ログを使った Finesse エージェントのログイン トレース

目次

概要

このドキュメントでは、ログ ファイルによる Finesse システムを介したエージェント ログインに関連したプロセスを説明します。 さまざまな Finesse コンポーネント、コンピュータ テレフォニー インテグレーション（CTI）サーバ、およびクライアント デスクトップの間のメッセージ フローを理解して、問題のトラブルシューティングができるようにすることが重要です。

前提条件

要件

このドキュメントの読者には Cisco Finesse システムおよび音声オペレーティング システム（VOS）の CLI コマンド プロンプトの知識を持っていることが推奨されます。

使用するコンポーネント

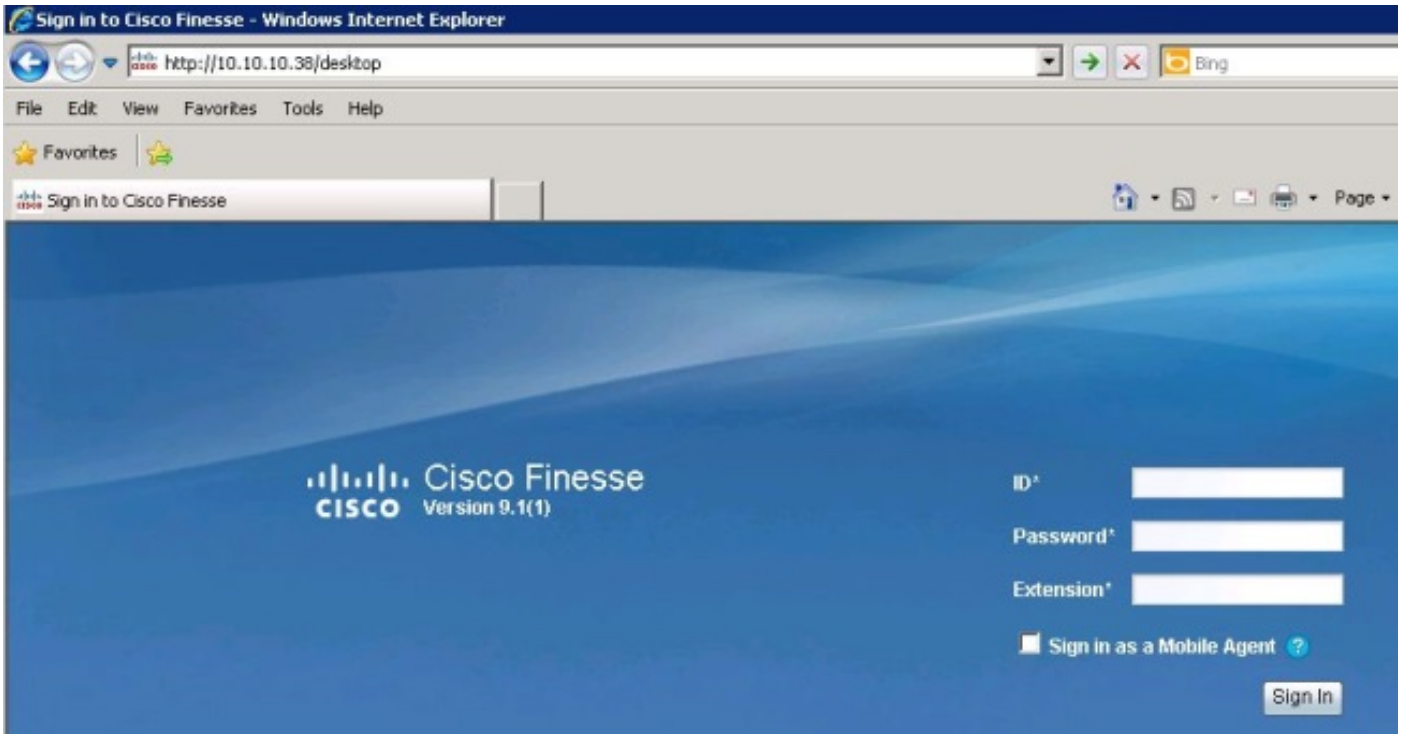
このドキュメントの情報は、Cisco Finesse バージョン 9.1(1) に基づいています。

本書の情報は、特定のラボ環境にあるデバイスに基づいて作成されたものです。 このドキュメントで使用するすべてのデバイスは、初期（デフォルト）設定の状態から起動しています。 稼働中のネットワークで作業を行う場合、コマンドの影響について十分に理解したうえで作業してくだ

さい。

Agent Desktop の起動

Agent Desktop を起動するには、次の URL を Web ブラウザにコピーします。 **http://<your finesse server>/desktop**. Finesse バージョン 9.1 では、HTTP または HTTPS がサポートされます。



Finesse は Web サーバとして Tomcat を使用します。 Web ブラウザを起動する際に、Agent Desktop を表示するための要求を Finesse に対して行います。 Cisco Tomcat の **localhose_access_log** コマンドは Agent Desktop をロードする要求を表示します。

```
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 のログイン クレデンシャル

Agent Desktop が表示されるので、ログイン クレデンシャルを入力します。 Finesse が CTI サーバにログイン要求を送信する前に、クライアントは Bidirectional-streams Over Synchronous HTTP (BOSH) 接続を確立する必要があります。 ボッシュ接続を確立するには、クライアントは最初に Finesse サーバからのシステム情報を要求します。

SystemInfo

クライアント デスクトップは次の URL に Representational State Transfer (REST) アプリケーション プログラミング インターフェイス (API) 要求を行いました： **//finesse/api/SystemInfo。 nocache=** を書き留めておきます。 システムを通じてこの要求をトレースするために、この一意の ID が使用されます。 **status=200 が返されていれば、**要求は正常に受信されています。

