THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2010–2017 Cisco Systems, Inc. All rights reserved.
# CONTENTS

## CHAPTER 1

**Introduction** 1

- What's New in Cisco Finesse 11.6(1) 1
- Cisco Finesse REST APIs 1
- JavaScript Library and Sample Gadgets 2
- Communication with the Cisco Finesse Web Service 3
  - Client Requests 3
  - HTTP Requests 5
  - HTTPS Requests 5
  - Real-Time Events 5
- API Parameter Types 6
- Cisco Finesse API Errors 7

## CHAPTER 2

**Lab Development Environment Validation with Cisco Finesse Web Services APIs** 9

- Environment and Tools 9
  - Postman 9
  - Pidgin for Windows 10
  - Adium for Mac OS X 13
- Cisco Finesse APIs 16
  - Sign In to Finesse 17
  - Change Agent State 18

## CHAPTER 3

**Cisco Finesse Desktop APIs** 21

- User 21
  - User APIs 22
    - User—Sign In to Finesse 22
    - User—Sign In as a Mobile Agent 24
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialog—Accept, Close, or Reject a Direct Preview Outbound Reservation</td>
<td>100</td>
</tr>
<tr>
<td>Dialog—Reclassify a Direct Preview Call</td>
<td>101</td>
</tr>
<tr>
<td>Dialog—Schedule or Cancel a Callback</td>
<td>103</td>
</tr>
<tr>
<td>Dialog API Parameters</td>
<td>104</td>
</tr>
<tr>
<td>State (Dialog) Parameter Values</td>
<td>112</td>
</tr>
<tr>
<td>Actions Parameter Values</td>
<td>115</td>
</tr>
<tr>
<td>State (Participant) Parameter Values</td>
<td>118</td>
</tr>
<tr>
<td>CTI Event Mappings for Dialog and Participant States</td>
<td>120</td>
</tr>
<tr>
<td>Outbound Call Types and BAStatus</td>
<td>128</td>
</tr>
<tr>
<td>Disposition Code Parameter Values for Nonvoice Tasks</td>
<td>130</td>
</tr>
<tr>
<td>Dialog API Errors</td>
<td>132</td>
</tr>
<tr>
<td>Queue</td>
<td>134</td>
</tr>
<tr>
<td>Queue APIs</td>
<td>134</td>
</tr>
<tr>
<td>Queue—Get Queue</td>
<td>134</td>
</tr>
<tr>
<td>Queue—Get List of Queues for User</td>
<td>136</td>
</tr>
<tr>
<td>Queue API Parameters</td>
<td>138</td>
</tr>
<tr>
<td>Configuring Queue Statistics</td>
<td>139</td>
</tr>
<tr>
<td>Queue API Errors</td>
<td>140</td>
</tr>
<tr>
<td>Team</td>
<td>140</td>
</tr>
<tr>
<td>Team APIs</td>
<td>141</td>
</tr>
<tr>
<td>Team—Get Team</td>
<td>141</td>
</tr>
<tr>
<td>Team API Parameters</td>
<td>143</td>
</tr>
<tr>
<td>Team API Errors</td>
<td>144</td>
</tr>
<tr>
<td>ClientLog</td>
<td>144</td>
</tr>
<tr>
<td>ClientLog—Post to Finesse</td>
<td>145</td>
</tr>
<tr>
<td>ClientLog API Parameters</td>
<td>146</td>
</tr>
<tr>
<td>ClientLog API Errors</td>
<td>146</td>
</tr>
<tr>
<td>Task Routing APIs</td>
<td>146</td>
</tr>
<tr>
<td>Media</td>
<td>147</td>
</tr>
<tr>
<td>Media APIs</td>
<td>147</td>
</tr>
<tr>
<td>Agent States for Nonvoice Media</td>
<td>156</td>
</tr>
<tr>
<td>Media API Parameters</td>
<td>159</td>
</tr>
<tr>
<td>Media API Errors</td>
<td>163</td>
</tr>
<tr>
<td>Dialog APIs for Nonvoice Tasks</td>
<td>163</td>
</tr>
</tbody>
</table>
User APIs for Nonvoice Tasks 165

Single Sign-On 166
  Single Sign-On APIs 166
    Single Sign-On—Test API 166
    Single Sign-On—Fetch Access Token 167
    Single Sign-On—Refresh Existing Access Token 169

CHAPTER 4

Cisco Finesse Configuration APIs 171
  SystemConfig 172
    SystemConfig APIs 172
      SystemConfig—Get 172
      SystemConfig—Set 173
    SystemConfig API Parameters 174
    SystemConfig API Errors 175
  ClusterConfig 176
    ClusterConfig APIs 176
      ClusterConfig—Get 176
      ClusterConfig—Set 177
    ClusterConfig API Parameters 178
    ClusterConfig API Errors 178
  EnterpriseDatabaseConfig 178
    EnterpriseDatabaseConfig APIs 179
      EnterpriseDatabaseConfig—Get 179
      EnterpriseDatabaseConfig—Set 180
    EnterpriseDatabaseConfig API Parameters 181
    EnterpriseDatabaseConfig API Errors 182
  LayoutConfig 183
    LayoutConfig APIs 186
      LayoutConfig—Get 186
      LayoutConfig—Set 187
    LayoutConfig API Parameters 188
    LayoutConfig API Errors 188
  ReasonCode 188
    ReasonCode APIs 189
PhoneBook—Import Contact List (CSV) 221
PhoneBook—Import Contact List (XML) 222
PhoneBook—Export Contact List 223
PhoneBook API Parameters 224
PhoneBook API Errors 225
Contact 225
Contact APIs 226
   Contact—Get 226
   Contact—Get List 227
   Contact—Create 228
   Contact—Update 229
   Contact—Delete 229
Contact API Parameters 230
Contact API Errors 231
Workflow 231
Workflow APIs 236
   Workflow—Get 236
   Workflow—Get List 239
   Workflow—Create 240
   Workflow—Update 241
   Workflow—Delete 243
Workflow API Parameters 244
Workflow API Errors 248
WorkflowAction 249
WorkflowAction APIs 251
   WorkflowAction—Get 251
   WorkflowAction—Get List 251
   WorkflowAction—Create 252
   WorkflowAction—Update 254
   WorkflowAction—Delete 255
WorkflowAction API Parameters 256
WorkflowAction API Errors 260
Team 260
Team APIs 261
Resources 289
  User Notification 289
  Dialog Notification 290
  Dialogs/Media Notification 297
  Dialog CTI Error Notification 300
  Team Notification 301
  Queue Notifications 302
  User/Queue Notification 304
  Media Notification 306
  Media and Dialogs/Media Asynchronous Error Notification 307
  Notification Parameters 311
Managing Notifications in Third-Party Applications 312
  Connect to XMPP over HTTP (BOSH) using Finesse EventTunnel 313
  Connect to XMPP over TCP 314

CHAPTER 7
Finesse High Availability 317
  Failure Scenarios 318
  Desktop Presence and Forced Logout 318
  Failure Handling for Task Routing Clients 320

CHAPTER 8
Finesse Desktop Gadget Development 323
  Finesse Gadgets 323
    Gadget Description 324
    Simple Example Gadget 326
    Import Finesse JavaScript API 327
    alternateHosts Configuration 328
  Supported OpenSocial Features 328
    Gadget Specification XML Features 328
      Required Module pref Feature 329
      Loading Indicator Feature 329
      APIs Available to Gadget JavaScript 330
      Gadget Preferences 331
    Caveats 332
  Gadget Caching 332
CHAPTER 1

Introduction

• What's New in Cisco Finesse 11.6(1), on page 1
• Cisco Finesse REST APIs, on page 1
• JavaScript Library and Sample Gadgets, on page 2
• Communication with the Cisco Finesse Web Service, on page 3
• API Parameter Types, on page 6
• Cisco Finesse API Errors, on page 7

What's New in Cisco Finesse 11.6(1)

• The Cisco Finesse Notification Service (OpenFire) has been upgraded from version 3.8.2 to version 4.0.3 as part of regular maintenance activity.

• For gadget development, Finesse server and client connections only support TLS 1.2 by default.

• Secondary call ID is added in the additional element of the primary dialog API for consult, transfer and conference calls scenarios.

• queueName and queueNumber fields are included in call variables.

• Callkeyid and callkeyprefix parameters now included in the Dialog API.

• Use CLI commands to enable and disable the queue statistics.

• systemCode attribute included in the ReasonCode object.

• status and statusReason attributes included in the SystemInfo object.

• Pending state included as a notification trigger.

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
- Task Routing
- Mobile Agents
- Workflows
- TeamMessages
- Desktop Chat

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 several sample gadgets to help jump-start your gadget development. The JavaScript library provides a substantial amount of fundamental code infrastructure that you would otherwise need to write yourself.

- You can access the JavaScript library at the following URL: http(s)://<FQDN>:<port>/desktop/assets/js/finesse.js
- You can access the JavaScript documentation at the following URL: http(s)://<FQDN>:<port>/desktop/assets/js/doc/index.html
- You can access JQuery at the following URL: http(s)://<FQDN>:<port>/desktop/assets/js/jquery.min.js.

**Note**

For the proper functioning of the JavaScript library, you must import both the Finesse JavaScript library and JQuery.

- If you have third-party gadgets loaded on Finesse, the third-party gadgets can access the JavaScript library at: /desktop/assets/js/finesse.js.
- The sample gadgets are available from Cisco DevNet at the following link: https://developer.cisco.com/site/finesse/.
Communication with the Cisco Finesse Web Service

The Cisco Finesse Notification Service name in the following diagram is specific to Unified CCE deployments. In a Unified CCX deployment, the notification service is named the Cisco Unified CCX Notification Service.

Figure 1: Finesse API and Event Flow

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/<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 is 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:

```xml
<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 Finesse Tomcat is running, Finesse rejects any API requests and returns an HTTP 503 error. The error appears as follows:

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

If the Cisco Finesse 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 YWdlbnRiYXJ0b3dza2k6Y2FybWljaGFibA=="
```

where "YWdlbnRiYXJ0b3dza2k6Y2FybWljaGFibA==" is the Base64-encoded string of "agentbartowski:carmichael" (agentbartowski being the username and carmichael being the password).
In case of Single Sign-On mode, the Authorization header would contain the following string:

```
Bearer <authtoken>
```

where the authtoken has to be fetched from IDS through the ADFS server.

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 8445. 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 URLs and example URLs. If you want to make HTTP requests in a Unified CCX deployment, include the port number in the URLs:

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.)
- Use port 8445.

**Note**

For gadget development, Finesse server and client connections only support TLS 1.2 by default.

**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) protocol. Applications that need to communicate with the Notification Service must use XMPP over the BOSH (Bidirectional-streams Over Synchronous HTTP)/WebSocket transport.

All real-time events are sent over HTTPS.

BOSH/WebSocket 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/WebSocket (XEP-0124).

Client applications can communicate with the Cisco Finesse Notification Service through BOSH/WebSocket over HTTPS, using the binding URI https://<FQDN>:7443/http-bind. Developers can create their own BOSH/WebSocket library or use any that are available publicly.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>String</td>
<td>The password used in the request header to make any Finesse API request. Finesse supports a &quot;Basic&quot; authorization scheme only and authorization is required for each Finesse API request.</td>
</tr>
<tr>
<td>username</td>
<td>String</td>
<td>The username used in the request header to make any Finesse API request. Finesse supports a &quot;Basic&quot; authorization scheme only and authorization is required for each Finesse API request.</td>
</tr>
</tbody>
</table>

Body Parameter

A body parameter (also known as a complex parameter) appears in the body of the message. In the following example, targetMediaAddress and requestedAction are body parameters.

```xml
<Dialog>
  <targetMediaAddress>1001001</targetMediaAddress>
  <requestedAction>HOLD</requestedAction>
</Dialog>
```

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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>A logical data type that has one of two values: true or false.</td>
</tr>
<tr>
<td>Integer</td>
<td>A 32-bit wide integer.</td>
</tr>
<tr>
<td>Long</td>
<td>A 64-bit wide integer.</td>
</tr>
<tr>
<td>String</td>
<td>A variable-length string. If a maximum length exists, it is listed with the parameter description.</td>
</tr>
</tbody>
</table>

**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 Cisco Unified Contact Center Enterprise*, which you can find at [https://developer.cisco.com/site/cti-protocol/documentation/](https://developer.cisco.com/site/cti-protocol/documentation/).

For Finesse deployments with Unified CCX, see the *Cisco Unified Contact Center Express CTI Protocol Developer Guide*.
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, on page 9
- Cisco Finesse APIs, on page 16

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.
- Tools: Postman and Pidgin for Windows or Adium for Mac OS X.

Note

Postman, Pidgin and Adium are meant to aid in development; however, they are not officially supported.

Postman

Postman 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 Postman from https://www.getpostman.com/.


To test an API in Postman, follow these steps:
1. Copy and paste the URI for the API request from this Developer Guide into a text editor. For example, to enter the URI for signing in, copy the URI from the User—Sign In to Finesse API. Examine the pasted code for case sensitivity and format and remove any carriage returns.

2. Update the URI with the IP address of your Cisco Finesse Web Services server.

3. Add any mandatory parameters for the request.

4. Enter the username and password for the agent you set up for these exercises.

5. For Content Type, enter application/xml.

6. Click the appropriate action (GET, PUT, or POST).

Figure 2: Postman Rest Client

Pidgin for Windows

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 Postman appear in the XMPP Console tool of the Pidgin application. For example, if you use Postman 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 Postman and Pidgin.

You can download Pidgin from [http://www.pidgin.im/download/](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 3: 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).

---

**Note**

When connecting to the secure port 5223:

1. Add the Finesse Notification Service certificate in the Pidgin certificate manager. Finesse Notification Service shares the same certificate with Cisco Finesse Tomcat.

2. To download the certificate:
   a. Sign in to the Cisco Unified Operating System Administration through the URL (https://FQDN:8443/cmplatform, where FQDN is the fully qualified domain name of the primary Finesse server and 8443 is the port number).
   b. Click **Security > Certificate Management**.
   c. Click **Find** to get the list of all the certificates.
   d. In the Certificate List screen, choose **Certificate** from the **Find Certificate List where** drop-down menu, enter **tomcat** in the **begins with** option and click **Find**.
   e. Click the FQDN link which appears in the **Common Name** column parallel to the listed tomcat certificate.
   f. In the pop-up that appears, click the option **Download .PEM File** to save the file on your desktop.

3. In the Pidgin Certificate Manager, go to the Connection Security drop-down menu and choose **Use old-style SSL**.

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.
The agent must be signed into Finesse through Postman or the browser interface to be signed into 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.

Adium for Mac OS X

Adium is a free open source instant messaging application for Mac OS X. You can use Adium to establish an XMPP connection and view XMPP messages published by the Cisco Finesse Notification Service.

You can download Adium from https://www.adium.im.

Perform the following steps to configure XMPP:

1. In Adium go to Preferences > Account > '+' > XMPP (Jabber).
2. For Jabber ID, enter the username for the agent along with the fully qualified domain name of the Cisco Finesse server.

3. For Password, enter the password of the agent.

*Figure 6: The Adium Interface*

4. Enable XMPP Advanced Features (Default: Off).
   
   To enable the XML Console menu run the following command in Terminal: `defaults write com.adiumX.adiumX AMXMPPShowAdvanced -bool YES`

5. In Adium go to File > Logged in User > XML Console.
The agent must be signed into Finess through Postman or the browser interface to be signed into the XMPP account on Adium.

Note

The XML 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 XML Console window fills with iq type notifications and becomes difficult to navigate, close and reopen it to refresh with a clean 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 Postman or RESTClient. No event is published by XMPPP 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.

GET, PUT, POST, and DELETE requests that fail Finesse server internal checks are synchronous. If a request fails, Postman or RESTClient display the error. No event is published by XMPP to Pidgin. However, if the request fails on CTI side, Finesse will send an api Error XMPP event back to client after receiving a failure confirmation response from the CTI Server.
For each object, Finesse maintains an internal request queue where each subsequent request for this object is processed only after a success or a failure confirmation response is received from the CTI Server for the previous request.

RequestId is a user provided unique string that is added to the request API header and used to correlate originating requests with the resulting XMPP notifications or errors.

---

**Note**

RequestId is a best effort request-response correlation and is not reliable.

XMPP event notifications that match the requested action are tagged with the requestId (if available) from the original request. If the originating request results in a system error, the corresponding XMPP error notifications also contain the requestId. Note that the request id is not sent in the case of synchronous responses to GET requests. Although not mandatory, using a unique requestId helps in tracking error messages and allows a user to debug issues faster, as messages with requestId are easily tracked in Finesse logs.

---

**Note**

The requestId facility is not implemented for Task routing APIs. For more information, see the section on *Task Routing APIs*.

The following sections provide instructions and examples for using the APIs with Postman 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
- requestId: xyz

---

**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 Postman (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:

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

5. (Optional) To add the requestId:
   a. Click **Headers**.
   b. In the Name field, enter `requestId`, and in the Value field, enter `xyz`.
   c. Click **Add/Change**

6. Click **PUT**.

Postman 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:

```xml
<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>xyz</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.
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 Postman, 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:

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

5. (Optional) To add the requestId:
   a. Click Headers.
   b. In the Name field, enter `requestId`, and in the Value field, enter `xyz`.
   c. Click Add/Change

6. Click PUT.

   Postman returns the following response:

   ```xml
   PUT on http://finesse1.xyz.com/finesse/api/User/1234
   Status 202: Accepted
   Finesse returns the following user notification:
   ```

   ```xml
   <!DOCTYPE Update PUBLIC "-//Finesse//DTD Update//EN" "">
   <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>xyz</requestId>
     <source>/finesse/api/User/1234</source>
   </Update>
   ```
<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>
        <wrapUpOnOutgoing></wrapUpOnOutgoing>
      </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>xyz</requestId>
  <source>/finesse/api/User/1234</source>
</Update>
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.

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:

```xml
<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>
  <mediaType>1</mediaType>
  <pendingState>NOT READY</pendingState>
</User>
```
<pendingStateReasonCode>
  <category>NOT_READY</category>
  <code>1725</code>
  <forAll>true</forAll>
  <id>489</id>
  <label>Lunch</label>
  <systemCode>false</systemCode>
  <uri>/finesse/api/ReasonCode/489</uri>
</pendingStateReasonCode>

<pendingState>
  <reasonCodeId>16</reasonCodeId>
  <ReasonCode>
    <category>NOT_READY</category>
    <code>10</code>
    <forAll>true</forAll>
    <label>Team Meeting</label>
    <systemCode>false</systemCode>
    <id>16</id>
  </ReasonCode>
</pendingState>

<settings>
  <wrapUpOnIncoming>OPTIONAL</wrapUpOnIncoming>
</settings>

<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.
If five consecutive sign-ins fail due to an incorrect password, Finesse blocks access to the user account for a period of 5 minutes.

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.

To sign in as a mobile agent, see [User—Sign In as a Mobile Agent, on page 24](#).
To sign in to nonvoice Media Routing Domains, see [Media - Sign in, on page 147](#).

| 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: |  
```xml
<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: |  
```xml
<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).

Asynchronous Errors

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Device</td>
<td>Attempt to sign in an agent with a multiline device without the correct Unified CM configuration for maximum calls and busy trigger for these devices.</td>
<td>All</td>
</tr>
<tr>
<td>Invalid Device</td>
<td>Attempt to sign in an agent with a device that does not exist.</td>
<td>All</td>
</tr>
<tr>
<td>Invalid Device</td>
<td>Attempt to sign in an agent with a device that is offline.</td>
<td>All</td>
</tr>
<tr>
<td>Invalid Device</td>
<td>Attempt to sign in an agent with an extension that is not associated with the Unified CCX Resource Manager provider.</td>
<td>All</td>
</tr>
<tr>
<td>Device Busy</td>
<td>Attempt to sign in an agent with a device that is already in use.</td>
<td>All</td>
</tr>
</tbody>
</table>

Related Topics

Dialog CTI Error Notification, on page 300

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.

If five consecutive sign-ins fail due to an incorrect password, Finesse blocks access to the user account for a period of 5 minutes.

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 Cisco Unified Contact Center Enterprise Features Guide.
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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234">http://finesse1.xyz.com/finesse/api/User/1234</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Users can only act on their own User objects.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td></td>
</tr>
</tbody>
</table>

```xml
<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:

```xml
<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

Asynchronous Errors

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>ErrorType</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode Not Allowed</td>
<td>Attempt to sign in as a mobile agent when that agent is not configured as a mobile agent.</td>
<td>Unified CCE</td>
</tr>
</tbody>
</table>

Related Topics

- Dialog CTI Error Notification, on page 300

User—Sign Out of Finesse

This API allows a user to sign out of the Finesse desktop.

When signing out of the desktop, the user can sign out of voice only or out of all Media Routing Domains. Finesse sends separate sign out requests to CCE for each MRD.

For nonvoice MRDs only, users can sign out with active tasks. The user's tasks are either transferred or closed, depending on the way the MRD was configured when the user signed in through the Media-Sign In API.

The desktop sign out fails only if the voice MRD LOGOUT fails; it is not impacted by nonvoice MRD LOGOUT failure.

Note

To sign out of nonvoice Media Routing Domains only, see Media—Change State or Sign Out, on page 149.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234">http://finesse1.xyz.com/finesse/api/User/1234</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Agents and supervisors can use this API. Users can only act on their own User objects.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
</tbody>
</table>
Content Type: Application/XML
Input/Output Format: XML
HTTP Request:
```xml
<User>
  <state>LOGOUT</state>
  <logoutAllMedia>true</logoutAllMedia>
</User>
```
Request Parameters:
- id (required): The ID of the user
- state (required): The new state that the user wants to be in (LOGOUT)
- logoutAllMedia (optional): Determines whether the user signs out of all Media Routing Domains, or only out of voice. If set to false or not specified, the user signs out of voice only.

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:
```xml
<ApiErrors>
  <ApiError>
    <ErrorType>Invalid Input</ErrorType>
    <ErrorData>state</ErrorData>
    <ErrorMessage>Invalid State specified for user</ErrorMessage>
  </ApiError>
</ApiErrors>
```
Notifications Triggered:
- User notification
- Media notification (for nonvoice MRDs)

**Note**

If a nonvoice MRD sign out operation results in an asynchronous error, the error is returned in a Media notification. The notification includes the error type, error code, and error constant. The ErrorMedia parameter indicates the Media RoutingDomain to which the error applies.

**Related Topics**

- Media and Dialogs/Media Asynchronous Error Notification, on page 307

**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.
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: | For Unified CCE: http://<FQDN>/finesse/api/User/<id>  
For Unified CCX: 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. To get the User object, a user must be signed in, or provide valid authorization credentials when challenged. |
| HTTP Method: | GET |
| Content Type: | — |
| 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:

```xml
<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>
    </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):

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

### Note

Mobile agent only applies to Unified CCE deployments.
User—Get List

This API allows an administrator to get a list of users.

<table>
<thead>
<tr>
<th>Example Failure Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;ApiErrors&gt;</td>
</tr>
<tr>
<td>&lt;ApiError&gt;</td>
</tr>
<tr>
<td>&lt;ErrorType&gt;User Not Found&lt;/ErrorType&gt;</td>
</tr>
<tr>
<td>&lt;ErrorMessage&gt;UNKNOWN_USER&lt;/ErrorMessage&gt;</td>
</tr>
<tr>
<td>&lt;ErrorData&gt;4023&lt;/ErrorData&gt;</td>
</tr>
<tr>
<td>&lt;/ApiError&gt;</td>
</tr>
<tr>
<td>&lt;/ApiErrors&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Example Failure</strong></th>
<th><strong>Response:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;ApiErrors&gt;</td>
<td>&lt;ApiError&gt;</td>
</tr>
<tr>
<td>&lt;ApiError&gt;</td>
<td>&lt;ErrorType&gt;User Not Found&lt;/ErrorType&gt;</td>
</tr>
<tr>
<td>&lt;ErrorMessage&gt;UNKNOWN_USER&lt;/ErrorMessage&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;ErrorData&gt;4023&lt;/ErrorData&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/ApiError&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/ApiErrors&gt;</td>
<td></td>
</tr>
</tbody>
</table>

| **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. |
| **To get a list of users, the administrator must be signed in or provide valid authorization credentials when challenged.** |
| **HTTP Method:**  | GET |
| **Content Type:** | — |
| **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> |
|     ... 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 (Voice Only by Default)**

This API allows an agent or administrator to get a list of dialogs associated with a particular user. By default, this API returns voice dialogs only. You can use the query parameters to include nonvoice dialogs.

The URI for this API contains two query parameters:

- **type**: (optional) Set the type to return voice or nonvoice dialogs for a user. You can include both types to return all dialogs for a user (type=voice&type=non-voice). If you do not include the type query parameter, only voice dialogs are returned.

- **media**: (optional) Use this parameter to filter nonvoice dialog results by a specific media id. This parameter is only applicable when the "type=non-voice" query parameter is used.

**URI:**

```
http://<FQDN>/finesse/api/User/<id>/Dialogs?type={voice|non-voice}&media={id}
```

**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. To get a list of dialogs, a user must be signed in, or provide valid authorization credentials when challenged.

**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
User—Get List of Dialogs (Nonvoice Only)

This API allows an agent or administrator to get a list of nonvoice dialogs associated with a particular user for a specific Media Routing Domain (MRD).

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;/Media/&lt;mrdId&gt;/Dialogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234/Media/5001/Dialogs">http://finesse1.xyz.com/finesse/api/User/1234/Media/5001/Dialogs</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Agents can only get a list of their own dialogs. Administrators can get a list of dialogs associated with any user. To get a list of dialogs, a user must be signed in or provide valid authorization credentials when challenged.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>
| HTTP Response: | 200: Success  
401: Authorization Failure  
500: Internal Server Error  
503: Service Unavailable |
**User—Get List of Reservation Dialogs**

This API allows an agent or administrator to get a list of reservation dialogs and is applicable for progressive and predictive outbound reservation calls.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;/ReservationDialogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234/ReservationDialogs">http://finesse1.xyz.com/finesse/api/User/1234/ReservationDialogs</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Agents can get a list of their outbound reservation dialogs. Administrators can get a list of outbound reservation dialogs for all the users. To get a list of outbound reservation dialogs, a user must be signed in or must have the valid authorization credentials.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>
### 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.

To change user state in a nonvoice Media Routing Domain, see Media—Change State or Sign Out, on page 149.

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.

---

**Example Failure Response:**

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

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>UNKNOWN</td>
<td>If the agent state is unknown, the state is UNKNOWN. This scenario is unlikely.</td>
</tr>
<tr>
<td>LOGOUT</td>
<td>LOGIN</td>
<td>To sign in to Finesse, the agent sets the state to LOGIN. LOGIN is a transient state and transitions to NOT_READY.</td>
</tr>
<tr>
<td>LOGIN</td>
<td>NOT_READY</td>
<td>After a successful LOGIN, the agent transitions to NOT_READY.</td>
</tr>
<tr>
<td>NOT_READY</td>
<td>LOGOUT</td>
<td>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.</td>
</tr>
<tr>
<td>From</td>
<td>To</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NOT_READY</td>
<td>NOT_READY</td>
<td>To change their Not Ready reason code, agents can set a NOT_READY state from NOT_READY.</td>
</tr>
<tr>
<td>NOT_READY</td>
<td>READY</td>
<td>To become available for incoming or Outbound Option calls, agents set their state to READY.</td>
</tr>
<tr>
<td>NOT_READY</td>
<td>TALKING</td>
<td>An agent who places a call while in NOT_READY state transitions to TALKING.</td>
</tr>
<tr>
<td>READY</td>
<td>RESERVED</td>
<td>An incoming call arrives at an agent.</td>
</tr>
<tr>
<td>READY</td>
<td>RESERVED_OUTBOUND</td>
<td>An outbound agent becomes reserved to handle an Outbound Option Progressive or Predictive call.</td>
</tr>
<tr>
<td>READY</td>
<td>RESERVED_OUTBOUND_PREVIEW</td>
<td>An outbound agent becomes reserved to handle an Outbound Option Preview call.</td>
</tr>
<tr>
<td>READY</td>
<td>NOT_READY</td>
<td>Agents can change to NOT READY to make themselves unavailable for incoming calls.</td>
</tr>
<tr>
<td>RESERVED</td>
<td>READY</td>
<td>An agent can become RESERVED but never take a call.</td>
</tr>
<tr>
<td>RESERVED</td>
<td>TALKING</td>
<td>When an agent answers an incoming call, the agent transitions to TALKING.</td>
</tr>
<tr>
<td>RESERVED_OUTBOUND</td>
<td>READY</td>
<td>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.</td>
</tr>
<tr>
<td>RESERVED_OUTBOUND</td>
<td>NOT_READY</td>
<td>An agent can change to NOT READY state to leave RESERVED_OUTBOUND.</td>
</tr>
<tr>
<td>RESERVED_OUTBOUND</td>
<td>TALKING</td>
<td>An agent transitions to TALKING when an Outbound Option call arrives at the agent.</td>
</tr>
<tr>
<td>RESERVED_OUTBOUND_PREVIEW</td>
<td>READY</td>
<td>An agent transitions to READY if the agent was in READY state before being reserved in an Outbound Option Preview campaign.</td>
</tr>
<tr>
<td>RESERVED_OUTBOUND_PREVIEW</td>
<td>NOT_READY</td>
<td>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 NOTREADY until the call is complete or the Outbound Option Preview reservation is closed or rejected.</td>
</tr>
<tr>
<td>RESERVED_OUTBOUND_PREVIEW</td>
<td>TALKING</td>
<td>An agent transitions to TALKING when an Outbound Option call arrives at the agent.</td>
</tr>
</tbody>
</table>
ToFromDescription
---|---|---
**READY** | TALKING | 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).

**NOT_READY** | TALKING | 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.

**WORK** | TALKING | 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.

**WORK_READY** | TALKING | If wrap-up is enabled, an agent enters WORK_READY state after a call is dropped.

**HOLD** | TALKING | An agent puts a call on hold and transitions to HOLD state.

**READY** | HOLD | 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).

**NOT_READY** | HOLD | 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).

**WORK** | HOLD | 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.

**WORK_READY** | HOLD | If wrap-up is enabled and an agent is connected to a held call that is dropped, the agent transitions to WORK_READY state.

**TALKING** | HOLD | When an agent retrieves a held call, the agent transitions to TALKING state.

**READY** | WORK | To leave WORK state, agents can set their state to READY.

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

**NOT_READY** | WORK_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.
ToFrom

After a successful LOGIN, the agent transitions to NOT READY.

NOT READY LOGIN 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.
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 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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234">http://finesse1.xyz.com/finesse/api/User/1234</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Agents can only act on their own User objects. Supervisors can act on the User objects of agents who belong to their team.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>
| 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 |
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.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change from LOGOUT to NOT_READY.</td>
<td>Stand-alone Finesse with Unified CCE:</td>
</tr>
<tr>
<td></td>
<td>&lt;data&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;apiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;apiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorData&gt;257&lt;/errorData&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorMessage&gt;CF_INVALID_PASSWORD_SPECIFIED&lt;/errorMessage&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorType&gt;Invalid State&lt;/errorType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/apiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/apiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/data&gt;</td>
</tr>
<tr>
<td></td>
<td>Coresident Finesse with Unified CCX:</td>
</tr>
<tr>
<td></td>
<td>&lt;data&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;apiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;apiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorData&gt;1010&lt;/errorData&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorMessage&gt;CF_INVALID_PARAMETER&lt;/errorMessage&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorType&gt;Invalid State&lt;/errorType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/apiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/apiErrors&gt;</td>
</tr>
<tr>
<td>Agent receives and answers a non-ICD call.</td>
<td>Stand-alone Finesse with Unified CCE:</td>
</tr>
<tr>
<td></td>
<td>Finesse sends a User notification with state=TALKING.</td>
</tr>
<tr>
<td></td>
<td>Coresident Finesse with Unified CCX:</td>
</tr>
<tr>
<td></td>
<td>Finesse does not send a User notification. The agent remains in NOT_READY state.</td>
</tr>
<tr>
<td>Agent puts an ICD call on hold.</td>
<td>Stand-alone Finesse with Unified CCE:</td>
</tr>
<tr>
<td></td>
<td>Finesse sends a User notification with state=HOLD.</td>
</tr>
<tr>
<td></td>
<td>Coresident Finesse with Unified CCX:</td>
</tr>
<tr>
<td></td>
<td>Finesse does not send a User notification. The agent remains in TALKING state.</td>
</tr>
<tr>
<td>Scenario</td>
<td>Response</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>While talking on an ICD call, the agent sets a pending state of READY.</td>
<td><strong>Stand-alone Finesse with Unified CCE:</strong></td>
</tr>
<tr>
<td></td>
<td>Agent transitions to READY state after the call ends.</td>
</tr>
<tr>
<td></td>
<td><strong>Coresident Finesse with Unified CCX:</strong></td>
</tr>
<tr>
<td></td>
<td>Unified CCX does not allow an agent to set a pending state of READY while</td>
</tr>
<tr>
<td></td>
<td>that agent is talking on an ICD call.</td>
</tr>
<tr>
<td></td>
<td>&lt;data&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;apiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;apiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorData&gt;265&lt;/errorData&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorMessage&gt;CF_INVALID_AGENT_WORKMODE&lt;/errorMessage&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorType&gt;Invalid State&lt;/errorType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/apiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/apiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/data&gt;</td>
</tr>
<tr>
<td>While talking on a non-ICD call (agent state can be TALKING in Unified</td>
<td><strong>Stand-alone Finesse with Unified CCE:</strong></td>
</tr>
<tr>
<td></td>
<td>Agent transitions to READY state after the call ends.</td>
</tr>
<tr>
<td></td>
<td><strong>Coresident Finesse with Unified CCX:</strong></td>
</tr>
<tr>
<td></td>
<td>Unified CCX does not allow an agent to set a pending state of READY while</td>
</tr>
<tr>
<td></td>
<td>that agent is talking on a non-ICD call.</td>
</tr>
<tr>
<td></td>
<td>&lt;data&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;apiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;apiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorData&gt;33&lt;/errorData&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorMessage&gt;CF_RESOURCE_BUSY&lt;/errorMessage&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorType&gt;Invalid State&lt;/errorType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/apiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/apiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/data&gt;</td>
</tr>
<tr>
<td>While talking on an ICD call, the agent attempts to change from a</td>
<td><strong>Stand-alone Finesse with Unified CCE:</strong></td>
</tr>
<tr>
<td></td>
<td>Agent transitions to NOT_READY state with reason code 2 after the call</td>
</tr>
<tr>
<td></td>
<td><strong>Coresident Finesse with Unified CCX:</strong></td>
</tr>
<tr>
<td></td>
<td>Unified CCX allows an agent to set a pending state of</td>
</tr>
<tr>
<td></td>
<td>NOT_READY only once during a call. Unified CCX does</td>
</tr>
<tr>
<td></td>
<td>not allow an agent to change from one Not Ready</td>
</tr>
<tr>
<td></td>
<td>reason code to another.</td>
</tr>
<tr>
<td></td>
<td>&lt;data&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;apiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;apiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorData&gt;265&lt;/errorData&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorMessage&gt;CF_INVALID_AGENT_WORKMODE&lt;/errorMessage&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;errorType&gt;Invalid State&lt;/errorType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/apiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/apiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/data&gt;</td>
</tr>
<tr>
<td>Scenario</td>
<td>Response</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| A supervisor changes the state of an agent on that supervisor’s team to NOT READY. | **Stand-alone Finesse with Unified CCE:**
|                                                                         | Finesses sends a hard-coded reason code of 999 to indicate the forced state change. |
|                                                                         | **Coresident Finesse with Unified CCX:**
|                                                                         | Finesses sends a hard-coded reason code of 33 to indicate the forced state change. |

Asynchronous Errors

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid State</td>
<td>Invalid state transition requested. For example, attempt to set Wrap-Up state on an agent that is not allowed to go to Wrap-Up, or attempt to change an agent's state from READY state to Wrap-up or WORK state.</td>
<td>All</td>
</tr>
<tr>
<td>Internal Server Error</td>
<td>Attempt to change an agent's state from RESERVED_OUTBOUND to any other state.</td>
<td>Unified CCX</td>
</tr>
</tbody>
</table>

Related Topics

Dialog CTI Error Notification, on page 300

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.

