Finesse Agent Login Trace with the Use of Logs

Document ID: 116255

Contributed by Linda Mordosky, Cisco TAC Engineer.
Jul 22, 2013

Contents

Introduction

This document describes the process involved in an agent login through the Finesse system with the log files. It is important to understand the message flow between the different Finesse Components, the Computer Telephony Integration (CTI) Server, and the client desktop so that you can successfully troubleshoot issues.

Prerequisites

Requirements

Cisco recommends that you have knowledge of Cisco Finesse and the Voice Operating System (VOS) CLI command prompt.

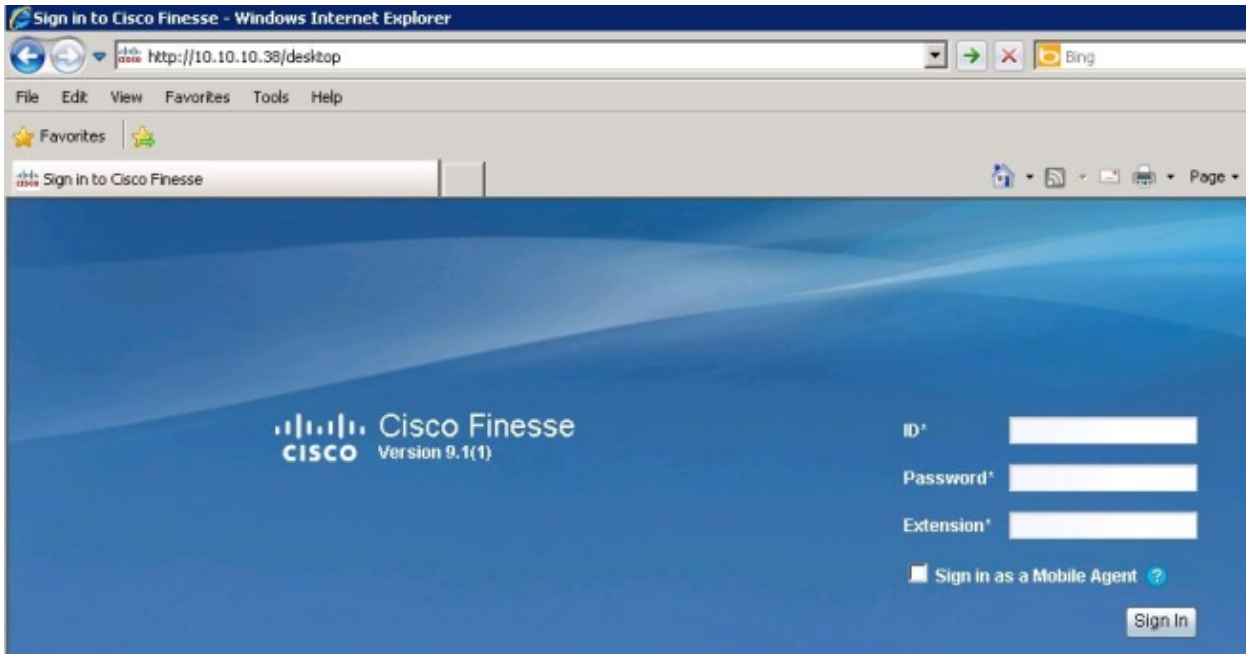
Components Used

The information in this document is based on Cisco Finesse Version 9.1(1).

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Launch the Agent Desktop

In order to launch the agent desktop, copy this URL into the web browser: ***http://<your finesse server>/desktop***. In Finesse Version 9.1, HTTP or HTTPS is supported.



