the agent on the outbound call.

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 Dialog collection for the user) /finesse/api/Dialog/{id} (when a Dialog within the Dialogs collection for the 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>
Sample Notification Payload:	

```
<Update>
   <data>
    <dialog>
      <associatedDialogUri></associatedDialogUri>
      <fromAddress>1081001</fromAddress>
      <id>16804377</id>
      <mediaProperties>
        <DNIS>1081002</DNIS>
        <callType>AGENT_INSIDE</callType>
        <callvariables>
          <CallVariable>
            <name>callVariable1</name>
            <value></value>
          ...
        </callvariables>
        <dialedNumber>1081002</dialedNumber>
      </mediaProperties>
      <mediaType>Voice</mediaType>
      <participants>
        <Participant>
          <actions>
            <action>TRANSFER_SST</action>
            <action>CONSULT_CALL</action>
            <action>HOLD</action>
            <action>UPDATE_CALL_DATA</action>
            <action>SEND_DTMF</action>
            <action>DROP</action>
          </actions>
          <mediaAddress>1081001</mediaAddress>
          <mediaAddressType>AGENT_DEVICE</mediaAddressType>
          <startTime>2014-02-04T15:33:16.653Z</startTime>
          <state>ACTIVE</state>
          <stateCause></stateCause>

<stateChangeTime>2014-02-04T15:33:16.653Z</stateChangeTime>
        </Participant>
        <Participant>
          <actions>
            <action>UPDATE_CALL_DATA</action>
            <action>DROP</action>
            <action>RETRIEVE</action>
          </actions>
          <mediaAddress>1081002</mediaAddress>
          <mediaAddressType>AGENT_DEVICE</mediaAddressType>
          <startTime>2014-02-04T15:33:16.653Z</startTime>
          <state>HELD</state>
          <stateCause></stateCause>

<stateChangeTime>2014-02-04T15:33:27.584Z</stateChangeTime>
        </Participant>
      </participants>
      <state>ACTIVE</state>
      <toAddress>1081002</toAddress>
      <uri>/finesse/api/Dialog/16804377</uri>
    </dialog>
  </data>
  <event>PUT</event>
  <requestId></requestId>
  <source>/finesse/api/Dialog/16804377</source>
</Update>
```

Notification Triggers:

- Incoming call
- Modification of participant state (for example, when a participant answers or hangs up a call)
- A new participant on the call
- Modification of the call data or actions

Dialog CTI Error Notification

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 asynchronous updates. 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>
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>
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

Team Notification

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>2012-03-01T17:58:21.123Z</stateChangeTime>

        <ReasonCode>
           <uri>finesse/api/ReasonCode/1</uri>
           <code>10</code>
           <label>Team Meeting</label>
           <category>NOT_READY</category>
           <id>1</id>
        </ReasonCode>
      </user>
   </data>
</Update>
```

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>1004</loginId>
               <firstName>Charles</firstName>
               <lastName>Norrad</lastName>
               <pendingState></pendingState>
               <state>LOGOUT</state>

<stateChangeTime>2012-03-01T17:58:21.123Z</stateChangeTime>
            </User>
            <User>
               <uri>/finesse/api/User/9876</uri>
               <loginId>9876</loginId>
               <firstName>Jack</firstName>
               <lastName>Brown</lastName>
               <state>NOT_READY</state>

<stateChangeTime>2012-03-01T17:58:21.134Z</stateChangeTime>
               <ReasonCode>
                  <uri>/finesse/api/ReasonCode/1</uri>
                  <code>10</code>
                  <label>Team Meeting</label>
                  <category>NOT_READY</category>
                  <id>1</id>
               </ReasonCode>
            </User>
            ... other users ...
         </users>
      </team>
   </data>
</Update>
```

Notification Triggers:

- Agent name is changed for an agent who belongs to the team
- Agent state is changed for an agent who belongs to the team

Queue Notifications

Finesse sends a queue notification every 10 seconds (if queue statistics change).

