Traza del login del agente de la delicadeza con el uso de los registros

Contenido

Introducción

Este documento describe el proceso implicado en un login del agente a través del sistema de la delicadeza con los archivos del registro. Es importante entender el flujo de mensajes entre los diversos componentes de la delicadeza, el servidor del Integración de telefonía de computadora (CTI), y el escritorio del cliente de modo que usted pueda resolver problemas con éxito los problemas.

Prerequisites

Requisitos

Cisco recomienda que usted tiene conocimiento de la delicadeza de Cisco y del prompt de comando CLI del sistema operativo de la Voz (VOS).

Componentes Utilizados

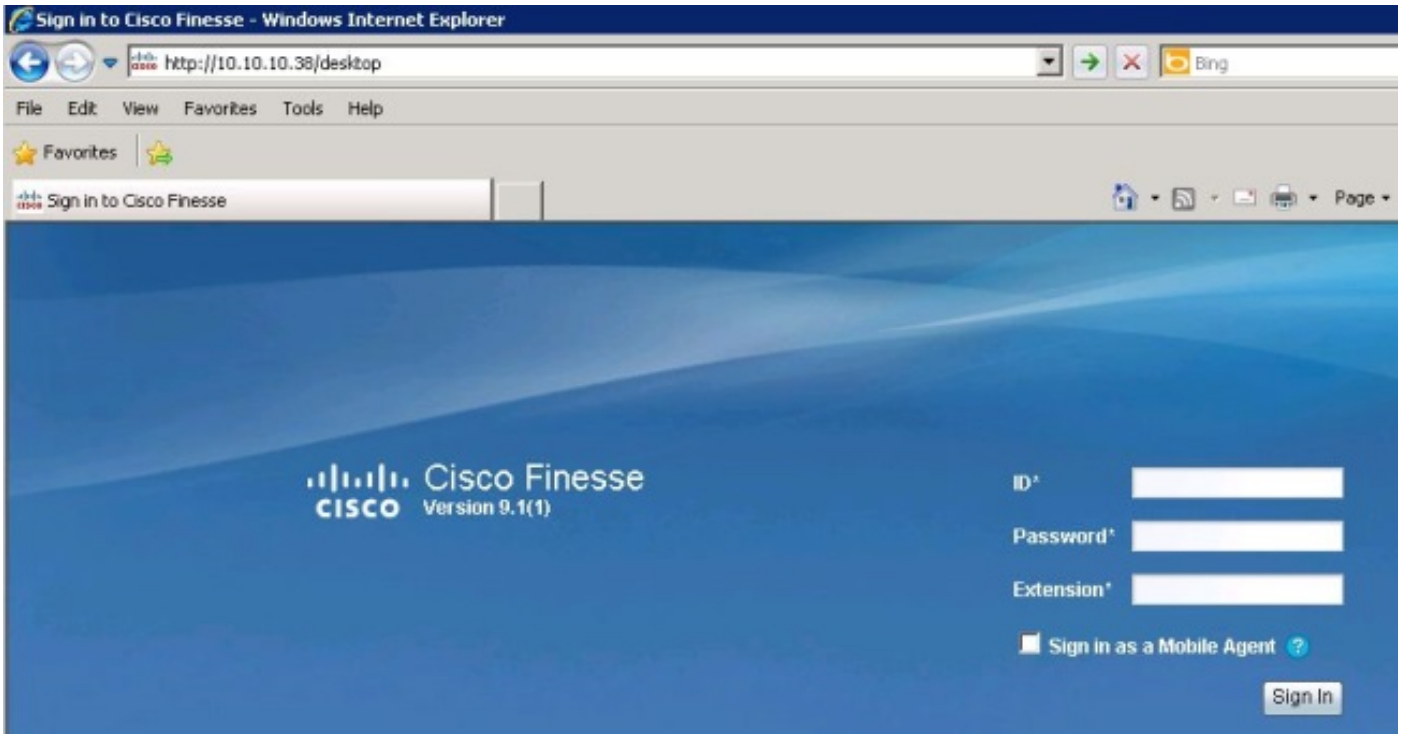
La información en este documento se basa en la versión 9.1(1) de la delicadeza de Cisco.