If the user is changing state to LOGOUT and is signing out of all Media Routing Domains, the same reason code is applied to all the Media Routing Domains.

**Note**

To change state with a reason code in a nonvoice Media Routing Domain only, see Media—Change Agent State with Reason Code, on page 151.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234">http://finesse1.xyz.com/finesse/api/User/1234</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Users can only act on their own User objects.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
</tbody>
</table>
### API Basics

**Input/Output Format:** XML

**HTTP Request:**

```xml
<User>
  <state>LOGOUT</state>
  <reasonCodeId>10</reasonCodeId>
  <logoutAllMedia>true</logoutAllMedia>
</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)
- **logoutAllMedia** (optional): This parameter can be included if changing the state to LOGOUT. It determines whether the user signs out of all Media Routing Domains, or only out of voice. If set to false or not specified, the user signs out of voice only.

**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:**

```xml
<ApiErrors>
  <ApiError>
    <ErrorType>Parameter Missing</ErrorType>
    <ErrorData>state</ErrorData>
    <ErrorMessage>State Parameter missing</ErrorMessage>
  </ApiError>
</ApiErrors>
```

**Notifications Triggered:**

- User notification
- Media notification (for nonvoice MRDS, when changing state to LOGOUT)

---

**Note**
If a nonvoice MRD sign out operation results in an asynchronous error, the error is returned in a Media notification. The notification includes the error type, error code, and error constant. The ErrorMedia parameter indicates the Media RoutingDomain to which the error applies.

**Related Topics**

[Media and Dialogs/Media Asynchronous Error Notification](#), on page 307

---

**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 section on ReasonCode.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Constraints:</td>
<td>Administrators, agents, and supervisors can use this API.</td>
</tr>
<tr>
<td></td>
<td>To get a reason code, a user must be signed in or provide valid authorization credentials when challenged.</td>
</tr>
<tr>
<td></td>
<td>The reason code must be global (forAll parameter set to true) or be assigned to a team to which the user belongs.</td>
</tr>
<tr>
<td></td>
<td>Only an administrator can get another user's reason codes.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td></td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>400: Bad Request</td>
</tr>
<tr>
<td></td>
<td>400: Finesse API Error (for example, the object does not exist, the object is stale, or violation of DB constraint)</td>
</tr>
<tr>
<td></td>
<td>401: Authorization Failure</td>
</tr>
<tr>
<td></td>
<td>401: Invalid Authorization User Specified</td>
</tr>
<tr>
<td></td>
<td>404: Not Found (for example, the reason code does not exist or has been deleted)</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
<tr>
<td>Example Response:</td>
<td>&lt;ReasonCode&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;uri&gt;finesse/api/ReasonCode/1&lt;/uri&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;category&gt;NOT READY&lt;/category&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;code&gt;12&lt;/code&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;label&gt;Lunch&lt;/label&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;forAll&gt;true&lt;/forAll&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ReasonCode&gt;</td>
</tr>
<tr>
<td>Example Failure Response:</td>
<td>&lt;ApiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ApiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorType&gt;Authorization Failure&lt;/ErrorType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorMessage&gt;UNAUTHORIZED&lt;/ErrorMessage&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorData&gt;1234&lt;/ErrorData&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ApiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ApiErrors&gt;</td>
</tr>
</tbody>
</table>
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 section on ReasonCode.

| URI: | http://<FQDN>/finesse/api/User/<id>/ReasonCodes?category=NOT_READY|LOGOUT |
|---|---|
| Security Constraints: | Administrators, agents and supervisors can use this API. To get a list of reason codes, a user must be signed in or provide valid authorization credentials when challenged. Only an administrator can get another user's list of reason codes. |
| HTTP Method: | GET |
| Content Type: | — |
| 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 500: Internal Server Error |
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 196.

| **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. |
| | To get a wrap-up reason, a user must be signed in, or provide valid authorization credentials when challenged. |
| | Only an administrator can get another user’s wrap-up reasons. |
| **HTTP Method:** | GET |
| **Content Type:** | — |
| **Input/Output Format:** | XML |
| **HTTP Request:** | — |
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 196.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;/WrapUpReasons</td>
</tr>
<tr>
<td>Example URI</td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234/WrapUpReasons">http://finesse1.xyz.com/finesse/api/User/1234/WrapUpReasons</a></td>
</tr>
<tr>
<td>Security Constraints</td>
<td>Administrators, agents, and supervisors can use this API.</td>
</tr>
<tr>
<td></td>
<td>To get a list of wrap-up reasons, a user must be signed in or provide valid authorization credentials when challenged.</td>
</tr>
<tr>
<td></td>
<td>Only an administrator can get another user's list of wrap-up reasons.</td>
</tr>
<tr>
<td>HTTP Method</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request</td>
<td>—</td>
</tr>
</tbody>
</table>
HTTP Response:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
</tr>
<tr>
<td>400</td>
<td>Finesse API Error (for example, the object does not exist, the object is stale, or violation of DB constraint)</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
</tr>
<tr>
<td>404</td>
<td>User Not Found</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

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 Default Media Properties Layout

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

**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.

To get the default MediaPropertiesLayout object, a user must be signed in or provide valid authorization credentials when challenged.

**HTTP Method:**

GET

**Content Type:**

---

**Input/Output Format:**

XML

**HTTP Request:**

---

**HTTP Response:**

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
<tr>
<td>Example Response:</td>
<td></td>
</tr>
</tbody>
</table>
```xml
<MediaPropertiesLayout>
  <header>
    <entry>
      <displayName>Call Variable 1</displayName>
      <mediaProperty>callVariable1</mediaProperty>
    </entry>
  </header>
  <column>
    <entry>
      <displayName>BA AccountNumber</displayName>
      <mediaProperty>BAAccountNumber</mediaProperty>
    </entry>
    <entry>
      <displayName>BA Campaign</displayName>
      <mediaProperty>BACampaign</mediaProperty>
    </entry>
    <entry>
      <displayName>Call Variable 1</displayName>
      <mediaProperty>callVariable1</mediaProperty>
    </entry>
    <entry>
      <displayName>Call Variable 2</displayName>
      <mediaProperty>callVariable2</mediaProperty>
    </entry>
    <entry>
      <displayName>Call Variable 3</displayName>
      <mediaProperty>callVariable3</mediaProperty>
    </entry>
    <entry>
      <displayName>Call Variable 4</displayName>
      <mediaProperty>callVariable4</mediaProperty>
    </entry>
    <entry>
      <displayName>Call Variable 5</displayName>
      <mediaProperty>callVariable5</mediaProperty>
    </entry>
  </column>
  <column>
    <entry>
      <displayName>BA Status</displayName>
      <mediaProperty>BAStatus</mediaProperty>
    </entry>
    <entry>
      <displayName>BA Response</displayName>
      <mediaProperty>BAResponse</mediaProperty>
    </entry>
    <entry>
      <displayName>Call Variable 6</displayName>
      <mediaProperty>callVariable6</mediaProperty>
    </entry>
    <entry>
      <displayName>Call Variable 7</displayName>
      <mediaProperty>callVariable7</mediaProperty>
    </entry>
    <entry>
      <displayName>Call Variable 8</displayName>
      <mediaProperty>callVariable8</mediaProperty>
    </entry>
    <entry>
      <displayName>Call Variable 9</displayName>
      <mediaProperty>callVariable9</mediaProperty>
    </entry>
    <entry>
      <displayName>Call Variable 10</displayName>
    </entry>
  </column>
</MediaPropertiesLayout>
```
User—Get Default Media Properties Layout

```
<mediaProperty>callVariable10</mediaProperty>
</entry>
</column>
</MediaPropertiesLayout/>
</name>Default Layout</name>
<description>Layout used when no other layout matches the user layout Custom/ECC Variable</description>
</type>DEFAULT</type>
</MediaPropertiesLayout>
<MediaPropertiesLayout>
<header>
<entry>
<displayName>Call Variable 1</displayName>
<mediaProperty>callVariable1</mediaProperty>
</entry>
</header>
<column>
<entry>
<displayName>BA AccountNumber</displayName>
<mediaProperty>BAAccountNumber</mediaProperty>
</entry>
<entry>
<displayName>BA Campaign</displayName>
<mediaProperty>BACampaign</mediaProperty>
</entry>
<entry>
<displayName>Call Variable 1</displayName>
<mediaProperty>callVariable1</mediaProperty>
</entry>
<entry>
<displayName>Call Variable 2</displayName>
<mediaProperty>callVariable2</mediaProperty>
</entry>
<entry>
<displayName>Call Variable 3</displayName>
<mediaProperty>callVariable3</mediaProperty>
</entry>
<entry>
<displayName>Call Variable 4</displayName>
<mediaProperty>callVariable4</mediaProperty>
</entry>
<entry>
<displayName>Call Variable 5</displayName>
<mediaProperty>callVariable5</mediaProperty>
</entry>
</column>
<column>
<entry>
<displayName>BA Status</displayName>
<mediaProperty>BAStatus</mediaProperty>
</entry>
<entry>
<displayName>BA Response</displayName>
<mediaProperty>BAResponse</mediaProperty>
</entry>
<entry>
<displayName>Call Variable 6</displayName>
<mediaProperty>callVariable6</mediaProperty>
</entry>
<entry>
<displayName>Call Variable 7</displayName>
<mediaProperty>callVariable7</mediaProperty>
</entry>
</column>
```

User—Get Media Properties Layout List

This API allows a user to get a list of all media properties layouts configured on the system, including the default media properties layout.

| URI: | http://<FQDN>/finesse/api/User/<UserId>/MediaPropertiesLayouts |
| Example URI: | http://finesse1.xyz.com/finesse/api/User/<UserId>/MediaPropertiesLayouts |
| Security Constraints: | Agents and supervisors can use this API. Any user can get a list of media properties layouts if they are signed in or they provide valid authorization credentials when challenged. |
| HTTP Method: | GET |
| Content Type: | — |
| 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  
500: Internal Server Error |
| --- | --- |

**Example Response:**

```
<MediaPropertiesLayouts>
  <MediaPropertiesLayout>
    ... Full MediaPropertiesLayout Object ...
  </MediaPropertiesLayout>
  <MediaPropertiesLayout>
    ... Full MediaPropertiesLayout Object ...
  </MediaPropertiesLayout>
  <MediaPropertiesLayout>
    ... Full MediaPropertiesLayout Object ...
  </MediaPropertiesLayout>
</MediaPropertiesLayouts>
```

**Example Failure Response:**

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

**Related Topics**

- MediaPropertiesLayout, on page 202

**User—Get List of Phone Books**

This API allows a user to get a list of phone books and the first 1500 associated contacts for that user. Contacts are retrieved from the global phone books first, followed by the team phone books, up to the maximum limit of 1500.

For more information about the PhoneBook object, see PhoneBook, on page 216.

**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 user can get a list of their own phone books if they are signed in or they provide valid authorization credentials when challenged.

**Additional Headers:**

"Range: objects=1-1500"

The range of contacts to retrieve.

**HTTP Method:**

GET
Content Type: —
Input/Output Format: XML
HTTP Request: —
HTTP Response: 200: Success
206: Partial Content
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
414: Invalid Range Specified. Range must be 1–1500 objects
500: Internal Server Error

Example Response:
<PhoneBooks>
  <PhoneBook>
    <name>PhoneBook1</name>
    <type>GLOBAL</type>
    <Contacts>
      <Contact>
        ...Full Contact Object...
      </Contact>
      ...Full Contact Object...
    </Contacts>
  </PhoneBook>
  <PhoneBook>
    <name>PhoneBook2</name>
    <type>TEAM</type>
    <Contacts>
      <Contact>
        ...Full Contact Object...
      </Contact>
      ...Full Contact Object...
    </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 231.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;/Workflows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234/Workflows">http://finesse1.xyz.com/finesse/api/User/1234/Workflows</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Any user can get their own workflows if they are signed in or they provide valid authorization credentials when challenged.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>
| 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>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>
  </Workflow>
</Workflows>
<Workflow>

  <ConditionSet>
    <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>
  </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=${CallVariable1}&amp;c=cat&amp;d=${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>

Cisco Finesse Web Services Developer Guide Release 11.6(1)
### Example Failure Response:

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

### User API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>The ID of the user.</td>
<td>—</td>
<td>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.</td>
</tr>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the object.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>roles</td>
<td>Collection</td>
<td>List of roles for this user.</td>
<td>Agent, Supervisor</td>
<td></td>
</tr>
<tr>
<td>--&gt;role</td>
<td>String</td>
<td>One of the roles assigned to this user.</td>
<td>Agent, Supervisor</td>
<td></td>
</tr>
<tr>
<td>loginId</td>
<td>String</td>
<td>The login ID for this user.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>loginName</td>
<td>String</td>
<td>The login name for this user.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>state</td>
<td>String</td>
<td>The state for this user.</td>
<td>LOGOUT, NOT_READY, READY, RESERVED, RESERVED_OUTBOUND, RESERVED_OUTBOUND_PREVIEW, TALKING, HOLD, WORK, WORK_READY, UNKNOWN</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Possible Values</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>stateChangeTime</td>
<td>String</td>
<td>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.</td>
<td>—</td>
<td>This parameter is empty if the time of the state change is not available (if no agent state change notification was received yet).</td>
</tr>
<tr>
<td>pendingState</td>
<td>String</td>
<td>The state to which the user will transition next.</td>
<td>LOGOUT</td>
<td>For Unified CCE deployments, this parameter is empty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Possible Values</td>
<td>Notes</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 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. |
<p>| ReasonCode  | Collection | Information about the reason code currently associated with this user.       |                                                                                 |                                                                                              |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>settings</td>
<td>Collection</td>
<td>The settings for this user.</td>
<td>—</td>
<td>The settings parameter is only present for Unified CCE deployments.</td>
</tr>
<tr>
<td>--&gt;wrapUpOnIncoming</td>
<td>String</td>
<td>Indicates whether this user required or allowed to enter wrap-up data on an incoming call.</td>
<td>REQUIRED, OPTIONAL, NOT_ALLOWED, REQUIRED_WITH_WRAP_UP_DATA</td>
<td>This parameter applies only to Unified CCE deployments.</td>
</tr>
<tr>
<td>extension</td>
<td>String</td>
<td>The extension that this user is currently using.</td>
<td>—</td>
<td>The extension must exist in Unified Communications Manager.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If the user is configured in Unified CCE, size is determined by Unified Communications Manager.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If the user is configured in Unified CCX, the size is determined by Unified CCX.</td>
</tr>
<tr>
<td>mobileAgent</td>
<td>Collection</td>
<td>Indicates that the user is a mobile agent.</td>
<td>—</td>
<td>This parameter is returned for mobile agents only. Finesse supports mobile agents only in Unified CCE deployments.</td>
</tr>
<tr>
<td>--&gt;mode</td>
<td>String</td>
<td>The work mode for the mobile agent</td>
<td>CALL_BY_CALL, NAILED_CONNECTION</td>
<td>This parameter is returned for mobile agents only. Finesse supports mobile agents only in Unified CCE deployments.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Possible Values</td>
<td>Notes</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>---&gt;dialNumber</td>
<td>String</td>
<td>The external number that the system calls to connect to the mobile agent.</td>
<td>—</td>
<td>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.</td>
</tr>
<tr>
<td>firstName</td>
<td>String</td>
<td>The first name of this user.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>lastName</td>
<td>String</td>
<td>The last name of this user.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>teamId</td>
<td>String</td>
<td>The ID of the team to which this user belongs.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>teamName</td>
<td>String</td>
<td>The name of the team to which this user belongs.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>dialogs</td>
<td>String</td>
<td>URI to the collection of dialogs that the user is a part of.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>teams</td>
<td>Collection</td>
<td>If the user has a role of Supervisor, a list of teams that the user supervises.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>---&gt;Team</td>
<td>Collection</td>
<td>Set of information for a team.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>---&gt;uri</td>
<td>String</td>
<td>The URI to get a new copy of the Team object.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>---&gt;id</td>
<td>String</td>
<td>The ID for the team.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>---&gt;name</td>
<td>String</td>
<td>The name of the team.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>mediaState</td>
<td>—</td>
<td>The state of the user on a manual outbound call from NOTREADY state.</td>
<td>BUSY, IDLE</td>
<td>This parameter is returned for Team API only and not for User API. This parameter applies only to Unified CCX deployments.</td>
</tr>
</tbody>
</table>
### User API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>The request is malformed or incomplete or the extension provided is invalid.</td>
</tr>
<tr>
<td>400</td>
<td>Generic Error</td>
<td>An unaccounted for error occurred. The root cause could not be determined.</td>
</tr>
<tr>
<td>400</td>
<td>Invalid Input</td>
<td>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)</td>
</tr>
<tr>
<td>400</td>
<td>Invalid State</td>
<td>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).</td>
</tr>
<tr>
<td>400</td>
<td>Parameter Missing</td>
<td>The extension, state, or requestedAction is not provided.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If signing in a mobile agent, the mode or dialNumber is not provided.</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>Unauthorized (for example, the user is not yet authenticated in the Web Session).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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).</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Authorization User Specified</td>
<td>The authenticated user tried to make a request for another user.</td>
</tr>
<tr>
<td>401</td>
<td>Invalid State</td>
<td>A user tried to change to a state that is not supported in the scenario.</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Supervisor</td>
<td>A supervisor tried to change the state of an agent who does not belong to that supervisor's team.</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>The resource specified is invalid or does not exist.</td>
</tr>
<tr>
<td>Status</td>
<td>Error Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>404</td>
<td>User Not Found</td>
<td>The user ID provided is invalid or is not recognized. No such user exists in CTI.</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
<tr>
<td>503</td>
<td>Service Unavailable</td>
<td>A dependent service is down (for example, the Cisco Finesse Notification Service or Cisco Finesse Database). Finesse is OUT_OF_SERVICE.</td>
</tr>
</tbody>
</table>

**Dialog**

The Dialog object represents a dialog with participants.

**Dialog Object for Voice Calls**

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 for voice calls:

```xml
<Dialog>
    <associatedDialogUri>/finesse/api/Dialog/321654</associatedDialogUri>
    <id>12345678</id>
    <secondaryId>12345679</secondaryId>
    <mediaType>Voice</mediaType>
    <fromAddress>2002</fromAddress>
    <toAddress>2000</toAddress>
    <mediaProperties>
        <dialedNumber>2000</dialedNumber>
        <callType>AGENT_INSIDE</callType>
        <DNIS>2000</DNIS>
        <queueNumber>5022</queueNumber>
        <queueName>UCM_PIM.Func.Agents.5G</queueName>
        <callKeyCallId>217</callKeyCallId>
        <callKeyPrefix>152018</callKeyPrefix>
        <wrapUpReason>Sales Call</wrapUpReason>
        <callVariables>
            <CallVariable>
                <name>callVariable1</name>
                <value>Chuck Smith</value>
            </CallVariable>
            <CallVariable>
                <name>callVariable2</name>
                <value>Cisco Systems, Inc.</value>
            </CallVariable>
            ...Other CallVariables ...
        </callVariables>
        <participants>
            <Participant>
                <actions>
                    <action>HOLD</action>
                    <action>DROP</action>
                </actions>
                <mediaAddress>2002</mediaAddress>
            </Participant>
        </participants>
    </mediaProperties>
</Dialog>
```
Dialog Object for Nonvoice Tasks

For nonvoice media types, this object represents a task. A participant represents an internal or external user's leg of the task.

The Dialog object is structured as follows for nonvoice tasks:

---

**Note**
Several Dialog parameters do not apply for nonvoice tasks, and are returned empty.
callKeyCallId and callKeyPrefix parameters apply only to Unified CCE deployments.

**Dialog APIs**

Finesse obtains the dialogId value from the CallID value defined for the calls by the CTI Server. With some call flows, the messaging between Finesse and the CTI Server refers to an updated CallID value. In most cases, the updated CallID value maintains a relationship to the original CallID value, and therefore Finesse maintains the same dialogId value for the duration of the call flows. However, there are some call flows in which the CallID and dialogId change permanently (for example, in a conference). If you require a better understanding of the relationship between the CallID and dialogId values, you can perform some test call flows and view the webservice logs.

**Dialog—Get Dialog**

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

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Dialog/&lt;dialogId&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Dialog/12345678">http://finesse1.xyz.com/finesse/api/Dialog/12345678</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Agents and administrators can use this API. Agents can only get their own Dialog object. Administrators can get any Dialog object.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>200</td>
<td>Success</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Authorization</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
<tr>
<td>Example Response:</td>
<td></td>
</tr>
</tbody>
</table>
<Dialog>
  <uri>/finesse/api/Dialog/12345678</uri>
  <mediaType>Voice</mediaType>
  <state>ACTIVE</state>
  <fromAddress>2002</fromAddress>
  <toAddress>2000</toAddress>
  <mediaProperties>
    <mediaId>1</mediaId>
    <dialedNumber>2000</dialedNumber>
    <callType>AGENT_INSIDE</callType>
    <DNIS>2000</DNIS>
    <queueNumber>5022</queueNumber>
    <queueName>UCM_PIM.Func.Agents.SG</queueName>
    <callKeyCallId>217</callKeyCallId>
    <callKeyPrefix>152018</callKeyPrefix>
    <wrapUpReason>Another satisfied customer</wrapUpReason>
    <callVariables>
      <CallVariable>
        <name>callVariable1</name>
        <valueChuck Smith</value>
      </CallVariable>
      <CallVariable>
        <name>callVariable2</name>
        <valueCisco Systems, Inc</value>
      </CallVariable>
      <CallVariable>
        <name>callVariable3</name>
        <valuechucksmith@cisco.com</value>
      </CallVariable>
      ...Other Call Variables (up to 10)
      <CallVariable>
        <name>ecc.user</name>
        <valuecsmith</value>
      </CallVariable>
      <CallVariable>
        <name>ecc.years[0]</name>
        <value1985</value>
      </CallVariable>
      <CallVariable>
        <name>ecc.years[1]</name>
        <value1995</value>
      </CallVariable>
    </callVariables>
  </mediaProperties>
  <participants>
    <Participant>
      <actions>
        <action>HOLD</action>
        <action>DROP</action>
      </actions>
      <mediaAddress>1081001</mediaAddress>
    </Participant>
    <Participant>
      <actions>
        <action>RETRIEVE</action>
        <action>DROP</action>
      </actions>
      <mediaAddress>1081002</mediaAddress>
    </Participant>
  </participants>
</Dialog>
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.

This API supports the use of any ASCII character in the toAddress. Finesse does not convert any entered letters into numbers, nor does it remove non-numeric characters (including parentheses and hyphens) from the toAddress.

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:**

```xml
<Dialog>
  <requestedAction>MAKE_CALL</requestedAction>
  <fromAddress>1001001</fromAddress>
  <toAddress>1002002</toAddress>
</Dialog>
```
**HTTP Request with Call Variables (Unified CCE only):**

```xml
<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

Asynchronous Errors

Note When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid State</td>
<td>Attempt to POST a Dialog when the agent is in an invalid state to make a call.</td>
<td>All</td>
</tr>
<tr>
<td>Invalid State</td>
<td>Supervisor attempts to POST a Dialog when that supervisor is silently monitoring another agent.</td>
<td>All</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt to POST a Dialog to a route point when there are no agents in Ready state in the queue corresponding to that route point.</td>
<td>All</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt to POST a Dialog in which the toAddress is an E164 extension.</td>
<td>Unified CCE</td>
</tr>
</tbody>
</table>
**Related Topics**

[Dialog CTI Error Notification](#), on page 300

## 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.

<table>
<thead>
<tr>
<th>URI</th>
<th>http://&lt;FQDN&gt;/finesse/api/Dialog/&lt;dialogId&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Dialog/54321">http://finesse1.xyz.com/finesse/api/Dialog/54321</a></td>
</tr>
<tr>
<td>Security Constraints</td>
<td>Agents can use this API. Agents can only act on a participant of a dialog when they are that participant.</td>
</tr>
<tr>
<td>HTTP Method</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format</td>
<td>XML</td>
</tr>
</tbody>
</table>

### HTTP Request

**For voice dialogs:**

```
<Dialog>
  <targetMediaAddress>1001001</targetMediaAddress>
  <requestedAction>ANSWER</requestedAction>
</Dialog>
```

voice dialog TRANSFER example:

```
<Dialog>
  <requestedAction>TRANSFER</requestedAction>
  <toAddress>1001002</toAddress>
  <targetMediaAddress>1001001</targetMediaAddress>
</Dialog>
```

voice dialog CONFERENCE example:

```
<Dialog>
  <requestedAction>CONFERENCE</requestedAction>
  <toAddress>1001002</toAddress>
  <targetMediaAddress>1001001</targetMediaAddress>
</Dialog>
```

### For nonvoice dialogs:

**Nonvoice dialog CLOSE example:**

```
<Dialog>
  <requestedAction>CLOSE</requestedAction>
  <mediaProperties>
    <wrapUpReason>Happy customer!</wrapUpReason>
  </mediaProperties>
</Dialog>
```

**Nonvoice dialog TRANSFER example:**

```
<Dialog>
  <requestedAction>TRANSFER</requestedAction>
  <target>scriptSelector</target>
</Dialog>
```
### Request Parameters:

**For voice dialogs:**
- 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

**For nonvoice dialogs:**
- dialogId (required): The ID of the dialog
- requestedAction (required): The action to take on the targeted participant
- mediaProperties (optional): A collection of media-specific properties for the dialog. This parameter can be used only when the action is CLOSE in order to set the wrapUpReason parameter.
- wrapUpReason (optional): A description of the task. This parameter can be used only when the action is CLOSE.
- target (required for TRANSFER): The Script Selector/dialed number to which the dialog is being transferred.

### 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
- 503: Service Unavailable (for example, the Notification Service is not running).

### Example Failure Response:

**For voice dialogs:**
```
<ApiErrors>
  <ApiError>
    <ErrorType>Invalid Input</ErrorType>
    <ErrorMessage>Invalid 'requestedAction' specified for dialog</ErrorMessage>
  </ApiError>
</ApiErrors>
```

**For nonvoice dialogs:**
```
<ApiErrors>
  <ApiError>
    <ErrorType>Agent is not logged in</ErrorType>
    <ErrorMessage>E_ARM_STAT_AGENT_NOT_LOGGED_IN</ErrorMessage>
  </ApiError>
</ApiErrors>
```
Notifications Triggered:

For voice dialogs:
- Dialog notification
- Dialog CTI error notification (if a CTI error occurs)

For nonvoice dialogs:
- Dialogs/Media notification
- Dialogs/Media asynchronous error notifications including CTI errors

Platform-Based API Differences

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

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

Asynchronous Errors for Voice Dialogs

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>ErrorType</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Error</td>
<td>Attempt a call transfer without an existing consult call.</td>
<td>All</td>
</tr>
</tbody>
</table>
### Error Type

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Error</td>
<td>Attempt a call transfer on the original call (a direct call) after the original call has already been retrieved.</td>
<td>All</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt to complete a conference on the original call after retrieving the original call.</td>
<td>All</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt to exceed the maximum allowed conference participants.</td>
<td>All</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt to RETRIEVE an incoming OutBoundPreview campaign call when the allowed actions are ACCEPT, CLOSE, and REJECT.</td>
<td>All</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Non-conference-controller attempts to conference in another party.</td>
<td>Unified CCE</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt to put the held call (a direct call) on hold again.</td>
<td>All</td>
</tr>
<tr>
<td>Invalid State</td>
<td>Non-conference-controller attempts to conference in another party.</td>
<td>Unified CCX</td>
</tr>
</tbody>
</table>

### Asynchronous Errors for Nonvoice Dialogs

If an error occurs after the initial validation of a nonvoice dialog is complete, the API send an error notification over XMPP to the Dialogs/Media notification. The error message has the format described in "Media and Dialogs/Media Asynchronous Error Notification.". The requestID is included in the response XML. The ErrorMedia parameter in the ApiService information indicates the Media Routing Domain to which the error applies.

For transfers, Finesse communicates asynchronously with SocialMiner to initiate task resubmission requests. The following types of errors can occur, and are returned asynchronously:

- SocialMiner can respond to the Finesse transfer request with an HTTP error response (for example 4XX or 5XX).
- The Finesse request to SocialMiner may time-out due to network issues.

If the request to SocialMiner fails, the API send an error notification over XMPP to the Dialogs/Media notification, and Finesse retains the dialog.

**Related Topics**
- Dialog CTI Error Notification, on page 300
- Media and Dialogs/Media Asynchronous Error Notification, on page 307

### 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.
With Unified CCX, 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:**
```xml
<Dialog>
  <requestedAction>UPDATE_CALL_DATA</requestedAction>
  <mediaProperties>
    <wrapUpReason>Happy customer!</wrapUpReason>
    <callvariables>
      <CallVariable>
        <name>callVariable1</name>
        <value>123456789</value>
      </CallVariable>
      ... Other call variables to be modified ...
      </CallVariable>
    </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:

```xml
<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)

**Asynchronous Errors**

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>ErrorType</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Input</td>
<td>The value of a call variable or ECC variable is longer than what is either allowed or configured as the maximum length for that variable.</td>
<td>All</td>
</tr>
<tr>
<td>Invalid Input</td>
<td>The value of an ECC variable that is configured as a scalar is set as an array.</td>
<td>All</td>
</tr>
<tr>
<td>Invalid Input</td>
<td>The value of an ECC variable that is configured as an array is set as a scalar.</td>
<td>All</td>
</tr>
<tr>
<td>Invalid Input</td>
<td>The value of an ECC variable that is configured as an array is set as an array but with an index greater than what is configured.</td>
<td>All</td>
</tr>
<tr>
<td>Call Variable is protected</td>
<td>Attempt to set call variables on a non-routed (direct) call.</td>
<td>All</td>
</tr>
</tbody>
</table>

**Related Topics**

- [Dialog CTI Error Notification](#), on page 300
ECC and Call Variable Error Handling

When a client makes an invalid update request for an 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.

<table>
<thead>
<tr>
<th>Error Scenario</th>
<th>CTI Server Response</th>
<th>Finesse Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. A request was sent that generates an error from the CTI server to Finesse.</td>
<td>The CTI server sends an error to Finesse.</td>
<td>Finesse forwards the error to the client.</td>
</tr>
<tr>
<td>2. The request payload contained no valid ECC or call variables.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. A request was sent that generates an error from the CTI server to Finesse.</td>
<td>1. The CTI server sends an error to Finesse.</td>
<td>1. Finesse forwards the error to the client.</td>
</tr>
<tr>
<td>2. The request payload contained a mix of valid and invalid ECC or call variables.</td>
<td>2. The CTI server does not send an UPDATE_CALL_DATA event to Finesse (that is, the CTI server fails the entire request).</td>
<td>2. The client does not receive an UPDATE_CALL_DATA event.</td>
</tr>
<tr>
<td>1. A request was sent that does not generate an error from the CTI server to Finesse.</td>
<td>The CTI server does not respond.</td>
<td>Finesse does not respond.</td>
</tr>
<tr>
<td>2. The request payload contained no valid ECC or call variables.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. A request was sent that does not generate an error from the CTI server to Finesse.</td>
<td>1. The CTI server does not send an error to Finesse.</td>
<td>1. Finesse does not forward an error to the client.</td>
</tr>
<tr>
<td>2. The request payload contained a mix of valid and invalid ECC or call variables.</td>
<td>2. The CTI server sends an UPDATE_CALL_DATA event to Finesse for the valid ECC and call variables.</td>
<td>2. Finesse forwards the UPDATE_CALL_DATA event to the client.</td>
</tr>
</tbody>
</table>

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.

CTI communication architecture has been optimized in Cisco Finesse Release 12.5(1), which has introduced changes in the Finesse API behavior. As a result of this change, it is suggested that call control requests for
the same device should not be sent to the Finesse server until the response to a previous call control request has been received. Multiple DTMF requests can however be send one after another, and the server queues them up for you without any error.

To prevent CTI errors, the Finesse desktop disables call control buttons such as **Hold, Transfer, Consult, Wrap-Up, and End** across all calls when the DTMF Keypad is opened until the responses to all of the DTMF requests have been completed or timed out. It is suggested that third-party clients follow the same design. The number of outstanding DTMF requests and the timeout duration can be configured using the Finesse CLI. For more information on CLIs, see the **Desktop Properties** section in *Cisco Finesse Administration Guide* at [https://www.cisco.com/c/en/us/support/customer-collaboration/finesse/products-maintenance-guides-list.html](https://www.cisco.com/c/en/us/support/customer-collaboration/finesse/products-maintenance-guides-list.html).

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Dialog/&lt;dialogId&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Dialog/54321">http://finesse1.xyz.com/finesse/api/Dialog/54321</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Agents can use this API. An agent must be a participant in the dialog to perform this action.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>
| 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:**

```xml
<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.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send a DTMF request with an alphanumeric dtmfString.</td>
<td><strong>Stand-alone Finesse with Unified CCE:</strong> Unified CCE accepts the alphanumeric dtmfString. <strong>Coresident Finesse with Unified CCX:</strong> Unified CCX allows only 0-9, *, or # in the dtmfString. Using any other values results in the following error:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;apiError&gt;</code>&lt;br&gt;  <code>(</code>&lt;errorData&gt;3&lt;/errorData&gt;<code>&lt;br&gt;  </code>&lt;errorMessage&gt;CF_VALUE_OUT_OF_RANGE&lt;/errorMessage&gt;<code>&lt;br&gt;  </code>&lt;errorType&gt;Generic Error&lt;/errorType&gt;<code>&lt;br&gt;  </code>&lt;/apiError&gt;`</td>
</tr>
</tbody>
</table>
Asynchronous Errors

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Error</td>
<td>Attempt to send a DTMF request with a valid requestedAction, a valid targetMediaAddress (agent's extension), and an alphanumeric dtmfString. Unified CCX allows only 0-9, *, and # for the dtmfString. Any other values result in the error.</td>
<td>Unified CCX</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt to send a DTMF request for a call when the participant in the dialog whose extension is the targetMediaAddress is in a HELD state.</td>
<td>ALL</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt a PUT request to send DTMF while a call is alerting.</td>
<td>ALL</td>
</tr>
</tbody>
</table>

Related Topics

- Dialog CTI Error Notification, on page 300

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.

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.

---

**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.

---

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Dialog/&lt;dialogId&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Dialog/54321">http://finesse1.xyz.com/finesse/api/Dialog/54321</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Agents can use this API.</td>
</tr>
<tr>
<td></td>
<td>An agent must be a participant in the dialog and the agent's extension must match the targetMediaAddress.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>CONSULT_CALL Example</td>
</tr>
<tr>
<td></td>
<td>&lt;Dialog&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;requestedAction&gt;CONSULT_CALL&lt;/requestedAction&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;toAddress&gt;1001002&lt;/toAddress&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;targetMediaAddress&gt;1001001&lt;/targetMediaAddress&gt;</td>
</tr>
<tr>
<td>Request Parameters:</td>
<td>dialogId (required): The ID of the dialog</td>
</tr>
<tr>
<td></td>
<td>requestedAction (required): The way in which the dialog is created (CONSULT_CALL)</td>
</tr>
<tr>
<td></td>
<td>toAddress (required): The destination for the call</td>
</tr>
<tr>
<td></td>
<td>targetMediaAddress (required): The extension of the agent, used to locate the participant to target with the requestedAction</td>
</tr>
</tbody>
</table>
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)

Asynchronous Errors

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>ErrorType</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Error</td>
<td>Attempt a CONSULT_CALL on an incoming OutBoundPreview campaign call while the allowed actions are ACCEPT, CLOSE, and REJECT.</td>
<td>ALL</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt a CONSULT_CALL while the call is alerting.</td>
<td>ALL</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt a CONSULT_CALL while the call is on HOLD.</td>
<td>ALL</td>
</tr>
</tbody>
</table>

Related Topics  
Dialog CTI Error Notification, on page 300

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.

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

Example URI:  
```
http://finesse1.xyz.com/finesse/api/Dialog/54321```
<table>
<thead>
<tr>
<th>Security Constraints:</th>
<th>Agents can use this API. An agent must be a participant in the dialog and the agent's extension must match the targetMediaAddress.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>
| 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                                                                           |
Asynchronous Errors

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Error</td>
<td>Attempt a TRANSFER_SST before the call gets answered.</td>
<td>Unified CCE</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt a TRANSFER_SST on an incoming OutBoundPreview campaign call while the allowed actions are ACCEPT, CLOSE, and REJECT.</td>
<td>Unified CCE</td>
</tr>
</tbody>
</table>

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.

