Traccia di accesso di Finesse Agent con utilizzo dei log

Sommario

Introduzione

Questo documento descrive il processo di accesso di un agente tramite il sistema Finesse con i file di log. È importante comprendere il flusso di messaggi tra i diversi componenti Finesse, il server CTI (Computer Telephony Integration) e il desktop del client in modo da poter risolvere correttamente i problemi.

Prerequisiti

Requisiti

Cisco consiglia di conoscere Cisco Finesse e il prompt dei comandi VOS (Voice Operating System) CLI.

Componenti usati

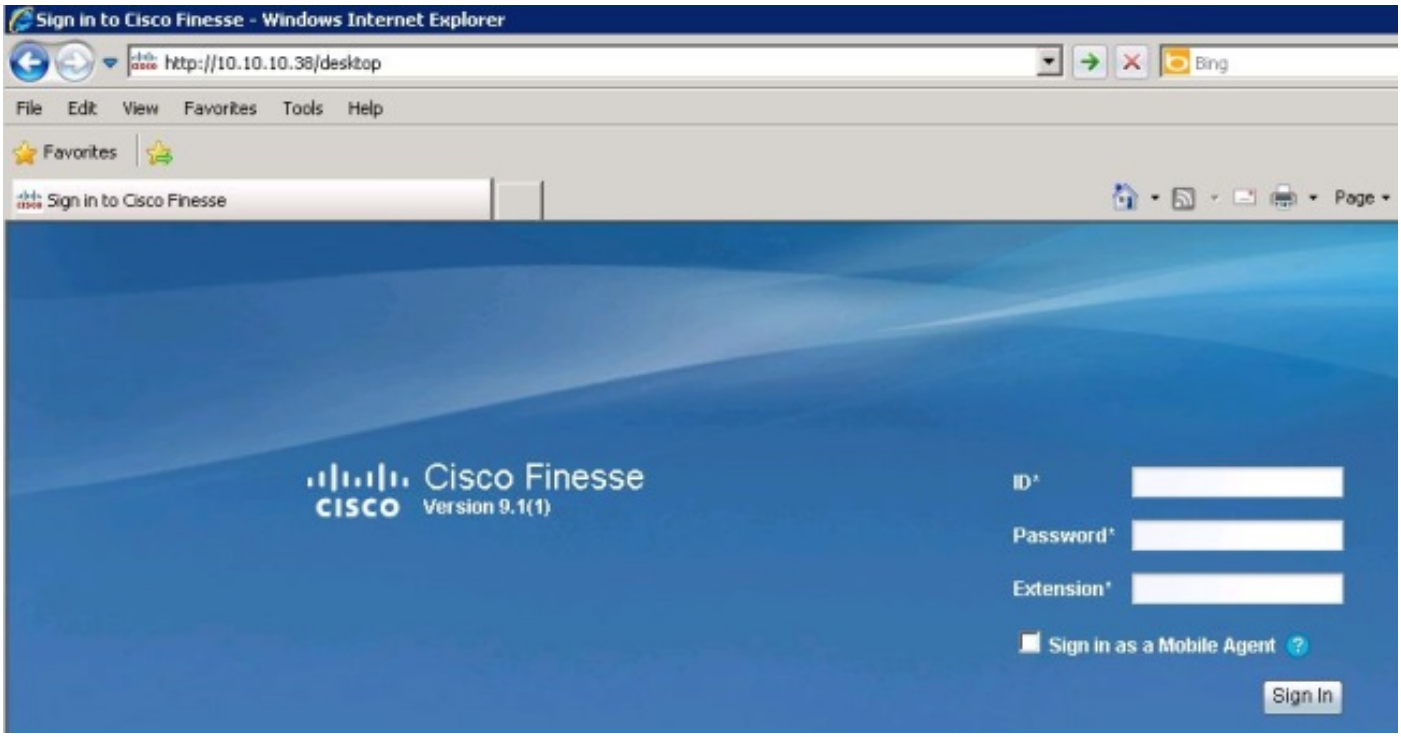
Il riferimento delle informazioni contenute in questo documento è Cisco Finesse versione 9.1(1).