Finesse uses Tomcat as the web server. When you launch your web browser, the request is made to Finesse to present the agent desktop to you. The Cisco Tomcat ***localhose_access_log*** command shows the request to load the agent desktop.

```
10.10.10.211 10.10.10.211 - - 80 GET /  HTTP/1.1 302 - 141
10.10.10.211 10.10.10.211 - - 80 GET /desktop/container/  HTTP/1.1 200 4541 185
10.10.10.211 10.10.10.211 - - 80 GET /desktop/theme/finesse/css/base.css
HTTP/1.1 200 3093 7
10.10.10.211 10.10.10.211 - - 80 GET /desktop/theme/finesse/css/login.css
HTTP/1.1 200 2185 0
10.10.10.211 10.10.10.211 - - 80 GET /desktop/js/Logon.js  HTTP/1.1 200 1745 1
10.10.10.211 10.10.10.211 - - 80 GET /desktop/js/utilities/Cookies.js  HTTP/1.1
200 2390 0
10.10.10.211 10.10.10.211 - - 80 GET /desktop/thirdparty/jquery/js/jquery.tools.
min.js  HTTP/1.1 200 15699 1
10.10.10.211 10.10.10.211 - - 80 GET /desktop/thirdparty/jquery/js/jquery-1.5.
min.js HTTP/1.1 200 84523 7
10.10.10.211 10.10.10.211 - - 80 GET /desktop/theme/finesse/images/sprite_
buttons.png HTTP/1.1 200 3297 0
10.10.10.211 10.10.10.211 - - 80 GET /desktop/theme/finesse/images/help.png
HTTP/1.1 200 830 0
10.10.10.211 10.10.10.211 - - 80 GET /desktop/theme/finesse/images/cisco_logo.
png HTTP/1.1 200 760 0 200 2205 1
10.10.10.211 10.10.10.211 - - 80 GET /desktop/theme/finesse/images/bg.jpg
HTTP/ 1.1 200 32222 4
```

Agent Login Credentials

Now that the agent desktop has been presented, you enter your login credentials. Before Finesse can send the login request to the CTI Server, the client needs to establish a Bidirectional−streams Over Synchronous HTTP (BOSH) Connection. In order to establish the BOSH Connection, the client first requests System Information from the Finesse Server.

SystemInfo

The client's desktop made a Representational State Transfer (REST) Application Programming Interface (API) request to this URL: ***/finesse/api/SystemInfo***. Take note of the ***nocache=***. This unique ID is used in

order to trace this request through the system. ***Returned with status=200*** indicates that the request was successfully received.

```
Container : [ClientServices] SystemInfo: requestId='undefined', Making REST
request: method=GET, url='/finesse/api/SystemInfo?nocache=1366756802163'18:40:03:
Container : [ClientServices] SystemInfo: requestId='undefined', Returned
with status=200
```

If you do not have clientlogs but you need to trace the request, you can search the Tomcat ***localhost_access_log*** in order to determine when the REST API request was made and to locate the unique identifier.

```
127.0.0.1 127.0.0.1 − − 80 GET /finesse/api/SystemInfo ?nocache=1366756802163
HTTP/1.1 200 336 120 10.10.10.211 10.10.10.211 2001 − 80 GET /gadgets/makeRequest
?refresh=3600&url=http%3A%2F%2Flocalhost%2Ffinesse%2Fapi%2FSystemInfo%3Fnocache%
3D1366756802163&httpMethod=GET&headers=Authorization%3DBasic%2520MjAwMToyMDAx%
26locale%3Den_US&postData=&authz=&st=&contentType=TEXT&numEntries=3&getSummaries
=false&signOwner=true&signViewer=true&gadget=undefined&container=default&
bypassSpecCache=&getFullHeaders=false HTTP/1.1 200 659 596
```

API_REQUEST

Tomcat sends this API request to the Finesse REST API Web Application Repository (WAR). In order to find the Finesse REST API logs, search the Finesse webservices log either by timestamp or the nocache ID in order to locate the API_REQUEST. This log shows the ***REQUEST_START***, the ***REQUEST_URL***, the ***REQUEST_END***, and the ***elapsed_time*** the system took to complete the request.

```
%CCBU_http−8080−7−6−REQUEST_START: %[method_name=GET][parameter_name=
{ nocache=1366756802163],  }][resource_name=/SystemInfo][usr=]: Request start
%CCBU_http−8080−7−6−API_REQUEST: %[REQUEST_URL=SystemInfo][agent_id=][request_
identifier=][request_method=systemInfo.GET][request_parameters=]: Request from
client to webservice api
%CCBU_http−8080−7−6−REGISTER_API_STATS_OBJECT: %[resource_name=com.cisco.ccbu:
category=WebAppStats,component0=SystemInfo−GET]: Registered new api stats object
for new request type. %CCBU_http−8080−7−6−REQUEST_END: %[elapsed_time=98]: Request complete
```