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 session and the agent must be in TALKING state.
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.

**Asynchronous Errors**

---

**Note**

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>88049</td>
<td>Attempt to POST Silent Monitor for an agent who is in Ready, Wrap-Up, Hold, or Not Ready state.</td>
<td>Unified CCX</td>
</tr>
<tr>
<td>13145</td>
<td>Attempt to POST Silent Monitor for an agent who is in Hold or Not Ready state.</td>
<td>Unified CCE</td>
</tr>
<tr>
<td>Invalid State</td>
<td>Attempt to POST Silent Monitor for an agent who is in Ready or Wrap-Up State.</td>
<td>Unified CCE</td>
</tr>
</tbody>
</table>

**Related Topics**

[Dialog CTI Error Notification](#), on page 300

---

### 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:**

```xml
<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). In case of CVP routed calls, the barge request is also possible for agents other than the conference controller. 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.
### Dialog—Make a Barge Call

<table>
<thead>
<tr>
<th><strong>URI:</strong></th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;/Dialogs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example URI:</strong></td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234/Dialogs">http://finesse1.xyz.com/finesse/api/User/1234/Dialogs</a></td>
</tr>
<tr>
<td><strong>Security Constraints:</strong></td>
<td>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 into any calls because they are not associated with a phone device.</td>
</tr>
<tr>
<td><strong>HTTP Method:</strong></td>
<td>POST</td>
</tr>
<tr>
<td><strong>Content Type:</strong></td>
<td>Application/XML</td>
</tr>
<tr>
<td><strong>Input/Output Format:</strong></td>
<td>XML</td>
</tr>
</tbody>
</table>
| **HTTPRequest:** | `<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:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>Successfully Accepted</td>
</tr>
</tbody>
</table>

**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 into that call. In a Finesse deployment with Unified CCE, the supervisor's state during the silent monitoring session is TALKING. When the supervisor barges into 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 into 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.
Asynchronous Errors

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Error</td>
<td>Supervisor attempts a barge call on an agent who is not the conference controller.</td>
<td>ALL</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Supervisor attempts a barge call on an agent who is on a Consult call.</td>
<td>ALL</td>
</tr>
</tbody>
</table>

Related Topics

Dialog CTI Error Notification, on page 300

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:

```xml
<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:**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>Successfully Accepted</td>
</tr>
<tr>
<td>400</td>
<td>Parameter Missing</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Authorization User Specified</td>
</tr>
<tr>
<td>404</td>
<td>Not Found (the dialog specified by the dialogId does not exist)</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

**Example Failure Response:**

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

### Notifications Triggered:
Dialog notification

### Asynchronous Errors

**Note**
When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."
<table>
<thead>
<tr>
<th>ErrorType</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Error</td>
<td>Supervisor barges in and attempts to drop a participant in a two-party call scenario.</td>
<td>ALL</td>
</tr>
</tbody>
</table>

**Related Topics**

*Dialog CTI Error Notification*, on page 300

**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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Dialog/&lt;dialogId&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Dialog/54321">http://finesse1.xyz.com/finesse/api/Dialog/54321</a></td>
</tr>
</tbody>
</table>
| 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:

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

Notifications Triggered: Dialog notification

Asynchronous Errors

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Error</td>
<td>Attempt to PUT a START_RECORDING when the only allowable action is TRANSFER_SST.</td>
<td>Unified CCX</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt to PUT a START_RECORDING when the only allowable action is ANSWER.</td>
<td>Unified CCX</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt to PUT a START_RECORDING with no MediaSense server.</td>
<td>Unified CCX</td>
</tr>
<tr>
<td>Generic Error</td>
<td>Attempt to PUT a START_RECORDING on a Unified CCE deployment type. This API is only supported with Unified CCX deployment type.</td>
<td>Unified CCE</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Dialog/&lt;dialogId&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Dialog/54321">http://finesse1.xyz.com/finesse/api/Dialog/54321</a></td>
</tr>
</tbody>
</table>
| 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:

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

Notifications Triggered: Dialog notification

Asynchronous Errors

**Note**
When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>ErrorType</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Error</td>
<td>Attempt to PUT a Dialog object using an action that is not allowed. For example, attempting a HOLD call when allowed actions are ACCEPT, REJECT, and CLOSE.</td>
<td>All</td>
</tr>
</tbody>
</table>

Related Topics
- Dialog CTI Error Notification, on page 300

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

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

**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:**
```xml
<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

### Example Failure Response:
```xml
<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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Dialog/&lt;dialogId&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Dialog/54321">http://finesse1.xyz.com/finesse/api/Dialog/54321</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Agents can use this API. Agents can only act on their own Dialog object.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>
### Dialog—Reclassify a Direct Preview Call

#### 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.

**Note**: The BUSY parameter is not supported in a Finesse deployment with Unified CCE. If used, it returns an invalid input error.

#### 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 reclassify 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
Asynchronous Errors

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in "Dialog CTI Error Notification."

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic error</td>
<td>Attempt to reclassify a dialog that is not generated by the outbound campaign.</td>
<td>All</td>
</tr>
</tbody>
</table>

**Related Topics**

Dialog CTI Error Notification, on page 300

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Parameter Provided</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the object.</td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Possible Values</td>
<td>Parameter Provided</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>associatedDialog Uri</td>
<td>String</td>
<td>The URI to a Dialog object that is associated with this Dialog object.</td>
<td>/finesse/api/Dialog/dialogId</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>secondaryId</td>
<td>Numeric</td>
<td>The call ID value assigned to the secondary call.</td>
<td>—</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>mediaType</td>
<td>String</td>
<td>The type of media under which this dialog is classified.</td>
<td>The enterprise name of the Media Routing Domain (MRD).</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>state</td>
<td>String</td>
<td>The last state of this dialog.</td>
<td>For a list of possible values, see State (Dialog) Parameter Values, on page 112.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>fromAddress</td>
<td>String</td>
<td>The calling line ID of the caller.</td>
<td>—</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>toAddress</td>
<td>String</td>
<td>The destination for the call.</td>
<td>—</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>mediaProperties</td>
<td>Collection</td>
<td>A collection of media-specific properties for the dialog.</td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>--&gt;mediaId</td>
<td>String</td>
<td>The ID of the MRD.</td>
<td>For voice, this value is always 1.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>--&gt;dialedNumber</td>
<td>String</td>
<td>The number dialed.</td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>queueNumber</td>
<td>Numeric</td>
<td>The queue ID of the call.</td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>queueName</td>
<td>String</td>
<td>The queue name of the call.</td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>callKeyIdCallId</td>
<td>Numeric</td>
<td>The unique number of the call routed on a particular day.</td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>callKeyPrefix</td>
<td>Numeric</td>
<td>Represents the day when the call is routed.</td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Notes:**
- **Voice Calls**: Indicates whether the parameter is provided for voice calls.
- **Nonvoice Tasks**: Indicates whether the parameter is provided for nonvoice tasks.

This parameter is empty for nonvoice tasks.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Parameter Provided</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>--&gt;callType</td>
<td>String</td>
<td>The type of call.</td>
<td>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</td>
<td>Yes</td>
<td>Nonvoice Yes</td>
</tr>
<tr>
<td>--&gt;DNIS</td>
<td>String</td>
<td>The DNIS provided with the call. For routed calls, the DNIS is the route point.</td>
<td>—</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>--&gt;wrapUpReason</td>
<td>String</td>
<td>A description of the call.</td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>--&gt;callVariables</td>
<td>Collection</td>
<td>A list of call variables associated with this dialog.</td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The maximum size of this parameter is 39 bytes (which equals 39 US English characters).
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Parameter Provided</th>
<th>Notes</th>
</tr>
</thead>
</table>
| - - - > 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 128. | Yes | Yes | 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. | — | Yes | Yes |
| - - - > Participant | Collection | Information about one participant in the dialog. | — | Yes | Yes |
| - - - > actions | Collection | A list of actions that are allowed for a participant. | For a list of possible values, see Actions Parameter Values, on page 115. | Yes | Yes |
### Dialog API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Parameter Provided</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>---&gt;mediaAddress</td>
<td>String</td>
<td>Point of contact for the participant.</td>
<td>Possible values include the extension of an agent or ANI for a caller who are participants in the call. For nonvoice dialogs, the value is the agent's id.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>---&gt;mediaAddressType</td>
<td>Collection</td>
<td>The device type specified by the mediaAddress.</td>
<td>AGENTDEVICE or empty string</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Possible Values</td>
<td>Parameter Provided</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>-------</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>startTime</strong></td>
<td>String</td>
<td>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.</td>
<td>The start time in the format YYYY-MM-DDThh:MM:ss.SSSZ or an empty string</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

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 agent state or call information. After failover occurs, Finesse sets the startTime parameter as an empty string.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Parameter Provided</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>---›state</td>
<td>String</td>
<td>The last participant state in a dialog.</td>
<td>For a list of possible values, see State (Participant) Parameter Values, on page 118.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>---›stateCause</td>
<td>String</td>
<td>The cause for the last participant state in a dialog.</td>
<td>BUSY, BAD_DESTINATION, OTHER</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>This parameter is normally associated with a FAILED participant state.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Possible Values</td>
<td>Parameter Provided</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>stateChangeTime</td>
<td>String</td>
<td>The UTC time when the participant changed to the current state. Finesse uses the Finesseserver timestamp (not the CTI even timestamp) to determine the stateChangeTime. A time difference may exist between the Finesseserver 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.</td>
<td>The state change time in the format YYYY-MM-DDThh:MM:ss.SSSZ or an empty string</td>
<td>Yes</td>
<td>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.</td>
</tr>
<tr>
<td>scheduledCallbackInfo</td>
<td>Collection</td>
<td>For Outbound Option campaigns, provides information about scheduled callbacks.</td>
<td>—</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This parameters is provided only if a callback is scheduled for this dialog.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Possible Values</td>
<td>Parameter Provided</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>--------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>--&gt;callbackTime</td>
<td>String</td>
<td>The callback time in the format YYYY-MM-DD HH: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 timezone, is ignored.</td>
<td>—</td>
<td>Yes No</td>
<td>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)</td>
</tr>
<tr>
<td>--&gt;callbackNumber</td>
<td>String</td>
<td>The phone number to call for the callback.</td>
<td>—</td>
<td>Yes No</td>
<td>This parameter is provided only if a callback number has been set. Value returned in the BAREponse: P#&lt;callbackNumber&gt; (for example, P#9780001)</td>
</tr>
<tr>
<td>dispositionCode</td>
<td>String</td>
<td>The reason the dialog ended.</td>
<td>For a list of possible values, see Disposition Code Parameter Values for Nonvoice Tasks, on page 130.</td>
<td>No Yes</td>
<td></td>
</tr>
</tbody>
</table>

State (Dialog) Parameter Values

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

<table>
<thead>
<tr>
<th>Dialog State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIATING</td>
<td>Indicates that the phone is off the hook at a device</td>
</tr>
<tr>
<td>INITIATED</td>
<td>Indicates that the phone is dialing at the device</td>
</tr>
</tbody>
</table>
### Dialog State Description

<table>
<thead>
<tr>
<th>Dialog State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALERTING</td>
<td>Indicates that the call is ringing at a device</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>Indicates that the dialog has at least one active participant</td>
</tr>
<tr>
<td>FAILED</td>
<td>Indicates that the dialog has failed</td>
</tr>
<tr>
<td>DROPPED</td>
<td>Indicates that the dialog has no active participants</td>
</tr>
<tr>
<td>ACCEPTED</td>
<td>Indicates the user has accepted the OUTBOUND_PREVIEW dialog</td>
</tr>
</tbody>
</table>

### Nonvoice States

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

<table>
<thead>
<tr>
<th>Dialog State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFERED</td>
<td>Indicates that the dialog has been offered to a user</td>
</tr>
<tr>
<td>ACCEPTED</td>
<td>Indicates that the user has accepted the offered dialog</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>Indicates that the dialog has at least one active participant; the user has started working on the accepted dialog</td>
</tr>
<tr>
<td>PAUSED</td>
<td>Indicates that an active dialog has been paused</td>
</tr>
<tr>
<td>WRAPPING_UP</td>
<td>Indicates that a user is performing wrap up activity for a dialog</td>
</tr>
<tr>
<td>INTERRUPTED</td>
<td>Indicates that the dialog has been interrupted by a dialog from another MRD. Dialogs can be interrupted if they are in the ACTIVE, PAUSED, or WRAPPING_UP states. While a dialog is interrupted, all actions for that dialog are disabled. This state is applicable only for interruptible MRDs with the Media API interruptAction parameter set to ACCEPT.</td>
</tr>
<tr>
<td>CLOSED</td>
<td>Indicates that the dialog ended. The disposition code indicates the reason the dialog closed. See Disposition Code Parameter Values for Nonvoice Tasks, on page 130.</td>
</tr>
</tbody>
</table>
### Dialog State Description

<table>
<thead>
<tr>
<th>Dialog State</th>
<th>Description</th>
</tr>
</thead>
</table>
| UNKNOWN      | After Finesse server or PG failure recovery, any dialogs in the INTERRUPTED state change UNKNOWN state when the dialog is no longer interrupted.  

For example, the following scenario results in a dialog in the UNKNOWN state:

1. The user accepts and starts a dialog in an interruptible media.
2. The user accepts and starts a dialog in a non-interruptible media.
3. The dialog in the interruptible media changes to the INTERRUPTED state.
4. The PG goes out of service.
5. Both dialogs are recovered and are in the correct state.
6. The user closes the dialog in the non-interruptible media.
7. The dialog in the interruptible media changes to the UNKNOWN state.  

When a dialog is in the UNKNOWN state, the user is only allowed to close the dialog. |

The following figure illustrates these allowed state transitions for nonvoice dialogs:
Actions Parameter Values

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

<table>
<thead>
<tr>
<th>Participant Allowable Action</th>
<th>Enabled Button on Desktop</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAKE_CALL</td>
<td>Make a New Call</td>
<td>Allows an agent to make an outgoing call.</td>
</tr>
<tr>
<td>ANSWER</td>
<td>Answer</td>
<td>Allows an agent to answer an incoming call.</td>
</tr>
<tr>
<td>HOLD</td>
<td>Hold</td>
<td>Allows an agent to hold a call that is currently active.</td>
</tr>
<tr>
<td>RETRIEVE</td>
<td>Retrieve</td>
<td>Allows an agent to retrieve a call that was on hold.</td>
</tr>
<tr>
<td>DROP</td>
<td>End</td>
<td>Allows an agent to drop the participant of a call.</td>
</tr>
<tr>
<td>Participant Allowable Action</td>
<td>Enabled Button on Desktop</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UPDATE_CALL_DATA</td>
<td>—</td>
<td>Allows an agent to set call data for the call.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> Finesse does not allow an agent to set call data from the desktop. A user can set call</td>
</tr>
<tr>
<td>SEND_DTMF</td>
<td>—</td>
<td>Allows an agent to send DTMF digits for the call.</td>
</tr>
<tr>
<td>CONSULT_CALL</td>
<td>Consult</td>
<td>Allows an agent to make a consult call for transfer or conference.</td>
</tr>
<tr>
<td>CONFERENCE</td>
<td>Conference</td>
<td>Allows an agent to start a conference between the selected held call and the existing active call on the desktop.</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>Transfer</td>
<td>Allows an agent to complete a transfer between the selected held call and the existing active call on the desktop.</td>
</tr>
<tr>
<td>TRANSFER_SST</td>
<td>Direct Transfer</td>
<td>Allows an agent to initiate a single-step transfer.</td>
</tr>
<tr>
<td>SILENT_MONITOR</td>
<td>Start Monitoring</td>
<td>Allows a supervisor to silent monitor an agent who is in TALKING state on an active call.</td>
</tr>
<tr>
<td>BARGE_CALL</td>
<td>Barge In</td>
<td>Allows a supervisor to barge in on an agent call that the supervisor is silently monitoring.</td>
</tr>
<tr>
<td>PARTICIPANT_DROP</td>
<td>Drop</td>
<td>Allows a supervisor to drop a participant from a conference call.</td>
</tr>
<tr>
<td>START_RECORDING</td>
<td>Start Recording</td>
<td>Allows an agent to start a recording (Unified CCX only; requires integration with MediaSense).</td>
</tr>
<tr>
<td>UPDATE_SCHEDULED_CALLBACK</td>
<td>Callback, Schedule</td>
<td>Allows an agent to update the details for a scheduled callback.</td>
</tr>
<tr>
<td>CANCEL_SCHEDULED_CALLBACK</td>
<td>Callback, Cancel</td>
<td>Allows an agent to cancel a scheduled callback.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Action</th>
<th>BAResponse Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPT</td>
<td>Accept</td>
<td>Performing the ACCEPT action while reserved in an Outbound Option Preview campaign instructs Unified CCE to establish a call with the customer.</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Reject-Close</td>
<td>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.</td>
</tr>
<tr>
<td>REJECT</td>
<td>Reject</td>
<td>Performing the REJECT action while reserved in an Outbound Option Preview campaign instructs Unified CCE to retry the previewed number later.</td>
</tr>
</tbody>
</table>

**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.

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 *Outbound Option Guide for Unified Contact Center Enterprise*.

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*.
### Nonvoice Actions

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

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPT</td>
<td>Allows an agent to accept an incoming task.</td>
</tr>
<tr>
<td>START</td>
<td>Allows an agent to start work on an accepted task.</td>
</tr>
<tr>
<td>PAUSE</td>
<td>Allows an agent to pause an active task.</td>
</tr>
<tr>
<td>RESUME</td>
<td>Allows an agent to resume a paused task.</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>Allows an agent to transfer an accepted, active, or paused task to another</td>
</tr>
<tr>
<td></td>
<td>Script Selector/dialed number.</td>
</tr>
<tr>
<td>WRAP_UP</td>
<td>Allows an agent to perform wrap up work for a task.</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Allows an agent to end a task.</td>
</tr>
</tbody>
</table>

### State (Participant) Parameter Values

The following table describes possible values for the state (participant) response parameter for voice calls:
<table>
<thead>
<tr>
<th>Participant State</th>
<th>Allowable Actions for the Participant State</th>
<th>Call State on Finesse Desktop</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIATING</td>
<td>DROP, UPDATE_CALL_DATA</td>
<td>Off Hook</td>
<td>Indicates that an outgoing call, not yet active, exists on the device</td>
</tr>
<tr>
<td>INITIATED</td>
<td>DROP, UPDATE_CALL_DATA</td>
<td>Dialing</td>
<td>Indicates that the phone is dialing at a device</td>
</tr>
<tr>
<td>ALERTING</td>
<td>ANSWER</td>
<td>Incoming</td>
<td>Indicates that an incoming call is ringing on the device</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>HOLD, DROP, UPDATE_CALL_DATA, CONSULT_CALL</td>
<td>Active</td>
<td>Indicates that the participant is active on the call</td>
</tr>
<tr>
<td>FAILED</td>
<td>DROP</td>
<td>Busy</td>
<td>Indicates that the call failed (BUSY)</td>
</tr>
<tr>
<td>FAILED</td>
<td>DROP</td>
<td>Error</td>
<td>Indicates that the call failed (BAD_DESTINATION)</td>
</tr>
<tr>
<td>FAILED</td>
<td>DROP</td>
<td>Error</td>
<td>Indicates that the call failed (OTHER)</td>
</tr>
<tr>
<td>HELD</td>
<td>RETRIEVE, DROP, UPDATE_CALL_DATA, TRANSFER (if active call exists), CONFERENCE (if active call exists)</td>
<td>Hold</td>
<td>Indicates that the participant has held their connection to the call</td>
</tr>
<tr>
<td>DROPPED</td>
<td>-</td>
<td>-</td>
<td>Indicates that the participant has dropped from the call</td>
</tr>
<tr>
<td>WRAP_UP</td>
<td>UPDATE_CALL_DATA</td>
<td>Active</td>
<td>Indicates that the participant is not in active state on the call but is wrapping up after the participant has dropped from the call</td>
</tr>
<tr>
<td>ACCEPTED</td>
<td>-</td>
<td>-</td>
<td>Indicates that the participant has accepted the dialog. This state is applicable to OUTBOUND_PREVIEW dialogs.</td>
</tr>
</tbody>
</table>
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.

**Nonvoice State (Participant) Parameter Values**

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

<table>
<thead>
<tr>
<th>Participant State</th>
<th>Allowable Actions for the Participant State</th>
<th>Dialog State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFERED</td>
<td>ACCEPT</td>
<td>OFFERED</td>
<td>Indicates that the participant has been offered the task.</td>
</tr>
<tr>
<td>ACCEPTED</td>
<td>START, CLOSE, TRANSFER</td>
<td>ACCEPTED</td>
<td>Indicates that the participant has accepted a task, but has not started working on the task.</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>PAUSE, WRAP_UP, CLOSE, TRANSFER</td>
<td>ACTIVE</td>
<td>Indicates that the participant is active in the task.</td>
</tr>
<tr>
<td>PAUSED</td>
<td>RESUME, CLOSE, TRANSFER, WRAP_UP</td>
<td>PAUSED</td>
<td>Indicates that the participant has paused the active task. The WRAP_UP action is not available if the task was PAUSED from the WRAPPING_UP state.</td>
</tr>
<tr>
<td>WRAPPING_UP</td>
<td>PAUSE, CLOSE</td>
<td>WRAPPING_UP</td>
<td>Indicates that the participant is performing wrap up work for a task.</td>
</tr>
<tr>
<td>INTERRUPTED</td>
<td>-</td>
<td>INTERRUPTED</td>
<td>Indicates that the participant has been interrupted in this MRD by a task from another MRD. This state is applicable only for interruptible MRDs with the interruptAction parameter set to ACCEPT.</td>
</tr>
<tr>
<td>CLOSED</td>
<td>-</td>
<td>-</td>
<td>Indicates that the participant ended the task.</td>
</tr>
</tbody>
</table>

**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.
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

<table>
<thead>
<tr>
<th>Scenario</th>
<th>CTI Event</th>
<th>Event Method</th>
<th>Dialog State</th>
<th>Participant State (Agent)</th>
<th>Participant State (Caller)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start the call</td>
<td>BEGIN_CALL_EVENT</td>
<td>POST (Caller)</td>
<td>INITIATING</td>
<td>Not a participant yet</td>
<td>INITIATING</td>
</tr>
<tr>
<td>Call arrives at agent</td>
<td>CALL_DELIVERED</td>
<td>POST (Agent), PUT (Caller)</td>
<td>ALERTING</td>
<td>ALERTING</td>
<td>INITIATED</td>
</tr>
<tr>
<td>Agent answers call</td>
<td>CALL_ESTABLISHED</td>
<td>PUT</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>Caller drops call</td>
<td>CALL_CONNECTION_CLEARED</td>
<td>PUT</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>DROPPED</td>
</tr>
<tr>
<td>Agent is dropped from call</td>
<td>CALL_CONNECTION_CLEARED</td>
<td>PUT</td>
<td>DROPPED</td>
<td>DROPPED</td>
<td>DROPPED</td>
</tr>
<tr>
<td>Call is cleared</td>
<td>CALL_CONNECTION_CLEARED</td>
<td>PUT</td>
<td>DROPPED</td>
<td>DROPPED</td>
<td>DROPPED</td>
</tr>
<tr>
<td>Call is removed</td>
<td>END_CALL_EVENT</td>
<td>DELETE</td>
<td>DROPPED</td>
<td>DROPPED</td>
<td>DROPPED</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Scenario</th>
<th>CTI Event</th>
<th>Event Method</th>
<th>Dialog State</th>
<th>Participant State (Caller)</th>
<th>Participant State (Recipient)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start of any call</td>
<td>BEGIN_CALL_EVENT</td>
<td>POST (Caller)</td>
<td>INITIATING</td>
<td>INITIATING</td>
<td>Not a participant yet</td>
</tr>
<tr>
<td>Scenario</td>
<td>CTI Event</td>
<td>Event Method</td>
<td>Dialog State</td>
<td>Participant State (Caller)</td>
<td>Participant State (Recipient)</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>--------------</td>
<td>--------------</td>
<td>-----------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Caller takes phone off-hook</td>
<td>CALL_SERVICE_INITIATED_EVENT</td>
<td>POST (Caller)</td>
<td>INITIATING</td>
<td>INITIATING</td>
<td>Not a participant yet</td>
</tr>
<tr>
<td>Caller dials number</td>
<td>CALL_ORIGINATED_EVENT</td>
<td>PUT (Caller)</td>
<td>INITIATED</td>
<td>INITIATED</td>
<td>Not a participant yet</td>
</tr>
<tr>
<td>Destination is busy</td>
<td>CALL_FAILED_EVENT (BUSY)</td>
<td>PUT (Caller)</td>
<td>FAILED</td>
<td>FAILED</td>
<td>Not a participant yet</td>
</tr>
<tr>
<td>Destination is bad</td>
<td>CALL_FAILED_EVENT (BAD_DESTINATION)</td>
<td>PUT (Caller)</td>
<td>FAILED</td>
<td>FAILED</td>
<td>Not a participant yet</td>
</tr>
<tr>
<td>Destination is recipient</td>
<td>CALL_DELIVERED</td>
<td>PUT (Caller), POST (Recipient)</td>
<td>ALERTING</td>
<td>INITIATED</td>
<td>ALERTING</td>
</tr>
<tr>
<td>Recipient answers call</td>
<td>CALL_ESTABLISHED</td>
<td>PUT</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>Caller drops call</td>
<td>CALL_CONNECTION_CLEARED</td>
<td>PUT</td>
<td>ACTIVE</td>
<td>Dropped</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>Recipient is dropped from call</td>
<td>CALL_CONNECTION_CLEARED</td>
<td>PUT</td>
<td>Dropped</td>
<td>Dropped</td>
<td>Dropped</td>
</tr>
<tr>
<td>Call is cleared</td>
<td>CALL_CLEARED_EVENT</td>
<td>PUT</td>
<td>Dropped</td>
<td>Dropped</td>
<td>Dropped</td>
</tr>
<tr>
<td>Call is removed</td>
<td>END_CALL_EVENT</td>
<td>DELETE</td>
<td>Dropped</td>
<td>Dropped</td>
<td>Dropped</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Scenario</th>
<th>CTI Event</th>
<th>Event Method</th>
<th>Dialog State</th>
<th>Participant State (Agent)</th>
<th>Participant State (Caller)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call arrives and is answered</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Agent holds call</td>
<td>CALL HELD</td>
<td>PUT</td>
<td>ACTIVE</td>
<td>HELD</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>Caller holds call</td>
<td>CALL HELD</td>
<td>PUT</td>
<td>ACTIVE</td>
<td>HELD</td>
<td>HELD</td>
</tr>
<tr>
<td>Agent retrieves call</td>
<td>CALL RETRIEVED</td>
<td>PUT</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>HELD</td>
</tr>
<tr>
<td>Caller retrieves call</td>
<td>CALL RETRIEVED</td>
<td>PUT</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Scenario</th>
<th>CTI Event (Original Call)</th>
<th>CTI Event (Consult Call)</th>
<th>Event Method</th>
<th>Dialog State</th>
<th>Participant State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent A starts consult call</td>
<td>CALL HELD</td>
<td>-</td>
<td>PUT (original call only)</td>
<td>Original call: ACTIVE</td>
<td>Caller: ACTIVE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Agent A: HELD (original call)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Agent B: Not yet a participant</td>
</tr>
<tr>
<td>Agent A takes phone off-hook</td>
<td>-</td>
<td>CALL SERVICE INITIATED EVENT</td>
<td>PUT (consult call only)</td>
<td>Original call: ACTIVE</td>
<td>Caller: ACTIVE</td>
</tr>
<tr>
<td>(BEGIN_CALL_ EVENT assumed)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Agent A: INITIATING (consult call)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Agent B: Not yet a participant</td>
</tr>
<tr>
<td>Agent A dials number</td>
<td>-</td>
<td>CALL_ORIGINATED EVENT</td>
<td>PUT (consult call only)</td>
<td>Original call: ACTIVE</td>
<td>Caller: ACTIVE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Agent A: INITIATED (consult call)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Agent B: Not yet a participant</td>
</tr>
<tr>
<td>Scenario</td>
<td>CTI Event (Original Call)</td>
<td>CTI Event (Consult Call)</td>
<td>Event Method</td>
<td>Dialog State</td>
<td>Participant State</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------</td>
<td>--------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Agent B receives the call</td>
<td>-</td>
<td>CALL_DELIVERED</td>
<td>PUT (consult call, on Agent A) POST (consult call on Agent B)</td>
<td>Original call: ACTIVE Consult call: ALERTING</td>
<td>Caller: ACTIVE Agent A: INITIATED (consult call) Agent B: ALERTING</td>
</tr>
<tr>
<td>Agent B answers the call</td>
<td>-</td>
<td>CALL_ESTABLISHED</td>
<td>PUT (consult call only)</td>
<td>Original call: ACTIVE Consult call: ACTIVE</td>
<td>Caller: ACTIVE Agent A: ACTIVE (consult call) Agent B: ACTIVE</td>
</tr>
<tr>
<td>Agent A completes the transfer of the caller to Agent B</td>
<td>CALL_TRANSFERRED_EVENT</td>
<td>-</td>
<td>DELETE (original call on Agent A) DELETE (consult call on Agent A) DELETE (consult call on Agent B) POST (original call on Agent B)</td>
<td>Original call: DROPPED (Agent A), ACTIVE (Agent B) Consult call: DROPPED (both Agent A and Agent B)</td>
<td>Caller: ACTIVE Agent A: DROPPED (original and consult call) Agent B: DROPPED (consult call), ACTIVE (original call)</td>
</tr>
</tbody>
</table>

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.
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

<table>
<thead>
<tr>
<th>Scenario</th>
<th>CTI Event (Silent Monitor Call)</th>
<th>Event Method</th>
<th>Dialog State (Original Call)</th>
<th>Dialog State (Silent Monitor Call)</th>
<th>Participant State (Caller)</th>
<th>Participant State (Agent A)</th>
<th>Participant State (Supervisor)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent call arrives and is answered</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Supervisor starts the silent monitor call</td>
<td>BEGIN_CALL</td>
<td>POST (SILENT MONITOR)</td>
<td>ACTIVE</td>
<td>INITIATING</td>
<td>ACTIVE (original call)</td>
<td>ACTIVE (original call)</td>
<td>INITIATING (silent monitor call)</td>
</tr>
<tr>
<td></td>
<td>CALL_SERVICE_INITIATED_EVENT</td>
<td>-</td>
<td>ACTIVE</td>
<td>INITIATING</td>
<td>ACTIVE (original call)</td>
<td>ACTIVE (original call)</td>
<td>INITIATING (silent monitor call)</td>
</tr>
<tr>
<td></td>
<td>CALL_DATA_UPDATE_EVENT</td>
<td>-</td>
<td>ACTIVE</td>
<td>INITIATED</td>
<td>ACTIVE (original call)</td>
<td>ACTIVE (original call)</td>
<td>INITIATED (silent monitor call)</td>
</tr>
<tr>
<td></td>
<td>CALL_ORIGINATED_EVENT</td>
<td>-</td>
<td>ACTIVE</td>
<td>INITIATED</td>
<td>ACTIVE (original call)</td>
<td>ACTIVE (original call)</td>
<td>INITIATED (silent monitor call)</td>
</tr>
<tr>
<td></td>
<td>CALL_DELIVERED_EVENT</td>
<td>-</td>
<td>ACTIVE</td>
<td>ALERTING</td>
<td>ACTIVE (original call)</td>
<td>ACTIVE (original call)</td>
<td>INITIATED (silent monitor call)</td>
</tr>
</tbody>
</table>
The following table provides a list of CTI call events and their mapping to the Dialog state and Participant state for a barge call.

### Table 6: Barge Call

<table>
<thead>
<tr>
<th>Scenario</th>
<th>CTI Event</th>
<th>Event Method</th>
<th>Dialog State (Original Call)</th>
<th>Dialog State (Silent Monitor Call)</th>
<th>Participant State (Caller)</th>
<th>Participant State (Agent A)</th>
<th>Participant State (Supervisor)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent call arrives and is answered</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>ACTIVE</td>
<td>ACTIVE (original call)</td>
<td>ACTIVE (original call)</td>
<td>ACTIVE (silent monitor call)</td>
</tr>
<tr>
<td>Supervisor silent monitors the call</td>
<td>-</td>
<td>-</td>
<td>ACTIVE (original call)</td>
<td>ACTIVE</td>
<td>ACTIVE (silent monitor call)</td>
<td>ACTIVE (passive, silent monitor call)</td>
<td>ACTIVE (silent monitor call)</td>
</tr>
<tr>
<td>Supervisor starts barge call</td>
<td>-</td>
<td>POST (BARGE)</td>
<td>ACTIVE (original call)</td>
<td>ACTIVE</td>
<td>ACTIVE (original call)</td>
<td>ACTIVE (passive, silent monitor call)</td>
<td>ACTIVE (silent monitor call)</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Scenario</th>
<th>CTI Event</th>
<th>Event Method</th>
<th>Dialog State</th>
<th>Participant State (Caller)</th>
<th>Participant State (Agent A)</th>
<th>Participant State (Supervisor)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finesse drops silent monitor call through Unified CCE</td>
<td>CALL_CONNECTION _CLEARED (silent monitor call)</td>
<td>-</td>
<td>ACTIVE (original call)</td>
<td>DROPPED (silent monitor call)</td>
<td>ACTIVE (original call)</td>
<td>DROPPED (silent monitor call)</td>
</tr>
<tr>
<td></td>
<td>CALL_CLEARED (silent monitor call)</td>
<td></td>
<td></td>
<td></td>
<td>ACTIVE (original call)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>END_CALL (silent monitor call)</td>
<td></td>
<td></td>
<td></td>
<td>ACTIVE (silent monitor call)</td>
<td></td>
</tr>
<tr>
<td>Unified CCE puts original call on hold</td>
<td>CALL_HELD (original call)</td>
<td>-</td>
<td>ACTIVE (original call)</td>
<td>ACTIVE (original call)</td>
<td>HELD (original call)</td>
<td>Not a participant yet</td>
</tr>
<tr>
<td>Unified CCE generates consult call</td>
<td>BEGIN_CALL (consult call)</td>
<td>-</td>
<td>ACTIVE (original call)</td>
<td>INITIATING (consult call)</td>
<td>ACTIVE (original call)</td>
<td>Not a participant yet</td>
</tr>
<tr>
<td></td>
<td>CALL_SERVICE_INITIATED EVENT (consult call)</td>
<td></td>
<td></td>
<td></td>
<td>INITIATING (consult call)</td>
<td></td>
</tr>
<tr>
<td>Unified CCE dials supervisor's extension</td>
<td>CALL_ORIGINATED EVENT (consult call)</td>
<td>-</td>
<td>ACTIVE (original call)</td>
<td>INITIATED (consult call)</td>
<td>ACTIVE (original call)</td>
<td>Not a participant yet</td>
</tr>
<tr>
<td>Agent receives the consult call</td>
<td>CALL_DELIVERED (consult call)</td>
<td>-</td>
<td>ACTIVE (original call)</td>
<td>INITIATED (consult call)</td>
<td>ACTIVE (original call)</td>
<td>Not a participant yet</td>
</tr>
<tr>
<td>Supervisor receives the consult call</td>
<td>CALL_DELIVERED (consult call)</td>
<td>-</td>
<td>ACTIVE (original call)</td>
<td>ALERTING (consult call)</td>
<td>ACTIVE (original call)</td>
<td>ALERTING</td>
</tr>
<tr>
<td>Unified CCE answers the consult call on behalf of the supervisor and</td>
<td>CALL_CONFERENCED</td>
<td>-</td>
<td>ACTIVE (original call)</td>
<td>ALERTING (consult call)</td>
<td>ACTIVE (original call)</td>
<td>ALERTING</td>
</tr>
<tr>
<td>changes the original agent call to a conference call</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 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.