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali

conseguenze derivanti dall'uso dei comandi.

Avvia Agent Desktop

Per avviare il desktop dell'agente, copiare questo URL nel browser Web: **http://<server finesse>/desktop**. In Finesse versione 9.1 è supportato HTTP o HTTPS.



Finesse utilizza Tomcat come server Web. Quando si avvia il browser Web, viene richiesta a Finesse la presentazione del desktop dell'agente. Il comando Cisco Tomcat **localhose_access_log** mostra la richiesta di caricare il desktop dell'agente.

```
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
```

Credenziali di accesso agente

Dopo aver presentato il desktop dell'agente, immettere le credenziali di accesso. Affinché Finesse possa inviare la richiesta di accesso al server CTI, il client deve stabilire una connessione BOSH (Over Synchronous HTTP) con flussi bidirezionali. Per stabilire la connessione BOSH, il client richiede prima System Information al server Finesse.

InformazioniSistema

Il desktop del client ha effettuato una richiesta dell'API (Application Programming Interface) REST (Representative State Transfer) a questo URL: **/finesse/api/SystemInfo**. Prendere nota di **nocache=**. Questo ID univoco viene utilizzato per tracciare la richiesta nel sistema. **Restituito con status=200** indica che la richiesta è stata ricevuta correttamente.

```
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
```

Se non si dispone dei log dei client ma è necessario tracciare la richiesta, è possibile eseguire una ricerca in Tomcat **localhost_access_log** per determinare quando è stata effettuata la richiesta dell'API REST e per individuare l'identificatore univoco.

```
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
```

RICHIESTA_API

Tomcat invia questa richiesta API a Finesse REST API Web Application Repository (WAR). Per trovare i registri API di Finesse REST, cercare il registro dei servizi Web di Finesse in base all'indicatore orario o all'ID nocache per individuare API_REQUEST. In questo log vengono mostrati **REQUEST_START**, **REQUEST_URL**, **REQUEST_END** e il **tempo trascorso** impiegato dal sistema per completare la richiesta.

```
%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
```

Di seguito è illustrato il contenuto restituito al client dalla richiesta API REST per il recupero di System Information. Queste informazioni si trovano nei log del client (agente).

```
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>'
```

Connessione BOSH

SystemInfo mostra i server Finesse primario e secondario, lo stato di Finesse come **IN_SERVICE**, l'oggetto **xmppDomain** e l'oggetto **xmppPubSubDomain**. Il client dispone ora di informazioni sufficienti per stabilire una connessione BOSH.

```
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!
```

Il client viene sottoscritto correttamente all'oggetto Finesse (nodo) **/finesse/api/User/2001** una volta stabilita la connessione BOSH.

Una volta stabilita la connessione BOSH del client, il log dei servizi Web riceve un messaggio **PRESENCE_NOTIFICATION** dal client. Questo **PRESENCE_TYPE** indica solo che il client è disponibile per ricevere **eventi XMPP** e non ha nulla a che fare con la disponibilità dell'agente in Unified Contact Center Enterprise (UCCE). Ricorda che l'agente non ha ancora effettuato l'accesso.

> **Nota:** I messaggi **PRESENCE_TYPE** vengono visualizzati solo quando un client stabilisce una connessione BOSH o quando una connessione BOSH di un client viene disconnessa. Quando la connessione BOSH del client viene disconnessa, **PRESENCE_TYPE** viene visualizzato come non disponibile.

Di seguito è riportato l'evento Notification nel registro dei servizi Web:

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

Accesso agente

Ora che il client ha stabilito la connessione BOSH, viene avviato il processo di accesso. Il client effettua un'altra richiesta dell'API REST per ottenere informazioni sull'utente corrente. Per effettuare questa richiesta, passare a questo URL: **/finesse/api/User/2001** e immettere method=**GET**.