The content returned to the client by the REST API request to retrieve System Information is shown here. This information is located in the client (agent) logs.

```
content='<SystemInfo>
  <primaryNode>
    <host>UCCEFINESSE91.vmload.cvp</host>
  </primaryNode>
  <secondaryNode>
    <host>UCCEFINESSE138.vmload.cvp</host>
  </secondaryNode>
  <status>IN_SERVICE</status>
  <xmppDomain>UCCEFINESSE138.vmload.cvp</xmppDomain>
  <xmppPubSubDomain>pubsub.UCCEFINESSE138.vmload.cvp</xmppPubSubDomain>
</SystemInfo>'
```

Establish the BOSH Connection

The SystemInfo shows the Primary and Secondary Finesse servers, the status of Finesse as ***IN_SERVICE***, the ***xmppDomain***, and the ***xmppPubSubDomain***. The client now has enough information in order to establish a BOSH connection.

```
18:40:03: Container : PageServices.init().onLoad: System info status: IN_SERVICE
18:40:03: Container : PageServices.init(): Establishing BOSH connection...
```

```
18:40:03: Container : PageServices.init(): Starting timeout and poller...
18:40:04: Container : [ClientServices] MasterPublisher._connInfoHandler()
− Connection status: connecting
18:40:04: Container : [ClientServices] MasterPublisher._connInfoHandler()
− Connection status: disconnected
18:40:04: Container : PageServices._onDisconnect(): retryCount=0, retrying...
18:40:04: Container : [ClientServices] MasterPublisher._connInfoHandler()
− Connection status: connecting
18:40:05: Container : [ClientServices] MasterPublisher._connInfoHandler()
− Connection status: connected
18:40:05: Container : PageServices.onLoad(): BOSH established!
```

The client is successfully subscribed to the Finesse Object (node) ***/finesse/api/User/2001*** once the BOSH connection is established.

When the client's BOSH connection is established, the webservices log recieves a ***PRESENCE_NOTIFICATION*** message from the client. This ***PRESENCE_TYPE*** only indicates that the client is available to recieve ***XMPP Events*** and has nothing to do with the agent availability in Unified Contact Center Enterprise (UCCE). Remember that the agent is not signed in yet.

Note: You only see the ***PRESENCE_TYPE*** messages when a client establishes a BOSH connection or when a client's BOSH connection is disconnected. When the client's BOSH connection is disconnected, the ***PRESENCE_TYPE*** displays as unavailable.

Here is the Notification event in the websservices log:

```
%CCBU_Smack Listener Processor (1)−6−PRESENCE_NOTIFICATION_RECIEVED:
%[FROM JID=2001@uccefinesse138.vmload.cvp/desktop]
[PRESENCE_TYPE=available]: Finesse received a presence notifcation
```

Agent Sign−in

Now that the client has established the BOSH connection, the sign−in process begins. The client makes another REST API request in order to obtain current user information. In order to make this request, navigate to this URL: ***/finesse/api/User/2001*** and enter method=***GET***.