```
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
```

クライアントログはないものの要求のトレースが必要な場合は、REST API 要求がいつ作成されたかを特定し、一意の ID を探すために、Tomcat の **localhost_access_log** を検索できます。

```
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 は Finesse REST API の Web アプリケーション リポジトリ (WAR) にこの API 要求を送信します。 Finesse REST API のログを検索するには、Finesse Web サービスのログをタイムスタンプまたは nocache ID のいずれかで検索して、API_REQUEST を見つけます。 このログは、**REQUEST_START**、**REQUEST_URL**、**REQUEST_END**、および要求を実行するためにシステムでかかった時間 **elapsed_time** を表示します。

```
%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
```

システム情報を取得するための REST API 要求によりクライアントに返された内容は、ここに表示されます。 この情報はクライアント (エージェント) ログにあります。

```
%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
```

BOSH 接続の確立

SystemInfo にはプライマリおよびセカンダリ Finesse サーバ、**IN_SERVICE** に Finnesse のステータス、**xmppDomain**、および **xmppPubSubDomain** が表示されます。 これで、クライアントは BOSH 接続を確立するために十分な情報を得ることができます。

```
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!
```

BOSH 接続が確立されると、クライアントは Finesse オブジェクト (ノード) **/finesse/api/User/2001** に対して正常にサブスクライブされます。

クライアントの BOSH 接続が確立されると、Web サービスのログはクライアントから **PRESENCE_NOTIFICATION** メッセージを受信します。 この **PRESENCE_TYPE** は、**XMPP イベント**を受信するためにクライアントが利用できることを示すだけで、Unified Contact Center Enterprise (UCCE) でのエージェントの可用性とはまったく関係ありません。 エージェントはまだサインインしていないことに注意してください。

> **注: PRESENCE_TYPE** メッセージが表示されるのは、クライアントが BOSH 接続を確立したとき、またはクライアントの BOSH 接続が切断されたときのみです。 クライアントの BOSH 接続が切断されると、**PRESENCE_TYPE** には unavailable と表示されます。

Web サービスのログの通知イベントを次に示します。

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

エージェントのサインイン

クライアントが BOSH 接続を確立したので、サインイン プロセスが開始されます。 クライアントは現在のユーザ情報を取得するために別の REST API 要求を行います。 この要求を行うためには、次の URL に移動します。 **//finesse/api/User/2001** そして、method=GET と入力します。

これは別の API 要求であるため、nocache ID は異なります。 したがって、この要求を追跡するためには、この新しい ID を使用する必要があります。

```
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,
```

必要ならば、この要求は Tomcat の localhost_access_log で見つけることができます。 次に、これを Web サービス ログで見つける方法を示します。

```
%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
```

これが、通知サービス ログにある要求です。 HTTP/1.1 200 ok をメモします。

> **注:** シスコの通知ログは情報提供のみを目的としています。 Cisco Finesse の通知ロギングを有効にすると、パフォーマンスに影響します。

```
>> "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]"
```

通知サービスに要求が届き、このユーザの情報が表示されます。 これが、クライアントに送られる通知サービス ログからの POST です。

```
>> "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]"
```

この XMPP イベント（この例では agent 2001 ）は、すべてのサブスクリプション クライアントに送信されます。 クライアントの JavaScript はこの XMPP イベントを受信し、そのイベントはクライアント内のガジェットに送信されます。 応答の内容を示すクライアントログを次に示します。

```
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>
```

ログインの実行

これで、クライアントはログインを実行する準備が整いました。 **RequestID** に注目してください。 RequestID は要求の本文で送信されます。 ログイン要求が **REST API > CTI > REST API > 通知サービス > 応答**の順でクライアントに戻るのを追跡するには、この RequestID を使用します。この要求は PUT です。つまり、クライアントは現在の状態の更新または変更を要求していることを意味します。

```
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>
```

Finesse REST API は、この要求をクライアントから受信します。 次に、API は **SetAgentStateReq** を CTI サーバに送信します。

```
%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
```

CTI サーバが要求を受信します。

```
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
```

ステータスが NOT_READY でエージェントがログインすると、CTI サーバは Finesse に AGENT_STATE-EVENT を送信します。

```
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
```

CTI サーバからイベントを受信した Web サービスのログを次に示します。 最初に CTI サーバの RAW メッセージが表示され、次に Decoded メッセージが表示されることを覚えておいてください。

```
%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
```

Finesse が CTI サーバから AgentStateEvent を受信したので、クライアントが UPDATE を受信するように、このイベントを通知サービスにパブリッシュする必要があります。 エージェントの状態が変更されたことをエージェント自身が知るための唯一の方法は、この XMPP イベントを受信することです。 Finesse は AgentStateEvent を XMPP に変換し、その XMPP を通知サービスに送信します。 このイベントは PUT であり、RequestID がペイロードにあることに注目してください。

```
%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
```

ここで、通知サービスが UPDATE を受信します。 「failed to route packet to JID」というメッセージが表示されても、イベントがパブリッシュされたというメッセージはユーザに送信されます。

```
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;
```

メッセージの本文を次に示します。

```
 &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>
```

前述のように、XMPP メッセージがクライアントによって受信され、クライアントのガジェットに配信されます。 クライアントはメッセージで、元の RequestID の付いたイベントを受け取ることに注意してください。

```
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>
```

これで、クライアントは正常にログインできます。

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

ログアウト コード、理由コード、電話帳

次に、クライアントはログアウト コード、理由コード、電話帳など、エージェント固有のデータを取得する必要があります。 その情報を求めるクライアントへの要求を次に示します。

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

これらの要求には同じロジックが適用されます。 Finesse の理由コードと電話帳は UCCE でなく、Finesse データベースに保存されていることに注意してください。