Note Finesse sends notifications for this node 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>
Sample Notification Payload (PUT):	<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 Notification Payload (DELETE):	<Update> <event>delete</event> <source>/finesse/api/Queue/1004</source> <requestId></requestId> <data> <Queue> <uri>/finesse/api/Queue/1004</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

User/Queue Notification

Finesse sends a User/Queues notification when a queue is added or removed from the user's list of queues or if a queue assigned to that user is removed from the system.

Note Finesse sends notifications for this node 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

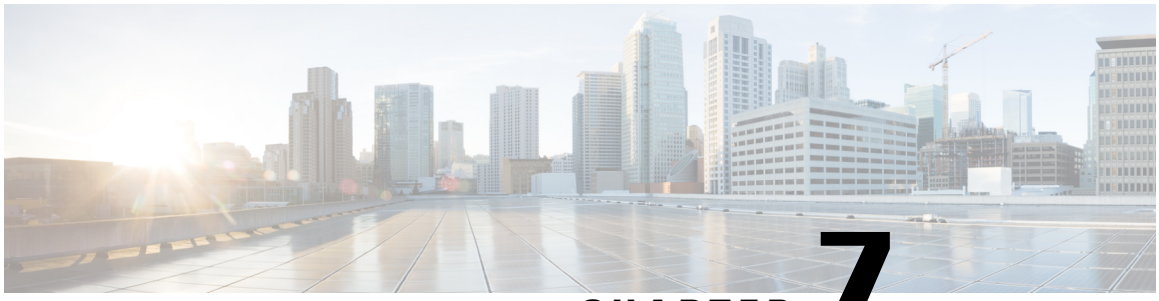
Payload (POST):	<Update> <event>{post}</event> <source>/finesse/api/User/{id}/Queues</source> <requestId></requestId> <data> <Queues> <Queue> <uri>/finesse/api/Queue/{id}</uri> <name>Sales</name> <statistics> <callsInQueue>3</callsInQueue> <startTimeOfLongestCallInQueue>2012-02-15 T17:58:21Z</startTimeOfLongestCallInQueue> <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>
Sample Notification Payload (POST):	

	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 Notification Payload (DELETE):	<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>
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

Name	Data Type	Description	Possible Values
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. For a Dialog Error notification, provides the list of ApiError objects that represent the failure conditions detected by the server.	The entire User, Team, Dialog, or Queue object in its most current, updated form. The Team object includes all of its agents. For the User/Queues object, specifies a list of queues that were added or deleted from the user's list.
Event	String	The type of modification that occurred to the User, Team, Dialog, Queue, or User/Queues object.	**PUT:** A property of the User, Dialog, Team, or Queue object that was modified. **DELETE:** A User, Dialog, Team, or Queue object has been deleted. For a User/Queues modification, the queues removed from the user's list of queues. **POST:** A User, Dialog, Team, or Queue object has been added. For a User/Queues modification, specifies the queues that were added to the user's list of queues.
Source	String	The resource location for the User, Dialog, Team, Queue, or User/Queues object 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
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.



CHAPTER 7

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 Finesse 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 244
- Desktop Presence and Forced Logout, page 245

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).

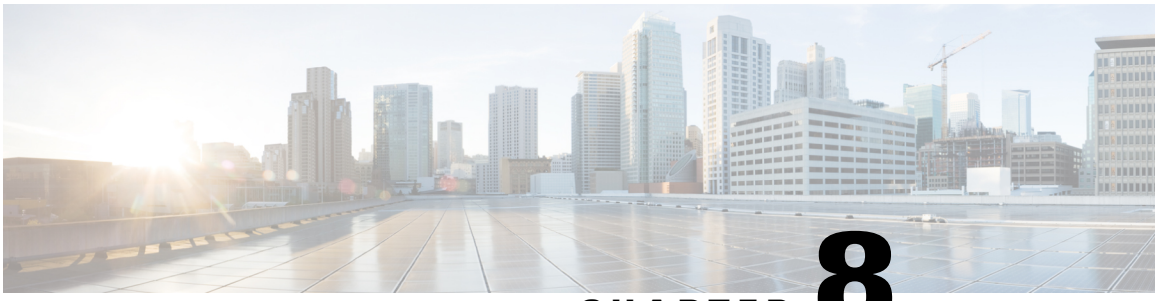
In a Unified CCX deployment, the agent is logged out.