Because this is a different API Request, the ***nocache ID*** is different. So, in order to track this request, you need to use this new ID.

```
Container : PageServices.onLoad(): BOSH established! Commencing sign−in process
Container : [ClientServices] User: requestId='undefined', Making REST request:
method=GET, url='/finesse/api/User/2001?nocache=1366756805180
'18:40:05: Container : [ClientServices] User: requestId='undefined',
Returned with status=200,
```

You can find this request in the Tomcat ***localhost_access_log*** if needed. Here is how you find it in the webservices log:

```
%CCBU_http−8080−7−6−REQUEST_START: %[method_name=GET][parameter_name={ nocache=
[1366756805180],  }][resource_name=/User/2001][usr=2001]: Request start
 %CCBU_http−8080−7−6−API_REQUEST: %[REQUEST_URL=User/2001][agent_id=2001]
[request_identifier=null][request_method=user.GET][request_parameters=2001]:
Request from client to webservice api
```

Here is the request in the Notification Services log. Take note of ***HTTP/1.1 200 ok***.

Note: The Cisco Notification Log is for informational purposes only. If you enable Cisco Finesse Notification logging, it impacts performance.

```
>> "GET /finesse/api/User/2001 HTTP/1.1[\r][\n]"
Adding Host request header
>> "Authorization: Basic MjAwMToyMDAx[\r][\n]"
>> "User-Agent: Jakarta Commons-HttpClient/3.1[\r][\n]"
>> "Host: localhost:8080[\r][\n]"
>> "[\r][\n]"
<< "HTTP/1.1 200 OK[\r][\n]"
<< "HTTP/1.1 200 OK[\r][\n]"
<< "Pragma: No-cache[\r][\n]"
<< "Cache-Control: no-cache[\r][\n]"
```

Now that the Notification Service has the request, it posts the information for this user. Here is the POST from the Notification Service Log that goes to the client:

```
Cookie accepted: "$Version=0; JSESSIONIDSSO=C11F62C59D0D0438CDEDEEB0DB12AA0B;
$Path=/"
Cookie accepted: "$Version=0; JSESSIONID=25FE81BD7DB73280A07B4CA4138E7680;
$Path=/finesse"
Buffering response body
<< "<User>[\n]"
<< "  <dialogs>/finesse/api/User/2001/Dialogs</dialogs>[\n]"
<< "  <extension></extension>[\n]"
<< "  <firstName>Mickey</firstName>[\n]"
<< "  <lastName>Mouse</lastName>[\n]"
<< "  <loginId>2001</loginId>[\n]"
<< "  <loginName>mmouse</loginName>[\n]"
<< "  <roles>[\n]"
<< "    <role>Agent</role>[\n]"
<< "  </roles>[\n]"
<< "  <state>LOGOUT</state>[\n]"
<< "  <stateChangeTime></stateChangeTime>[\n]"
<< "  <teamId>5000</teamId>[\n]"
<< "  <teamName>Minnies_Team</teamName>[\n]"
<< "  <uri>/finesse/api/User/2001</uri>[\n]"
<< "</User>"
```

This ***XMPP Event***, which is ***agent 2001*** in this example, is sent to all subscription clients. The JavaScript at the client recieves the ***XMPP Event***, and the Event is sent to the Gadget within the client. Here are the clientlogs that show the content of the response:

```
Commencing sign-in process18:40:05: Container : [ClientServices] User: requestId=
'undefined', Maurl='/finesse/api/User/2001?nocache=1366756805180'18:40:05:
Container : [ClientServices] User: requestId='undefined', Returned with status=200,
 content='<User> king REST request: method=GET,
  <dialogs>/finesse/api/User/2001/Dialogs</dialogs>
  <extension></extension>
  <firstName>Mickey</firstName>
  <lastName>Mouse</lastName>
  <loginId>2001</loginId>
  <loginName>mmouse</loginName>
<roles>
    <role>Agent</role>
  </roles>
  <state>LOGOUT</state>
<stateChangeTime></stateChangeTime>
  <teamId>5000</teamId>
  <teamName>Minnies_Team</teamName>
  <uri>/finesse/api/User/2001</uri>
</User>
```