Poiché si tratta di una richiesta API diversa, l'**ID nocache** è diverso. Per tenere traccia della richiesta, è necessario utilizzare il nuovo 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,
```

Èpossibile trovare questa richiesta in Tomcat **localhost_access_log** se necessario. Di seguito è riportata una descrizione del log dei servizi Web:

```
%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
```

Di seguito è riportata la richiesta nel registro di Notification Services. Prendere nota di **HTTP/1.1 200 ok**.

> **Nota:** Il registro delle notifiche Cisco ha scopo puramente informativo. L'abilitazione della registrazione di Cisco Finesse Notification influisce sulle prestazioni.

```
>> "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]"
```

Ora che il servizio di notifica ha ricevuto la richiesta, pubblica le informazioni per questo utente. Di seguito è riportato il POST dal log del servizio di notifica che viene inviato al 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>"
```

Questo **evento XMPP**, che in questo esempio è l'**agente 2001**, viene inviato a tutti i client di sottoscrizione. Il codice JavaScript sul client riceve l'**evento XMPP** e l'evento viene inviato al gadget all'interno del client. Di seguito sono riportati i log client che mostrano il contenuto della risposta:

```
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>
```

Esegui login

A questo punto il client è pronto per eseguire l'accesso. Notare il valore **RequestID**. RequestID viene inviato nel corpo della richiesta. Utilizzare questo RequestID per seguire la richiesta di accesso all'**API REST > CTI > API REST > Servizio di notifica > Risposta** al client. Questa richiesta è un PUT, ovvero il client richiede un UPDATE o una modifica allo stato corrente.

```
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>
```

L'API REST Finesse riceve questa richiesta dal client. L'API invia quindi un **SetAgentStateReq** al server CTI.

```
%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
```

Il server CTI riceve la richiesta.

```
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
```

Una volta che l'agente ha eseguito l'accesso con lo stato NOT_READY, il server CTI invia AGENT_STATE-EVENT a 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
```

Di seguito è riportato il registro dei servizi Web che ha ricevuto l'evento dal server CTI. Tenere presente che prima viene visualizzato il messaggio RAW dal server CTI, quindi il messaggio Decoded.

```
%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
```

Ora che Finesse ha ricevuto l'evento AgentStateEvent dal server CTI, l'evento deve essere pubblicato al servizio di notifica in modo che il client riceva l'aggiornamento. L'unico modo per l'agente di sapere che il proprio stato è cambiato consiste nel ricevere questo evento XMPP. Finesse converte l'evento AgentStateEvent in XMPP e invia l'evento XMPP al servizio di notifica. Notare che l'evento è un PUT e l'ID richiesta è nel 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
```

In questo caso, il servizio di notifica riceve l'aggiornamento. Anche se nel messaggio viene indicato che non è stato possibile instradare il pacchetto a JID, all'utente viene inviato un messaggio che indica che un evento è stato pubblicato.

```
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;
```

Ecco il testo del messaggio:

```
 &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<;/requestId&gt;
&lt;source&gt;/finesse/api/User/2001&lt;/source&gt;
&lt;/Update&gt;</notification></item></items></event></message>
```

Come in precedenza, il messaggio XMPP viene ricevuto dal client e consegnato al relativo gadget. Si noti che il client riceve l'evento con il RequestID originale nel messaggio.

```
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>
```

Ora il client ha completato l'accesso.

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

Codici di disconnessione, codici motivo, rubrica

Ora il client deve recuperare i dati specifici dell'agente, come i codici di disconnessione, i codici motivo e la rubrica. Ecco la richiesta di informazioni fatta al cliente.

```
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
```

La stessa logica si applica a queste richieste. Tenere presente che i codici motivo Finesse e la rubrica sono memorizzati nel database Finesse e non in UCCE.