#### Scenario

<table>
<thead>
<tr>
<th>Scenario</th>
<th>CTI Event</th>
<th>Event Method</th>
<th>Dialog State</th>
<th>Participant State (Caller)</th>
<th>Participant State (Agent A)</th>
<th>Participant State (Supervisor)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unified CCE ends the consult call</td>
<td>END_CALL (consult call)</td>
<td></td>
<td></td>
<td>ACTVE (original call)</td>
<td>HELD (original call)</td>
<td>DROPPED (consult call)</td>
</tr>
<tr>
<td>Unified CCE changes the original call type to conference</td>
<td>CALL_DATA_UPDATE (original call)</td>
<td></td>
<td></td>
<td>ACTVE (original call)</td>
<td>ACTVE (original call, callType=15 =Conference)</td>
<td>DROPPED (consult call)</td>
</tr>
<tr>
<td>Unified CCE answers call on behalf of supervisor</td>
<td>CALL_ESTABLISHED (original call)</td>
<td></td>
<td></td>
<td>ACTVE (original call)</td>
<td>ACTVE (original call)</td>
<td>ACTVE</td>
</tr>
</tbody>
</table>

If the caller is also an agent, the caller receives a dialog update (PUT) with an updated participant list on the conference.

#### 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

<table>
<thead>
<tr>
<th>Call Type</th>
<th>Progressive</th>
<th>Predictive</th>
<th>Preview</th>
<th>Direct Preview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reservation Call</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer Call</td>
<td>callType: OUTBOUND DIRECT PREVIEW BAStatus: PROGRESSIVE OUTBOUND</td>
<td>callType: OUTBOUND DIRECT PREVIEW BAStatus: PREDICTIVE OUTBOUND</td>
<td>callType: OUTBOUND DIRECT PREVIEW BAStatus: OUTBOUND RESERVATION</td>
<td></td>
</tr>
<tr>
<td>Callback Reservation Call</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Callback Customer Call</td>
<td>callType: OUTBOUND CALLBACK BAStatus: PROGRESSIVE OUTBOUND</td>
<td>callType: OUTBOUND CALLBACK BAStatus: PREDICTIVE OUTBOUND</td>
<td>callType: OUTBOUND CALLBACK BAStatus: OUTBOUND RESERVATION</td>
<td></td>
</tr>
<tr>
<td>Personal Callback Reservation Call</td>
<td>callType: OUTBOUND PERSONAL CALLBACK PREVIEW BAStatus: PERSONAL CALLBACK OUTBOUND RESERVATION</td>
<td>callType: OUTBOUND PERSONAL CALLBACK PREVIEW BAStatus: PERSONAL CALLBACK OUTBOUND RESERVATION</td>
<td>callType: OUTBOUND PERSONAL CALLBACK PREVIEW BAStatus: PERSONAL CALLBACK OUTBOUND RESERVATION</td>
<td></td>
</tr>
</tbody>
</table>
### Table 8: Outbound Call Types and BAStatus for Finesse with Unified CCX

<table>
<thead>
<tr>
<th></th>
<th>Progressive</th>
<th>Predictive</th>
<th>Preview</th>
<th>Direct Preview</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Personal Callback</strong></td>
<td>callType: OUTBOUND_</td>
<td>callType: OUTBOUND_</td>
<td>callType: OUTBOUND_</td>
<td>callType: OUTBOUND_</td>
</tr>
<tr>
<td><strong>Customer Call</strong></td>
<td>PERSONAL_CALLBACK</td>
<td>PERSONAL_CALLBACK</td>
<td>PERSONAL_CALLBACK</td>
<td>PERSONAL_CALLBACK</td>
</tr>
<tr>
<td><strong>BAStatus</strong></td>
<td>BAStatus: PERSONAL_CALLBACK</td>
<td>BAStatus: PERSONAL_CALLBACK</td>
<td>BAStatus: PERSONAL_CALLBACK</td>
<td>BAStatus: PERSONAL_CALLBACK</td>
</tr>
<tr>
<td><strong>OUTBOUND</strong></td>
<td>OUTBOUND</td>
<td>OUTBOUND</td>
<td>OUTBOUND</td>
<td>OUTBOUND</td>
</tr>
<tr>
<td><strong>DIRECT_PREVIEW</strong></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td><strong>CALLBACK</strong></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

### Disposition Code Parameter Values for Nonvoice Tasks

The following table describes possible values for the dispositionCode response parameter for nonvoice tasks:
<table>
<thead>
<tr>
<th>Type of Code</th>
<th>Disposition Code Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal End</td>
<td>CD_NORMAL_END_TASK</td>
<td>The task ended normally.</td>
</tr>
<tr>
<td>Transfer</td>
<td>CD_TASK_TRANSFER</td>
<td>The task was transferred. The initiating application sends a new task request to CCE for routing that includes the task id of the first task.</td>
</tr>
<tr>
<td></td>
<td>CD_TASK_TRANSFERRED_ON_AGENT_LOGOUT</td>
<td>The task was transferred because the agent logged out during the task.</td>
</tr>
<tr>
<td>RONA</td>
<td>CD_RING_NO_ANSWER</td>
<td>The task timed out while waiting to be accepted by an agent. The task was redirected to another agent.</td>
</tr>
<tr>
<td>Task Lifetime Exceeded</td>
<td>CD_MAX_DIALOG_LIFETIME_EXCEEDED</td>
<td>The dialog ended because it exceeded the maximum task duration for the MRD.</td>
</tr>
<tr>
<td>Customer Abandoned</td>
<td>CD_TASK_ABANDONED_WHILE_OFFERED</td>
<td>The customer cancelled the task before the agent began working on the task.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In this case, the Finesse user sees the offered dialog but the dialog is deleted before the user can accept it.</td>
</tr>
</tbody>
</table>
### Dialog API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>20700 (conference resource limit violation)</td>
<td>The barge call will cause the total number of parties on the conference call to exceed the allowed resource limit for the conference bridge.</td>
</tr>
<tr>
<td>400</td>
<td>20999 (Barge via a non-conference-controller)</td>
<td>The agent specified in the toAddress is not the controller of the conference call or the agent already has an outstanding conference call.</td>
</tr>
<tr>
<td>400</td>
<td>Generic Error</td>
<td>An unaccounted for error occurred. The root cause could not be determined.</td>
</tr>
<tr>
<td>Status</td>
<td>Error Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 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, BStatus, BADialedListID, BATimeZone, BABuddyName, BCustomerNumber (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 fromAddress 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). |
| 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. |
<p>| 401    | Invalid State | The targetMediaAddress in a Dialog—Start Recording request specifies an extension of a participant in HELD state. |</p>
<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>Invalid Supervisor</td>
<td>A supervisor tried to change the state of an agent who does not belong to that supervisor's team.</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>The resource specified is invalid or does not exist.</td>
</tr>
<tr>
<td>404</td>
<td>Dialog Not Found</td>
<td>The dialogId provided is invalid or no such dialog exists.</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
<tr>
<td>501</td>
<td>Not Implemented</td>
<td>A user attempted to use the API in a deployment where it is not supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, a recording attempt was made in a Unified CCE deployment.</td>
</tr>
<tr>
<td>503</td>
<td>Service Unavailable</td>
<td>The required service is unavailable. For example, the Notification Service is not running.</td>
</tr>
</tbody>
</table>

**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:

```xml
<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.

This API is only supported for a stand-alone Finesse deployment with Unified CCE and not applicable for coresident Finesse deployment with Unified CCX.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Queue/ id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Queue/10">http://finesse1.xyz.com/finesse/api/Queue/10</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Any user can use this API to retrieve information about a specific queue. The user does not need to belong to that queue.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>401: Authorization Failure</td>
</tr>
<tr>
<td></td>
<td>404: Not Found</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
<tr>
<td>Example Response:</td>
<td>&lt;Queue&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;uri&gt;/finesse/api/Queue/10&lt;/uri&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;name&gt;Sales&lt;/name&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;statistics&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;callsInQueue&gt;3&lt;/callsInQueue&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;startTimeOfLongestCallInQueue&gt;2012-02-15T17:58:21Z&lt;/startTimeOfLongestCallInQueue&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;agentsReady&gt;1&lt;/agentsReady&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;agentsNotReady&gt;2&lt;/agentsNotReady&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;agentsTalkingInbound&gt;3&lt;/agentsTalkingInbound&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;agentsTalkingOutbound&gt;4&lt;/agentsTalkingOutbound&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;agentsTalkingInternal&gt;5&lt;/agentsTalkingInternal&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;agentsWrapUpNotReady&gt;6&lt;/agentsWrapUpNotReady&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;agentsWrapUpReady&gt;7&lt;/agentsWrapUpReady&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/statistics&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Queue&gt;</td>
</tr>
<tr>
<td>Example Failure Response:</td>
<td>&lt;ApiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ApiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorType&gt;Authorization Failure&lt;/ErrorType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorMessage&gt;UNAUTHORIZED&lt;/ErrorMessage&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorData&gt;jsmith&lt;/ErrorData&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ApiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ApiErrors&gt;</td>
</tr>
</tbody>
</table>
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

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.

Note

This API is only supported for a stand-alone Finesse deployment with Unified CCE and not applicable for coresident Finesse deployment with Unified CCX.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;/Queues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234/Queues">http://finesse1.xyz.com/finesse/api/User/1234/Queues</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>All users can use this API to retrieve a list of queues for any user.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>401: Authorization Failure</td>
</tr>
<tr>
<td></td>
<td>404: User Not Found</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
</tbody>
</table>
Precision Queues are not visible for a logged out agent.

**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
## Queue API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the Queue object.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td>A unique identifier for the queue. This identifier is the PeripheralNumber from t_Skill_Group in AWDB.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the queue.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>statistics</td>
<td>Collection</td>
<td>A list of statistics for the queue.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>--callsInQueue</td>
<td>Integer</td>
<td>The number of calls currently queued to this queue.</td>
<td>—</td>
<td>If the queue is not assigned to an agent or supervisor, this value is -1.</td>
</tr>
<tr>
<td>--startTimeOfLongestCallInQueue</td>
<td>String</td>
<td>The start time of the longest call in the queue. The format for this parameter is YYYY-MM-DDThh:MM:ssZ.</td>
<td>—</td>
<td>If the queue is not assigned to an agent or supervisor, this value is -1.</td>
</tr>
<tr>
<td>--agentsReady</td>
<td>Integer</td>
<td>The number of agents assigned to the queue who are in READY state.</td>
<td>—</td>
<td>If the queue is not assigned to an agent or supervisor, this value is -1.</td>
</tr>
<tr>
<td>--agentsNotReady</td>
<td>Integer</td>
<td>The number of agents assigned to the queue who are in NOT_READY state.</td>
<td>—</td>
<td>If the queue is not assigned to an agent or supervisor, this value is -1.</td>
</tr>
<tr>
<td>--agentsTalkingInbound</td>
<td>Integer</td>
<td>The number of agents assigned to the queue who are in TALKING state on inbound calls.</td>
<td>—</td>
<td>If the queue is not assigned to an agent or supervisor, this value is -1.</td>
</tr>
</tbody>
</table>
### Configuring Queue Statistics

The Queue Statistics gadget is enabled by default as part of Cisco Finesse new installation (Unified CCE only). When performing a system upgrade from Cisco Finesse 11.5(1), the desktop custom layout needs to be modified by the administrator for the Queue Statistics gadget to be displayed on the Agent and Supervisor desktop.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>--&gt;agentsTalking Outbound</td>
<td>Integer</td>
<td>The number of agents assigned to the queue who are in TALKING state on outbound calls.</td>
<td>—</td>
<td>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.</td>
</tr>
<tr>
<td>--&gt;agentsTalking Internal</td>
<td>Integer</td>
<td>The number of agents assigned to the queue who are in Talking state on internal calls. Internal calls are consult calls. When an agent on a routed call initiates an internal consult call, this statistic is incremented for the queue associated with the original call.</td>
<td>—</td>
<td>If the queue is not assigned to an agent or supervisor, this value is -1.</td>
</tr>
<tr>
<td>--&gt;agentsWrapUp NotReady</td>
<td>Integer</td>
<td>The number of agents assigned to the queue who are in Work Not Ready state.</td>
<td>—</td>
<td>If the queue is not assigned to an agent or supervisor, this value is -1.</td>
</tr>
<tr>
<td>--&gt;agentsWrapUp Ready</td>
<td>Integer</td>
<td>The number of agents assigned to the queue who are in Work Ready state.</td>
<td>—</td>
<td>If the queue is not assigned to an agent or supervisor, this value is -1.</td>
</tr>
</tbody>
</table>
Use the following CLI commands to enable and disable the queue statistics polling or check the status of the queue statistics polling:

- `utils finesse queue_statistics enable`
- `utils finesse queue_statistics disable`
- `utils finesse queue_statistics status`

After performing a system upgrade, during switch-version the queue statistics polling will be enabled by default. The procedure to disable the queue statistics polling remains the same.

---

**Note**

When enabled, Queue Statistics supports a maximum of 1500 users (Agents and Supervisors).

### Queue API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>Unauthorized (for example, the user is not yet authenticated in the Web Session).</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>The resource specified is invalid or does not exist.</td>
</tr>
<tr>
<td>404</td>
<td>User Not Found</td>
<td>The user ID provided is invalid or is not recognized. No such user exists in CTI.</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
</tbody>
</table>

### 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*.

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>
    </User>
  </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.

The URI for this API contains the parameter includeLoggedOutAgents. This parameter is optional and can be set to:

- **True** or **Empty**: Includes all the agents of that team in the list (with the logged out agents).
- **False**: Includes only the logged in agents in the list.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Team/&lt;id&gt;?includeLoggedOutAgents=true</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Team/10?includeLoggedOutAgents=true">http://finesse1.xyz.com/finesse/api/Team/10?includeLoggedOutAgents=true</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Supervisors can use this API to get a list of users assigned to their team.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>Request Parameters:</td>
<td>id (required): The ID of the user includeLoggedOutAgents (optional): Returns the list with all the agents in that team</td>
</tr>
</tbody>
</table>
| HTTP Response: | 200: Success  
401: Authorization Failure  
404: Not Found  
500: Internal Server Error |
|-----------------|-----------------|
| **Example Response for Unified CCE deployment:** | `<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 Response for Unified CCX deployment:

```xml
<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>
      <mediaState>BUSY</mediaState>
      <dialogs>/finesse/api/User/1234/Dialogs</dialogs>
      <extension>1001001</extension>
      <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>Brown</lastName>
      <dialogs>/finesse/api/User/1235/Dialogs</dialogs>
      <extension>1001002</extension>
      <reasonCode>
        <category>NOT_READY</category>
        <code>12</code>
        <label>Lunch Break</label>
        <id>1</id>
        <uri>/finesse/api/ReasonCode/1</uri>
      </reasonCode>
      <state>NOT_READY</state>
      <stateChangeTime>2012-03-01T18:22:25.123Z</stateChangeTime>
    </User>
  </users>
</Team>
```

Example Failure Response:

```xml
<ApiErrors>
  <ApiError>
    <ErrorType>Authorization Failure</ErrorType>
    <ErrorMessage>UNAUTHORIZED</ErrorMessage>
    <ErrorData>j smith</ErrorData>
  </ApiError>
</ApiErrors>
```

## Team API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the Team object.</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td>The unique identifier for the team.</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the team.</td>
<td></td>
<td>—</td>
</tr>
</tbody>
</table>
### Team API Errors

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

### 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:

```xml
<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.

<table>
<thead>
<tr>
<th><strong>URI:</strong></th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;/ClientLog</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example URI:</strong></td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234/ClientLog">http://finesse1.xyz.com/finesse/api/User/1234/ClientLog</a></td>
</tr>
<tr>
<td><strong>HTTP Method:</strong></td>
<td>POST</td>
</tr>
<tr>
<td><strong>Content Type:</strong></td>
<td>Application/XML</td>
</tr>
<tr>
<td><strong>Input/Output Format:</strong></td>
<td>XML</td>
</tr>
</tbody>
</table>

**HTTP Request:**

```xml
<ClientLog>
  <logData>
    xxxxxxxxxxxxxxx
    xxxxxxxxxxxxxxx
  </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:**

```xml
<ApiErrors>
  <ApiError>
    <ErrorType>User Not Found</ErrorType>
    <ErrorMessage>UNKNOWN_USER</ErrorMessage>
    <ErrorData>4023</ErrorData>
  </ApiError>
</ApiErrors>
```
ClientLog API Parameters

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

ClientLog API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Parameter Missing</td>
<td>The logData parameter is not present.</td>
</tr>
<tr>
<td>400</td>
<td>Invalid Input</td>
<td>The size of the logData exceeds 1,048,576 characters.</td>
</tr>
<tr>
<td>400</td>
<td>Operation Failure</td>
<td>The POST client log operation failed.</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>The user is not yet authenticated in the Web Session.</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Authorization User Specified</td>
<td>The authenticated user tried to make a request for another user.</td>
</tr>
<tr>
<td>405</td>
<td>Method Not Allowed</td>
<td>GET or PUT HTTP method not allowed for client-side log collection.</td>
</tr>
</tbody>
</table>

Task Routing APIs

Task Routing APIs provide a standard way to request, queue, route, and handle third-party multichannel tasks in CCE.

Contact Center customers or partners can develop applications using SocialMiner and Finesse APIs in order to use Task Routing. The SocialMiner Task API enables applications to submit nonvoice task requests to CCE. The Finesse APIs enable agents to sign into different types of media and handle the tasks. Agents sign into and manage their state in each media independently.

Cisco partners can use the sample code available on Cisco DevNet as a guide for building these applications (https://developer.cisco.com/site/task-routing/).

For Finesse, the APIs used for Task Routing include the Media APIs and some of the Dialog and User APIs.
This API is only supported for a stand-alone Finesse deployment with Unified CCE and not applicable for coresident Finesse deployment with Unified CCX.

Related Topics
  Failure Handling for Task Routing Clients, on page 320

Media

The Media object represents a user's state in a Media Routing Domain (MRD). The Media object is structured as follows:

```xml
<Media>
  <uri>/finesse/api/User/1001004/Media/5000</uri>
  <description>Chat MRD</description>
  <dialogLogoutAction>CLOSE</dialogLogoutAction>
  <id>5000</id>
  <interruptible>true</interruptible>
  <maxDialogLimit>10</maxDialogLimit>
  <name>Cisco_Chat_MRD</name>
  <ReasonCode>
    <category>NOT_READY</category>
    <code>10</code>
    <forAll>true</forAll/>
    <id>16</id>
    <label>Team Meeting</label>
    <uri>/finesse/api/ReasonCode/16</uri>
  </ReasonCode>
  <routable>true</routable>
  <state>NOT_READY</state>
</Media>
```

Media APIs

Media - Sign in

The Media—Sign in API allows a user to sign in to an individual non-voice Media Routing Domain (MRD) on CCE. If the response is successful, the user is signed in to Finesse and is automatically placed in NOT_READY state and made routable for that MRD. Routable means that CCE is allowed to assign an agent tasks in the MRD.

If five consecutive sign-ins fail due to an incorrect password, Finesse blocks access to the user account for a period of 5 minutes.

If a user is already signed in and attempts to sign in again, the user receives an error.

Some parameters used in this API are only known to the Finesse side on which the user signed in. If the user switches sides, the user must sign in again to have this functionality work correctly.
Finesse does not support a user staying signed into both Finesse servers at the same time, through either the REST API or xmpp subscriptions.

The user xmpp presence determines which side a user is signed into, in order to perform actions on the user’s behalf. These actions include transferring nonvoice dialogs automatically and either accepting or ignoring interrupts. Finesses transfers nonvoice dialogs automatically if an agent does not accept a dialog within the StartTimeout threshold for the MRD, and if the agent is set to transfer dialogs on sign out in the MRD.

| URI: | http://<FQDN>/finesse/api/User/<id>/Media/<mrdId> |
| Example URI: | http://finesse1.xyz.com/finesse/api/User/1234/Media/5001 |
| Security Constraints: | Users can only act on their own Media objects. |
| HTTP Method: | PUT |
| Content Type: | Application/XML |
| Input/Output Format: | XML |
| HTTP Request: | `<Media>`  
| | `<maxDialogLimit>10</maxDialogLimit>`  
| | `<state>LOGIN</state>`  
| | `<interruptAction>ACCEPT</interruptAction>`  
| | `<dialogLogoutAction>CLOSE</dialogLogoutAction>` |
| Request Parameters: | id (required): The ID of the user  
| | mrdId (required): The ID of the MRD  
| | maxDialogLimit (required): The maximum number of concurrent dialogs this user is allowed to handle in the MRD. Each dialog represents a task.  
| | state (required): The new state that the user wants to be in (LOGIN)  
| | interruptAction (required): Defines the behavior when an agent is handling a task in an interruptible MRD and is interrupted by a task or call from a non-interruptible MRD. Finesse can ACCEPT the interrupt; the agent is put into INTERRUPTED state and cannot work on dialogs in the interrupted MRD. Finesse can IGNORE the interrupt; the agent's state does not change and the agent can continue to work on the dialogs in the MRD.  
| | dialogLogoutAction(optional): Determines whether to TRANSFER or CLOSE active tasks when an agent logs out of the MRD. If not specified, this parameter is set to CLOSE. |
| Header Parameters: | requestId: A user provided unique string used to correlate originating request with the resulting HTTP response. This parameter is not part of the resulting event/events. |
HTTP Response: 202: Successfully Accepted

Note: The requestId is included in the response header if provided.
This response only indicates successful completion of the request. The request is processed and the actual response is sent as part of a media notification.

400: Bad Request (for example, malformed or incomplete request)
400: Parameter Missing
401: Unauthorized (for example, the user is not authenticated in the Web Session)
404: Not Found (for example, the user ID or mrdId is not known)
503: Service Unavailable (for example, the Notification Service is not running)

Example Failure Response:

```xml
<ApiErrors>
  <ApiError>
    <ErrorData>1</ErrorData>
    <ErrorMedia>5001</ErrorMedia>
    <ErrorMessage>E_ARM_STAT_AGENT_ALREADY_LOGGED_IN</ErrorMessage>
    <ErrorType>Agent already logged into MRD</ErrorType>
  </ApiError>
</ApiErrors>
```

Notifications Triggered: Media notification

Asynchronous Errors

If an error occurs after the initial validation is complete, an error notification is sent over XMPP to the Media notification. The ErrorMedia parameter in the ApiError information indicates the Media Routing Domain to which the error applies.

Related Topics
- Media and Dialogs/Media Asynchronous Error Notification, on page 307

Media—Change State or Sign Out

This API allows a user to change state in or sign out of an individual nonvoice Media Routing Domain.

See Agent States for Nonvoice Media, on page 156 for information about the agent states you can set with this API.

Users can sign out with active tasks. The user's tasks are either automatically transferred or closed, depending on the way the MRD was configured when the user signed in through the Media - Sign In API. To transfer tasks, Finesse resubmits the tasks into the system as new tasks.

URI: http://<FQDN>/finesse/api/User/<id>/Media/<mrdId>

Example URI: http://finesse1.xyz.com/finesse/api/User/1234/Media/5001

Security Constraints:
- Agents and supervisors can use this API.
- Users can only act on their own Media objects.
HTTP Method: PUT
Content Type: Application/XML
Input/Output Format: XML

**HTTP Request:**
```xml
<Media>
  <state>LOGOUT</state>
</Media>
```

**Request Parameters:**
id (required): The ID of the user
mrdId (required): The ID of the MRD
state (required): The new state that the user wants to be in (READY, NOT_READY, LOGIN, or LOGOUT)

**Header Parameters:**
requestId: A user provided unique string used to correlate originating request with the resulting HTTP response or asynchronous error. This parameter is not part of the resulting event/events.

**HTTP Response:**
202: Successfully Accepted

*Note* The requestId is included in the response header if provided.
This response only indicates successful completion of the request. The request is processed and the actual response is sent as part of a media notification.

400: Bad Request (for example, malformed or incomplete request)
401: Unauthorized (for example, the user is not authenticated in the Web Session)
404: Not Found (for example, the user ID or mrdId is not known)
503: Service Unavailable (for example, the Notification Service is not running)

**Example Failure Response:**
```xml
<ApiErrors>
  <ApiError>
    <ErrorData>6</ErrorData>
    <ErrorMedia>5001</ErrorMedia>
    <ErrorMessage>E_ARMSTAT_AGENT_NOT_LOGGED_IN</ErrorMessage>
    <ErrorType>Agent is not logged in</ErrorType>
  </ApiError>
</ApiErrors>
```

**Notifications Triggered:**
Media notification

*Note* The system ignores requests to change agent state from READY to READY; these requests do not trigger a notification.

**Asynchronous Errors**
If an error occurs after the initial validation is complete, an error notification is sent over XMPP to the Media notification. The requestId is included in the response XML. The ErrorMedia parameter in the ApiError information indicates the Media Routing Domain to which the error applies.
Media—Change Agent State with Reason Code

This API allows a user to change the agent state in an individual non-voice Media Routing Domain, 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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;/Media/&lt;mrdId&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234/Media/5001">http://finesse1.xyz.com/finesse/api/User/1234/Media/5001</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Agents and supervisors can use this API. Users can only act on their own Media objects.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>
| HTTP Request: | <Media>
  <state>NOT_READY</state>
  <reasonCodeId>1001</reasonCodeId>
</Media> |
| Request Parameters: | id (required): The ID of the user
mrdId (required): The ID of the Media Routing Domain
reasonCodeId (required if reason codes are configured for the given state): The database ID for the reason code
state (required): The new state that the user wants to be in (NOT_READY or LOGOUT) |
| Header Parameters: | requestId: A user provided unique string used to correlate originating request with the resulting HTTP response or asynchronous error. This parameter is not part of the resulting event/events. |
HTTP Response: | 202: Successfully Accepted  
| Note | The requestId is included in the response header if provided.  
|      | This response only indicates successful completion of the request. The request is processed and the actual response is sent as part of a media notification.  
| 400: Bad Request (for example, malformed or incomplete request)  
| 400: Parameter Missing  
| 401: Unauthorized (for example, the user is not authenticated in the Web Session)  
| 404: Not Found (for example, the user ID or mrdId is not known)  
| 503: Service Unavailable (for example, the Notification Service is not running)  

Example Failure Response:  

| <ApiErrors>  
| <ApiError>  
|     <ErrorData>1</ErrorData>  
|     <ErrorMedia>5001</ErrorMedia>  
|     <ErrorMessage>E_ARM_STAT_AGENT_ALREADY_LOGGED_IN</ErrorMessage>  
|     <ErrorType>Agent already logged into MRD</ErrorType>  
| </ApiError>  
| </ApiErrors>  

Notifications Triggered: | Media notification  

Asynchronous Errors  

If an error occurs after the initial validation is complete, an error notification is sent over XMPP to the Media notification. The requestId is included in the response XML. The ErrorMedia parameter in the ApiError information indicates the Media Routing Domain to which the error applies.

Related Topics  

- Media and Dialogs/Media Asynchronous Error Notification, on page 307  
- Agent States for Nonvoice Media, on page 156

Media—Change Agent to Routable/Not Routable  

The Media—Change Agent to Routable/Not Routable API allows a user to set an agent's routable mode in a Media Routing Domain. Routable mode determines whether CCE can route tasks to an agent in a Media Routing Domain.

When the routable parameter is set to true, the agent is routable. CCE can assign task to the agent in that MRD.

When the routable parameter is set to false, the agent is not routable. CCE cannot assign tasks to the agent in that MRD.

Make the agent not routable to stop sending tasks to the agent without changing the agent's state to NOT_READY. If an agent changes to NOT_READY state while still working on tasks, those tasks appear ended in CCE reports; time spent working on the tasks after going Not Ready is not counted. You may want to make the agent not routable near the end of the agent's shift, to allow the agent to finish final tasks without being assigned more tasks and to report accurately on those final tasks.
In a RONA situation, in which a task is resubmitted because an agent does not accept a task within the MRD's Start Timeout threshold, Finesse automatically makes the agent not routable.

If a user sets the agent's mode to not routable when an agent has pending incoming tasks or has not started an accepted task, the agent's mode does not change until the agent has started these tasks.

The agent's mode is set to routable automatically when the agent signs in, and when the agent changes to READY state.

| **URI:** | http://<FQDN>/finesse/api/User/<id>/Media/<mrdId> |
| **Example URI:** | http://finesse1.xyz.com/finesse/api/User/1234/Media/5001 |
| **Security Constraints:** | Users can only act on their own Media objects. |
| **HTTP Method:** | PUT |
| **Content Type:** | Application/XML |
| **Input/Output Format:** | XML |
| **HTTP Request:** | <Media>
  <routable>true</routable>
</Media> |
| **Request Parameters:** | id (required): The ID of the user
  mrdId (required): The ID of the MRD
  routable (required): Indicates whether CCE can route tasks to the user in the MRD. |
| **Header Parameters:** | requestId: A user provided unique string used to correlate originating request with the resulting HTTP response or asynchronous error. This parameter is not part of the resulting event/events. |
| **HTTP Response:** | 202: Successfully Accepted

**Note**

The requestId is included in the response header if provided.

This response only indicates successful completion of the request. The request is processed and the actual response is sent as part of a media notification.

400: Bad Request (for example, invalid input for parameters)
400: Parameter Missing
401: Unauthorized (for example, the user is not authenticated in the Web Session)
404: Not Found (for example, the user ID or mrdId is not known)
500: Internal Server Error
Example Failure Response:

```xml
<ApiErrors>
  <ApiError>
    <ErrorData>1</ErrorData>
    <ErrorMedia>5001</ErrorMedia>
    <ErrorMessage>E_ARM_STAT_ALREADY_IN_REQUESTED_AGENT_MODE</ErrorMessage>
    <ErrorType>Agent already in requested mode</ErrorType>
  </ApiError>
</ApiErrors>
```

Notifications Triggered: Media notification

Asynchronous Errors

If an error occurs after the initial validation is complete, an error notification is sent over XMPP to the Media notification. The requestId is included in the response XML. The ErrorMedia parameter in the ApiError information indicates the Media Routing Domain to which the error applies.

Media—Get Media

This API allows a user to get a copy of a Media object for a specified agent. This API can be used to return only nonvoice Media objects.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/User/&lt;id&gt;/Media/&lt;mrdId&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/User/1234/Media/5001">http://finesse1.xyz.com/finesse/api/User/1234/Media/5001</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Users can only act on their own Media objects.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>Request Parameters:</td>
<td>id (required): The ID of the user mrdId (required): The ID of the Media Routing Domain</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success 400: Bad Request (for example, malformed or incomplete request) 400: Parameter Missing 401: Unauthorized (for example, the user is not authenticated in the Web Session) 404: Not Found (for example, the user ID or mrdId is not known)</td>
</tr>
</tbody>
</table>
Response if the agent is assigned to skill groups in the Media Routing Domain:

```
<Media>
  <uri>/finesse/api/User/1001004/Media/5000</uri>
  <description>Chat MRD</description>
  <dialogLogoutAction>CLOSE</dialogLogoutAction>
  <id>5000</id>
  <interruptible>false</interruptible>
  <maxDialogLimit>10</maxDialogLimit>
  <name>Cisco_Chat_MRD</name>
  <ReasonCode>
    <category>NOT_READY</category>
    <code>10</code>
    <forAll>true</forAll/>
    <id>16</id>
    <label>Team Meeting</label>
  </ReasonCode>
  <reasonCodeId>16</reasonCodeId>
  <routable>true</routable>
  <state>NOT_READY</state>
  <interruptAction>IGNORE</interruptAction>
</Media>
```

Response if the agent is not assigned to skill groups in the Media Routing Domain:

```
<Media>
  <uri>/finesse/api/User/1001004/Media/5002</uri>
  <description>Chat MRD</description>
  <id>5002</id>
  <interruptible>false</interruptible>
  <name>Cisco_Chat_MRD2</name>
</Media>
```

Example Failure Response:

```
@apiErrors
  @apiError
    @errorMsgData>1002001</errorMsgData>
    @errorMsg>The user specified in the authentication credentials and the uri don't match</errorMsg>
    @errorType>Invalid Authorization User Specified</errorType>
  </apiError
</apiErrors>
```

Media—Get List

This API allows a user to get a list of Media objects for all nonvoice Media Routing Domains (MRDs) configured on Unified CCE.

If the agent belongs to a skill group in the MRD, the media object includes the agent's state information for that MRD.

**URI:**  
`http://<FQDN>/finesse/api/User/<id>/Media`

**Example URI:**  
`http://finesse1.xyz.com/finesse/api/User/1234/Media`

**Security Constraints:**  
Users can only act on their own Media objects.

**HTTP Method:**  
GET

**Content Type:**  
—
Agent States for Nonvoice Media

Users can set the following states with the Media APIs:

- LOGIN
- READY
- NOT_READY
- LOGOUT

Users enter the following states automatically while on a task. Users cannot place themselves in these states. For example, agents enter ACTIVE state when they accept a task.

- RESERVED
- ACTIVE
- PAUSED
- INTERRUPTED
- WORK_READY
The agent enters WORK_NOT_READY state automatically if the Finesses server on which the agent is signed in disconnects. When agent signs in again or Finesses side reconnects to CCE, the agent is moved out of the WORK_NOT_READY state. This state cannot be set from the agent desktop.

If an agent is configured to work on a maximum of one task in an MRD, the agent's state in the MRD reflects the agent's activity on that task. However, an agent can be configured to work on several tasks at once in an MRD. The following state hierarchy determines the agent's state in that MRD:

1. LOGIN/LOGOUT
2. READY/NOT_READY
3. INTERRUPTED
4. ACTIVE
5. WORK_READY
6. PAUSED
7. RESERVED

Consider this state hierarchy example. An agent is handling three tasks in an interruptible MRD:

- Task 1 = PAUSED
- Task 2 = WORK_READY
- Task 3 = ACTIVE

Based on the state hierarchy, the agent's overall state in the MRD is ACTIVE. If a task from another MRD then interrupts this MRD, the agent's state in this MRD changes to INTERRUPTED.

The table describes the agent states for nonvoice MRDs.

<table>
<thead>
<tr>
<th>State</th>
<th>State Information</th>
<th>Allowed Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGIN</td>
<td>The agent's state immediately after signing in. No tasks are assigned to an agent while in this state. The LOGIN state is a transitive state; LOGIN triggers a change that results in a new state (NOT READY).</td>
<td>None; the user transitions to NOT_READY automatically</td>
</tr>
<tr>
<td>NOT_READY</td>
<td>The agent won't be assigned tasks. The agent enters NOT_READY state automatically after signing in. For accurate task durations in reports, do not change agents to NOT_READY state while they have active tasks. Instead, make the agent not routable to stop assigning tasks to the agent. An agent cannot change to NOT READY state if the agent has a pending incoming task. The agent has a pending task if Finesse has an offered dialog for that agent.</td>
<td>• READY • LOGOUT</td>
</tr>
<tr>
<td>State</td>
<td>State Information</td>
<td>Allowed Actions</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| READY            | The agent will be assigned tasks. The agent currently doesn't have any tasks.  
The agent is automatically made routable when the agent enters READY state.  
When an agent completes all tasks in the MRD, the agent's state returns to the READY.                             | • NOT_READY  
• LOGOUT |
| INTERRUPTED      | The agent has been interrupted in this MRD by a task from another MRD.  
An agent can be interrupted from ACTIVE, WORK READY, PAUSED, and RESERVED states.  
The agent cannot perform dialog actions while INTERRUPTED.  
This state is only applicable for interruptible MRDs in which the agent was configured to accept interrupts when signing into the MRD. | • NOT_READY  
• LOGOUT |
| ACTIVE           | The agent has accepted at least one offered task.  
The agent can also have one or more of the following:  
• Paused tasks  
• Offered tasks  
• Tasks for which the agent is performing wrap-up work | • NOT_READY  
• LOGOUT |
| WORK READY       | The agent is performing wrap-up work for all tasks, or is performing wrap-up work for at least one task and has one or more paused tasks.                                                                                           | • NOT_READY  
• LOGOUT |
| PAUSED           | The agent has paused all tasks.                                                                                                                                                                                      | • NOT_READY  
• LOGOUT |
| RESERVED         | The agent has been assigned one or more tasks by CCE, but has not accepted the tasks. The agent does not have active or paused tasks, and is not performing wrap-up work for any tasks.                                              | • NOT_READY  
• LOGOUT |
| LOGOUT           | The agent signed out of the MRD.  
If the agent signs out with active tasks, Finesse either closes or transfers the tasks depending on how the dialogLogoutAction parameter was set for the MRD when the agent signed in. | LOGIN           |
Allowed Actions

State Information

WORK_NOT_READY
The Finesse server on which the agent is signed in disconnected.

When an agent fails over to the secondary Finesse server, the agent must sign in to the media again.
The agent's state after signing in is determined based on the state of the agent's assigned tasks. If the agent doesn't have tasks, the agent is put in NOT_READY state.

Media API Parameters

For parameters specified when a user signs in, including maxDialogLimit, interruptAction, and dialogLogoutAction, the setting for the parameter is correct only when the user is signed in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the Media object.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>A description of the Media Routing Domain (MRD)</td>
<td>—</td>
<td>Any special XML characters in the description are escaped. For example, &quot;&lt;&quot; is replaced with &quot;&lt;&quot;.</td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td>The ID of the user.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>mrdId</td>
<td>String</td>
<td>The ID of the MRD.</td>
<td>—</td>
<td>The size is determined by Unified CCE.</td>
</tr>
<tr>
<td>interruptible</td>
<td>Boolean</td>
<td>Whether a task in this MRD can be interrupted by a task from another MRD.</td>
<td>true, false</td>
<td>—</td>
</tr>
<tr>
<td>maxDialogLimit</td>
<td>Integer</td>
<td>The maximum number of concurrent dialogs this user is allowed to handle in this MRD. Each dialog represents a task.</td>
<td>1 through 10</td>
<td>The maximum value for this parameter is 10.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the MRD.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Possible Values</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>-------</td>
</tr>
<tr>
<td>requestId</td>
<td>String</td>
<td>The earlier sentence was a user provided unique string used to correlate originating request with the resulting HTTP response or asynchronous error. This parameter is not part of the resulting event/events.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReasonCode</td>
<td>Collection</td>
<td>Information about the reason code currently associated with this user.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--&gt;category</td>
<td>String</td>
<td>The category of the reason code.</td>
<td>NOT READY</td>
<td></td>
</tr>
<tr>
<td>--&gt;code</td>
<td>Integer</td>
<td>CTI code associated with this reason code.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--&gt;forAll</td>
<td>Boolean</td>
<td>Whether the reason code is global (true) or non-global (false).</td>
<td>true, false</td>
<td></td>
</tr>
<tr>
<td>--&gt;id</td>
<td>Integer</td>
<td>The ID of the reason code.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--&gt;label</td>
<td>String</td>
<td>The label associated with this reason code.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--&gt;uri</td>
<td>String</td>
<td>The full URI for the reason code.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Possible Values</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| reasonCodeld | Integer | The database ID for the reason code that indicates why the user is in the current state in this MRD. | 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:  
- The agent logged out.  
- 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. |
<p>| routable    | Boolean | Indicates whether CCE can route the tasks to the user in this MRD. When the agent is routable (true), CCE can route tasks to the user. When the agent is not routable (false), CCE cannot route tasks to the agent. | true, false                                                                   | —                                                                                       |
| state       | String  | The state for this user in this MRD.                                       | LOGIN, NOT_READY, READY, LOGOUT, RESERVED, ACTIVE, PAUSED, WORK READY, INTERRUPTED, WORK NOT READY | —                                                                                       |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>stateChangeTime</td>
<td>String</td>
<td>The time at which the state of the user changed to the current state in this MRD. The format for this parameter is YYYY-MM-DDTh:h:MM:ss.SSSZ.</td>
<td>—</td>
<td>This parameter is empty if the time of the state change is not available (if no agent state change notification was received yet).</td>
</tr>
<tr>
<td>interruptAction</td>
<td>String</td>
<td>This parameter only applies to interruptible MRDs. It is ignored for noninterruptible MRDs. An agent setting that defines the behavior when an agent is handling a task in an interruptible MRD and is interrupted by a task or call from a non-interruptible MRD. ACCEPT: The MRD accepts the interrupt event. The agent state is INTERRUPTED in the interruptible MRD and the agent cannot perform any actions on dialogs in that MRD. IGNORE: The MRD does not accept the interrupt event. The agent state does not change in the interruptible MRD and the agent can continue to perform actions on dialogs in that MRD.</td>
<td>ACCEPT, IGNORE</td>
<td>This parameter reflects the configured setting only if you are performing a GET on the Finesse server that the user is signed in to.</td>
</tr>
<tr>
<td>dialogLogoutAction</td>
<td>String</td>
<td>An agent setting that determines whether active tasks are closed or transferred when an agent logs out of an MRD. CLOSE (default): Active tasks are closed when an agent logs out. Finesse sends SocialMiner the task handled events. CCE determines the correct disposition codes for the closed task. TRANSFER: Active tasks are transferred using SocialMiner when an agent logs out. Finesse puts the dialogs in the CLOSED state with the CD_TASK_TRANSFERRED_AGENT_LOGOUT disposition code.</td>
<td>CLOSE, TRANSFER</td>
<td>This parameter reflects the configured setting only if you are performing a GET on the Finesse server that the user is signed in to.</td>
</tr>
</tbody>
</table>
Media API Errors

For synchronous errors, the Media APIs include the requestId in the error response.

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>The request is malformed or incomplete.</td>
</tr>
<tr>
<td>400</td>
<td>Generic Error</td>
<td>An unaccounted for error occurred. The root cause could not be determined.</td>
</tr>
<tr>
<td>400</td>
<td>Invalid Input</td>
<td>One of the parameters provided as part of the user input is invalid or not recognized.</td>
</tr>
<tr>
<td>400</td>
<td>Parameter Missing</td>
<td>The state or requested Action is not provided.</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>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).</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Authorization User Specified</td>
<td>The authenticated user tried to make a request for another user.</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Supervisor</td>
<td>A supervisor tried to change the state of an agent who does not belong to that supervisor’s team.</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>The resource specified is invalid or does not exist.</td>
</tr>
<tr>
<td>503</td>
<td>Service Unavailable</td>
<td>A dependent service is down (for example, the Cisco Finesse Notification Service or Cisco Finesse Database). Finesse is OUT_OF_SERVICE.</td>
</tr>
</tbody>
</table>

Dialog APIs for Nonvoice Tasks

Supported Functionality for Voice and Nonvoice Dialogs

The following are the major differences between supported functionality for voice and nonvoice dialogs:

- Users cannot initiate nonvoice dialogs; nonvoice dialogs are always incoming.
- Nonvoice dialogs can be blind transferred only. Direct transfer is not supported.
- Nonvoice dialogs support only one agent participant. Consult and conference are not supported.

Dialog Object and Parameters for Nonvoice Tasks

The same Dialog object is used for voice calls and nonvoice tasks. The Dialog object includes mediaId and mediaType parameters that indicate the Media Routing Domain with which the dialog is associated.

Some of the Dialog parameters used for voice calls, such as callType and mediaAddressType, are not applicable for nonvoice tasks; these parameters are not returned.
The dialog id format is different for voice calls and nonvoice tasks. The nonvoice dialog id contains underscores (for example, 151635_312_1). Voice dialog ids do not contain underscores (for example, 16804377).

The Dialog section of the Finesse Desktop APIs chapter describes the differences in the Dialog object for voice calls and nonvoice tasks. It also explains the parameters and parameter values used for nonvoice tasks.

**Dialog APIs for Nonvoice Tasks**

Most Dialog APIs are restricted to voice media.

You can use **Dialog - Take Action on Participant API** to handle nonvoice dialogs. This API supports the following allowable actions for nonvoice tasks.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPT</td>
<td>Allows an agent to accept an incoming task.</td>
</tr>
<tr>
<td>START</td>
<td>Allows an agent to start work on an accepted task.</td>
</tr>
<tr>
<td>PAUSE</td>
<td>Allows an agent to pause an active task.</td>
</tr>
<tr>
<td>RESUME</td>
<td>Allows an agent to resume a paused task.</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>Allows an agent to transfer an accepted, active, or paused task to another Script Selector/dialed number.</td>
</tr>
<tr>
<td>WRAP_UP</td>
<td>Allows an agent to perform wrap up work for a task.</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Allows an agent to end a task.</td>
</tr>
</tbody>
</table>

---

**Important**

For nonvoice tasks, dialog actions result only in Finesse reporting the state to CCE. The application is responsible for enforcing that state within the application. For example, if a user pauses an email dialog using the Dialog - Take Action on Participant API, the dialog state PAUSED is reported to CCE. However, if the application still displays the user interface to work on the email, the agent can continue to work on the email. The application must enforce the PAUSED state by preventing agent from working on the email in the user interface.

---

**Notifications**

Finesse sends a Dialogs/Media notification when information (or an action) changes for a nonvoice task to which the user belongs.

If a nonvoice dialog operation results in an asynchronous error, the error is returned in a Dialogs/Media notification. The notification includes the error type, error code, and error constant. The ErrorMedia parameter indicates the Media RoutingDomain to which the error applies.

---

**Important**

For an interruptible Media Routing Domain configured to accept interrupts, Finesse sends only a Media state change when an agent is interrupted in that MRD. It does not send Dialogs/Media notifications with the action list modified to reflect the fact that actions not permitted on the tasks in that media. The state change is the only indication to the Finesse applications that no actions are allowed on the interrupted dialogs.
Interactions with SocialMiner

Finesse connects to SocialMiner in order to resubmit tasks into the system for these reasons:

- The agent transfers a task.
- A task RONAs while waiting to be accepted by an agent. Finesse automatically resubmits the task to SocialMiner.
- An agent signs out with tasks. The agent was configured to transfer tasks on logout. Finesse automatically resubmits the task to SocialMiner.

The original dialog is closed with an appropriate disposition code, and the task is resubmitted as a new task request.

For automatic task resubmissions due to RONA and agent logout, the Finesse server on which the agent was last signed in initiates the request.

Related Topics

- Dialog
- Dialog—Take Action on Participant, on page 75
- Dialog API Parameters
- Disposition Code Parameter Values for Nonvoice Tasks, on page 130
- Dialogs/Media Notification
- Media and Dialogs/Media Asynchronous Error Notification, on page 307

User APIs for Nonvoice Tasks

Most User APIs are restricted to voice media. Several of them, described here, can be used with nonvoice media.

User- Get List of Dialogs APIs

You can use User - Get List of Dialogs (Nonvoice Only) to get a list of only nonvoice dialogs for a user.

To get a list of both voice and nonvoice dialogs for a user, use the User - Get List of Dialogs (Voice Only by Default) API.

User - Sign Out and User - Change State with Reason Code APIs

You can sign a user out of all Media Routing Domains when the user signs out of the desktop, using either the User - Sign Out API or the User - Change State with Reason Code API.

The desktop sign out fails only if the voice MRD sign out fails; it is not impacted by nonvoice MRD sign out failure.

Related Topics

- User—Get List of Dialogs (Nonvoice Only), on page 32
- User—Get List of Dialogs (Voice Only by Default), on page 31
- User—Sign Out of Finesse, on page 26
- User—Change Agent State With Reason Code, on page 43
- Media Notification, on page 306
- Media and Dialogs/Media Asynchronous Error Notification, on page 307
Single Sign-On

The Single Sign-On (SSO) APIs are used in the Finess desktop for token related operations and are ready to use in an out of the box Finess deployment. Third-party desktop applications have to use these APIs independently for SSO token related operations.

For more information about SSO Solution overview, see https://developer.cisco.com/docs/contact-center-express/#cisco-identity-service-client-sdk-overview.

For more information about the third-party integrations, see https://developer.cisco.com/docs/contact-center-express/#cisco-identity-service-client-sdk-guide/overview.

Single Sign-On APIs

Single Sign-On—Test API

This SSO Test API is used to test the SSO authentication and authorization setup with Finess.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http(s)://&lt;FQDN&gt;/desktop/sso/test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td>http(s)://finesse1.xyz.com/desktop/sso/test</td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Agents and supervisors can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>HTML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>Request Parameters:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
</tbody>
</table>
Response body returned after the SSO test contains an HTML displaying information about the user and token. This HTML also contains a JavaScript that sends the SSO test status, via window postMessage API, to the parent or opener window.

To get the status of SSO test on an older versions of Internet Explorer or any third-party non-browser clients that do not have this API, use the cookie set as part of HTTP response.

```
COOKIES set as part of response:
ssotest=true
Post message to parent window with below object:
{
    status: "true",
    errorMessage: ""
}
```

---

### Example Failure Response:

```
COOKIES set as part of response:
ssotest=false
Post message to parent window with below object:
{
    status: "false",
    errorMessage: "AUTH_ERROR"/"NO TOKEN"
}
```

---

**Single Sign-On—Fetch Access Token**

This API gets the access token from the Finess server.

**Note**

Invoking this API might involve browser redirect to Cisco Identity Server and Cisco Identity Provider.

---

**URI:** http(s)://<FQDN>/desktop/sso/token

**Example URI:** http(s)://finesse1.xyz.com/desktop/sso/token

**Security Constraints:** Agents and supervisors can use this API.

**HTTP Method:** GET

**Content Type:** —

**Input/Output Format:** JSON

**HTTP Request:** —

**Request Parameters:** (Optional) return_user=yes

**HTTP Response:**

- 200: Success
- 400: Bad Request
- 401: Unauthorized
- 500: Internal Server Error
Cisco Finesse Desktop APIs
Single Sign-On—Fetch Access Token

Example Response:

Response without parameter:
{"token":"eyJhbGciOiJkaXIiLCJjdHkiOiJKV1QiLCJlbm
MiOiJBMTI4Q0JDLUhTMjU2In0..lDXjaqAsM89uhdc
Qt364LA.qXBMK_y58Hkz19k-B8ealJ9LOalB0yNnm9
vOvKExf8slCpXAPPlJLnNXGD9_-YTGdjs7lPtEcdI—
hSuDmwxxOhdGZc7ekbAadJ6EItZhOGykCYk_CBF
mEHKU8-pHV3bdbsUGrCTponA8BMw04-S-N5iuI3v
u8fuihcNAeRY_9tjl5jvlhHEnD6zrYLDFH8KcO-V2f9
bcFdxHn3BrZk9tMasrsAJNhm8Uo_kg06PXq9omrTb
UEKm3f1_lMb3bwqZGXfOO6WLOngsADRTuHren_C
Tp5gR8r94LpsbXV7gRaEqsCu9kWo3pfxQsu88LNPR
W6RPcjozupw0A4-jrHBOf_X2XaDquanEbBkZIt9VIJh
jr6p8bTO5zlH9Z_x7vdMIfEt2pcjqcXKP3NiHlXOaB-tni
PX_zN8ckGqIKR7L4wBxYmXUj82cnjBNMkcUsbvP9W
Mb7ihJw0wazl1Tq6WnhtTGeOf0cnorjPm8DOZrcAAjJc
SDCpudfj5CgE-OwikeSdWURgYTg_k6Kcct71I3olVLT
c6nFRGcYvclvjCfTc1_ooBQ6ZKI_thq0Apnof235l6drDxG
sDMPiyop69hWCuMoRRK-KKAXr8xK3fiqKjSse-KMLMG
rMLZkUsr2Y_Q0YwiEIJk1FJ4n5Qgn-ismhKi-A_Vg3ZicG
J-YyIcYgcslJGDeqSB10Y0uThqOuMA9eGEHKSlZGLcZ
BfX5MGv23dEOOxN9_wLkqazF75m5H_23ycLyN0v9d8u
F7_fe7IWB97cI9nDAhaNBdHBR3XYU5GPSbRRS7GknD
oWZM_8eTgzc-gFTfYfAJveg_pPr1sSKvWnabqLXUuLDm
vcVbgA-5UI2Y4HEGKzW85fNOHE9WPpo3cQdxFdRQyH
fvFCBdTAOiFcIz_uP2nCDB_8oPT7qycm6b58BRJ5EzaTc
WapskB73w8no1YJadliQ20OYHrDKSs_LJYDeB2iBROS
UoVocYlW6GwTv0Ko7NsLv3OtGc_I.Fre8fhy_Y4u11tIfNo6
fIA","expires_in":300}

Response with parameter:
{"token":"eyJhbGciOiJkaXIiLCJjdHkiOiJKV1QiLCJlbm
MiOiJBMTI4Q0JDLUhTMjU2In0..lDXjaqAsM89uhdcQ
t364LA.qXBMK_y58Hkz19k-B8ealJ9LOalB0yNnm9v
OvKExf8slCpXAPPlJLnNXGD9_-YTGdjs7lPtEcdI—
hSuDmwxxOhdGZc7ekbAadJ6EItZhOGykCYk_CBFm
EHKU8-pHV3bdbsUGrCTponA8BMw04-S-N5iuI3vu8
fuihcNAeRY_9tjl5jvlhHEnD6zrYLDFH8KcO-V2f9bc
FdxHn3BrZk9tMasrsAJNhm8Uo_kg06PXq9omrTbUE
Km3f1_lMb3bwqZGXfOO6WLOngsADRTuHren_CTp
5gR8r94LpsbXV7gRaEqsCu9kWo3pfxQsu88LNPR
W6RPcjozupw0A4-jrHBOf_X2XaDquanEbBkZIt9V
IJhjr6p8bTO5zlH9Z_x7vdMIfEt2pcjqcXKP3NiHlXOaBtniPX_zN8ckGqIKR7L4wBxYmXUj82cnjBNMkcUsbvP
9WMb7ihJw0wazl1Tq6WnhtTGeOf0cnorjPm8DOZrcA
AjJcSDCpudfj5CgE-OwikeSdWURgYTg_k6Kcct71I3ol
VLTc6nFRGcYvclvjCfTc1_ooBQ6ZKI_thq0Apnof235l6
drDxGsDMPiyop69hWCuMoRRK-KKAXr8xK3fiqKjSseKMLMGrMLZkUsr2Y_Q0YwiEIJk1FJ4n5Qgn-ismhKi-A
_Vg3ZicGJ-YyIcYgcslJGDeqSB10Y0uThqOuMA9e
GEHKSlZGLcZBfX5MGv23dEOOxN9_wLkqazF75m5H
_23ycLyN0v9d8uF7_fe7IWB97cI9nDAhaNBdHBR3XYU
5GPSbRRS7GknDoWZM_8eTgzc-gFTfYfAJveg_pPr1s
SKvWnabqLXUuLDmvcVbgA-5UI2Y4HEGKzW85fNO
HE9WPpo3cQdxFdRQyHfvFCBdTAOiFcIz_uP2nCDB_8
oPT7qycm6b58BRJ5EzaTcWapskB73w8no1YJadliQ20O
YHrDKSs_LJYDeB2iBROSUoVocYlW6GwTv0Ko7NsLv3
OtGc_I.Fre8fhy_Y4u11tIfNo6fIA",
"expires_in":49,
"user_id":"1001001",
"realm":"finesse.com",
"user_principal":"1001001@finesse.com"}

Cisco Finesse Web Services Developer Guide Release 11.6(1)
168


Example Failure Response:
{"error":"invalid_redirectUri","error_description":"Invalid Redirect URI."}

Single Sign-On—Refresh Existing Access Token

This API allows a user to refresh an existing access token that is about to expire.

- Third-party applications have to refresh the access token after 75% of the token expiry time is elapsed.
- Invoking this API might involve browser redirect to Cisco Identity Service and Identity Provider.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http(s)://&lt;FQDN&gt;/desktop/sso/token</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td>http(s)://finesse1.xyz.com/desktop/sso/token</td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Agents and supervisors can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>POST</td>
</tr>
<tr>
<td>Content Type:</td>
<td>application/x-www-form-urlencoded</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>JSON</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>token=&lt;token value&gt;</td>
</tr>
<tr>
<td>Request Parameters:</td>
<td>(Optional) return_user=yes</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>400: Bad Request</td>
</tr>
<tr>
<td></td>
<td>401: Unauthorized</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
</tbody>
</table>
If the token was initially fetched with the `return_refresh_token=true` query parameter, then the refresh token in request payload is mandatory.
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

If a user repeatedly passes an invalid password in the basic authorization header to a configuration API, on the fifth invalid attempt, Finesse blocks the user's access to all configuration APIs for 5 minutes. This lock period differs from the 30-minute lock period implemented for the Finesse administrator console.

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, on page 172
- ClusterConfig, on page 176
- EnterpriseDatabaseConfig, on page 178
- LayoutConfig, on page 183
- ReasonCode, on page 188
- WrapUpReason, on page 196
- MediaPropertiesLayout, on page 202
- PhoneBook, on page 216
- Contact, on page 225
SystemConfig

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

Note

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

The SystemConfig object is structured as follows:

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

Note

Any changes made to the settings through the SystemConfig API will require a Cisco Finess Tomcat restart.

SystemConfig APIs

SystemConfig—Get

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>http://&lt;FQDN&gt;/finesse/api/SystemConfig</td>
</tr>
<tr>
<td>Example URI</td>
<td><a href="http://finesse1.xyz.com/finesse/api/SystemConfig">http://finesse1.xyz.com/finesse/api/SystemConfig</a></td>
</tr>
<tr>
<td>Security Constraints</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format</td>
<td>XML</td>
</tr>
</tbody>
</table>
### 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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/SystemConfig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/SystemConfig">http://finesse1.xyz.com/finesse/api/SystemConfig</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>
**SystemConfig API Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the SystemConfig object.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>cti</td>
<td>Collection</td>
<td>Information about the CTI server settings.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>host</td>
<td>String</td>
<td>The hostname or IP address of the primary (A Side) CTI server.</td>
<td>—</td>
<td>No special characters allowed except “.” and “-“.</td>
</tr>
</tbody>
</table>
### SystemConfig API Errors

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

**Note**

Any changes made to the settings through the ClusterConfig API will require a Cisco Finesse Tomcat restart.

**ClusterConfig APIs**

**ClusterConfig—Get**

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

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/ClusterConfig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/ClusterConfig">http://finesse1.xyz.com/finesse/api/ClusterConfig</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>
### HTTP Response:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
</tr>
<tr>
<td>400</td>
<td>Invalid Input</td>
</tr>
<tr>
<td>400</td>
<td>Parameter Missing</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>
ClusterConfig API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the ClusterConfig object.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>secondaryNode</td>
<td>Collection</td>
<td>Information about secondary Finesse node.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>host</td>
<td>String</td>
<td>The hostname or IP address of the secondary Finesse node.</td>
<td>—</td>
<td>No special characters allowed except “.” and “-”.</td>
</tr>
</tbody>
</table>

ClusterConfig API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Invalid Input</td>
<td>One of the parameters provided as part of the user input is invalid or not recognized.</td>
</tr>
<tr>
<td>400</td>
<td>Parameter Missing</td>
<td>A required parameter was not provided in the request.</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>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).</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
</tbody>
</table>

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.
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:

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

Any changes made to the settings through the EnterpriseDatabaseConfig API will require a Cisco Finesse Tomcat restart.

**EnterpriseDatabaseConfig APIs**

**EnterpriseDatabaseConfig—Get**

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

<table>
<thead>
<tr>
<th><strong>URI:</strong></th>
<th>http://&lt;FQDN&gt;/finesse/api/EnterpriseDatabaseConfig</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example URI:</strong></td>
<td><a href="http://finesse1.xyz.com/finesse/api/EnterpriseDatabaseConfig">http://finesse1.xyz.com/finesse/api/EnterpriseDatabaseConfig</a></td>
</tr>
<tr>
<td><strong>Security Constraints:</strong></td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td><strong>HTTP Method:</strong></td>
<td>GET</td>
</tr>
<tr>
<td><strong>Content Type:</strong></td>
<td>—</td>
</tr>
<tr>
<td><strong>Input/Output Format:</strong></td>
<td>XML</td>
</tr>
<tr>
<td><strong>HTTP Request:</strong></td>
<td>—</td>
</tr>
</tbody>
</table>
| **HTTP Response:** | 200: Success  
401: Unauthorized  
403: Forbidden  
500: Internal Server Error |
EnterpriseDatabaseConfig—Set

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

**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>
```

**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
EnterpriseDatabaseConfig API Parameters

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

HTTP Request:

```xml
<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 (optional): Domain of the AWDB
- username (required): Username to sign in to the AWDB. If there is a domain specified, this must be a domain user. Otherwise it must be an SQL user.
- 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:

```xml
<ApiErrors>
  <ApiError>
    <ErrorType>Invalid Input</ErrorType>
    <ErrorMessage>port</ErrorMessage>
    <ErrorData>65536</ErrorData>
  </ApiError>
</ApiErrors>
```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>backupHost</td>
<td>String</td>
<td>The hostname or IP address of the backup AWDB server.</td>
<td>—</td>
<td>No special characters allowed except “.” and “-”.</td>
</tr>
<tr>
<td>port</td>
<td>Integer</td>
<td>The port of the AWDB server.</td>
<td>1–65535</td>
<td>—</td>
</tr>
<tr>
<td>databaseName</td>
<td>String</td>
<td>The name of the AWDB (for example, ucceinstance_awdb).</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>domain</td>
<td>String</td>
<td>The domain of the AWDB.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>username</td>
<td>String</td>
<td>The username required to sign in to the AWDB. If there is a domain specified, this must be a domain user. Otherwise it must be an SQL user.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>password</td>
<td>String</td>
<td>The password required to sign in to the AWDB.</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

### EnterpriseDatabaseConfig API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Invalid Input</td>
<td>One of the parameters provided as part of the user input is invalid or not recognized.</td>
</tr>
<tr>
<td>400</td>
<td>Parameter Missing</td>
<td>A required parameter was not provided in the request. For example, if the backupPort is provided but the backupHost is missing.</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>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).</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
</tbody>
</table>
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:

```plaintext
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 a relative path, as shown in the following example:

```plaintext
/ab/def<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:

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

The LayoutConfig object is structured as follows:

```xml
<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>
                <!-- The following Gadget is only for temporary use and has been superseded by the Live Data gadgets.
                Remove the Queue gadget once you have configured Live Data and the Live Data gadgets
                The following Gadget (Agent Queue Statistics) is *not* supported in Packaged CCE deployment.
                -->
                <gadget>/desktop/gadgets/QueueStatistics.jsp</gadget>
              </column>
            </columns>
          </tab>
        </tabs>
      </layout>
    </finesseLayout>
  </layoutxml>
</LayoutConfig>
```
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.

```html
<!-- HTTPS Version of LiveData Gadgets -->
<!-- "Agent" Report -->
<!-- <gadget>https://my-cuic-server:8444/cuic/gadget/LiveData/LiveDataGadget.jsp?gadgetHeight=310&viewId=99E6C8E210000141000000D80A0006C4&filterId=agent.id=CL%20teamName</gadget> -->

<!-- "Agent Skill Group" Report -->
<!-- <gadget>https://my-cuic-server:8444/cuic/gadget/LiveData/LiveDataGadget.jsp?gadgetHeight=310&viewId=9AB7848B10000141000001C50A0006C4&filterId=agent.id=CL%20teamName</gadget> -->

<!-- "Agent All Fields" Report -->
<!-- <gadget>https://my-cuic-server:8444/cuic/gadget/LiveData/LiveDataGadget.jsp?gadgetHeight=310&viewId=9A08E2351000014100001230A0006C4&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=A30EC2581000014100003A60A0006C4&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=99E6C8E210000141000000D80A0006C4&filterId=agent.id=CL%20teamName</gadget> -->

<!-- "Agent Skill Group" Report -->
<!-- <gadget>http://my-cuic-server:8081/cuic/gadget/LiveData/LiveDataGadget.jsp?gadgetHeight=310&viewId=9AB7848B10000141000001C50A0006C4&filterId=agent.id=CL%20teamName</gadget> -->

<!-- "Agent All Fields" Report -->
<!-- <gadget>http://my-cuic-server:8081/cuic/gadget/LiveData/LiveDataGadget.jsp?gadgetHeight=310&viewId=9A08E2351000014100001230A0006C4&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=A30EC2581000014100003A60A0006C4&filterId=agent.id=CL%20teamName</gadget> -->
```
viewId=A30EC25810000141000003A60A0006C4
filterId=agent.id=CL%20teamName</gadget> -->
</gadgets>
</column>
</column>
</tab>
</id>manageCall/id>
</label>finesse.container.tabs.agent.manageCallLabel</label>
</tab>
</gadgets>
</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> -->
</layout>
<layout>
<role>Supervisor</role>
<page>
<gadget>/desktop/gadgets/CallControl.jsp</gadget>
</page>
</tabs>
<tab>
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: | — |
| 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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/LayoutConfig/default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/LayoutConfig/default">http://finesse1.xyz.com/finesse/api/LayoutConfig/default</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>
| 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the LayoutConfig object.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>layoutxml</td>
<td>String</td>
<td>The XML data that determines the layout of the Finesse desktop.</td>
<td>—</td>
<td>Must be valid XML and must conform to the Finesse schema.</td>
</tr>
</tbody>
</table>

### LayoutConfig API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Invalid Input</td>
<td>The submitted XML is invalid or does not conform to the Finesse schema.</td>
</tr>
<tr>
<td>400</td>
<td>Parameter Missing</td>
<td>The layout XML file was not provided.</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>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).</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
</tbody>
</table>

### 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. For more information about predefined reason codes for Unified CCE, see the Cisco Unified Contact Center Enterprise Reporting User Guide (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*.

### Note

System reason codes are defined by Unified CCE and Unified CCX. These reason codes are used by Finesse but not listed in the ReasonCode APIs.

The ReasonCode object is structured as follows:

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

### ReasonCode APIs

#### ReasonCode—Get

The following GET APIs allow an administrator or an agent to get a copy of the ReasonCode object.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/ReasonCode/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Constraints:</td>
<td>Administrators and agents can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>401: Authorization Failure</td>
</tr>
<tr>
<td></td>
<td>401: Invalid Authorization User Specified</td>
</tr>
<tr>
<td></td>
<td>403: Forbidden</td>
</tr>
<tr>
<td></td>
<td>404: Not Found</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
<tr>
<td>ReasonCode—Get</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td></td>
</tr>
</tbody>
</table>

**Example Response:**

```
<ReasonCode>
    <uri>/finesse/api/ReasonCode/45</uri>
    <category>NOT_READY</category>
    <code>10</code>
    <label>Team Meeting</label>
    <forAll>true</forAll>
    <systemCode>true</systemCode>
</ReasonCode>
```

**Example Failure Response:**

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

**URI:**


**Example URI:**


**Security Constraints:**

Administrators and agents can use this API.

**HTTP Method:**

GET

**Content Type:**

—

**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>
    <systemCode>true</systemCode>
</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, sign out reason codes or both. If the category parameter is missing, the API returns an error.

| URI: | http://<FQDN>/finesse/api/ReasonCodes?category=NOT_READY|LOGOUT|ALL |
|-----|----------------------------------------------------------|
| Example URI: | http://finesse1.xyz.com/finesse/api/ReasonCodes?category=ALL |
| Security Constraints: | Only administrators can use this API. |
| HTTP Method: | GET |
| Content Type: | — |
| 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="ALL"> |
| | <ReasonCode> |
| | <uri>/finesse/api/ReasonCode/1</uri> |
| | <category>NOT_READY</category> |
| | ... Rest of ReasonCode Object .. |
| | </ReasonCode> |
| | <ReasonCode> |
| | <uri>/finesse/api/ReasonCode/2</uri> |
| | <category>LOGOUT</category> |
| | ... Rest of 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/ |
| Security Constraints: | Only administrators can use this API. |
| HTTP Method: | POST |
| Content Type: | Application/XML |
| Input/Output Format: | XML |

**HTTP Request:**

```xml
<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: Finesses 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: Finesses 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, Finesses 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>
```

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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/ReasonCode/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
</tbody>
</table>
ReasonCode—Delete

This API allows an administrator to delete an existing reason code.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/ReasonCode/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>DELETE</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>

**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>
```
HTTP Response:

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
<td>Success</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>Authorization Failure</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Authorization User Specified</td>
<td>Invalid Authorization User Specified</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>Forbidden</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>Not Found</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

Example Failure Response:

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

### ReasonCode API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the ReasonCode object.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>category</td>
<td>String</td>
<td>The category of the reason code.</td>
<td>NOT READY, LOGOUT</td>
<td></td>
</tr>
<tr>
<td>code</td>
<td>Integer</td>
<td>The code for the reason code.</td>
<td>Unified CCE: 1–65535, Unified CCX: 1–999</td>
<td>The combination of code and category must be unique.</td>
</tr>
</tbody>
</table>
| 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     |                                                                     |
| systemCode| Boolean | The reserved status of the reason code.           | true, false     |                                                                     |

### ReasonCode API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>One of the required parameters was not provided or is invalid</td>
</tr>
</tbody>
</table>
### 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:

```xml
<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 |
WrapUpReason—Get List

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

| Content Type: | — |
| 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> |

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: —

Input/Output Format: XML

HTTP Request: —
### 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 1500 non-global wrap-up reasons, for each category, with the restriction that a maximum of 100 non-global wrap-up reasons can be assigned to a single team.

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

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URI:</strong></td>
<td>http://&lt;FQDN&gt;/finesse/api/WrapUpReason/</td>
</tr>
<tr>
<td><strong>Example URI:</strong></td>
<td><a href="http://finesse1.xyz.com/finesse/api/WrapUpReason/">http://finesse1.xyz.com/finesse/api/WrapUpReason/</a></td>
</tr>
<tr>
<td><strong>Security Constraints:</strong></td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td><strong>HTTP Method:</strong></td>
<td>POST</td>
</tr>
<tr>
<td><strong>Content Type:</strong></td>
<td>Application/XML</td>
</tr>
<tr>
<td><strong>Input/Output Format:</strong></td>
<td>XML</td>
</tr>
</tbody>
</table>
### 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:

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

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.

| 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>

### URI:

- **http://<FQDN>/finesse/api/WrapUpReason/<id>**
- **http://finesse1.xyz.com/finesse/api/WrapUpReason/43**
### WrapUpReason—Delete

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

<table>
<thead>
<tr>
<th>Security Constraints:</th>
<th>Only administrators can use this API.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>&lt;WrapUpReason&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;label&gt;Sales Call&lt;/label&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/WrapUpReason&gt;</td>
</tr>
<tr>
<td>Request Parameters:</td>
<td>id (required): The database ID for the wrap-up reason</td>
</tr>
<tr>
<td></td>
<td>label (required): The UI label for the reason code</td>
</tr>
<tr>
<td></td>
<td>forAll (required): Whether the reason code is global (true) or non-global (false)</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>400: Finesse API Error</td>
</tr>
<tr>
<td></td>
<td>401: Authorization Failure</td>
</tr>
<tr>
<td></td>
<td>401: Invalid Authorization User Specified</td>
</tr>
<tr>
<td></td>
<td>403: Forbidden</td>
</tr>
<tr>
<td></td>
<td>404: Not Found</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
<tr>
<td>Example Failure Response:</td>
<td>&lt;ApiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ApiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorType&gt;Authorization Failure&lt;/ErrorType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorMessage&gt;UNAUTHORIZED&lt;/ErrorMessage&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorData&gt;jsmith&lt;/ErrorData&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ApiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ApiErrors&gt;</td>
</tr>
</tbody>
</table>

**Uri:**

http://<FQDN>/finesse/api/WrapUpReason/</id>

**Example URI:**

http://finesse1.xyz.com/finesse/api/WrapUpReason/23
HTTP Response:
200: Success
401: Authorization Failure
401: Invalid Authorization User Specified
403: Forbidden
404: Not Found
500: Internal Server Error

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

WrapUpReason API Parameters

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

WrapUpReason API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>The request body is invalid</td>
</tr>
<tr>
<td>400</td>
<td>Finesse API Error</td>
<td>API error such as duplicated wrap-up reason or the wrap-up reason does not exist.</td>
</tr>
<tr>
<td>400</td>
<td>Maximum Exceeded</td>
<td>The maximum number of items has been exceeded.</td>
</tr>
</tbody>
</table>
## 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 create and customize multiple layouts for 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:

```xml
<MediaPropertiesLayout>
  <uri>/finesse/api/MediaPropertiesLayout/{id}</uri>
  <name>Layout name</name>
  <description>Layout description</description>
  <type>DEFAULT|CUSTOM</type>
  <header>
    <entry>
```

### Table: Error Types and Descriptions

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

**MediaPropertiesLayout APIs**

**MediaPropertiesLayout—Get**

This API allows an administrator to get a copy of the media properties layout associated with the specified ID.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/MediaPropertiesLayout/{id}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/15">http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/15</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>
| HTTP Response: | 200: Success  
401: Authorization Failure  
403: Forbidden  
500: Internal Server Error |
**MediaPropertiesLayout—Get Default Layout**

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

**Note**
Cisco Finesse supports this API for backward compatibility, but to get the default layout, developers must specify the default MediaPropertiesLayout ID in the MediaPropertiesLayout—Get API.

| 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>
  <uri>/finesse/api/MediaPropertiesLayout/{id}</uri>
  <name>Default</name>
  <description>This is the default layout</description>
  <type>DEFAULT</type>
  <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>
```

### MediaPropertiesLayout—Get List

This API allows an administrator to list all the media properties layouts configured in the system.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/MediaPropertiesLayouts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/MediaPropertiesLayouts">http://finesse1.xyz.com/finesse/api/MediaPropertiesLayouts</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>
HTTP Response:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
</tr>
<tr>
<td>400</td>
<td>Finesse API error</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Authorization User Specified</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

Example Response:

```xml
<MediaPropertiesLayouts>
  <MediaPropertiesLayout>
    ... Full MediaPropertiesLayout Object ...
  </MediaPropertiesLayout>
  <MediaPropertiesLayout>
    ... Full MediaPropertiesLayout Object ...
  </MediaPropertiesLayout>
  <MediaPropertiesLayout>
    ... Full MediaPropertiesLayout Object ...
  </MediaPropertiesLayout>
</MediaPropertiesLayouts>
```

Example Failure Response:

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

Note

If the Finessedatabase is down or if there is a problem retrieving themedia properties layout from the database, then a GET on http://<server>/finesse/api/MediaPropertiesLayouts (or on http://<server>/finesse/api/MediaPropertiesLayout/default) returns the system defined default mediaproperties layout with an ID of 0.