Perform Login

Now the client is ready to perform login. Notice the ***RequestID***. The RequestID is sent in the body of the request. You use this RequestID in order to follow the login request to the ***REST API*** > ***CTI*** > ***REST API*** > ***Notification Service*** > ***Response*** back to the client. This request is a PUT, which means that the client is requesting an UPDATE or a change to its current state.

```
Container : SignIn.handleUserLoad(): Performing login: extn=2003 18:40:05:
Container : [ClientServices] User: requestId='6e210ca9-5786-43bc-babf-
64a397a6057f',
 </data>
  <event>PUT</event>
  <requestId>6e210ca9-5786-43bc-babf-64a397a6057f</requestId>
  <source>/finesse/api/User/2001</source>
</Update>
```

The Finesse REST API receives this request from the client. Then, the API sends a ***SetAgentStateReq*** to the CTI Server.

```
%CCBU_http-8080-7-6-API_REQUEST: %[REQUEST_URL=User/2001][agent_id=2001]
[request_identifier=6e210ca9-5786-43bc-babf-64a397a6057f][request_method=
user.PUT][request_parameters= extension:2003 state:LOGIN]: Request from
client to webservice api
%CCBU_http-8080-7-6-REGISTER_API_STATS_OBJECT: %[resource_name=com.cisco.ccbu:
category=WebAppStats,component0=User-[id]-PUT]: Registered new api stats object
for new request type.
%CCBU_http-8080-7-6-REQUEST_END: %[elapsed_time=8]: Request complete
%CCBU_pool-5-thread-4-6-MESSAGE_TO_CTI_SERVER: %[cti_message=Invoke id :20 ,
agentstate : 0, workmode : 0, reason code: -15532, forceflag :1, agentcapacity:
0, agentext: 2003, agentid: 2001][cti_message_name=SetAgentStateReq]:
Message going to the backend cti server
```

The CTI Server receives the request.

```
Trace: AGENT_EVENT: ID=2001 Periph=5001 Ext=2003 Inst=2003 Sig=
Trace:  SkgState=LOGIN SkgDuration=0 OverallState=NOT_READY OverallDuration=0
Reason=0
Trace:  MRDID=1 NumTasks=0 MaxTaskLimit=1 AgtMode=1 AGTAvailabilityStatus=0
ICMAgtID=5001
Trace:  SkTgtID=5001 SkGrpNo=0x0 SkGrpID=5006 NumLines=0 CurLine=0 ClientStatus=
0x0 Direction=0
```

Once the agent is logged in with a status of ***NOT_READY***, the CTI Server sends ***AGENT_STATE-EVENT*** to Finesse.

```
MsgType:AGENT_STATE_EVENT (MonitorID:0 PeripheralID:5001 SessionID:0x0
PeripheralType:EnterpriseAgent SkillGroupState:LOGIN StateDuration:0
SkillGroupNumber:85881 SkillGroupID:5000 SkillGroupPriority:0 AgentState:
NOT_READY EventReasonCode:0 MRDID:1 NumTasks:0 AgentMode:1 MaxTaskLimit:1
ICMAgentID:5001 AgentAvailabilityStatus:0 NumFltSkillGroups:0 Direction:0
ClientSignature:""AgentID:"2001" AgentExtension:"2003" AgentInstrument:"2003"
RemaskNumMasks:1 RemaskInstrument:"2003" RemaskExtension:"2003" RemaskCallId:
0xffffffff RemaskFunctionFlag:<0x38> <LogoutCodeReq,NotRdyCodeReq,WrapDataReq>
RemaskCallMask:<0x21000000> <MC,Emerg> RemaskAgentMask:<0x0a000000> <
Logout,Avail> )Trace: AGENT_EVENT: ID=2001 Periph=5001 Ext=2003 Inst=2003 Sig=
Trace:  SkgState=LOGIN SkgDuration=0 OverallState=NOT_READY OverallDuration=0
Reason=0 Trace:  MRDID=1 NumTasks=0 MaxTaskLimit=1 AgtMode=1
AGTAvailabilityStatus=0 ICMAgtID=5001
```

