Understand and Troubleshoot Finesse BOSH Implementation

Contents

Introduction

This document describes the architecture behind Finesse connections that use BOSH and how BOSH connection problems can be diagnosed.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Cisco Finesse
- Unified Contact Center Enterprise (UCCE)
- Unified Contact Center Express (UCCX)
- Web browser developer tools
- Windows and/or Mac administration

Components Used

The information in this document is based on these software and hardware versions:

- Cisco Finesse 9.0(1) - 11.6(1)
- UCCX 10.0(1) - 11.6(2)

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

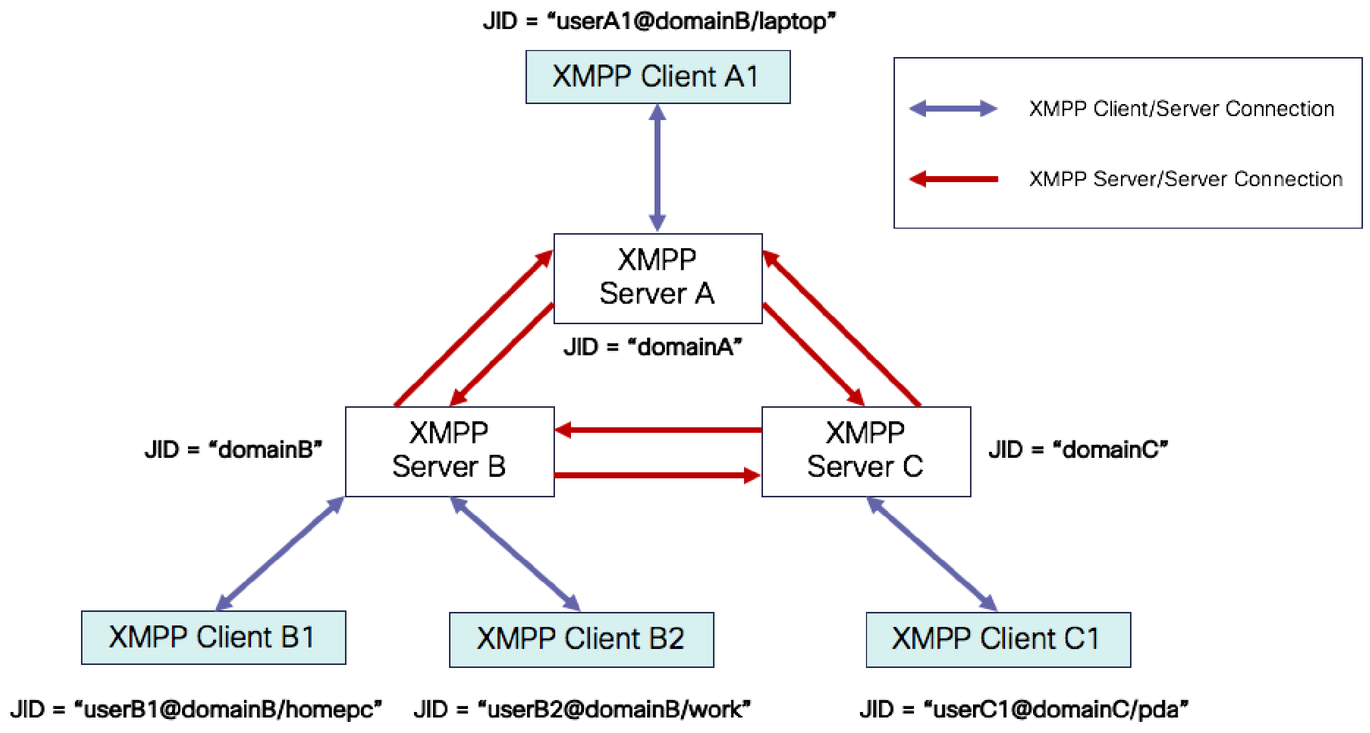
Connections that use Bidirectional-streams Over Synchronous HTTP are called BOSH.

Understand Finesse BOSH Implementation

Understand XMPP

Extensible Messaging and Presence Protocol (XMPP) (also known as Jabber) is a stateful protocol in a client-server model. XMPP allows for fast delivery of small pieces of structured eXtensible Markup Language (XML) data from one entity to another. XMPP/Jabber is extensively used in instant messaging (IM) and presence applications.

All XMPP entities are identified by their Jabber ID (JID).


JID = "userA1@domainB/laptop"
XMPP Client A1
XMPP Client/Server Connection
XMPP Server/Server Connection
XMPP Server A
JID = "domainA"
JID = "domainB"
XMPP Server B
XMPP Server C
JID = "domainC"
XMPP Client B1
XMPP Client B2
XMPP Client C1
JID = "userB1@domainB/homepc"
JID = "userB2@domainB/work"
JID = "userC1@domainC/pda"

JID Addressing Scheme: user@domain/resource

user	client username on the XMPP server or name of the conference room
domain	XMPP server fully qualified domain name (FQDN)
resource	identifier of the user's specific entity/endpoint (for example, laptop, smartphone, and so on), a session identifier, or pubsub node name

Note: All three JID components are not used in all cases. A server would typically just be defined by the domain, a conference room defined by user@domain, and a client by user@domain/resource.

