精良与使用的座席登录Trace日志

目录

简介

本文描述在座席登录涉及的进程通过精良系统用日志文件。了解区别精良组件、计算机电话集成(CTI)服务器和客户端桌面之间的消息流是重要的，以便您能顺利地排除故障问题。

先决条件

要求

Cisco建议您有Cisco精良和语音操作系统的(VOS) CLI命令提示符知识。

使用的组件

本文档中的信息根据Cisco精良版本9.1(1)。

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您使用的是真实网络，请确保您已经了解所有命令的潜在影响。

启动Agent Desktop

为了启动Agent Desktop，请复制此URL到Web浏览器：**http:// <your精良server>/desktop**。在精良

版本9.1，支持HTTP或HTTPS。

精良使用Tomcat作为Web服务器。当您启动您的Web浏览器时，请求被做对精良提交Agent Desktop对您。思科Tomcat **localhose_access_log**命令显示请求装载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
```

座席登录凭证

即然提交了Agent Desktop，您输入您的登录凭证。在精良能发送登录请求到CTI服务器前，客户端需要设立在同步HTTP (炉腹)连接的双向数据流。为了首先建立炉腹连接，从精良服务器的客户端请求系统信息。

SystemInfo

客户端的桌面使一代表状态转接(其余) Application Programming Interface (API)请求到此URL：**/finesse/api/SystemInfo**。注意到**nocache=**。此唯一的ID用于为了通过系统跟踪此请求。**返回与status=200**表明请求顺利地接收。

```
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
```

如果没有clientlogs，但是需要跟踪请求，您能搜索Tomcat **localhost_access_log**为了确定，当其余API请求被做了和找出唯一标识符。

```
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发送此API请求到精良其余API Web应用程序信息库(战争)。为了查找精良其余API日志，请搜索精良webservices由时间戳或nocache ID记录为了找出API_REQUEST。此日志显示 **REQUEST_START**， **REQUEST_URL**， **REQUEST_END**，并且系统采取完成请求的 **elapsed_time**。

```
%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
```

内容返回对客户端由其余API请求获取系统信息显示此处。此信息在客户端(代理程序)日志查找。

```
%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
```

建立炉腹连接

SystemInfo显示主要的和附属精良服务器、精良状况作为**IN_SERVICE**， **xmppDomain**和 **xmppPubSubDomain**。客户端当前有足够的信息为了建立炉腹连接。

```
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!
```

客户端顺利地订阅对精良对象(节点) **/finesse/api/User/2001**，一旦炉腹连接被建立。

当客户端的炉腹连接被建立时， webservices日志收到从客户端的一个 **PRESENCE_NOTIFICATION**消息。此**PRESENCE_TYPE**在Unified Contact Center企业(UCCE)中只表明客户端是可用接收**XMPP事件**并且与座席可用性无关。切记代理程序没有签到。

Note:您只看到PRESENCE_TYPE消息，当客户端建立炉腹连接时或，当客户端的炉腹连接被断开时。当客户端的炉腹连接被断开， PRESENCE_TYPE显示如不可用。

这是在websservices日志的通知事件：

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

代理程序签到

即然客户端建立了炉腹连接，签到进程开始。客户端做另一其余API请求为了得到当前用户信息。为了做此请求，请导航对此URL ：/finesse/api/User/2001和回车method=GET。

由于这是一不同的API请求， nocache ID不同的。因此，为了跟踪此请求，您需要使用此新建的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,
```

若需要您能找到在Tomcat localhost_access_log的此请求。这是您如何在webservices日志查找它：

```
%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
```

这是在通知服务日志的请求。注意到HTTP/1.1 200 ok。

Note:思科通知日志只是作为提供情报的目的。如果启用思科精良通知记录日志，影响性能。

```
>> "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]"
```

即然通知服务有请求，张贴此用户的信息。这是从去客户端的通知服务日志的POST ：

```
>> "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]"
```

此**XMPP事件**，是在本例中的**代理程序2001年**，发送给所有订阅客户端。在客户端的Javascript接收**XMPP事件**，并且事件发送到在客户端内的小配件。这是显示答复的内容的clientlogs ：

```
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>
```

执行洛金

现在客户端准备执行登录。注意**RequestID**。RequestID在请求的正文发送。您使用此RequestID为了跟随登录请求到**其余API > CTI >其余API >通知服务>答复**回到客户端。此请求是PUT，因此意味着客户端请求一次更新或一更改向其当前状态。

```
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>
```

精良其余API收到从客户端的此请求。然后， API发送**SetAgentStateReq**对CTI服务器。

```
%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服务器收到请求。

```
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
```

一旦代理程序登陆以NOT_READY状态， CTI服务器发送AGENT_STATE-EVENT对精良。

```
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
```

这是接收从CTI服务器的事件的webservices日志。切记您首先看到从CTI服务器的**原始**消息，您然后看到**解码的**消息。

```
%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
```

即然精良接收从CTI服务器的AgentStateEvent，事件需要**发布**到通知服务，以便客户端接收更新。代理程序的唯一方法能知道状态更改是通过接收此**XMPP事件**。精良转换AgentStateEvent对XMPP并且发送XMPP对通知服务。注意事件是PUT和RequestID在有效负载。

```
%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
```

这里，通知服务接收更新。即使消息说**失败路由数据包到JID**，信息事件发布传送给用户。

```
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">
<Update>

这是消息的正文：

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

和前面，XMPP消息由客户端接收并且传送到客户端的小配件。注意客户端接收与原始RequestID的事件在消息。

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>

现在客户端顺利地登陆。

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

注销代码，原因代码，电话簿

现在客户端需要获取代理程序特定数据，例如注销代码、原因代码和电话簿。这是要求该信息做对客户端。

Container : SignIn._triggerLoggedIn(): **Successfully logged in**!18:40:05
同样逻辑应用对这些请求。记住精良原因代码和电话簿在精良数据库存储，不在UCCE。