Here is the webservices log that received the event from the CTI Server. Remember that you see the ***RAW*** message from the CTI Server first, and then you see the ***Decoded*** message.

```
%CCBU_CTIMessageEventExecutor-0-6-DECODED_MESSAGE_FROM_CTI_SERVER: %[cti_message
=CTIAgentStateEvent [skillGroupState=0 (LOGIN), stateDuration=0, skillGroupNumber
=85881, skillGroupPriority=0, agentState=2 (NOT_READY), eventReasonCode=0,
numFltSkillGroups=0,CTIClientSignature=, agentID=2001, agentExtension=2003,
agentInstrument=2003, agentID_ Long=null, duration=null, nextAgentState=null,
fltSkillGroupNumberList=[], fltSkill GroupIDList=[], fltSkillGroupPriorityList=[],
fltSkillGroupStateList=[]]CTIMessageBean [invokeID=null, msgID=30, timeTracker=
{"id":"AgentStateEvent","CTI_MSG_RECEIVED":1366756808374,
"CTI_MSG_DISPATCH":1366756808375}, msgName=AgentStateEvent, deploymentType=CCE]]
[cti_response_time=1]: Decoded Message to Finesse from backend cti server
```

Now that Finesse received the ***AgentStateEvent*** from the CTI server, the event needs to be ***Published*** to the Notification Service so that the client receives the UPDATE. The only way for the agent to know that his/her state has changed is by receiving this ***XMPP Event***. Finesse converts the ***AgentStateEvent*** to ***XMPP*** and sends the ***XMPP*** to the Notification Service. Notice that the Event is a ***PUT***, and the RequestID is in the Payload.

```
%CCBU_pool-5-thread-5-6-XMPP_PUBLISH_ASYNCHRONOUS: %[NodeId=/finesse/api/User/
2001][Payload=<Update><data><user><dialogs>/finesse/api/User/2001/Dialogs
</dialogs><extension>2003</extension><firstName>Mickey</firstName><lastName>
Mouse</lastName><loginId>2001</loginId><loginName>mmouse</loginName>
<reasonCodeId>-1</reasonCodeId><roles><role>Agent</role></roles><state>NOT_READY
</state><stateChangeTime>2013-04-23T22:40:08Z</stateChangeTime><teamId>5000
</teamId><teamName>Minnies_Team</teamName><uri>/finesse/api/User/2001
</uri></user></data><event>PUT</event><requestId>6e210ca9-5786-43bc-babf-
64a397a6057f </requestId><source>/finesse/api/User/2001</source></Update>]:
Publishing XMPP Message Asynchronously
```

Here, the Notification Service receives the UPDATE. Even though the message says ***failed to route packet to JID***, a message that an event has been published is sent to the user.

```
RoutingTableImpl: Failed to route packet to JID: 2001@uccefinesse138.vmload.cvp/
User packet: <message from="pubsub.uccefinesse138.vmload.cvp" to=
"2001@uccefinesse138.vmload.cvp/ User" id="/finesse/api/User/
2001__2001@uccefinesse138.vmload.cvp__VI1B2"><event xmlns=
"http://jabber.org/protocol/pubsub#event"><items node="/finesse/api/User/2001">
<item id="1su0Keff8M2irdS"><notification xmlns="http://jabber.org/protocol/pubsub">
&lt;Update&gt;
```

