Suivi de connexion de l'agent de finesse avec l'utilisation des logs

Contenu

Introduction

Ce document décrit le processus impliqué dans une procédure de connexion d'agent par le système de finesse des fichiers journal. Il est important de comprendre le flux des messages entre les différents composants de finesse, le serveur du couplage de la téléphonie et de l'informatique (CTI), et l'appareil de bureau de client de sorte que vous puissiez avec succès dépanner des questions.

Conditions préalables

Conditions requises

Cisco recommande que vous ayez la connaissance du Cisco Finesse et de l'invite de commande du système d'exploitation CLI de Voix (VOS).

Composants utilisés

Les informations dans ce document sont basées sur la version 9.1(1) de Cisco Finesse.

Les informations contenues dans ce document ont été créées à partir des périphériques d'un environnement de laboratoire spécifique. Tous les périphériques utilisés dans ce document ont

démarré avec une configuration effacée (par défaut). Si votre réseau est opérationnel, assurez-vous que vous comprenez l'effet potentiel de toute commande.

Lancez l'Agent Desktop

Afin de lancer l'Agent Desktop, copiez cet URL dans le navigateur Web : **finesse server>/desktop de <your de http://**. Dans la version 9.1 de finesse, le HTTP ou le HTTPS est pris en charge.

La finesse utilise Tomcat comme web server. Quand vous lancez votre navigateur Web, la demande est faite à la finesse pour te présenter l'Agent Desktop. La commande de **localhose_access_log de** Cisco Tomcat affiche la demande de charger l'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
```

Qualifications de connexion de l'agent

Maintenant que l'Agent Desktop a été présenté, vous entrez dans vos qualifications de procédure de connexion. Avant que la finesse puisse envoyer la demande de procédure de connexion au serveur CTI, le client doit établir des Bidirectionnel-flots au-dessus de la connexion synchrone de HTTP (BÊTISES). Afin d'établir la connexion de BÊTISES, le client demande d'abord les informations système du serveur de finesse.

SystemInfo

L'appareil de bureau du client a fait un état figurative transférer la demande de l'interface de programmation (de REPOS) (API) vers cet URL : **/finesse/api/SystemInfo**. Notez le **nocache=**. Cet identificateur unique est utilisé afin de tracer cette demande par le système. **Retourné avec status=200** indique que la demande a été avec succès reçue.

```
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 vous n'avez pas des clientlogs mais vous devez tracer la demande, vous pouvez rechercher le **localhost_access_log de** Tomcat afin de déterminer quand la demande du REPOS API a été faite et localiser l'identifiant unique.

```
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 envoie cette demande API au référentiel d'application Web du REPOS API de finesse (GUERRE). Afin de trouver les logs du REPOS API de finesse, recherchez le log de webservices de finesse par l'horodateur ou l'ID de nocache afin de localiser l'API_REQUEST. Ce log affiche le **REQUEST_START**, le **REQUEST_URL**, le **REQUEST_END**, et l'**elapsed_time que le** système a pris pour se terminer la demande.

```
%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
```

Le contenu retourné au client par la demande du REPOS API de récupérer les informations système est affiché ici. Ces informations se trouvent dans les logs de client (agent).

```
%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
```

Établissez la connexion de BÊTISES

Le SystemInfo affiche les serveurs primaires et secondaires de finesse, le statut de finesse comme **IN_SERVICE**, le **xmppDomain**, et le **xmppPubSubDomain**. Le client a maintenant assez d'informations afin d'établir une connexion de BÊTISES.