La información que contiene este documento se creó a partir de los dispositivos en un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en

funcionamiento con una configuración verificada (predeterminada). Si la red está funcionando, asegúrese de haber comprendido el impacto que puede tener cualquier comando.

Inicie el escritorio del agente

Para iniciar el escritorio del agente, copie este URL en el buscador Web: **delicadeza server>/desktop del <your de http://**. En la versión 9.1 de la delicadeza, se soporta el HTTP o el HTTPS.



La delicadeza utiliza Tomcat como el servidor Web. Cuando usted pone en marcha a su buscador Web, la petición se hace a la delicadeza para presentarle el escritorio del agente. El comando del **localhose_access_log de** Cisco Tomcat muestra la petición de cargar el escritorio del 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
```

Credenciales del login del agente

Ahora que se ha presentado el escritorio del agente, usted ingresa sus credenciales del login. Antes de que la delicadeza pueda enviar el pedido de registro al CTI Server, el cliente necesita establecer las Bidireccional-secuencias sobre la conexión síncrona HTTP (PALABRERÍA). Para establecer la conexión de la PALABRERÍA, el cliente primero pide la información del sistema del servidor de la delicadeza.

SystemInfo

El escritorio del cliente hizo una petición representativa de la interfaz de programación de aplicaciones de la transferencia del estado (RESTO) (API) a este URL: **/finesse/api/SystemInfo**. Tome la nota del **nocache=**. Este ID único se utiliza para localizar esta petición a través del sistema. **Vuelto con status=200** indica que la petición fue recibida con éxito.

```
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
```

Si usted no tiene clientlogs sino que usted necesita localizar la petición, usted puede buscar el **localhost_access_log del** tomcat para determinar cuando la petición del RESTO API fue hecha y localizar el Identificador único.

```
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 envía esta petición API al repositorio de la aplicación de Web del RESTO API de la delicadeza (GUERRA). Para encontrar los registros del RESTO API de la delicadeza, busque los webservices de la delicadeza registran por el grupo fecha/hora o el nocache ID para localizar el API_REQUEST. Este registro muestra el **REQUEST_START**, el **REQUEST_URL**, el **REQUEST_END**, y el **elapsed_time que el** sistema tomó para completar la petición.

```
%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
```

El contenido vuelto al cliente por la petición del RESTO API de extraer la información del sistema se muestra aquí. Esta información se encuentra en los registros del cliente (agente).

```
%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
```

Establezca la conexión de la PALABRERÍA

El SystemInfo muestra los servidores primarios y secundarios de la delicadeza, el estatus de la delicadeza como **IN_SERVICE**, el **xmppDomain**, y el **xmppPubSubDomain**. El cliente ahora tiene bastante información para establecer una conexión de la PALABRERÍA.

```
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!
```

El cliente está inscrito con éxito al objeto de la delicadeza (nodo) **/finesse/api/User/2001** una vez que se establece la conexión de la PALABRERÍA.

Cuando se establece la conexión de la PALABRERÍA del cliente, el registro de los webservices recibe un mensaje **PRESENCE_NOTIFICATION del** cliente. Este **PRESENCE_TYPE** indica solamente que el cliente está disponible recibir los **eventos XMPP** y no tiene nada hacer con la disponibilidad del agente en la empresa unificada del Centro de contacto (UCCE). Recuerde que el agente no está ingresado todavía.

> **Note**: Usted ve solamente los mensajes **PRESENCE_TYPE** cuando un cliente establece una conexión de la PALABRERÍA o cuando la conexión de la PALABRERÍA de un cliente es disconnected. Cuando la conexión de la PALABRERÍA del cliente es disconnected, las visualizaciones **PRESENCE_TYPE** como inasequibles.