Here is the body of the message:

```
 &lt;data&gt;
    &lt;user&gt;
      &lt;dialogs&gt;/finesse/api/User/2001/Dialogs&lt;/dialogs&gt;
      &lt;extension&gt;2003&lt;/extension&gt;
      &lt;firstName&gt;Mickey&lt;/firstName&gt;
      &lt;lastName&gt;Mouse&lt;/lastName&gt;
      &lt;loginId&gt;2001&lt;/loginId&gt;
      &lt;loginName&gt;mmouse&lt;/loginName&gt;
      &lt;reasonCodeId&gt;-1&lt;/reasonCodeId&gt;
      &lt;roles&gt;
        &lt;role&gt;Agent&lt;/role&gt;
      &lt;/roles&gt;
      &lt;state&gt;NOT_READY&lt;/state&gt;
      &lt;stateChangeTime&gt;2013-04-23T22:40:08Z&lt;/stateChangeTime&gt;
      &lt;teamId&gt;5000&lt;/teamId&gt;
      &lt;teamName&gt;Minnies_Team&lt;/teamName&gt;
      &lt;uri&gt;/finesse/api/User/2001&lt;/uri&gt;
    &lt;/user&gt;
  &lt;/data&gt;
  &lt;event&gt;PUT&lt;/event&gt;
  &lt;requestId&gt;6e210ca9-5786-43bc-babf-64a397a6057f&lt;/requestId&gt;
  &lt;source&gt;/finesse/api/User/2001&lt;/source&gt;
```

```
&lt;/Update&gt;</notification></item></items></event></message>
```

As before, the XMPP message is received by the client and delivered to the client's Gadget. Notice the client receives the event with the original RequestID in the message.

```
Returned with status=202, content=''18:40:05: Container : [ClientServices]
 MasterPublisher._eventHandler() − Received event on node '/finesse/api/User/
 2001': <Update>
  <data>
    <user>
      <dialogs>/finesse/api/User/2001/Dialogs</dialogs>
      <extension>2003</extension>
      <firstName>Mickey</firstName>
      <lastName>Mouse</lastName>
      <loginId>2001</loginId>
      <loginName>mmouse</loginName>
      <reasonCodeId>−1</reasonCodeId>
      <roles>
        <role>Agent</role>
      </roles>
      <state>NOT_READY</state>
      <stateChangeTime>2013−04−23T22:40:08Z</stateChangeTime>
      <teamId>5000</teamId>
      <teamName>Minnies_Team</teamName>
      <uri>/finesse/api/User/2001</uri>
    </user>
  </data>
  <event>PUT</event>
  <requestId>6e210ca9−5786−43bc−babf−64a397a6057f</requestId>
  <source>/finesse/api/User/2001</source>
</Update>
```

Now the client is successfully logged in.

```
Container : SignIn._triggerLoggedIn(): Successfully logged in!18:40:05
```

Logout Codes, Reason Codes, PhoneBook

Now the client needs to retrieve agent−specific data, such as Logout codes, Reason Codes, and Phonebook. Here is the request for that information made to the client.

```
Container : SignIn._triggerLoggedIn(): Successfully logged in!18:40:05:
Container : [ClientServices] Dialogs: requestId='undefined', Making REST
request: method=GET, url='/finesse/api/User/2001/Dialogs?nocache=
1366756805355?

18:40:05: Container : [ClientServices] User: requestId='undefined',
Making REST request: method=GET, url='/finesse/api/User/2001/ReasonCodes?
category=LOGOUT&nocache=1366756805356'18:40:05: Container : [ClientServices]
User: requestId='undefined', POST_DATA=''18:40:05: Container : _displayUserData
(): User's current state is: NOT_READY

'18:40:05: Container : [ClientServices] User: requestId='undefined', Making REST
request: method=GET, url='/finesse/api/User/2001/ReasonCodes?category=NOT_READY&
nocache=1366756805358

18:40:05: Container : [ClientServices] User: requestId='undefined', POST_DATA=
''18:40:05: Header : The client logger has been initialize for the header
18:40:05: Header : _displayUserData(): User's current state is: NOT_READY

18:40:05: Header : Container._initGadgetContainer(): Initializing gadget
container.
18:40:05: Header : FailoverMonitor.startListening(): Listening for triggers
```

```
18:40:05: Header : PageServices.stopTimeoutPoller(): Cancelling connection
timeout and poller...
18:40:05: Header : [ClientServices] id=2001: TypeError: 'this._listenerCallback
[...].callback' is null or not an object
```

The same logic applies to these requests. Keep in mind that the Finesse Reason Codes and PhoneBook are stored in the Finesse database, not in UCCE.

Updated: Jul 22, 2013 Document ID: 116255