MediaPropertiesLayout—Create

This API allows an administrator to create a custom mediaproperties layout. Finessesupports up to 200 mediaproperties layouts (1 default and 199 custom mediaproperties layouts).

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

Example URI: http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/

Security Constraints: Only administrators can use this API.

HTTP Method: POST

Content Type: Application/XML
<table>
<thead>
<tr>
<th>Input/Output Format:</th>
<th>XML</th>
</tr>
</thead>
</table>

| HTTP Request: |  
|---------------|--------------------------------------------------|
|               | <MediaPropertiesLayout>                         |
|               |   <name>Layout name</name>                      |
|               |   <description>Layout description</description>|
|               |   <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: | name (required): Name of the media properties layout  |
|                    | description (optional): Description of the media properties layout  |
|                    | header (optional): Mapping for a single mediaProperty to be displayed with a label on the call details in the agent or supervisor desktop  |
|                    | column (optional): Grouping of mediaProperties for agent or supervisor desktops  |
|                    | entry (optional): Contains a displayName and mediaProperty combination  |
|                    | displayName (required): Name of the field to be displayed to the agent or supervisor  |
|                    | mediaProperty (required): Value of the entry to be displayed to the agent or supervisor matched with the displayName in the same entry  |
MediaPropertiesLayout—Update

This API allows an administrator to update the media properties layout associated with the specified ID.

| URI: | http://<FQDN>/finesse/api/MediaPropertiesLayout/{id} |
| Example URI: | http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/15 |
| Security Constraints: | Only administrators can use this API. |
| HTTP Method: | PUT |
| Content Type: | Application/XML |
| Input/Output Format: | XML |

**HTTP Response:**

- **200: Success**
- **Note:** Finesse successfully created the new media properties layout. The response contains an empty response body and a location header that denotes the absolute URL of the newly created MediaPropertiesLayout object.
- **400: Parameter Missing**
- **400: Invalid Input**
- **401: Authorization Failure**
- **403: Forbidden**
- **500: Internal Server Error**

**Example Failure Response:**

```xml
<ApiErrors>
  <ApiError>
    <ErrorType>Authorization Failure</ErrorType>
    <ErrorMessage>UNAUTHORIZED</ErrorMessage>
    <ErrorData>jsmith</ErrorData>
  </ApiError>
</ApiErrors>
```
HTTP Request:

```xml
<MediaPropertiesLayout>
  <name>Layout name</name>
  <description>Layout description</description>
  <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:
- **name** (required): Name of the media properties layout
- **description** (optional): Description of the media properties layout
- **header** (optional): Mapping for a single mediaProperty to be displayed with a label on the call details in the agent or supervisor desktop
- **column** (optional): Grouping of mediaProperties for agent or supervisor desktops
- **entry** (optional): Contains a displayName and mediaProperty combination
- **displayName** (required): Name of the field to be displayed to the agent or supervisor
- **mediaProperty** (required): Value of the entry to be displayed to the agent or supervisor matched with the displayName in the same entry

HTTP Response:
- 200: Success
- 400: Parameter Missing
- 400: Invalid Input
- 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>
```

MediaPropertiesLayout—Update Default Layout

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

Note

Cisco Finesse supports this API for backward compatibility, but to update the default layout, developers must specify the default MediaPropertiesLayout ID in the MediaPropertiesLayout—Update API.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/MediaPropertiesLayout/default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/default">http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/default</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>
### HTTP Request:

```xml
<MediaPropertiesLayout>
  <name>Default</name>
  <description>default layout</description>
  <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:

- **name** (required): Name of the media properties layout
- **description** (optional): Description of the media properties layout
- **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
### MediaPropertiesLayout—Delete

This API allows an administrator to delete the custom media properties layout with the specified ID.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/MediaPropertiesLayout/{id}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/15">http://finesse1.xyz.com/finesse/api/MediaPropertiesLayout/15</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td></td>
<td>Administrators can only delete a media properties layout of type CUSTOM.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>DELETE</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>400: Bad Request</td>
</tr>
<tr>
<td></td>
<td>401: Unauthorized</td>
</tr>
<tr>
<td></td>
<td>403: Forbidden</td>
</tr>
<tr>
<td></td>
<td>404: Not Found</td>
</tr>
<tr>
<td></td>
<td>500: Runtime exception</td>
</tr>
</tbody>
</table>

**Example Failure Response:**

```xml
<ApiErrors>
  <ApiError>
    <ErrorType>Authorization Failure</ErrorType>
    <ErrorMessage>UNAUTHORIZED</ErrorMessage>
    <ErrorData>jsmith</ErrorData>
  </ApiError>
</ApiErrors>
```
If you attempt to delete the default media properties layout, the system responds with one of the following errors, depending on the API you use for the operation:

<table>
<thead>
<tr>
<th>API Used to Delete the Default Layout</th>
<th>HTTP Response</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>http://&lt;FQDN&gt;/finesse/api/</td>
<td>403 Forbidden</td>
<td>DELETE of the default media properties layout is forbidden with this API.</td>
</tr>
<tr>
<td>MediaPropertiesLayout/{id}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>http://&lt;FQDN&gt;/finesse/api/</td>
<td>405 Method Not Allowed</td>
<td>DELETE is not a supported operation with this API.</td>
</tr>
<tr>
<td>MediaPropertiesLayout/default</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### MediaPropertiesLayout API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The id maps to the primary key of the media properties layout entry.</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the media properties layout.</td>
<td>—</td>
<td>Max length of 40 characters</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>The description of the media properties layout.</td>
<td>—</td>
<td>Max length of 128 characters</td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>The type of media properties layout.</td>
<td>DEFAULT, CUSTOM</td>
<td></td>
</tr>
<tr>
<td>header</td>
<td>Object</td>
<td>Contains a single entry (combination of displayName and mediaProperty) that appears in the call header on the desktop for each call.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Possible Values</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>column</td>
<td>Object</td>
<td>Grouping of media properties for agent and supervisor desktops. Contains a list of entry objects</td>
<td>—</td>
<td>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.</td>
</tr>
<tr>
<td>--- &gt; entry</td>
<td>Object</td>
<td>A displayName and mediaProperty combination.</td>
<td>—</td>
<td>Each entry must contain one displayName and one mediaProperty. The displayName can be empty.</td>
</tr>
<tr>
<td>--- &gt; displayName</td>
<td>String</td>
<td>Part of an entry. A label that describes the mediaProperty for that entry (for example, Customer Name). The label appears on the Finesse desktop.</td>
<td>—</td>
<td>Maximum of 50 characters.</td>
</tr>
</tbody>
</table>
### MediaPropertiesLayout API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>Request parameter is invalid.</td>
</tr>
<tr>
<td>400</td>
<td>Finesse API error</td>
<td>API error, such as: object is stale, violation of database constraint, and so on.</td>
</tr>
<tr>
<td>400</td>
<td>Invalid Input</td>
<td>At least one of the parameters provided is not valid.</td>
</tr>
<tr>
<td>400</td>
<td>Parameter Missing</td>
<td>At least one of the required parameters was not provided.</td>
</tr>
<tr>
<td>400</td>
<td>Maximum Exceeded</td>
<td>The maximum number of items has been exceeded.</td>
</tr>
<tr>
<td>400</td>
<td>Invalid Input</td>
<td>The user has selected more than five call variables when configuring call pop-over for a layout.</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>Unauthorized (for example, the user is not yet authenticated in the Web Session).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The user is not authorized to use the API (the user is not an administrator).</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Authorization User Specified</td>
<td>The authenticated user tried to use the identity that is not their own.</td>
</tr>
</tbody>
</table>
### Status | Error Type | Description
--- | --- | ---
403 | Forbidden | The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server. The default media properties layout may not be deleted.
404 | Not Found | Could not find the call variables layout with the specified ID.
405 | Method Not Allowed | Unsupported operation is performed against an API. For example, if a DELETE or POST is attempted on: http://<FQDN>/finesse/api/MediaPropertiesLayout/default (which only supports GET and PUT).
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 300 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 |
### 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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/PhoneBooks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/PhoneBooks">http://finesse1.xyz.com/finesse/api/PhoneBooks</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>
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:**

```xml
<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:  
```xml
<?xml version="1.0"?>
<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.

<table>
<thead>
<tr>
<th><strong>URI:</strong></th>
<th>http://&lt;FQDN&gt;/finesse/api/PhoneBook/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example URI:</strong></td>
<td><a href="http://finesse1.xyz.com/finesse/api/PhoneBook/45">http://finesse1.xyz.com/finesse/api/PhoneBook/45</a></td>
</tr>
<tr>
<td><strong>Security Constraints:</strong></td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td><strong>HTTP Method:</strong></td>
<td>PUT</td>
</tr>
<tr>
<td><strong>Content Type:</strong></td>
<td>Application/XML</td>
</tr>
<tr>
<td><strong>Input/Output Format:</strong></td>
<td>XML</td>
</tr>
</tbody>
</table>
| **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`). To import:

1. Create a Web Form HTML file and save it on your desktop. Example: phonebook.html

   ```html
   <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>
   ```

2. Create a CSV file with the phonebook content you want to upload. Example: pb.csv. (Also saved to the Desktop).

   "First Name","Last Name","Phone Number","Notes"
   "Agent","10001","20001","Sales"
   "Agent","10002","20002","Service"
   "Agent","10003","20011","Supervisor"
   "VVB","090011","HelloWorld"
   "Survivability","090011","To HelloWorld"

3. Run the phonebook.html file. A browser window will open.

4. Click Browse and select the pb.csv file.

5. Click Import.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/PhoneBook/&lt;1&gt;/Contacts/csvFileContent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contacts/csvFileContent">http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contacts/csvFileContent</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>POST</td>
</tr>
<tr>
<td>Content Type:</td>
<td>text/CSV</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>text/plain, text/CSV</td>
</tr>
</tbody>
</table>
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:

```xml
<Contacts>
  <Contact>
    ...Full Contact Object...
  </Contact>
  <Contact>
    ...Full Contact Object...
  </Contact>
  <Contact>
    ...Full Contact Object
  </Contact>
</Contacts>
```

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:

```xml
<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
PhoneBook API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the PhoneBook object.</td>
<td>—</td>
<td>The id in the URI maps to the primary key of the phone book entry.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the phone book.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>The type of phone book.</td>
<td>GLOBAL, TEAM</td>
<td></td>
</tr>
</tbody>
</table>
## PhoneBook API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Finesse API Error</td>
<td>API error such as the object is stale or does not exist.</td>
</tr>
<tr>
<td>400</td>
<td>Invalid Input</td>
<td>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 over written.</td>
</tr>
<tr>
<td>400</td>
<td>In Use</td>
<td>The phone book is assigned to a team. You cannot change a team phone book to a global phone book if it is in use. You cannot delete a phone book if it is in use.</td>
</tr>
<tr>
<td>400</td>
<td>Maximum Exceeded</td>
<td>The maximum number of phone books or contacts has been exceeded.</td>
</tr>
<tr>
<td>400</td>
<td>Parameter Missing</td>
<td>A required parameter was not present in the request.</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>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).</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Authorization User Specified</td>
<td>The authenticated user tried to use the identity of another user.</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>The specified resource cannot be found.</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
</tbody>
</table>

## 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 50,000 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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/PhoneBook/{phoneBookId}/Contact/{id}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contact/785">http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contact/785</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>
| 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> |
**Contact—Get List**

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

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/PhoneBook/&lt;phoneBookId&gt;/Contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contacts">http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contacts</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>
| 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>  
...Full Contact Object...  
</Contacts> |
| Example Failure Response: | <ApiErrors>  
  <ApiError>  
    <ErrorType>Authorization Failure</ErrorType>  
    <ErrorMessage>UNAUTHORIZED</ErrorMessage>  
    <ErrorData>jsmith</ErrorData>  
  </ApiError>  
</ApiErrors> |
This API allows an administrator to create a new phone book contact.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/PhoneBook/&lt;phoneBookId&gt;/Contact/</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contact/">http://finesse1.xyz.com/finesse/api/PhoneBook/34/Contact/</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>POST</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>&lt;Contact&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;firstName&gt;Jerry&lt;/firstName&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;lastName&gt;Green&lt;/lastName&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;phoneNumber&gt;5554444&lt;/phoneNumber&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;description&gt;Product Expert&lt;/description&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Contact&gt;</td>
</tr>
<tr>
<td>Request Parameters:</td>
<td>phoneBookId (required): Maps to the primary key of the phone book to which the contact belongs</td>
</tr>
<tr>
<td></td>
<td>firstName (optional): The first name of the contact</td>
</tr>
<tr>
<td></td>
<td>lastName (optional): The last name of the contact</td>
</tr>
<tr>
<td></td>
<td>phoneNumber (required): The phone number of the contact</td>
</tr>
<tr>
<td></td>
<td>description (optional): A description for the contact</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td>Note</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>400: Bad Request</td>
</tr>
<tr>
<td></td>
<td>400: Finesse API Error</td>
</tr>
<tr>
<td></td>
<td>401: Authorization Failure</td>
</tr>
<tr>
<td></td>
<td>403: Forbidden</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
<tr>
<td>Example Failure Response:</td>
<td>&lt;ApiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ApiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorType&gt;Authorization Failure&lt;/ErrorType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorMessage&gt;UNAUTHORIZED&lt;/ErrorMessage&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorData&gt;jsmith&lt;/ErrorData&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ApiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ApiErrors&gt;</td>
</tr>
</tbody>
</table>
## 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the Contact object.</td>
<td>—</td>
<td>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.</td>
</tr>
<tr>
<td>firstName</td>
<td>String</td>
<td>The first name of the contact.</td>
<td>—</td>
<td>Maximum of 128 characters.</td>
</tr>
<tr>
<td>lastName</td>
<td>String</td>
<td>The last name of the contact.</td>
<td>—</td>
<td>Maximum of 128 characters.</td>
</tr>
<tr>
<td>phoneNumber</td>
<td>String</td>
<td>The phone number for the contact.</td>
<td>—</td>
<td>Maximum of 32 characters.</td>
</tr>
</tbody>
</table>
### Contact API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>The request body is invalid.</td>
</tr>
<tr>
<td>400</td>
<td>Finesse API Error</td>
<td>API error such as the object is stale or does not exist.</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>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).</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>The specified resource cannot be found.</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
</tbody>
</table>

### 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>
  <media></media>
  <TriggerSet>
    <type></type>
    <name></name>
    <allowOverlappingCallWorkflow></allowOverlappingCallWorkflow>
    <triggers>
      <Trigger>
        <Variable>
          <name></name>
          <node></node>
          <type></type>
        </Variable>
        <comparator></comparator>
```

---

**Parameter** | **Type** | **Description** | **Possible Values** | **Notes**
---|---|---|---|---
description | String | A description of the contact. | — | Maximum of 128 characters.
The following SYSTEM TriggerSets are defined by the Finesses 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**

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

<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[.='${extension}']/../state</node>
    <type>CUSTOM</type>
  </Variable>
</Trigger>

<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, 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>
  </triggers>
</TriggerSet>
</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>
</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, 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
            [.='${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>
      </Variable>
    </Trigger>
  </triggers>
</TriggerSet>


```
<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[.='${extension}']/../state</node>
  <type>CUSTOM</type>
</Variable>
<comparator>IS_IN_LIST</comparator>
<value>ALERTING,INITIATED,FAILED,ACTIVE,HELD</value>
</Trigger>
<Trigger>
<Variable>
  <name>fromAddress</name>
  <node>//Dialog/fromAddress</node>
  <type>CUSTOM</type>
</Variable>
<comparator>IS_EQUAL</comparator>
<value>${extension}</value>
</Trigger>
</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_IN_LIST</comparator>
<value>OUTBOUND_PREVIEW,OUTBOUND_CALLBACK_PREVIEW,OUTBOUND_DIRECT_PREVIEW,OUTBOUND_PERSONAL_CALLBACK_PREVIEW</value>
</Trigger>
```
Workflow APIs

Workflow—Get

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

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Workflow/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Workflow/195">http://finesse1.xyz.com/finesse/api/Workflow/195</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>400: Bad Request</td>
</tr>
<tr>
<td></td>
<td>400: Finesse API Error</td>
</tr>
<tr>
<td></td>
<td>401: Authorization Failure</td>
</tr>
<tr>
<td></td>
<td>403: Forbidden</td>
</tr>
<tr>
<td></td>
<td>404: Not Found</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
</tbody>
</table>
Example Response:
<Workflow>
  <uri>/finesse/api/Workflow/195</uri>
  <name>Workflow A</name>
  <description>Workflow description</description>
  <media>Media Channel</media>
  <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>CONTAINS</comparator>
        <value>1234</value>
      </Condition>
    </conditions>
  </ConditionSet>
</Workflow>
Workflow—Get List

This API allows an administrator to get a list of workflows.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Workflows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Workflows">http://finesse1.xyz.com/finesse/api/Workflows</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>
**HTTP Response:**

200: Success  
400: Bad Request  
400: Finesse API Error  
401: Authorization Failure  
403: Forbidden  
500: Internal Server Error

**Example Response:**

```xml
<Workflows>
  <Workflow>...Full Workflow Object...</Workflow>
  <Workflow>...Full Workflow Object...</Workflow>
  <Workflow>...Full Workflow Object...</Workflow>
</Workflows>
```

**Example Failure Response:**

```xml
<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:**

```xml
<Workflow>...Full Workflow Object...</Workflow>
```
Request Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id (required)</td>
<td>Maps to the primary key of the workflow entry</td>
</tr>
<tr>
<td>name (required)</td>
<td>The name of the workflow</td>
</tr>
<tr>
<td>description (optional)</td>
<td>A description of the workflow</td>
</tr>
<tr>
<td>Media (optional)</td>
<td>The media of the workflow</td>
</tr>
<tr>
<td>TriggerSet (required)</td>
<td>A set of events that cause the conditions to be evaluated</td>
</tr>
<tr>
<td>ConditionSet (optional)</td>
<td>A set of conditions that determine if the workflow is executed</td>
</tr>
<tr>
<td>workflowActions (optional)</td>
<td>A list of workflow actions to execute if the trigger and conditions are satisfied</td>
</tr>
</tbody>
</table>

HTTP Response:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
</tr>
<tr>
<td>400</td>
<td>Finesse API Error</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

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.

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>
```
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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Workflow/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Workflow/769">http://finesse1.xyz.com/finesse/api/Workflow/769</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>
| HTTP Request: | `<Workflow>  
...Workflow Object...  
</Workflow>` |
| Request Parameters: | id (required): Maps to the primary key of the workflow entry  
named (optional): The name of the workflow  
description (optional): A description of the workflow  
Media (optional): The media 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 |
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the Workflow object.</td>
<td>—</td>
<td>The id in the URI maps to the primary key of the workflow.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the workflow.</td>
<td>—</td>
<td>Must be unique. Maximum of 40 characters.</td>
</tr>
<tr>
<td>media</td>
<td>String</td>
<td>Media channel of the workflow</td>
<td>—</td>
<td>Media channel maps to the media id.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note</strong> Domain List can be obtained from the MediaDomain API.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• For Unified CCE, you can define custom media channels for Voice</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>and Digital Channels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• For Unified CCX, the media channels are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Voice with media id as 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Chat with media id as Chat.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Email with media id as Email.</td>
</tr>
<tr>
<td>TriggerSet</td>
<td>Object</td>
<td>A set of events that cause the conditions to be</td>
<td>—</td>
<td><strong>Note</strong> If no media channels are specified, Voice is set as the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>evaluated.</td>
<td></td>
<td>default media.</td>
</tr>
</tbody>
</table>

---

Domain List can be obtained from the MediaDomain API.

- For Unified CCE, you can define custom media channels for Voice and Digital Channels.

- For Unified CCX, the media channels are:
  - Voice with media id as 1.
  - Chat with media id as Chat.
  - Email with media id as Email.
### Workflow API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConditionSet</td>
<td>Object</td>
<td>A set of conditions that determine whether the workflow executes.</td>
<td>—</td>
<td>You can assign up to five conditions to a workflow.</td>
</tr>
<tr>
<td>workflowActions</td>
<td>Object</td>
<td>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.</td>
<td>—</td>
<td>You can assign up to five workflow actions to a workflow.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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 249.</td>
</tr>
</tbody>
</table>

#### ConditionSet Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyMethod</td>
<td>String</td>
<td>Determines whether any or all of the conditions must be met for the workflow to execute.</td>
<td>ANY, ALL</td>
<td></td>
</tr>
<tr>
<td>conditions</td>
<td>Object</td>
<td>A list of conditions for the workflow.</td>
<td>—</td>
<td>Maximum of five conditions for a workflow.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A workflow with no conditions is specified by a conditions parameter with no Condition elements.</td>
</tr>
<tr>
<td>Condition</td>
<td>Object</td>
<td>Information about a workflow condition.</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>
### Workflow API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Object</td>
<td>A piece of data from the Trigger event used to filter the event.</td>
<td>—</td>
<td>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.</td>
</tr>
<tr>
<td>comparator</td>
<td>String</td>
<td>The operator used to compare the event variable to the desired value.</td>
<td>IS_EQUAL, IS_NOT_EQUAL, BEGINS_WITH, ENDS_WITH, CONTAINS, IS_EMPTY, IS_NOT_EMPTY, IS_IN_LIST, IS_NOT_IN_LIST</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to compare the event variable with.</td>
<td>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).</td>
<td></td>
</tr>
</tbody>
</table>

### TriggerSet Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>The type of TriggerSet.</td>
<td>SYSTEM</td>
<td></td>
</tr>
</tbody>
</table>
### Workflow API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the TriggerSet</td>
<td>When type is SYSTEM, valid values are CALL_ARRIVES, CALL_ANSWERED, CALL_ENDS, CALL_IS_MADE, and CALL_IS_PREVIEWED.</td>
<td></td>
</tr>
</tbody>
</table>

| allow Overlapping CallWorkflow | Boolean | Indicates whether workflow for a second simultaneous call can fire 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Object</td>
<td>A piece of data from the trigger event to be used to filter the event. Contains a name, node, and type.</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

| 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. | —               |       |
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.

### Notes
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.

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

### Workflow API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>The request body is invalid.</td>
</tr>
<tr>
<td>400</td>
<td>Finesse API Error</td>
<td>API error such as the object is stale or does not exist.</td>
</tr>
<tr>
<td>Status</td>
<td>Error Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>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).</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>The user attempted to run the API against the secondary Finesseserver. Configuration APIs cannot be run against the secondary Finesseserver.</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>The specified resource cannot be found.</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
</tbody>
</table>

**WorkflowAction**

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

The WorkflowAction object is structured as follows:

```xml
<WorkflowAction>
  <uri>/finesse/api/WorkflowAction/{id}</uri>
  <name>DuckDuckGo</name>
  <type>HTTP_REQUEST</type>
  <handledBy>FINESSE_DESKTOP</handledBy>
  <params>
    <Param>
      <name>name</name>
      <value>DuckDuckGo</value>
    </Param>
  </params>
</WorkflowAction>
```

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

The BROWSER_POP type is structured as follows:

```xml
<WorkflowAction>
  <uri>/finesse/api/WorkflowAction/{id}</uri>
  <name>DuckDuckGo</name>
  <type>BROWSER_POP</type>
  <handledBy>FINESSE_DESKTOP</handledBy>
  <params>
```
The HTTP REQUEST type is structured as follows:

```xml
<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>
</WorkflowAction>
```
WorkflowAction APIs

WorkflowAction—Get

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

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/WorkflowAction/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/WorkflowAction/674">http://finesse1.xyz.com/finesse/api/WorkflowAction/674</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>400: Bad Request</td>
</tr>
<tr>
<td></td>
<td>400: Finesse API Error</td>
</tr>
<tr>
<td></td>
<td>401: Authorization Failure</td>
</tr>
<tr>
<td></td>
<td>403: Forbidden</td>
</tr>
<tr>
<td></td>
<td>404: Not Found</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
<tr>
<td>Example Response:</td>
<td>&lt;WorkflowAction&gt;</td>
</tr>
<tr>
<td></td>
<td>...Full WorkflowAction Object...</td>
</tr>
<tr>
<td></td>
<td>&lt;/WorkflowAction&gt;</td>
</tr>
<tr>
<td>Example Failure Response:</td>
<td>&lt;ApiErrors&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ApiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorData&gt;Action 674 not found.&lt;/ErrorData&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorType&gt;Not Found&lt;/ErrorType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ErrorMessage&gt;HTTP Status code:404 (Not Found)</td>
</tr>
<tr>
<td></td>
<td>Api Error Type: Not Found</td>
</tr>
<tr>
<td></td>
<td>Error Message: Workflow not found with an id of 674</td>
</tr>
<tr>
<td></td>
<td>&lt;/ErrorMessage&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ApiError&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ApiErrors&gt;</td>
</tr>
</tbody>
</table>

WorkflowAction—Get List

This API allows an administrator to get a list of workflow actions.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/WorkflowActions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/WorkflowActions">http://finesse1.xyz.com/finesse/api/WorkflowActions</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>400: Bad Request</td>
</tr>
<tr>
<td></td>
<td>400: Finesse API Error</td>
</tr>
<tr>
<td></td>
<td>401: Authorization Failure</td>
</tr>
<tr>
<td></td>
<td>403: Forbidden</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
</tbody>
</table>

**Example Response:**

```
<WorkflowActions>
  <WorkflowAction>
    <name>WorkflowAction 1</name>
    <type>HTTP</type>
    <uri>/finesse/api/WorkflowAction/{id}</uri>
  </WorkflowAction>
  <WorkflowAction>
    <name>WorkflowAction 2</name>
    <type>DELAY</type>
    <uri>/finesse/api/WorkflowAction/{id}</uri>
  </WorkflowAction>
</WorkflowActions>
```

**Example Failure Response:**

```
<ApiErrors>
  <ApiError>
    <ErrorData>Database read/write error</ErrorData>
    <ErrorType>Bad Request</ErrorType>
    <ErrorStatus>HTTP status code: 400 (Bad Request)
      Api Error Type:  Bad Request
      Error Message: Database read/write error</ErrorStatus>
  </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 action
- **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 action
- **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:

```xml
<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:

```xml
<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
### WorkflowAction—Delete

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

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/WorkflowAction/&lt;id&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/WorkflowAction/768">http://finesse1.xyz.com/finesse/api/WorkflowAction/768</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>DELETE</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>

#### 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>
```
**HTTP Response:**

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
</tr>
<tr>
<td>400</td>
<td>Finesse API Error</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the WorkflowAction object.</td>
<td>—</td>
<td>The id in the URI maps to the primary key of the WorkflowAction.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the workflow action.</td>
<td>—</td>
<td>Must be unique. Maximum of 64 characters.</td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>The type of workflow action</td>
<td>BROWSER_POP, HTTP_REQUEST</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Possible Values</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>--------------------------------------------------</td>
<td>-----------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>handledBy</td>
<td>String</td>
<td>Indicates what handles the action when it is triggered by a workflow.</td>
<td>FINESSE_DESKTOP, OTHER</td>
<td>For FINESSE_DESKTOP, the Finesses workflow engine executes the action. For OTHER, the action event is published on the OpenAJAX hub but is not executed by the Finesses desktop. This allows a third-party gadget to execute the action.</td>
</tr>
<tr>
<td>params</td>
<td>Object</td>
<td>A list of Param subobjects.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--&gt;Param</td>
<td>Object</td>
<td>Includes a name and value pair.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--&gt;name</td>
<td>String</td>
<td>The name of the parameter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--&gt;value</td>
<td>String</td>
<td>The value of the parameter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>actionVariables</td>
<td>Object</td>
<td>List of ActionVariable subobjects.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--&gt;ActionVariable</td>
<td>Object</td>
<td>Set of information about one ActionVariable.</td>
<td></td>
<td>You can assign up to five ActionVariable parameters to a workflow.</td>
</tr>
<tr>
<td>--&gt;name</td>
<td>String</td>
<td>The name of the variable.</td>
<td></td>
<td>Maximum of 32 characters.</td>
</tr>
</tbody>
</table>
### WorkflowAction API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>node</td>
<td>String</td>
<td>The XPath to extract from the dialog XML.</td>
<td>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.</td>
</tr>
<tr>
<td>---</td>
<td>type</td>
<td>String</td>
<td>Indicates the type of variable</td>
<td>CUSTOM, SYSTEM</td>
</tr>
<tr>
<td>---</td>
<td>testValue</td>
<td>String</td>
<td>The value used to test the variable.</td>
<td>Maximum of 128 characters.</td>
</tr>
</tbody>
</table>

### Param Values (BROWSER_POP)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
<th>Size</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>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: <a href="http://www.example.com?q=$%7BcallVariable1%7D">http://www.example.com?q=${callVariable1}</a> 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, \1). A literal backslash must be escaped with another backslash (\).</td>
<td>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: <a href="http://www.example.com?q=$%7BcallVariable1%7D">http://www.example.com?q=${callVariable1}</a> 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, \1). A literal backslash must be escaped with another backslash (\).</td>
<td>500</td>
<td>Yes</td>
</tr>
<tr>
<td>windowName</td>
<td>The window name to pop open The window name is passed to the browser Window Open method by the workflow 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.</td>
<td>The window name is passed to the browser Window Open method by the workflow 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.</td>
<td>40</td>
<td>No</td>
</tr>
</tbody>
</table>

### Param (HTTP_REQUEST)
## WorkflowAction API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
<th>Size</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>The path to use in the HTTP_REQUEST action</td>
<td>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: <a href="http://www.example.com?q=$%7BcallVariable1%7D">http://www.example.com?q=${callVariable1}</a> 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</td>
<td>500</td>
<td>Yes</td>
</tr>
<tr>
<td>method</td>
<td>The method to use in the HTTP_REQUEST</td>
<td>PUT, POST</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>authenticationType</td>
<td>The authentication type to use in the HTTP_REQUEST</td>
<td>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.</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>location</td>
<td>Defines if the HTTP_REQUEST is to Finesse or to a third party application</td>
<td>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.</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>contentType</td>
<td>The value of the content type header to send with the HTTP_REQUEST</td>
<td>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.</td>
<td>500</td>
<td>No</td>
</tr>
</tbody>
</table>
WorkflowAction API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>The request body is invalid.</td>
</tr>
<tr>
<td>400</td>
<td>Finesse API Error</td>
<td>API error such as the object is stale or does not exist.</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
<td>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).</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>The user attempted to run the API against the secondary Finesse server. Configuration APIs cannot be run against the secondary Finesse server.</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>The specified resource cannot be found.</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
</tbody>
</table>

Team

The Team object represents a team and the resources associated with that team. For more information, see Team, on page 140.
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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Teams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Teams">http://finesse1.xyz.com/finesse/api/Teams</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
</tbody>
</table>

**Example Response:**

```xml
<Teams>
  <Team>
    ...Summary Team Object...
  </Team>
  <Team>
    ...Summary Team Object...
  </Team>
  <Team>
    ...Summary Team Object...
  </Team>
</Teams>
```

**Example Failure Response:**

```xml
<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`.

<table>
<thead>
<tr>
<th><strong>URI:</strong></th>
<th>http://&lt;FQDN&gt;/finesse/api/Team/&lt;id&gt;/ReasonCodes?category=&lt;category&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example URI:</strong></td>
<td><a href="http://finesse1.xyz.com/finesse/api/Team/574/ReasonCodes?category=NOT_READY">http://finesse1.xyz.com/finesse/api/Team/574/ReasonCodes?category=NOT_READY</a></td>
</tr>
<tr>
<td><strong>Security Constraints:</strong></td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td><strong>HTTP Method:</strong></td>
<td>GET</td>
</tr>
<tr>
<td><strong>Content Type:</strong></td>
<td>—</td>
</tr>
<tr>
<td><strong>Input/Output Format:</strong></td>
<td>XML</td>
</tr>
<tr>
<td><strong>HTTP Request:</strong></td>
<td>—</td>
</tr>
<tr>
<td><strong>HTTP Response:</strong></td>
<td>200: Success 400: Bad Request 400: Finesse API Error 401: Authorization Failure 403: Forbidden 404: Not Found 500: Internal Server Error</td>
</tr>
</tbody>
</table>
| **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.

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).

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).

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Team/&lt;Id&gt;/ReasonCodes?category=&lt;category&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Team/574/ReasonCodes?category=NOT_READY">http://finesse1.xyz.com/finesse/api/Team/574/ReasonCodes?category=NOT_READY</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>Request Parameters:</td>
<td>id (required): The database ID for the team category (required): The category of reason code (NOT_READY or LOGOUT)</td>
</tr>
</tbody>
</table>
### 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 196**.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Team/&lt;id&gt;/WrapUpReasons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Team/574/WrapUpReasons">http://finesse1.xyz.com/finesse/api/Team/574/WrapUpReasons</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>400: Bad Request</td>
</tr>
<tr>
<td></td>
<td>400: Finesse API Error</td>
</tr>
<tr>
<td></td>
<td>401: Authorization Failure</td>
</tr>
<tr>
<td></td>
<td>403: Forbidden</td>
</tr>
<tr>
<td></td>
<td>404: Not Found</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
</tbody>
</table>
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.