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 8

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.

Name	Parameters	Description
<static> gadgets.views.requestNavigateTo (view)	view (string)-The view type to which the gadget is requesting to change.	Sets the view type of the gadget. If the parameter value equals "canvas", the gadget is requesting to be maximized within the tab on which it resides. If any other value is provided, the gadget is requesting to be restored to its default view.

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 in Finesse High Availability, on page 243 . 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 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 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.

Finesse Container Timer

Because too many timers that run concurrently can cause issues for JavaScript, you should not use setTimeout() or setInterval() directly. The Finesse container provides a service (the TimerTickEvent) that you can leverage for your third-party gadgets.

Finesse publishes the TimerTickEvent to the OpenAJAX hub every 1000 milliseconds. To use this service:

- Have the gadget subscribe to the TimerTickEvent:

  ```
  finesse.containerservices.ContainerServices.addHandler(finesse.containerservices.ContainerServices.Topics.
  TIMER_TICK_EVENT, callback);
  ```

- Define a callback method (see boilerplate gadget tick code - _timerTickHandler()) and, optionally, an update method (see boilerplate gadget tick code - _processTick()).

Cisco provides a boilerplate gadget tick code that you can use to define the callback method.

Boilerplate gadget tick code:

```
//Gadget defined field: _lastProcessedTimerTick
_lastProcessedTimerTick = null,

//Gadget defined field: _maxTimerCallbackThreshold
_maxTimerCallbackThreshold = 500,

//Gadget defined field: _forceTickProcessingEvery (10 seconds)
_forceTickProcessingEvery = 10000,

/**
  * Processes a timer tick - updating the UI.
  * @param start is the time that the tick was received
  * @returns {boolean} true
  */
_processTick = function (start) {
```

```
   //Developer's add UI update logic here
   //...
   //...

   _lastProcessedTimerTick = start;

  return true;
},

/**
  * Timer tick callback handler.
  * @param data
  */
_timerTickHandler  = function (timerTickEvent) {
   var start, end, diff, discardThreshold, processed;

  start = (new Date()).getTime();
  processed = false;

  //Prevent starvation of timer logic
  if (_lastProcessedTimerTick === null) {
     processed = _processTick(start);
  } else {
     if ((_lastProcessedTimerTick + _forceTickProcessingEvery) <= start) {
       //Force processing at least every _forceTickProcessingEvery milliseconds
       processed = _processTick(start);
     }
  }

  if (!processed) {
    //Allow Javascript to catch up and prevent lots of queued messages.
    discardThreshold = timerTickEvent.getDateQueued().getTime() +
(2*timerTickEvent.getTickFrequency());
    if (start <= discardThreshold) {
       _processTick(start);
    }

    //Otherwise - we're dropping this tick event
  }

  end = (new Date()).getTime();
  diff = end - start;
  if (diff > _maxTimerCallbackThreshold) {
   _clientLogs.log("GadgetXYZ took too long to process timer tick (_maxTimerCallbackThreshold
 exceeded).");
  }
},
```

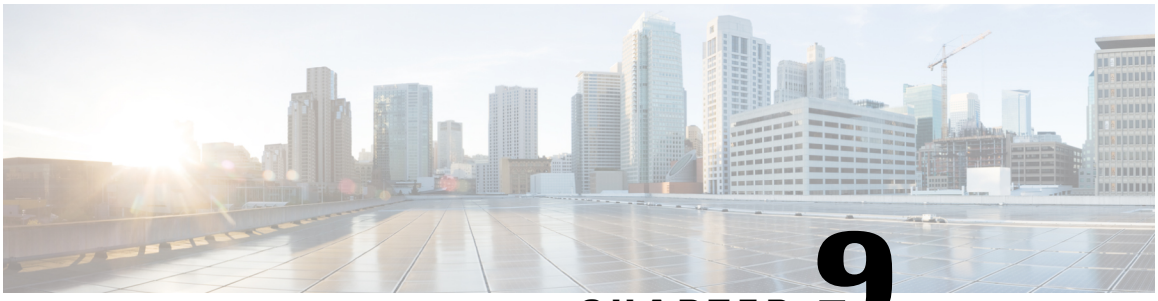
If you choose not to use the boilerplate gadget tick code, you should ensure the following:

- Callback calculates entry and exit time.
- Callback for timer tick is quick (log when callback takes to long - only when exceeding threshold).
- Callback provides discard capability (as outlined in the boilerplate gadget tick code) to prevent events from piling up.
- Callback adds a _lastProcessedTimerTick and uses it to force an update to occur at regular intervals (such as every 10 seconds). The intent is to prevent starvation in a heavily-loaded system that cannot respond quickly enough, such that all events are being discarded.

Note Because the timer callback triggers every 1 second and the JavaScript engine is single-threaded, it is important to process as quickly as possible. Using the boilerplate code makes gadget development issues more obvious and easier to debug.

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 9

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 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

Permissions

If a newly uploaded third-party gadget does not render via the desktop layout or when you launch it directly in a browser, the gadget files may not have the correct permissions. If gadget files do not have read permissions for everyone else (for example, the file permission is 770), Cisco Tomcat cannot read them. The minimum file permission should be 644.

If a gadget file does not have the correct permissions, when you launch it directly in the browser, you receive a 404 “Resource not available” error. When you try to launch the gadget via the desktop layout, you receive an error message that states the requested resource is not available.

To change file permissions on the Finesse server, use SFTP (CLI or client program) as shown in the following example:

```
$ sftp 3rdpartygadget@172.27.184.59
3rdpartygadget@172.27.184.59's password:
Connected to 172.27.184.59.
sftp> cd files
sftp> ls -l
---------- 1 751 751 0 Dec 6 19:40 MyGadget.xml
sftp> chmod 644 MyGadget.xml
Changing mode on /files/MyGadget.xml
sftp> ls -l
-rw-r--r-- 1 751 751 0 Dec 6 19:40 MyGadget.xml
sftp>
```

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.

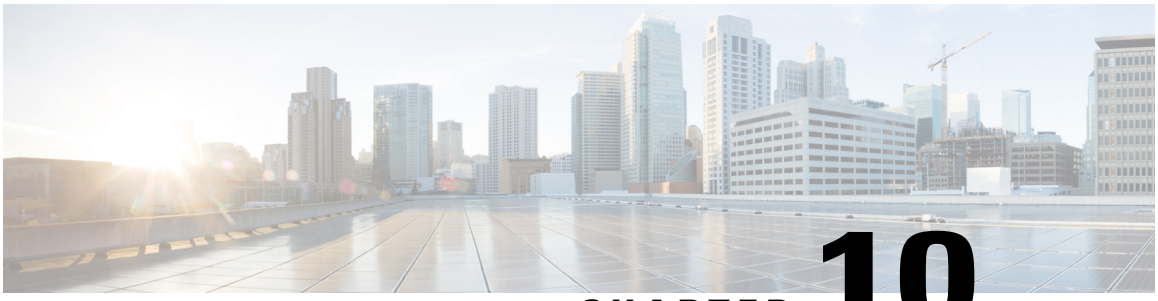
The 3rdpartygadget account password is not migrated across upgrades. After an upgrade, you must reset the password for the 3rdpartygadget account before you can make changes to third-party gadgets. You must reset the password on both the primary and secondary Finesse servers.

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 10

Documents and Documentation Feedback

Documents

The Cisco Finesse Web Services Developer Guide is available from Cisco DevNet at the following link:

https://developer.cisco.com/site/collaboration/contact-center/finesse/documentation/

If you have development questions, you can post them to the Cisco Finesse forums on Cisco DevNet, located at the following link: https://communities.cisco.com/community/developer/collaboration/contact-center/finesse.

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 Cisco DevNet at the following link: https://developer.cisco.com/site/collaboration/contact-center/finesse/documentation/

Documentation Feedback

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

We appreciate your comments.