Aquí está el evento de notificación en el registro de los websservices:

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

Registro del agente

Ahora que el cliente ha establecido la conexión de la PALABRERÍA, el proceso del ingresar

comienza. El cliente hace otra petición del RESTO API para obtener la información de Usuario usuario actual. Para hacer esta petición, navegue a este URL: **/finesse/api/User/2001** y ingresa el method=GET.

Porque esto es una diversa petición API, el **nocache ID** es diferente. Así pues, para seguir esta petición, usted necesita utilizar este nuevo 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,
```

Usted puede encontrar esta petición en el **localhost_access_log de** Tomcat si es necesario. Aquí es cómo usted lo encuentra en el registro de los webservices:

```
%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
```

Aquí está la petición en el registro de los Servicios de notificación. Tome la nota de la **autorización HTTP/1.1 200**.

> **Note**: El registro de la notificación de Cisco está sólo con fines informativos. Si usted habilita el registro de la notificación de la delicadeza de Cisco, afecta el funcionamiento.

```
>> "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]"
```

Ahora que el servicio de notificación tiene la petición, fija la información para este usuario. Aquí está el POSTE del registro del servicio de notificación que va al cliente:

```
>> "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]"
```

Este **evento XMPP**, que es el **agente 2001** en este ejemplo, se envía a todos los clientes de la suscripción. El Javascript en el cliente recibe el **evento XMPP**, y el evento se envía al gadget dentro del cliente. Aquí están los clientlogs que muestran el contenido de la respuesta:

```
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>
```

Realice el login

El cliente está listo ahora para realizar el login. Note el **RequestID**. El RequestID se envía en el cuerpo de la petición. Usted utiliza este RequestID para seguir el pedido de registro al **RESTO API > CTI > RESTO API > servicio de notificación > respuesta** de nuevo al cliente. Esta petición es HABER PUESTO, así que significa que el cliente está pidiendo una ACTUALIZACIÓN o un cambio a su estado actual.

```
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>
```

El RESTO API de la delicadeza recibe esta petición del cliente. Entonces, el API envía un **SetAgentStateReq al** 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
```

El CTI Server recibe la petición.

```
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 vez que el agente se abre una sesión con un estatus de **NOT_READY**, el CTI Server envía **AGENT_STATE-EVENT a la** delicadeza.

```
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
```

Aquí está el registro de los webservices que recibió el evento del CTI Server. Recuerde que usted ve el **mensaje sin procesar del** CTI Server primero, y después usted vea el mensaje **decodificado**.

```
%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
```

Ahora que la delicadeza recibió el **AgentStateEvent del** CTI Server, el evento necesita **ser publicado al** servicio de notificación de modo que el cliente reciba la ACTUALIZACIÓN. La única forma para que el agente sepa que su estado ha cambiado está recibiendo este **evento XMPP**. La delicadeza convierte el **AgentStateEvent a XMPP** y envía el **XMPP al** servicio de notificación. Note que el evento es **HABER PUESTO**, y el RequestID está en el 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
```

Aquí, el servicio de notificación recibe la ACTUALIZACIÓN. Aunque el mensaje dice **fallado al paquete de Routes a JID**, un mensaje que se ha publicado un evento se envía al usuario.

```
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;
```

Aquí está el cuerpo del mensaje:

```
 &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>
```

Como antes, el mensaje XMPP es recibido por el cliente y entregado al gadget del cliente. Note que el cliente recibe el evento con el RequestID original en el mensaje.

```
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>
```

Ahora abren una sesión al cliente con éxito.

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

Códigos del logout, códigos de motivo, guía telefónica

Ahora el cliente necesita extraer los datos agente-específicos, tales como códigos del logout, códigos de motivo, y guía telefónica. Aquí está la petición esa información hecha al cliente.

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

La misma lógica aplica a éstos las peticiones. Tenga presente que los códigos de motivo y la guía telefónica de la delicadeza se salvan en la base de datos de la delicadeza, no en UCCE.