XMPP messages are called stanzas. There are three core stanzas in XMPP:

1. <message>: one direction, one recipient

2. <presence>: one direction, publish to many

3. <iq>: info/query - request/response

All stanzas have to and from addresses and most stanzas also have type, id, and xml:langattributes.

Stanza Attribute	**Purpose**
to	destination JID

from	source JID
type	purpose of the message
id	unique identifier used to link a request with a response for <iq> stanzas
xml:lang	defines the default language for any human readable XML in the stanza

Example XMPP Message

```
<message to='person1@example' from='person2@example' type='chat'>
  <subject> Team meeting </subject>
  <body>Hey, when is our meeting today? </body>
  <thread>A4567423</thread>
</message>
```

XMPP Implementation with Finesse

If a web application needs to work with XMPP, multiple issues arise. Browsers don't support XMPP over Transmission Control Protocol (TCP) natively, so all XMPP traffic must be handled by a program which runs inside the browser. Web servers and browsers communicate via HyperText Transfer Protocol (HTTP) messages, so Finesse and other web applications wrap XMPP messages inside of HTTP messages.

The first difficulty with this approach is that HTTP is a stateless protocol. This means that each HTTP request is not related to any other request. However, this problem can be addressed by applicative means-- for example through the use of cookies/post data.

The second difficulty is the unidirectional behavior of HTTP. Only the client sends requests, and the server can only respond. The server's inability to push data makes it unnatural to implement XMPP over HTTP.

This problem does not exist in the original XMPP Core specification (RFC 6120), where XMPP is bound to TCP. However, if you want to address the problem with XMPP bound to HTTP, for example, because Javascript can send HTTP requests, there are two possible solutions. Both require a bridge between HTTP and XMPP.

The proposed solutions are:

1. Polling (legacy protocol): repeated HTTP requests asking for new data defined in XEP-0025: Jabber HTTP Polling

2. Long polling also is known as BOSH: transport protocol that emulates the semantics of a long-lived, bidirectional TCP connection between two entities by efficiently using multiple synchronous HTTP request/response pairs without requiring the use of frequent polling defined in XEP-0124: HTTP Binding and extended by XEP-0206: XMPP Over BOSH

Finesse implements BOSH as it is quite efficient from the server load point of view and traffic-wise. The reason to use BOSH is to cover up the fact that the server does not have to respond as soon there is a request. The response is delayed up to a specified time until the server has data for the client, and then it is sent as a response. As soon as the client gets the response, the client makes a new request and so forth.

The Finesse desktop client (web application) establishes a stale BOSH connection over TCP port 7443 every 30 seconds. After 30 seconds, if there are no updates from the Finesse Notification Service, the Notification service sends an HTTP reply with a 200 OK and a (nearly) empty response body. If the Notification Service has an update on the presence of an agent or a dialog (call) event, for example, the data is sent immediately to the Finesse web client.

Example Finesse XMPP Reqeust/Response

This example shows the first XMPP message request response shared between the Finesse client and Finesse server to set up the BOSH connection.

```
Finesse client request:
<body xmlns="http://jabber.org/protocol/httpbind" xml:lang="en-US" xmlns:xmpp="urn:xmpp:xbosh" hold="1"

Finesse server response:
<body xmlns="http://jabber.org/protocol/httpbind" xmlns:stream="http://etherx.jabber.org/streams" authi
```

To summarize:

1. The Finesse web client has a stale HTTP connection (http-bind) set up to the Finesse server via TCP port 7443. This is known as a BOSH long poll.

2. The Finesse Notification Service is a presence service that posts updates regarding the state of an agent, call, and so on.

3. If the Notification service has an update, it replies to the http-bind request with the state update as an XMPP message in the HTTP response body.

4. If there are no state updates 30 seconds after receiving the http-bind request, the Notification Service replies without any state updates to allow the Finesse web client to send another http-bind request. This serves as a way for the Notification service to know that the Finesse web client is still able to connect to the Notification Service and that the agent did not close their browser or put their computer to sleep, and so on.

Understand Finesse XMPP Messages and XMPP Nodes

Finesse also implements XMPP specification XEP-0060: Publish-Subscribe. The purpose of this specification is to allow the XMPP server (Notification service) to get information published to XMPP nodes (topics) and then to send XMPP events to entities subscribed to the node. In the case of Finesse, the Computer Telephony Integration (CTI) server sends CTI messages to the Finesse web service to tell Finesse about configuration updates such as, but not limited to, agent or Contact Service Queue (CSQ) creation or information about a call. This information is then converted into an XMPP message that the Finesse web service publishes to the Finesse Notification service. The Finesse Notification service then sends XMPP over BOSH messages to agents that are subscribed to certain XMPP nodes.