```
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!
```

Le client est avec succès abonné à l'objet de finesse (noeud) **/finesse/api/User/2001** une fois que la connexion de BÊTISES est établie.

Quand la connexion des BÊTISES du client est établie, le log de webservices reçoit un message **PRESENCE_NOTIFICATION du** client. Ce **PRESENCE_TYPE** indique seulement que le client est disponible pour recevoir des **événements XMPP** et n'a rien à faire avec la Disponibilité d'agent dans Unified Contact Center Enterprise (UCCE). Souvenez-vous que l'agent n'est pas connecté encore.

> Remarque: Vous voyez seulement les messages **PRESENCE_TYPE** quand un client établit une connexion de BÊTISES ou quand la connexion des BÊTISES d'un client est déconnectée. Quand la connexion des BÊTISES du client est déconnectée, les affichages **PRESENCE_TYPE** comme indisponibles.

Voici l'événement de notification dans le log de 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
```

Connexion d'agent

Maintenant que le client a établi la connexion de BÊTISES, le processus de connexion commence. Le client fait une autre demande du REPOS API afin d'obtenir les informations d'utilisateur courant. Afin de faire cette demande, naviguez vers cet URL : **/finesse/api/User/2001** et écrivent le method=GET.

Puisque c'est une demande différente API, l'**ID de nocache** est différent. Ainsi, afin de dépister cette demande, vous devez utiliser ce nouvel 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,
```

Vous pouvez trouver cette demande dans le **localhost_access_log de** Tomcat si nécessaire. Voici comment vous le trouvez dans le log de 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
```

Voici la demande dans le log de Services de notification. Notez l'**ok HTTP/1.1 200**.

> Remarque: Le log de notification de Cisco est à des fins d'information seulement. Si vous activez la notification de Cisco Finesse se connectant, elle affecte la représentation.

```
>> "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]"
```

Maintenant que le service de notification a la demande, il signale les informations pour cet utilisateur. Voici le POST du log de service de notification qui va au client :

```
>> "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]"
```

Cet **événement XMPP**, qui est l'**agent 2001** dans cet exemple, est envoyé à tous les clients d'abonnement. Le Javascript au client reçoit l'**événement XMPP**, et l'événement est envoyé à l'instrument chez le client. Voici les clientlogs qui affichent le contenu de la réponse :

```
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>
```

Exécutez la procédure de connexion

Maintenant le client est prêt à exécuter la procédure de connexion. Notez le **RequestID**. Le RequestID est introduit le corps de la demande. Vous employez ce RequestID afin de suivre la demande de procédure de connexion au **REPOS API > CTI > REPOS API > service > réponse de notification** de nouveau au client. Cette demande est MISE, ainsi il signifie que le client demande une MISE À JOUR ou une modification à son état actuel.

```
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>
```

Le REPOS API de finesse reçoit cette demande du client. Puis, l'API envoie un **SetAgentStateReq au** serveur 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
```

Le serveur CTI reçoit la demande.

```
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
```

Une fois que l'agent est ouvert une session avec un état de **NOT_READY**, le serveur CTI envoie **AGENT_STATE-EVENT à la** 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
```

Voici les webservices se connectent que reçu l'événement du serveur CTI. Souvenez-vous que vous voyez le message **CRU du** serveur CTI d'abord, et alors vous voyez le message **décodé**.

```
%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
```

Maintenant que la finesse a reçu l'**AgentStateEvent du** serveur CTI, l'événement doit **être édité au** service de notification de sorte que le client reçoive la MISE À JOUR. La seule manière pour que l'agent sache que son état a changé est en recevant cet **événement XMPP**. La finesse convertit l'**AgentStateEvent** en **XMPP** et envoie le **XMPP au** service de notification. Notez que l'événement est **MISE**, et le RequestID est dans la charge utile.

```
%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
```

Ici, le service de notification reçoit la MISE À JOUR. Quoique le message indique **pour conduire le paquet à JID**, un message qu'un événement a été édité est envoyé à l'utilisateur.

```
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;
```

Voici le corps du 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>
```

Comme avant, le message XMPP est reçu par le client et fourni à l'instrument du client. Notez que le client reçoit l'événement avec le RequestID d'origine dans le 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>
```

Le client est maintenant avec succès ouvert une session.

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

Codes de déconnexion, codes de raison, répertoire

Maintenant le client doit récupérer des données d'agent-particularité, telles que des codes de déconnexion, des codes de raison, et le répertoire. Voici la demande de ces informations faites au client.

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

La même logique s'applique à ces derniers des demandes. Maintenez dans l'esprit que codes de raison et le répertoire de finesse sont enregistré dans la base de données de finesse, pas dans UCCE.