This API restricts the maximum number of non-global wrap-up reasons that can be assigned to a single team to 100.

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.

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>
```

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
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 216.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Team/&lt;id&gt;/PhoneBooks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Team/574/PhoneBooks">http://finesse1.xyz.com/finesse/api/Team/574/PhoneBooks</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
</tbody>
</table>
**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.

---

**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>
```
<table>
<thead>
<tr>
<th>HTTP Method:</th>
<th>PUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>
| HTTP Request: | `<PhoneBooks>`<PhoneBook>`
| | `<uri>/finesse/api/PhoneBook/123</uri>`
| | `<PhoneBook>`
| | `<uri>/finesse/api/PhoneBook/456</uri>`
| | `<PhoneBook>`
| | `<uri>/finesse/api/PhoneBook/789</uri>`
| | `....`<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.

<table>
<thead>
<tr>
<th>URI:</th>
<th>http://&lt;FQDN&gt;/finesse/api/Team/&lt;id&gt;/LayoutConfig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URI:</td>
<td><a href="http://finesse1.xyz.com/finesse/api/Team/574/LayoutConfig">http://finesse1.xyz.com/finesse/api/Team/574/LayoutConfig</a></td>
</tr>
<tr>
<td>Security Constraints:</td>
<td>Only administrators can use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>Security Constraints:</th>
<th>Only administrators can use this API.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Method:</td>
<td>PUT</td>
</tr>
<tr>
<td>Content Type:</td>
<td>Application/XML</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
</tbody>
</table>

**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
**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 231.

### 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:
—

### 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:
```xml
<Workflows>
  <Workflow>
    ... Summary Workflow Object ...
  </Workflow>
  <Workflow>
    ... Summary Workflow Object ...
  </Workflow>
  <Workflow>
    ... Summary Workflow Object ...
  </Workflow>
...
</Workflows>
```
### 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.

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 271).

| 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>`  
| | `<uri>`finesse/api/Workflow/456</uri>`  
| | `<Workflow>`  
| | `<uri>`finesse/api/Workflow/789</uri>`  
| | ```
| | ....
| | </Workflows>` |
| Request Parameters: | id (required): The database ID for the team |
HTTP Response:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
</tr>
<tr>
<td>400</td>
<td>Finess API Error</td>
</tr>
<tr>
<td>401</td>
<td>Authorization Failure</td>
</tr>
<tr>
<td>401</td>
<td>Invalid Authorization User Specified</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

Example Failure Response:

```xml
<ApiErrors>
  <ApiError>
    <ErrorData>574</ErrorData>
    <ErrorType>finess.api.team.team_assignment_invalid_team</ErrorType>
    <ErrorMessage>HTTP Status code: 404 (Not Found)
     Api Error Type:finess.api.team.team_assignment_invalid_team Error Message:
     This is not a valid team</ErrorMessage>
  </ApiError>
</ApiErrors>
```

Team API Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The URI to get a new copy of the Team, ReasonCode, WrapUpReason, LayoutConfig, or Workflow object.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td>The unique identifier for the team.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the team.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>category</td>
<td>String</td>
<td>Specifies the type of reason code.</td>
<td>NOT_READY, LOGOUT</td>
<td>—</td>
</tr>
<tr>
<td>useDefault</td>
<td>Boolean</td>
<td>Determines whether to use the default desktop layout for this team.</td>
<td>true, false</td>
<td>—</td>
</tr>
<tr>
<td>layoutxml</td>
<td>String</td>
<td>The XML data that determines the desktop layout.</td>
<td>—</td>
<td>If useDefault is set to true and the layoutxml is provided in a request, the layoutxml is ignored.</td>
</tr>
</tbody>
</table>
## Team API Errors

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

## 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
<table>
<thead>
<tr>
<th>Security Constraints:</th>
<th>Only administrators can use this API.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
<tr>
<td></td>
<td>401: Authorization Failure</td>
</tr>
<tr>
<td></td>
<td>403: Forbidden</td>
</tr>
<tr>
<td></td>
<td>500: Internal Server Error</td>
</tr>
</tbody>
</table>
SystemVariable—List

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[.="BAAccountNumber"]/../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>
  <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>
No API errors are returned. Responses are 401/403/404 Errors.

### SystemVariable API Parameters

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

### SystemVariable API Errors

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

Note

If a user repeatedly passes an invalid password in the basic authorization header to a serviceability API, on the fifth invalid attempt, Finesse blocks the user's access to all serviceability APIs for 5 minutes. This lock period differs from the 30-minute lock period implemented for the Finesse administrator console.

- SystemInfo, on page 279
- Diagnostic Portal APIs, on page 282

SystemInfo

The SystemInfo object represents the Finesse system and includes the deployment type (whether Finesse is deployed with Unified CCE or Unified CCX), the peripheral ID (for Unified CCE only), the installed license (for Unified CCX only), the current system state, the XMPP server and pubSub domains, the system auth mode (whether Non-SSO, SSO, or Hybrid) where Hybrid is for Unified CCE only, and the hostnames of the primary and secondary (if configured) Finesse nodes.

The SystemInfo object is structured as follows:

```xml
<SystemInfo>
  <currentTimestamp></currentTimestamp>
  <deploymentType></deploymentType>
  <license/>
  <peripheralId></peripheralId>
  <primaryNode>
    <host></host>
  </primaryNode>
  <secondaryNode>
    <host></host>
  </secondaryNode>
  <status></status>
  <statusReason></statusReason>
  <systemAuthMode>NON_SSO</systemAuthMode>
  <timezoneOffset></timezoneOffset>
  <uri>/finesse/api/SystemInfo</uri>
  <xmppDomain></xmppDomain>
  <xmppPubSubDomain></xmppPubSubDomain>
</SystemInfo>
```
SystemInfo—Get

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>deploymentType</td>
<td>String</td>
<td>The type of deployment for Finesse.</td>
<td>UCCE, UCCX</td>
<td></td>
</tr>
</tbody>
</table>

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>
  <peripheralId>5001</peripheralId>
  <license></license>
  <status>IN_SERVICE</status>
  <statusReason></statusReason>
  <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>
```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>peripheralId</td>
<td>String</td>
<td>The ID of the Unified CCE peripheral to which Finesse is connected.</td>
<td></td>
<td>This parameter is blank for Unified CCX deployments.</td>
</tr>
<tr>
<td>license</td>
<td>String</td>
<td>The Unified CCX license.</td>
<td>STANDARD, ENHANCED, or PREMIUM</td>
<td>This parameter is blank for Unified CCE deployments.</td>
</tr>
<tr>
<td>currentTimeStamp</td>
<td>String</td>
<td>The current time (GMT time) in the following format:</td>
<td>YYYY-MM-DDThh:MM:ss.SSZ</td>
<td>—</td>
</tr>
<tr>
<td>statusReason</td>
<td>String</td>
<td>The reason for which Finesse system is down.</td>
<td>Possible out-of-service scenarios returned by Finesse system:</td>
<td>This parameter is blank when Finesse system is IN_SERVICE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Cisco Finesse Database is down.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Cisco Finesse Notification Service is down.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Finesse connection to CTI Server is down.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• CTI Peripheral ID xxx is down.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• System is initializing.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Local Unified CCX Engine is not in Service.</td>
<td>(Unified CCX only)</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>String</td>
<td>The state of the Finesse system.</td>
<td>IN_SERVICE: The system is in service and normal operations are accepted.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OUT_OF_SERVICE: The system is out of service and normal operations result in a 503 Service Unavailable response.</td>
<td></td>
</tr>
<tr>
<td>timezoneOffset</td>
<td>Integer</td>
<td>The difference (in minutes) between the server time and GMT time.</td>
<td>—</td>
<td>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.</td>
</tr>
<tr>
<td>xmppDomain</td>
<td>String</td>
<td>The XMPP server domain.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xmppPubSubDomain</td>
<td>String</td>
<td>The XMPP server pubsub domain.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>primaryNode-host</td>
<td>String</td>
<td>The hostname or IP address of the primary Finesse node.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secondaryNode-host</td>
<td>String</td>
<td>The hostname or IP address of the secondary Finesse node.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SystemInfo API Errors

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
</tbody>
</table>

### 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.
The Diagnostic Portal APIs are not usable unless Finesse has initially gone IN_SERVICE, after which Finesse can go OUT_OF_SERVICE and the APIs should continue to work.

### Diagnostic Portal—Get Performance Information

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

<table>
<thead>
<tr>
<th>URI:</th>
<th><a href="https://FQDN/finesse-dp/rest/DiagnosticPortal/GetPerformanceInformation">https://FQDN/finesse-dp/rest/DiagnosticPortal/GetPerformanceInformation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Constraints:</td>
<td>A user must be signed in as an administrator to use this API.</td>
</tr>
<tr>
<td>HTTP Method:</td>
<td>GET</td>
</tr>
<tr>
<td>Content Type:</td>
<td>—</td>
</tr>
<tr>
<td>Input/Output Format:</td>
<td>XML</td>
</tr>
<tr>
<td>HTTP Request:</td>
<td>—</td>
</tr>
<tr>
<td>HTTP Response:</td>
<td>200: Success</td>
</tr>
</tbody>
</table>

**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
Diagnostic Portal—Get Product Version

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

**URI:**

**Example URI:**

**Security Constraints:**
A user must be signed in as an administrator to use this API.

**HTTP Method:**
GET

**Content Type:**
—

**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
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <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
<?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

<table>
<thead>
<tr>
<th>Status</th>
<th>Error Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>Authorization Error</td>
<td>The user is not authorized to access this API.</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>The resource is not found (for example, the DiagnosticPortal has been deleted).</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Any runtime exception is caught and responded with this error.</td>
</tr>
</tbody>
</table>
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-0060 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
- Media - /finesse/api/User/{id}/Media/{mrd-id}
• 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:

```xml
<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:

```xml
<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.

```xml
<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 Finesse 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.

<table>
<thead>
<tr>
<th>Format:</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node:</td>
<td>/finesse/api/User/{id}</td>
</tr>
<tr>
<td>Source:</td>
<td>/finesse/api/User/{id}</td>
</tr>
<tr>
<td>Data:</td>
<td>User</td>
</tr>
</tbody>
</table>
| Payload: | <Update>  
  <event>{put|delete}</event>  
  <source>/finesse/api/User/{id}</source>  
  <data>  
    <user>  
      <!-- full User object -->  
    </user>  
  </data>  
</Update> |
Sample Notification Payload:

```xml
<Update>
  <event>put</event>
  <source>/finesse/api/User/csmith</source>
  <data>
    <User>
      <dialogs>/finesse/api/User/1001001/Dialogs</dialogs>
      <firstName>AGENT</firstName>
      <lastName>1001001</lastName>
      <loginId>1001001</loginId>
      <reasonCodeId>2</reasonCodeId>
      <Role>Agent</Role>
      <state>LOGOUT</state>
    </User>
  </data>
</Update>
```

Notification Triggers:

- Addition of a user
- Deletion of a user
- State change
- First or last name change
- Role change
- Pending state change

Dialog Notification

Finesse sends a Dialog notification when information (or an action) changes for a call to which the user belongs or when the user adds or removes a dialog.

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.

When an agent who is on a call places a consult call, the original call will be on hold and the consult call will
be active. Once the call is complete where the agent either transfers or places the call on conference, the
surviving call's dialog notifications will contain the dropped call's dialog id in the secondary id field.

During Dialog notifications, there are two types of notifications that get sent to the Dialog node.

• When a dialog is added or removed from the Dialog collection of the user.

<table>
<thead>
<tr>
<th>Format:</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node:</td>
<td>/finesse/api/User/{id}/Dialogs</td>
</tr>
</tbody>
</table>
| Source:  | /finesse/api/User/{id}/Dialogs (when a Dialog is added or removed from the Dialog
collection for the user) |
| Data:    | Dialogs |
| Payload: | &lt;Update&gt;
  &lt;data&gt;
    &lt;dialogs&gt;
      &lt;Dialog&gt;
        <!-- full Dialog object -->
      &lt;/Dialog&gt;
    &lt;/dialogs&gt;
  &lt;/data&gt;
&lt;event&gt;{POST|DELETE}&lt;/event&gt;
&lt;requestId&gt;xxxxxxxx&lt;/requestId&gt;
&lt;source&gt;/finesse/api/User/{id}/Dialogs&lt;/source&gt;
&lt;/Update&gt; |
| Sample Notification Payload: |  |
<Update>
  <data>
    <dialogs>
      <Dialog>
        <associatedDialogUri/>
        <fromAddress>1112554</fromAddress>
        <id>2130715746</id>
        <secondaryId>2130715747</secondaryId>
        <mediaProperties>
          <mediaId>1</mediaId>
          <DNIS>90101</DNIS>
          <callType>CONSULT</callType>
          <dialedNumber>90101</dialedNumber>
        </mediaProperties>
        <callvariables>
          <CallVariable>
            <name>callVariable1</name>
            <value>1</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable2</name>
            <value>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable3</name>
            <value>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable4</name>
            <value>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable5</name>
            <value>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable6</name>
            <value>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable7</name>
            <value>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable8</name>
            <value>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable9</name>
            <value>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable10</name>
            <value>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable11</name>
            <value>0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
          </CallVariable>
        </callvariables>
      </Dialog>
    </dialogs>
  </data>
</Update>
Cisco Finesse Notifications

**Dialog Notification**

```xml
<callVariables>
    <queueNumber>5022</queueNumber>
    <queueName>UCM_PIM.Func.Agents.SG</queueName>
    <callKeyCallId>217</callKeyCallId>
    <callKeyPrefix>152018</callKeyPrefix>
</callVariables>

<mediaProperties>
    <mediaType>Voice</mediaType>
</mediaProperties>

<participants>
    <Participant>
        <actions>
            <action>UPDATE_CALL_DATA</action>
            <action>DROP</action>
        </actions>
        <mediaAddress>1112554</mediaAddress>
        <mediaAddressType>AGENT_DEVICE</mediaAddressType>
        <startTime>2016-05-03T21:49:36.512Z</startTime>
        <state>INITIATING</state>
        <stateChangeTime>2016-05-03T21:49:36.512Z</stateChangeTime>
    </Participant>
</participants>

<state>INITIATING</state>
<toAddress>90101</toAddress>
<uri>/finesse/api/Dialog/2130715746</uri>
</Dialog>
</dialogs>
</data>
<event>POST</event>
:requestId>edc7064f-1178-11e6-8bd0-005056000005</requestId>
<source>/finesse/api/User/112554/Dialogs</source>
</Update>
```

**Notification Triggers:**
- Incoming call
- Ending a call

- When dialog properties associated with the specified Dialog id is modified.

<table>
<thead>
<tr>
<th>Format:</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node:</td>
<td>/finesse/api/User/{id}/Dialogs</td>
</tr>
<tr>
<td>Source:</td>
<td>/finesse/api/Dialog/{id} (when a Dialog within the Dialogs collection for the user is modified)</td>
</tr>
<tr>
<td>Data:</td>
<td>Dialog</td>
</tr>
</tbody>
</table>
| Payload: | `<Update>
            <data>
            <dialog>
            <!-- full Dialog object -->
            </dialog>
            </data>
            </event>{PUT}</event>
          <requestId>xxxxxxxx</requestId>
          <source>/finesse/api/Dialog/16804377</source>
        </Update>` |
<table>
<thead>
<tr>
<th>Sample Notification Payload:</th>
<th></th>
</tr>
</thead>
</table>
<Update>
  <data>
    <dialog>
      <associatedDialogUri>1081001</associatedDialogUri>
      <fromAddress>1081001</fromAddress>
      <id>16804377</id>
      <mediaProperties>
        <mediaId>1</mediaId>
        <DNIS>1081002</DNIS>
        <callType>AGENT_INSIDE</callType>
        <callVariables>
          <CallVariable>
            <name>callVariable1</name>
            <value></value>
            <queueNumber>5022</queueNumber>
            <queueName>UCM_PIM.Func.Agents.SG</queueName>
            <callKeyCallId>217</callKeyCallId>
            <callKeyPrefix>152018</callKeyPrefix>
            <dialedNumber>1081002</dialedNumber>
          </CallVariable>
        </callVariables>
        <mediaType>Voice</mediaType>
      </mediaProperties>
      <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>
    </dialog>
  </data>
  <event>PUT</event>
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

### Dialogs/Media Notification

Finesse sends a Dialogs/Media notification when information (or an action) changes for a nonvoice dialog to which the user belongs.

For an interruptible Media Routing Domain configured to accept interrupts, Finesse sends only a Media state change when an agent is interrupted in that MRD. It does not send Dialogs/Media notifications with the action list modified to reflect the fact that actions not permitted on the tasks in that media. The state change is the only indication to the Finesse applications that no actions are allowed on the interrupted dialogs.

During Dialog notifications, there are two types of notifications that get sent to the Dialog node.

- When a dialog is added or removed from the Dialog collection of the user.

<table>
<thead>
<tr>
<th>Format:</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node:</td>
<td>/finesse/api/User/{id}/Dialogs/Media</td>
</tr>
<tr>
<td>Source:</td>
<td>/finesse/api/User/{id}/Media/{mrdId}/Dialogs (when a Dialog is added or removed from the Dialog collection for the user, for example offered or closed)</td>
</tr>
<tr>
<td>Data:</td>
<td>Dialogs</td>
</tr>
</tbody>
</table>
| Payload:| `<Update>
  <data>
    <dialogs>
      <Dialog>
        <!-- full Dialog object -->
      </Dialog>
    </dialogs>
  </data>
  <event>{POST|DELETE}</event>
  <RequestId>xxxxxxxxx</RequestId>
  <source>/finesse/api/User/{id}/Media(mrdId)/Dialogs</source>
</Update>` |
### Sample Notification Payload

```
<Update>
  <data>
    <dialogs>
      <Dialog>
        <associatedDialogUri>/finesse/api/Dialog/3216_5432_1</associatedDialogUri>
        <id>1234_5423_1</id>
        <mediaType>Cisco_Chat_MRD</mediaType>
        <mediaProperties>
          <mediaId>5002</mediaId>
          <dialedNumber></dialedNumber>
          <callvariables>
            <CallVariable>
              <name>callVariable1</name>
              <value>Chuck Smith</value>
            </CallVariable>
            <CallVariable>
              <name>callVariable2</name>
              <value>Cisco Systems, Inc.</value>
            </CallVariable>
            ...Other CallVariables ...
          </callvariables>
          <queueNumber>5022</queueNumber>
        </mediaProperties>
        <participants>
          <Participant>
            <actions>
              <action>ACCEPT</action>
            </actions>
            <mediaAddress>1001001</mediaAddress>
            <startTime>2015-11-19T06:04:27.864Z</startTime>
            <state>OFFERED</state>
            <stateChangeTime>2015-11-19T06:04:27.864Z</stateChangeTime>
          </Participant>
        </participants>
        <state>OFFERED</state>
        <uri>/finesse/api/Dialog/1234_5423_1</uri>
      </Dialog>
    </dialogs>
  </data>
</Update>
```

### Notification Triggers:

- Incoming dialog
- When dialog properties associated with the specified Dialog id is modified.

### Format:
- XML

### Node:
- `/finesse/api/User/{id}/Dialogs/Media`

### Source:
- `/finesse/api/Dialog/{id}` (when a Dialog within the Dialogs collection for the user is modified, for example accepted, started, paused, or wrapped up)
## Payload:

```xml
<Update>
  <data>
    <dialog>
      <!-- full Dialog object -->
    </dialog>
  </data>
  <event>{PUT}</event>
  <requestId>xxxxxxxxx</requestId>
  <source>/finesse/api/Dialogs{id}</source>
</Update>
```

### Sample Notification Payload

```
Update>
  <data>
    <dialog>
      <associatedDialogUri/>
      <id>151705_33542697_1</id>
      <mediaProperties>
        <mediaId>5000</mediaId>
        <dialedNumber>mark_test_dn</dialedNumber>
        <callvariables>
          <CallVariable>
            <name>callVariable1</name>
            <value>cv1_value</value>
          </CallVariable>
          <CallVariable>
            <name>callVariable2</name>
            <value>cv2_value</value>
          </CallVariable>
          <CallVariable>
            <name>user.finesse.ecc1</name>
            <value>ecc1</value>
          </CallVariable>
        </callvariables>
        <queueNumber>5022</queueNumber>
        <queueName>UCM_PIM.Func.Agents.SG</queueName>
        <callKeyCallId>217</callKeyCallId>
        <callKeyPrefix>152018</callKeyPrefix>
      </mediaProperties>
      <mediaType>Cisco_Chat_MRD</mediaType>
      <participants>
        <Participant>
          <actions>
            <action>START</action>
            <action>CLOSE</action>
            <action>TRANSFER</action>
          </actions>
          <mediaAddress>1001010</mediaAddress>
          <startTime>2016-05-10T20:25:12.302Z</startTime>
          <state>ACCEPTED</state>
          <stateChangeTime>2016-05-10T20:25:17.372Z</stateChangeTime>
        </Participant>
      </participants>
      <uri>/finesse/api/Dialog/151705_33542697_1</uri>
    </dialog>
  </data>
  <event>PUT</event>
  <requestId/>
  <source>/finesse/api/Dialogs{id}</source>
</Update>
```
### 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.
Asynchronous Errors

When accessing the Finesse REST API through the Finesse JavaScript library, asynchronous errors have a status code of 400. When receiving the asynchronous error directly through XMPP, the error message has the format described in the description above for Dialog CTI Error Notification.

<table>
<thead>
<tr>
<th>ErrorType</th>
<th>Reason</th>
<th>Deployment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call Operation Failure</td>
<td>Attempt to exceed maximum allowed conference participants.</td>
<td>Unified CCE</td>
</tr>
</tbody>
</table>

Note

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:

```xml
<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>
        </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>2012-02-15T17:58:21Z</startTimeOfLongestCallInQueue>
        <agentsReady>1</agentsReady>
        <agentsNotReady>2</agentsNotReady>
        <agentsTalkingInbound>3</agentsTalkingInbound>
        <agentsTalkingOutbound>4</agentsTalkingOutbound>
        <agentsTalkingInternal>5</agentsTalkingInternal>
        <agentsWrapUpNotReady>6</agentsWrapUpNotReady>
        <agentsWrapUpReady>7</agentsWrapUpReady>
      </statistics>
    </Queue>
  </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>2012-02-15T17:58:21Z</startTimeOfLongestCallInQueue>
        <agentsReady>1</agentsReady>
        <agentsNotReady>2</agentsNotReady>
        <agentsTalkingInbound>3</agentsTalkingInbound>
        <agentsTalkingOutbound>4</agentsTalkingOutbound>
        <agentsTalkingInternal>5</agentsTalkingInternal>
        <agentsWrapUpNotReady>6</agentsWrapUpNotReady>
        <agentsWrapUpReady>7</agentsWrapUpReady>
      </statistics>
    </Queue>
  </data>
</Update>
```
**Sample Notification Payload (DELETE):**

```xml
<Update>
  <event>delete</event>
  <source>/finesse/api/Queue/1004</source>
  <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>2012-02-15T17:58:21Z</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>
  </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):

```xml
<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>2012-02-15T17:58:21Z</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>
  </data>
</Update>
```

### Sample Notification Payload (DELETE):

```xml
<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.

## Media Notification

Finesse sends a Media notification when information about a user in a Media Routing Domain changes.

<table>
<thead>
<tr>
<th>Format</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node:</td>
<td>/finesse/api/User/{id}/Media</td>
</tr>
<tr>
<td>Source:</td>
<td>/finesse/api/User/{id}/Media/{mrdId}</td>
</tr>
</tbody>
</table>
Media and Dialogs/Media Asynchronous Error Notification

If an operations performed on a Media or nonvoice Dialog results in an asynchronous error, the error notifications include the error type, error code, and error constant. The ErrorMedia parameter indicates the Media Routing Domain to which the error applies.

<table>
<thead>
<tr>
<th>Format:</th>
<th>XML</th>
</tr>
</thead>
</table>
| Node: | /finesse/api/User/{id}/Media  
/finesse/api/User/{id}/Dialogs/Media |
**Media and Dialogs/Media Error Code Descriptions**

**Errors for Agent State and Mode Changes**

**Common Agent State and Mode Change Errors**

This table describes common errors returned if agent state or mode changes fail.

---

### Source:

<table>
<thead>
<tr>
<th>/finesse/api/User/{id}/Media/{mrdId}</th>
</tr>
</thead>
<tbody>
<tr>
<td>/finesse/api/User/{id}/Media/{mrdId}/Dialogs (when a Dialog is added or removed from the Dialog collection for the user, for example offered or closed.)</td>
</tr>
<tr>
<td>/finesse/api/Dialog/{id} (when a Dialog within the Dialogs collection for the user is modified, for example accepted, started, paused, or wrapped up.)</td>
</tr>
</tbody>
</table>

### Data:

| Media |
| Dialog |

### Payload:

```xml
<Update>
  <data>
    <apiErrors>
      <apiError>
        <errorData>&lt;Error Code&gt;</errorData>
        <errorMedia>5001</errorMedia>
        <errorMessage>&lt;Error Constant&gt;</errorMessage>
        <errorType>&lt;Error Type&gt;</errorType>
      </apiError>
    </apiErrors>
  </data>
  <event>PUT</event>
  <requestId>xxx-xxxx</requestId>
  <source>/finesse/api/User/{id}/Media/{mrdId}</source>
</Update>
```

### Sample Notification Payload:

```xml
<Update>
  <data>
    <apiErrors>
      <apiError>
        <errorData>1</errorData>
        <errorMedia>5001</errorMedia>
        <errorMessage>E_ARM_STAT_AGENT_ALREADY_LOGGED_IN</errorMessage>
        <errorType>Agent already logged into MRD</errorType>
      </apiError>
    </apiErrors>
  </data>
  <event>PUT</event>
  <requestId>xxx-xxxx</requestId>
  <source>/finesse/api/User/1001001/Media/5001</source>
</Update>
```

### Notification Triggers:

The notification system returns this error if an operation on a Media or nonvoice Dialog results in an asynchronous error.
### Error Constants

<table>
<thead>
<tr>
<th>Error Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_ARM_STAT_AGENT_NOT_FOUND</td>
<td>2</td>
<td>The specified agent is not configured in CCE.</td>
</tr>
<tr>
<td>E_ARM_STAT_MRD_LIST_ENTRY_NOT_FOUND</td>
<td>3</td>
<td>The specified Media Routing Domain is not configured in CCE.</td>
</tr>
<tr>
<td>E_ARM_STAT_AGENT_NOT_LOGGED_IN</td>
<td>6</td>
<td>The specified agent is not logged into the MRD. This error is not returned when logging the agent into an MRD.</td>
</tr>
</tbody>
</table>

### Agent Login Errors

<table>
<thead>
<tr>
<th>Error Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_ARM_STAT_AGENT_ALREADY_LOGGED_IN</td>
<td>1</td>
<td>The specified agent is already logged in to this MRD.</td>
</tr>
<tr>
<td>E_ARM_STAT_CANT_LOGIN_TO_VOICE_MRD</td>
<td>11</td>
<td>The agent cannot log in to the voice MRD. The application attempted to log an agent into the voice MRD using the Media API instead of the User API.</td>
</tr>
<tr>
<td>E_ARM_STAT_LOGIN_NOT_ALLOWED_FOR_APP_PATH</td>
<td>27</td>
<td>The MRD and peripheral specified in the agent login request are not members of the application path associated with the Finesse server that sent the request.</td>
</tr>
<tr>
<td>E_ARM_STAT_PERFORMANCE_LIMIT_EXCEEDED</td>
<td>34</td>
<td>This code is used in the Packaged CCE deployment. When the PG reaches the Maximum Concurrent Number of Logged in Agents for that peripheral, all the ARMMediaLoginReqs for that Peripheral are rejected with this status code.</td>
</tr>
<tr>
<td>E_ARM_STAT_CC_OFFLINE</td>
<td>36</td>
<td>The log in request failed because the Central Controller is offline.</td>
</tr>
<tr>
<td>E_ARM_STAT_LOGIN_TIMEOUT</td>
<td>37</td>
<td>The log in request timed out.</td>
</tr>
<tr>
<td>E_ARM_STAT_PQ_LOGIN_FAILED</td>
<td>38</td>
<td>The agent log in request to the precision queue failed.</td>
</tr>
<tr>
<td>E_ARM_STAT_LOGIN_REQUEST_ALREADY_PENDING</td>
<td>41</td>
<td>There is already a pending request for the agent to log in to the Media Routing Domain.</td>
</tr>
</tbody>
</table>
### Agent Not Ready Errors

<table>
<thead>
<tr>
<th>Error Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_ARM_STAT_ALREADY_HAVE_PENDING_MAKE_AGENT_NOT_READY</td>
<td>9</td>
<td>There is already a pending request to make this agent Not Ready in this Media Routing Domain.</td>
</tr>
<tr>
<td>E_ARM_STAT_DO_THIS_WITH_TASK_SENT_RECENTLY</td>
<td>14</td>
<td>The agent cannot be made Not Ready because the agent has a pending incoming task; Finesse has received an offered dialog for the agent.</td>
</tr>
<tr>
<td>E_ARM_STAT_ALREADY_IN_REQUESTED_AGENT_STATE</td>
<td>39</td>
<td>The specified agent is already in the Not Ready state. If reason codes are enabled, then an agent state change from Not Ready to Not Ready with a different reason code is allowed.</td>
</tr>
</tbody>
</table>

### Agent Mode Change Errors

<table>
<thead>
<tr>
<th>Error Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_ARM_STAT_ALREADY_HAVE_PENDING_MAKE_AGENT_NOT_ROUTABLE</td>
<td>8</td>
<td>There is already a pending request to make this agent Not Routable in this Media Routing Domain.</td>
</tr>
<tr>
<td>E_ARM_STAT_ALREADY_IN_REQUESTED_AGENT_MODE</td>
<td>40</td>
<td>The agent is already in the requested mode.</td>
</tr>
</tbody>
</table>

### Internal Errors

If you receive these errors, Contact Cisco Technical Support for assistance.

<table>
<thead>
<tr>
<th>Error Constant</th>
<th>Error Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_ARM_STAT_NO_ACTIVE_SKILL_GROUPS_IN_MRD_LIST_ENTRY</td>
<td>5</td>
</tr>
</tbody>
</table>

### Errors for Dialogs

#### Common Dialog Errors

This table describes common errors returned if Dialog actions fail.

<table>
<thead>
<tr>
<th>Error Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_ARM_STAT_AGENT_NOT_FOUND</td>
<td>2</td>
<td>The specified agent is not configured in CCE.</td>
</tr>
<tr>
<td>E_ARM_STAT_MRD_LIST_ENTRY_NOT_FOUND</td>
<td>3</td>
<td>The specified Media Routing Domain is not configured in CCE.</td>
</tr>
<tr>
<td>E_ARM_STAT_AGENT_NOT_LOGGED_IN</td>
<td>6</td>
<td>The specified agent is not logged into the MRD.</td>
</tr>
</tbody>
</table>
Error Constant | Error Code | Description
--- | --- | ---
E_ARM_STAT_TASK_OBJECT_NOT_FOUND | 18 | The specified dialog cannot be found.
E_ARM_STAT_INCONSISTENT_MEDIA_ROUTING_DOMAIN_IDS | 20 | The Media Routing Domain ID does not match the MRD ID for this skill, service, or dialog.
E_ARM_STAT_NOT_VALID_AFTER_INTERRUPT_ADVISORY_ACCEPT | 30 | The dialog has been interrupted by a dialog in a different MRD. Typically, this code indicates that a voice call interrupted the agent working on a chat or an email.
INVALID_DIALOG_ID: <DIALOG ID> | 6030 | The dialog API request is made and the synchronous response received but the dialog is removed before contacting CCE.

Internal Errors

If you receive these errors, Contact Cisco Technical Support for assistance.

<table>
<thead>
<tr>
<th>Error Constant</th>
<th>Error Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_ARM_STAT_INVALID_MESSAGE_SEQUENCE</td>
<td>19</td>
</tr>
<tr>
<td>E_ARM_STAT_NO_OFFER_OR_PRE_CALL_RECEIVED</td>
<td>21</td>
</tr>
<tr>
<td>E_ARM_STAT_INCONSISTENT_AGENT_IDS</td>
<td>22</td>
</tr>
<tr>
<td>E_ARM_STAT_SKILL_GROUP_NOT_FOUND</td>
<td>32</td>
</tr>
<tr>
<td>E_ARM_STAT_SERVICE_NOT_FOUND</td>
<td>33</td>
</tr>
</tbody>
</table>

Notification Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Object</td>
<td>Provides the new representation of the modified User, Team, Dialog, Queue, User/Queues, or Media object. This information is not provided when a user is deleted. For a Dialog or Media Error notification, provides the list of ApiError objects that represent the failure conditions detected by the server.</td>
<td>The entire User, Team, Dialog, Queue, or Media 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.</td>
</tr>
<tr>
<td>Name</td>
<td>Data Type</td>
<td>Description</td>
<td>Possible Values</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Event      | String    | The type of modification that occurred to the User, Team, Dialog, Queue, User/Queues, or Media object. | **PUT:** A property of the User, Dialog, Team, Queue, or Media object that was modified.  
**DELETE:** A User, Dialog, Team, Queue, or Media object has been deleted. For a User/Queues modification, the queues removed from the user's list of queues.  
**POST:** A User, Dialog, Team, Queue, or Media 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, User/Queues, or Media object that was modified. | /finesse/api/User/{id}  
/finesse/api/Dialog/{id}  
/finesse/api/Team/{id}  
/finesse/api/User/{id}/Dialogs  
/finesse/api/User/{id}/Dialogs/Media  
/finesse/api/Queue/{id}  
/finesse/api/User/{id}/Queues  
/finesse/api/User/{id}/Media |
| 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 |

**Managing Notifications in Third-Party Applications**

For applications that are neither gadgets in the Cisco Finesse Desktop nor in a third-party OpenSocial container, you can use one of the following methods to establish a BOSH XMPP connection with the Cisco Finesse Notification Service:

- Any client-side XMPP library such as Jabberwerx
- Cisco Finesse Desktop EventTunnel (for browser applications only)

This section describes how to use the Cisco Finesse Desktop EventTunnel. This method requires knowledge of how to use postMessage to pass messages between different frames in the browser.
The EventTunnel.js file is located at http://<hostname>:<port>/tunnel/EventTunnel.js (where hostname is the hostname of the Cisco Finesse server and the port is either 7071 for HTTP or 7443 for HTTPS). This class is designed to be loaded within an iframe in the browser application and uses postMessage to communicate between frames.

Using the EventTunnel, the application can perform the following operations:

- Establish the BOSH connection
- Subscribe to XMPP nodes
- Unsubscribe from XMPP nodes

The following is a sample file you can use to instantiate and initialize the EventTunnel in the iframe:

```html
<!DOCTYPE HTML>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <script type="text/javascript">
        // Set the JabberWerx connect to unsecure because the custom authentication on the XMPP server does not support encrypted credentials.
        var jabberwerx_config = {unsecureAllowed: true};
    </script>
    <script type="text/javascript" src="thirdparty/jquery/jquery-1.5.min.js"></script>
    <script type="text/javascript" src="thirdparty/jabberwerx/jabberwerx.js"></script>
    <script type="text/javascript" src="thirdparty/util/converter.js"></script>
    <script type="text/javascript" src="EventTunnel.js"></script>
    <script type="text/javascript">
        jQuery(document).ready(function () {
            var tunnel = new finesse.EventTunnel();
            tunnel.init();
        });
    </script>
</head>
</html>
```

**Connect to XMPP over HTTP (BOSH) using Finesse EventTunnel**

To initialize the BOSH connection, the following information must be passed to the EventTunnel before it can proceed:

1. Agent ID
2. XMPP Domain
3. Agent Password
4. XMPP PubSub Domain
5. Agent XMPP Resource (Optional)

The postMessage payload has the following message structure:

```javascript
message = type + "|" + message;
```

where type is a number that has the following mapping:
### Connect to XMPP over TCP

Any third party XMPP client can connect to the Finesse Notification Service through TCP sockets for sending and receiving notifications. You can connect to ports 5222 (non-secure connection) and 5223 (secure connection).

---

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT</td>
<td>0</td>
<td>XMPP events received by the EventTunnel and published out to the parent frame</td>
</tr>
<tr>
<td>ID</td>
<td>1</td>
<td>Agent XMPP ID</td>
</tr>
<tr>
<td>PASSWORD</td>
<td>2</td>
<td>Agent XMPP password</td>
</tr>
<tr>
<td>RESOURCEID</td>
<td>3</td>
<td>Agent XMPP resource</td>
</tr>
<tr>
<td>STATUS</td>
<td>4</td>
<td>Status of the BOSH connection published by the EventTunnel</td>
</tr>
<tr>
<td>XMPPDOMAIN</td>
<td>5</td>
<td>Domain of the XMPP service</td>
</tr>
<tr>
<td>PUBSUBDOMAIN</td>
<td>6</td>
<td>PubSub domain of the XMPP service</td>
</tr>
<tr>
<td>SUBSCRIBE</td>
<td>7</td>
<td>Request to subscribe to an XMPP node</td>
</tr>
<tr>
<td>UNSUBSCRIBE</td>
<td>8</td>
<td>Request to unsubscribe form an XMPP node</td>
</tr>
<tr>
<td>PRESENCE</td>
<td>9</td>
<td>Request to subscribe to XMPP presence</td>
</tr>
<tr>
<td>DISCONNECT_REQ</td>
<td>11</td>
<td>Request to disconnect the BOSH connection. This request attempts to unsubscribe the application from all nodes to which it subscribed during the session and then disconnects the session.</td>
</tr>
</tbody>
</table>

For example, a postMessage call to send the agent ID is as follows:

```javascript
message = "1|1001001" // 1 - type: ID, 1001001 - agent ID
tunnelFrame.postMessage(message, tunnelOrigin); // where tunnelFrame is the frame corresponding to the iframe hosting the EventTunnel and tunnelOrigin is the URL of the EventTunnel i.e. http://<host>:<port> where host is the host of the Cisco Finesse server and port is the port of // the Cisco Finesse Notification Service, either 7071 for http or 7443 for https
```

Be sure to also wire up a callback to receive messages using postMessage from the EventTunnel frame, for example:

```javascript
if (window.addEventListener) { //Firefox, Opera, Chrome, Safari
    window.addEventListener("message", cb, false);
} else { //Internet Explorer
    window.attachEvent("onmessage", cb);
}
```

where cb is the callback that handles any messages received using postMessage and that can parse the messages sent by the EventTunnel.
Cisco Finesse, Release 12.5(1) onward, the 5222 port (non-secure connection) is disabled by default. Set the `utils finesse set_property webservices enableInsecureOpenfirePort` to `true` to enable this port.


---

**Connect to Secure Port 5223 over SSL/TLS**

Third party clients need to add the Finesse Notification certificate to their respective trust stores. Finesse Notification Service shares the same certificate with Cisco Finesse Tomcat. To download the certificate:

1. Sign in to the Cisco Unified Operating System Administration through the URL (https://FQDN:8443/cmplatform, where FQDN is the fully qualified domain name of the primary Finesse server and 8443 is the port number).
2. Click Security > Certificate Management.
3. Click **Find** to get the list of all the certificates.
4. In the Certificate List screen, choose **Certificate** from the **Find Certificate List where** drop-down menu, enter **tomcat** in the **begins with** option and click **Find**.
5. Click the FQDN link which appears in the Common Name column parallel to the listed tomcat certificate.
6. In the pop-up that appears, click the option **Download .PEM File** to save the file on your desktop.
Connect to XMPP over TCP
Finesse High Availability

Availability of a Finesseserver 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 Finesses Notification Service:

   In a Unified CCX deployment, this service is called the Unified CCX Notification Service.

   An active BOSH connection to the Cisco Finesses 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 'finesses' BOSH user:
   Presence indicates whether Finesses has an active connection to the Cisco Finesses Notification Service (Unified CCE) or the Cisco Unified CCX Notification Service (Unified CCX). An UNAVAILABLE presence for the 'finesses' BOSH user may mean that the connection is lost or that the Finesses web app crashed.

A Finesses 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 'finesses' BOSH user is AVAILABLE.

Ensure 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 'finesses' BOSH user cannot be checked without a BOSH connection. A BOSH connection is not useful if the server is OUT_OF_SERVICE.

• Failure Scenarios, on page 318
• Desktop Presence and Forced Logout, on page 318
• Failure Handling for Task Routing Clients, on page 320
Failure Scenarios

The following table lists possible failure scenarios and describes how a client can determine when a failure occurs.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Notification mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco Finesse Notification Service goes down.</td>
<td>Client loses BOSH connection to the Cisco Finesse Notification Service.</td>
</tr>
<tr>
<td><strong>Note</strong> Cisco Unified CCX</td>
<td>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).</td>
</tr>
<tr>
<td>deployment, this service is called the Cisco Unified CCX Notification Service.</td>
<td></td>
</tr>
<tr>
<td>Cisco Finesse Tomcat goes down.</td>
<td>The 'finesse' user presence becomes UNAVAILABLE (if BOSH is still connected to the Cisco Finesse Notification Service).</td>
</tr>
<tr>
<td>Finesses web app goes down.</td>
<td>The 'finesse' user presence becomes UNAVAILABLE (if BOSH is still connected to the Cisco Finesse Notification Service).</td>
</tr>
<tr>
<td>Finesses loses connection to the CTI server.</td>
<td>Finesses sends a SystemInfo notification of status OUT_OF_SERVICE (if BOSH is still connected to the Cisco Finesses Notification Service).</td>
</tr>
</tbody>
</table>

Recovery

When any of the preceding failure scenarios are detected, the 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 Finesses server subscribes to the presence of the XMPP users of the Finesses desktop to monitor the health of the connection between the server and desktop.

Under certain conditions, Finesses 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:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Desktop Behavior</th>
<th>Server Action</th>
<th>Race Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>The client closes, the browser</td>
<td>When you close the browser or navigate away from the</td>
<td>Finesse receives a presence notification of <em>Unavailable</em> from the client.</td>
<td>1. The agent closes the browser window. Finesse receives a presence notification of <em>Unavailable</em> for the user. Finesse tries to sign the agent out; however, that agent is already signed out.</td>
</tr>
<tr>
<td>crashes, or the agent clicks the</td>
<td>Finesse desktop makes a best-effort attempt to notify</td>
<td>Finesse waits 60 seconds, and then sends a forced logout request to the</td>
<td>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.</td>
</tr>
<tr>
<td>Back button on the browser.</td>
<td>the server.</td>
<td>the CTI server.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. If the Finesse desktop is running over a slower network connection, Finesse may not always receive an <em>Unavailable</em> presence notification from the client browser. In this situation, the behavior mimics a browser crash, as described in the preceding condition.</td>
</tr>
<tr>
<td>The client refreshes the browser</td>
<td>Finesse receives a presence notification of <em>Unavailable</em> 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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scenario</td>
<td>Description</td>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>The client encounters a network glitch (Finesse is in service)</td>
<td>Because the connection to the Finesse server temporarily goes down, the client fails over to the secondary Finesse server.</td>
<td>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 unavailable from the client, the primary Finesse server sends the forced logout request to the CTI server for the agent.</td>
<td></td>
</tr>
<tr>
<td>In a Unified CCE deployment, when Refresh Token has expired</td>
<td>The primary Finesse server forwards the forced logout request to the CTI server.</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

### Load parameter = 0
- When the agent's current state is Not Ready, Ready or Wrap-Up, the agent's state after force logout is changed to Not Ready – Force Not Ready.
- When the agent's current state is Talking, the Agent goes into Not-Ready – Force Not Ready state after the call ends.

### Load parameter = 1
- When the agent's current state is Not Ready, Ready or Wrap-Up, the agent goes to Logged Out – System Failure.
- When the agent's current state is Talking, the Agent goes to Logged Out – System Failure immediately even though the call is still active.

---

## Failure Handling for Task Routing Clients

Task Routing applications that use the Finesse APIs must be able to handle failure scenarios involving Finesse and CCE services.

To recover REST and XMPP/BOSH connections, follow the steps described for failure recovery earlier in this chapter.

Once you recover the connections, perform more actions to recover nonvoice media state and nonvoice dialogs. The actions you perform depend on whether your application is built with the Finesse REST APIs or the finesse.js javascript library.

### Recovery Actions for Finesse REST APIs

If your application is built with Finesse REST APIs, perform these actions to recover nonvoice media state and nonvoice dialogs:
• Use the Media GET API to synchronize your application with the state of the agent in the application's media. For example:
https://finesse_server/finesse/api/User/userId/Media/mediaId.

• If the maxDialogLimit, interruptAction, or dialogLogoutAction settings do not match the settings set by your application at sign-in time, use the Media Sign In API to reset the settings. The Sign In API returns an "agent already logged in" error. This error is expected. The API call does not affect the agent's state in the media. The call does, however, reset the agent's maxDialogLimit, interruptAction, and dialogLogoutAction settings in the media.

• Use the nonvoice Dialog LIST method to synchronize the application with the set of dialogs that the agent currently is assigned. For example:
https://finesse_server/finesse/api/User/userId/Media/mediaId/Dialogs.
Typically, the set of dialogs does not change when you use this command. However, in some failure cases, such as double PG failures, the set of dialogs changes when you use this method.

Recovery Actions for Finesse.js Javascript Library

Media settings (maxDialogLimit, interruptAction, and dialogLogoutAction) can become out of sync after a failure.

If your application is built with finesse.js, when getting the media object for the application, tell the media object the media options. The finesse.js library uses these settings to ensure that the media object associated with your application's agent has the correct settings after recovering from a failure.

For example:

```javascript
media = _mediaList.getMedia({
  id: mrdID,
  onLoad: handleMediaLoad,
  onError: handleMediaError,
  onChange: handleMediaChange,
  mediaOptions: {
    maxDialogLimit: 3,
    interruptAction: "IGNORE",
    dialogLogoutAction: "CLOSE"
  }
});
```

Related Topics

- Failure Scenarios, on page 318
- Media - Sign in, on page 147
- Media—Get Media, on page 154
- User—Get List of Dialogs (Nonvoice Only), on page 32
Finesse Desktop Gadget Development

- Finesse Gadgets, on page 323
- Supported OpenSocial Features, on page 328
- Gadget Caching, on page 332
- Notifications on Finesse Desktop, on page 333
- Finesse Notifications in Third-Party Containers, on page 333
- Finesse Topics, on page 333
- Finesse Container Timer, on page 339
- Handling Special Characters in CSS, on page 341
- Subscription Management on Finesse Desktop, on page 342

Finesse Gadgets

Gadgets are web-based software components based on HTML, CSS, and JavaScript. They allow developers to write web applications that work anywhere on the web without modification. They are defined using a declarative XML syntax that is processed by a gadget server into a format that allows them to be embedded into the following contexts:

- standalone web pages
- web applications
- other gadgets

Note

Do not use the following JavaScript methods as they block the Finesse agent desktop until the pop up is dismissed. The Finesse backend process can also be interrupted by these methods which may lead to unexpected behavior.

- window.alert()
- window.prompt()
- window.confirm()
- window.showModalDialog()
Prerequisites to Develop Gadgets

For Finesse Gadget development, a basic understanding of the following is necessary:

- How web applications work
- XML
- HTML
- JavaScript

Gadget Description

The gadgets API consists of simple building blocks:

XML: is a general purpose markup language. It describes structured data in a way that both humans and computers can read and write.

XML is the language used to write gadget specifications. A gadget is an XML file, placed on the internet where Google can find it. The XML file that specifies a gadget contains instructions on how to process and render the gadget. The XML file contains all data and code for the gadget, or it can have references (URLs) on where to find the rest of the elements.

HTML: is the markup language used to format pages on the internet. The static content of a gadget is written in HTML. HTML looks similar to XML, but is used to format web documents rather than to describe structured data.

JavaScript: is a scripting language used to add dynamic behavior to your gadgets.

Gadget XML

A gadget and its XML are synonymous. The gadget XML contains all information needed to identify and render a web application. The XML gadget specification consists of the following:

Content

The <Content> section specifies the programming logic and the HTML elements that determine the appearance of the gadget. It defines the type of content, and either holds the content itself or has a link to external content. The gadget attributes and user preferences are combined with programming logic and formatting information to become a running gadget.

<Content> provides the actual HTML, CSS, and JavaScript to be rendered by the gadget. Code is provided directly in the gadget XML content section for rendering and control flow. The code is processed by a gadget server and rendered in an IFRAME.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Module>
  <ModulePrefs title="Sample Gadget"/>
  ...
</ModulePrefs>
  <UserPref name="scheme" display_name="scheme" default_value="" datatype="hidden"/>
  <UserPref name="host" display_name="host" default_value="" datatype="hidden"/>
  <UserPref name="hostPort" display_name="hostPort" default_value="" datatype="hidden"/>

  <Content type="html">
    <![CDATA[
      <!DOCTYPE html>
      ...
    ]]>DOCTYPE html>
User Preferences

The `<UserPrefs>` section allows you to pass custom properties to the gadget from the gadget XML. The custom properties have to be suffixed with the datatype attribute as hidden.

For example, `<UserPref name="myname" display_name="Name" required="true" datatype="hidden" />

The user preferences are defined in the XML specifications as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Module>
  <ModulePrefs title="Sample Gadget" ...
  <UserPref name="scheme" display_name="scheme" default_value="" datatype="hidden"/>
  <UserPref name="host" display_name="host" default_value="" datatype="hidden"/>
  <UserPref name="hostPort" display_name="hostPort" default_value="" datatype="hidden"/>
</Module>
```

Note that for each User Preference, “hangman variables” can be substituted into supported gadget specification fields. Hangman variables are of the form `__<TYPE>_<ID>__`, and are replaced with string values. For each provided User Pref with key foo and value bar, hangman expansion `__UP_foo__ = bar`. Hence, in the above code user preference scheme is available as `__UP_scheme__`. Similarly, for other User Preferences the hangman variables are dynamically substituted. Also, as the datatype value is specified as hidden, the user preferences pop up for the agent to enter their own data does not show up on the gadget.

User preferences are accessed from your gadget using the user preferences JavaScript API, for example:

```javascript
var prefs = new gadgets.Prefs();
var someStringPref = prefs.getString("StringPrefName");
var someIntPref = prefs.getInt("IntPrefName");
var someBoolPref = prefs.getBool("BoolPrefName");
```
Gadget JavaScript

Contains the business logic for the gadget. It can be written inside the gadget XML under the content section or an external JavaScript file can be created which can then be referred to using the src attribute in the `<script>` tag.

Gadget CSS

Contains the complete styling of the gadget. Similar to the JavaScript, CSS can also be referred to as an external file using href attribute in `<link>` tag.

Gadget Behavior

Rendering a gadget at the page level removes the title bar from the gadget layout.

Simple Example Gadget

Do the following to create and deploy a gadget:

- Use any text editor to write your gadget specification.
- Host the gadget on any web server. See Enable or Reset 3rdpartygadget Account, on page 343.
- Add the gadget to the Finesse Container which can run gadgets. See Upload Third-Party Gadgets, on page 344.

Example Gadget

Use the following lines of code to build a simple gadget. This gadget displays the message "Hello, world!". Copy the following lines of code into a new file named hello_world.xml:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs title="hello world example" />
  <Content type="html">
    <![CDATA[
      Hello, world!
    ]]>  
  </Content>
</Module>
```

Note the following about the "Hello World" example:

- Gadgets are specified in XML. The first line is the standard way to start an XML file. This must be the first line in the file.
- The `<Module>` tag indicates that this XML file contains a gadget.
- The `<ModulePrefs>` tag contains information about the gadget such as its title, description, author, and other optional features.
- The line `<Content type="html">` indicates that the gadget's content type is HTML.
- `<![CDATA[ ...insert HTML here... ]]>` is used to enclose HTML when a gadget's content type is html. It tells the gadget parser that the text within the CDATA section should not be treated as XML. The CDATA section typically contains HTML and JavaScript.
- `</Content>` signifies the end of the Content section.
• </Module> signifies the end of the gadget definition.

For a Finessespecific example, download the LearningSampleGadget from https://github.com/CiscoDevNet/finesse-sample-code/tree/master/LearningSampleGadget, which provides step by step instructions in learning some of the objects in the finesse.js library.

Portions of this page are reproduced from work created and shared by Google, see https://developers.google.com/terms/site-policies and used according to terms described in the Creative Commons 3.0 Attribution License, see https://creativecommons.org/licenses/by/3.0/. For more information about OpenSocial gadgets, see https://developers.google.com/gadgets/docs/overview. Note that not all OpenSocial gadget features are available in the Finesse container.

**Import Finesses JavaScript API**

For gadgets to work properly, they need to import the Finesses JavaScript library hosted on the Finesseserver.

**Hosting Third-Party Gadgets on Web Server**

To import the JavaScript library, the Finesses FQDN needs to be provided inside the import statement. For building the finesse.js URL, we need to retrieve the following properties from the gadget preferences:

1. **scheme**: http or https
2. **hostname**: FQDN of the Finesseserver
3. **port**: port of the Finesseservice

These properties are inside the gadget preferences as part of Finesses container initialization. In your gadget XML:

• Define the user preferences that will be used for building the finesse.js import statement.

```xml
<UserPref name="scheme" display_name="scheme" default_value="" datatype="hidden"/>
<UserPref name="host" display_name="host" default_value="" datatype="hidden"/>
<UserPref name="hostPort" display_name="hostPort" default_value="" datatype="hidden"/>
```

• Import the finesse.js file.

```html
<script type="text/javascript"
    src="__UP_scheme__://__UP_host__:__UP_hostPort__/desktop/assets/js/finesse.js">
</script>
```

**Hosting Third-Party Gadgets on Finesseserver**

Third-party gadgets can be hosted on the Finesseserver inside the 3rdpartygadget directory. See Upload Third-Party Gadgets, on page 344.

Since the third-party gadget is hosted on the Finesseserver, you can import the Finesses JavaScript API with a relative URL.
alternateHosts Configuration

The `<gadget>` element in the Finesse Layout XML provides an attribute to specify alternate hosts from which the gadget can be loaded. This allows the Cisco Finesse desktop to load the gadget using a different host if the primary server is unavailable.

The `alternateHosts` attribute contains a comma-separated list of FQDNs that will be used if the primary-host-FQDN is unavailable.

```xml
<gadget alternateHosts="host1,host2,host3,...">
  https://<primary-host-FQDN>/<gadget-URL>
</gadget>
```

The `alternateHosts` attribute is only applicable for gadgets with an absolute URL. That is URLs containing the FQDN of a host, an optional port, and the complete URL path to the gadget. For example: `<gadget alternateHosts="host1,host2">http://primary host/relative_path</gadget>`

If loading the gadget from the primary-host fails, the Cisco Finesse container attempts to load the gadget from the alternate hosts in the order specified in the `alternateHosts` attribute.

It is possible that under certain circumstances, the Cisco Finesse desktop fails to load the gadget even if some of the hosts are reachable. In such cases, refresh the Cisco Finesse desktop.

When the gadget is specified with a relative URL, for example: `<gadget>/3rdpartygadgets/relative_path</gadget>`, the `alternateHosts` attribute does not apply and are ignored by the Cisco Finesse desktop.

---

**Note**

If the host serving the gadget fails after the Cisco Finesse desktop was successfully loaded, the desktop must be refreshed in order to load the gadget from an alternate host. The gadget does not implement its own failover mechanism.

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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locale</td>
<td>The <code>&lt;Locale&gt;</code> 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.</td>
</tr>
</tbody>
</table>
The Scroll 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, use the gadgets.window.adjustHeight API to dynamically resize the gadget as needed instead of using this feature.

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.

Displays a loading message while the gadget is loading.

**Required Module pref Feature**

Finesse requires that all gadgets use the following module pref feature:

```xml
<Require feature="pubsub-2"/>
```

This feature is required for the gadget to load in the OpenAjax Hub.

Before you can access the authorization string through the gadget prefs, you must first import the Finesse JavaScript library.

**Loading Indicator Feature**

The loading indicator is an OpenSocial feature that displays a loading message over gadgets while they are loading. This feature allows you to provide a consistent user experience within Finesse.

**Requesting the Loading Indicator**

Use the following to request the loading indicator in the gadget ModulePrefs:

```xml
<ModulePrefs>
  <Require feature="loadingindicator">
    <Param name="manual-dismiss">false</Param>
    <Param name="loading-timeout">10</Param>
  </Require>
</ModulePrefs>
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Possible</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>loading-timeout</td>
<td>Integer</td>
<td>The number of seconds to wait before displaying the Retry button. If the loading indicator is dismissed within this time, the Retry button does not appear. Set this to a number that is appropriate for your gadget.</td>
<td>integers</td>
<td>Optional parameter. Default is 10.</td>
</tr>
</tbody>
</table>
### APIs Available to Gadget JavaScript

The following table lists the available APIs and methods.

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;static&gt;</code> gadgets.window.adjustHeight(opt_height)</td>
<td>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.</td>
<td>Adjusts the height of the gadget.</td>
</tr>
<tr>
<td><code>&lt;static&gt;</code> gadgets.window.setTitle(title)</td>
<td>title (string)—Preferred title of the gadget.</td>
<td>Sets the title of the gadget.</td>
</tr>
<tr>
<td>Name</td>
<td>Parameters</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| `<static> gadget.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. |
| `<static> gadget.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. |
| `<static> gadget.loadingindicator.dismiss()` | None | Dismisses the loading indicator so that the message is no longer visible. |
| `<static> gadget.loadingindicator.showLoading()` | None | Displays a loading indicator message over the gadget. |
| `<static> gadget.loadingindicator.showRetry()` | None | Displays an error message over the gadget stating that the gadget failed to load, along with a Retry button. When the user clicks the Retry button, the container reloads the gadget. |
| `<static> gadget.loadingindicator.updateLoadingMessage (text)` | text (string)—Text to display as the loading message. | Changes the message that appears when the gadget is loading. |
| `<static> gadget.loadingindicator.updateTimeoutMessage (text)` | text (string)—Text to display when the gadget loading has timed out. | Changes the message that appears when the gadget loading times out. |
| `<static> gadget.loadingindicator.updateRetryButtonText (text)` | text (string)—Text to display on the Retry button. | Changes the message that appears on the Retry button. |

## 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.

```java
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.
```javascript
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:

```javascript
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 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 Finesse Tomcat to clear the cache.

If you make changes to the code of an existing gadget, you can restart Cisco Finesse 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. OpenSocial Core Gadget Specification 1.1 supports an intergadget notification system that is based on the OpenAjax Hub 2.0 Specification.

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

If the BOSH connection is disconnected, the Finesse desktop attempts to recover based on the recovery strategy described in Finesse High Availability, on page 317. If the BOSH connection cannot be re-established, the Finesse Desktop triggers a failover to the alternate Finesse server.

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

<table>
<thead>
<tr>
<th>Topic Name</th>
<th>finesse.info.connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic Type</td>
<td>Gadgets subscribe to this topic.</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Topic Name</th>
<th>finesse.api.[resourceObject].[resourceID]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic Type</td>
<td>Gadgets subscribe to this topic.</td>
</tr>
</tbody>
</table>

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 Finesses notifications, see Cisco Finesse Notifications and look under the "node" property. These notifications are structured as follows:

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

**Sample Notification Payload**

```javascript
{
    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`.

```javascript
{
    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**

<table>
<thead>
<tr>
<th>Topic Name</th>
<th>finesse.info.requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic Type</td>
<td>Gadgets publish to this topic.</td>
</tr>
</tbody>
</table>

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:

```javascript
{
    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.

---

### 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:

```json
{
  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:

```json
{
  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:

```json
{
    type: "SubscribeNodeReq",
    data: {
        node: "/finesse/api/Team/{id}/Users" // the node of interest
    },
    invokeID: "xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxxxxx"
}
```

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:

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

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

<table>
<thead>
<tr>
<th>Topic Name</th>
<th>finesse.info.responses.{invokeID}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic Type</td>
<td>Gadgets subscribe to this topic.</td>
</tr>
</tbody>
</table>
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:

```javascript
// 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**

<table>
<thead>
<tr>
<th>Topic Name</th>
<th>finesse.containerservices.workflowActionEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic Type</td>
<td>Gadgets subscribe to this topic.</td>
</tr>
</tbody>
</table>

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:

```javascript
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:

```javascript
{
  uri: string,
nname: string,
type: string,
params: [
```
{  
    name: string,  
    value: string,  
    expandedValue: string  
},  
actionVariables: [  
    {  
        name: string,  
        node: string,  
        type: string,  
        testValue: string,  
        actualValue: string  
    }  
]

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>In the uri, the id maps to the primary key of the WorkflowAction entry.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the workflow action.</td>
</tr>
<tr>
<td>type</td>
<td>The type of workflow action. Possible value is BROWSER_POP.</td>
</tr>
<tr>
<td>params</td>
<td>List of Param subobjects (see below).</td>
</tr>
<tr>
<td>actionVariables</td>
<td>List of ActionVariable subobjects (see below). There can be at most 5 Action Variable subobjects assigned to a workflow action.</td>
</tr>
</tbody>
</table>

The Param subobject uses the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the parameter.</td>
</tr>
<tr>
<td>value</td>
<td>The value of the parameter.</td>
</tr>
<tr>
<td>expandedValue</td>
<td>The value of the parameter with variables substituted with their values.</td>
</tr>
</tbody>
</table>

The ActionVariable subobject uses the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the variable.</td>
</tr>
<tr>
<td>node</td>
<td>The XPath to extract from the dialogs XML.</td>
</tr>
<tr>
<td>type</td>
<td>Indicates if this is a SYSTEM or CUSTOM variable.</td>
</tr>
<tr>
<td>testValue</td>
<td>The value used to test the variable.</td>
</tr>
<tr>
<td>actualValue</td>
<td>The actual value of the variable in context of the events used by the workflow evaluation.</td>
</tr>
</tbody>
</table>

**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:
  ```javascript
  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:**

```javascript
//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);
    }
  }

  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.

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.

**Handling Special Characters in CSS**

When using CSS in a gadget, the Finesse Desktop Gadget Container restricts the following special characters:

@ ^ $ * :: ~

If the CSS contains any of the special characters listed above, copy the following JavaScript code into your gadget’s *.js file:

```javascript
/**
 * Injects css or js files into DOM dynamically.
 * This is to bypass gadget container's restriction for special chars in CSS 3 files.
 * E.g. @Keyframes
 */
injectResource : function (url){
  var node = null;
  // url null? do nothing
  if(!url) {
    return;
  }
  // creates script node for .js files
  else if(url.lastIndexOf('.js')=== url.length-3){
    node = document.createElement("script");
    node.async = false;
    node.setAttribute('src', url);
  }
  // creates link node for css files
  else if(url.lastIndexOf('.css')== url.length-4){
    node = document.createElement("link");
    node.setAttribute('href', url);
    node.setAttribute('rel', 'stylesheet');
  }
  // inserts the node into dom
  if(node) {
    document.getElementsByTagName('head')[0].appendChild(node);
  }
}
```
In your gadget’s *.xml file, call the injectResource function that you have copied above. The parameter to the injectResource function is the path to your css file:

```javascript
<script type="text/javascript">
    <your gadget namespace>.injectResource('<path to CSS file>/<CSS filename>.css');
</script>
```

**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.
Third-Party Gadgets

Cisco Finesse provides a mechanism for you to upload third-party gadgets to the Finess server. This mechanism allows one user in the Finess 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).

- Enable or Reset 3rdpartygadget Account, on page 343
- CSS Requirements, on page 344
- Upload Third-Party Gadgets, on page 344
- Permissions, on page 346
- Replication, on page 346
- Migration, on page 347
- Backup and Restore, on page 347
- Restrictions, on page 347
- CORS Support for Finesse REST API, on page 347

Enable or Reset 3rdpartygadget Account

Use the following CLI command to enable (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

You must enable or reset the password for the 3rdpartygadget account on install. The password must be between 5 and 32 characters long and must not contain spaces or double quotes (").
CSS Requirements

By default, Finesse rewrites the linked CSS in your gadget, which in some cases is not desirable as it results in a loss of functionality if the CSS you are loading refers to other asynchronous elements. As a result, for all third-party gadgets, you can bypass the content rewriting for CSS by including the following in your gadget XML:

1. Add the optional feature "content-rewrite" to disable the CSS rewrite:

   ```xml
   <Optional feature="content-rewrite">
     <Param name="expires">86400</Param>
     <Param name="include-url">.*</Param>
     <Param name="exclude-url">.css</Param>
   </Optional>
   ```

2. Include UserPref for "externalServerHost":

   ```xml
   <UserPref name="externalServerHost"/>
   ```

3. To reference the CSS file, perform one of the following:

   • If the gadget is hosted on the Finesseserver, reference the CSS file using externalServerHost:

   ```html
   <link rel="stylesheet" href="__UP_internalServerHost__/3rdpartygadget/files/<yourgadgetname>/<path to CSS file>/<CSS filename>.css" type="text/css" />
   ```

   where you must update `<yourgadgetname>` to the filename of your gadget under the 3rdpartygadget /files folder and update the remaining path variables to the location of the CSS file for your gadget.

   • If the gadget is hosted on a server external to Finesses, reference the CSS file using the URL:

   ```html
   <link rel="stylesheet" href="[http://|https://]<hostname>/<path to CSS file>/<CSS filename>.css" type="text/css" />
   ```

   where you must update the URL variables to the location of the CSS file on your external server, and where specifying the protocol (http or https) is optional. (If you omit the protocol, Finesses uses the default protocol of the page.)

---

**Note**

Finesse Desktop Gadget Container restrains special characters while loading a CSS3 file. See Handling Special Characters in CSS, on page 341

---

Upload Third-Party Gadgets

After you set the password for the 3rdpartygadget account, you can use SFTP to upload third-party gadgets to the Finesseserver, as illustrated in the following example.
Finesse allows you to upload third-party gadgets to your own web server, however, you must ensure that the Finesse server has access to your web server.

Note

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
sftp> exit

After you upload a gadget, it is available under the following URL:

http://<finesse>/3rdpartygadget/files/

Note

For Unified CCX deployments you must specify port 8082.

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. For more information on adding third party gadgets to the Finesse desktop layout, see the section Manage Desktop Layout in the Cisco Finesse Administration Guide.

To include the gadget that was uploaded in the previous example in the desktop layout, add the following XML (highlighted) to the layout:

```xml
<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>
```
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 Finesse 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.

CORS Support for Finesse REST API

If third-party web applications instantiated from a third-party web server need to make calls to Finesse Desktop APIs, they require Cross-Origin Resource Sharing (CORS) support. In order to enable this support, Finesse expects them to send a specific header that contains the Origin Host name.

Header name: Origin.

The Host name value in Origin is used by Finesse to populate the Response Header named Access-Control-Allow-Origin.

Note

When responding to a credentialed request, Finesse server must specify a domain and cannot use wild card characters, else the request will fail.
CORS Support for Finesse REST API
Log Collection

These commands prompt you to specify a secure FTP (SFTP) server location to which the files will be uploaded.

To obtain logs:

- **Install log**: `file get install desktop-install.log`
  
  Use this command to see the installation log after the system is installed.

  This log is written to the SFTP server and stored as a text file written to this path: `<IP Address>\<date time stamp>\install\desktop-install.log`

- **Desktop logs**: `file get activelog desktop recurs compress`
  
  Use this command to obtain logs for the Finesse web applications. This command uploads a zip file that contains the following directories:

  - **webservices**: contains the logs for the Finesse backend that serves the Finesse REST APIs. The maximum size of an uncompressed desktop log file is 100 MB. The maximum size of this directory is approximately 4.5 GB. After a log file reaches 100 MB, that file is compressed and a new log file is generated. Output to the last compressed desktop log file wraps to the log file created next. The log file wrap-up duration can vary, based on the number of users on the system. Timestamps are placed in the file name of each desktop log.

  - **desktop**: contains logs from the Finesse agent desktop gadget container that holds the Finesse desktop gadgets. Any container-level errors with Finesse agent desktop will appear in these log files.

  - **admin**: contains logs from the Finesse administration gadget container that holds the administration gadgets. Any container-level errors with the Finesse administration console appear in these log files.

    - **audit-log**: Audit logs contain all admin operations (including Finesse admin UI and REST client operations). The maximum size of an uncompressed audit log file is 100 MB. The maximum size of total audit log files (including compressed log files) is approximately 1 GB. After a log file reaches 100 MB, that file is compressed and a new log file is generated. The log file wrap-up duration can vary, based on the number of users on the system. The log contains the following parameters:
• Timestamp
• User Id of the administrator
• Method of operation (PUT, POST, DELETE). GET operations will not be logged
• URL
• Payload

• **clientlogs:** contains the client-side logs submitted from the Finesse agent desktop to the Finesse server. Each log file is no larger than 1.5 MB and contains a timestamp and the agent ID of the agent who submitted the file. A new log file is created each time an agent submits client-side logs (the data is not appended to an existing log file). The maximum size of this directory is 100 MB. The directory holds a maximum number of 25000 clientlog files. When the directory exceeds the size limit or the file count, the oldest files are deleted.

• **openfireservice:** contains startup and shutdown-related information logs for the Cisco Finesse Notification Service.

• **openfire:** contains limited error and information logs for the Cisco Finesse Notification Service.

• **realm:** contains the logs for authentication requests from clients that are handled by the Finesse backend.

• **db:** contains the logs pertaining to the Finesse database.

• **/finesse/logs:** contains the logs for the Cisco Finesse Tomcat service.

• **fippa:** contains logs for the Finesse IP Phone Agent (IPPA) application.

• **finesse-auth:** contains the logs for Finesse authentication with the Cisco Context Service.

• **jmx:** contains the JMX counters data generated by the JMX logger process. It contains important jmx counters exposed by Finesse and openfire.

These logs are stored to the following path on the SFTP server: `<IP address>/<date time stamp>/active_nnn.tgz`, where nnn is timestamp in long format.

• Context Service registration log: **file get activelog ccbu/logs/fusion-mgmt-connector**

Use this command to obtain the fusion-mgmt-connector logs generated by Finesse during the registration and deregistration with Cisco Context Service.

These logs are stored to the following path on the SFTP server: `<IP address>/<date time stamp>/active_nnn.tgz`, where nnn is timestamp in long format.

• Servm log: **file get activelog platform/log/servm*. * compress**

Use this command to obtain logs generated by the platform service manager that manages the starting and stopping of the Finesse services.

The desktop and servm logs are compressed to one set of files.

These logs are stored to the following path on the SFTP server: `<IP address>/<date time stamp>/active_nnn.tgz`, where nnn is timestamp in long format.

• Platform Tomcat logs: **file get activelog tomcat/logs recurs compress**
These logs are stored to the following path on the SFTP server: `<IP address>/<date time stamp>/active_nnn.tgz`, where nnn is timestamp in long format.

- **Install log**: `file get install install.log`

  These logs are stored to the following path on the SFTP server: `<IP address>/<date time stamp>/active_nnn.tgz`, where nnn is timestamp in long format.

---

**Note**

Log collection may fail when you use the compress flag if there are a lot of log files. If collection fails, run the command again without the compress flag.

---
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/finesse/

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/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/finesse/

Documentation Feedback

You can provide comments about this document by sending email to the following address:

contactcenterproducts_docfeedback@cisco.com

We appreciate your comments.