Some of Finesse API objects that are defined in the Finesse Web Services Developer Guide are XMPP nodes. Agent and supervisor Finesse web clients can subscribe to event updates for some of these XMPP Nodes in order to have up-to-date information about real-time events (such as call events, state events, and so on). This table shows the XMPP nodes that are pubsub enabled.

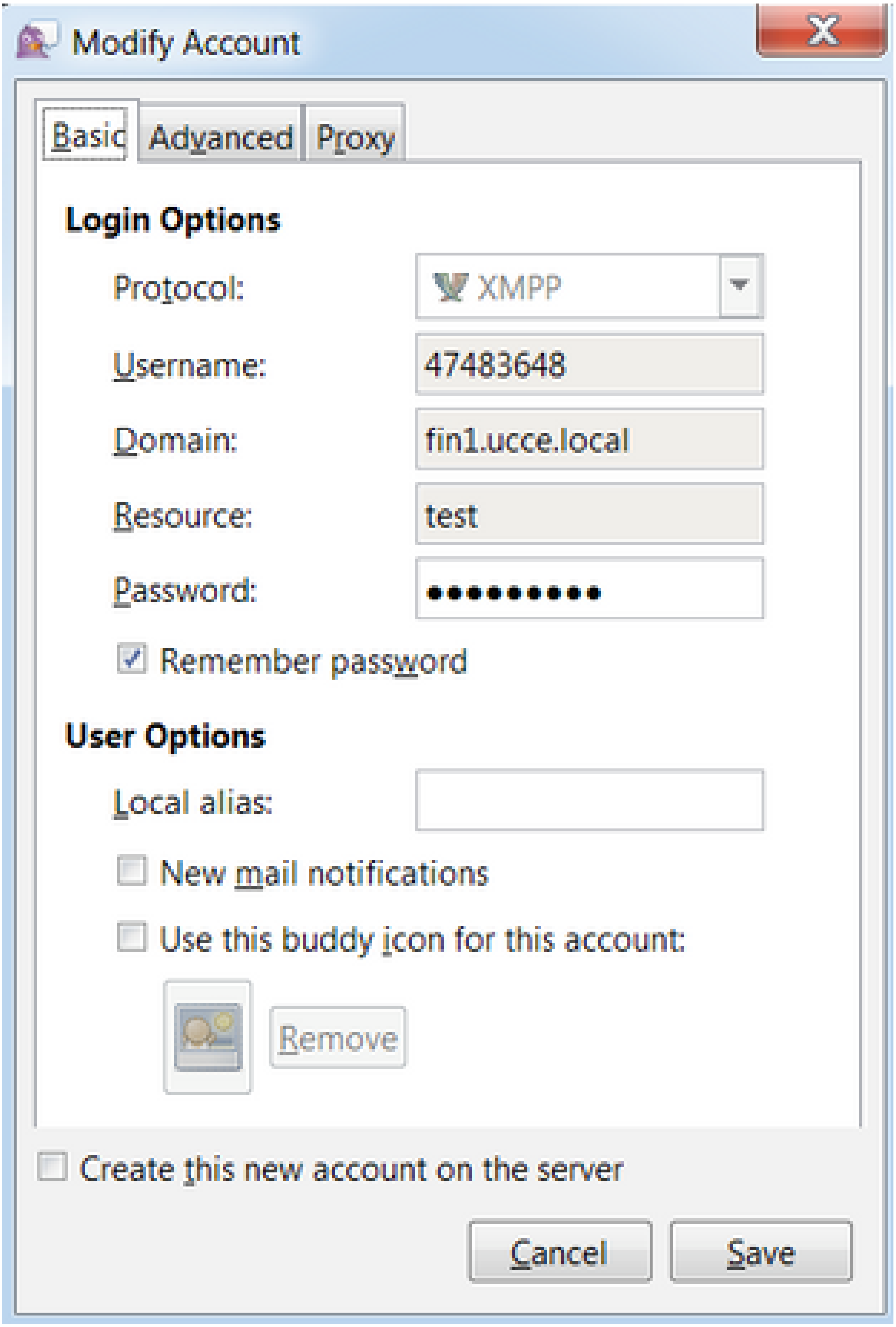
Finesse API Object	**Purpose**	**Subscription**
/finesse/api/User/<LoginID>	Shows the state and team mapping of the agent	Agents and Supervisors
/finesse/api/User/<LoginID>/Dialogs	Shows the call(s) handled by the agent	Agents and Supervisors
/finesse/api/User/<LoginID>/ClientLog	Used to capture client logs from the **Send Error Report** button	Agents and Supervisors
/finesse/api/User/<LoginID>/Queue/<queueID>	Shows queue statistics data (if enabled)	Agents and Supervisors
/finesse/api/Team/<TeamID>/Users	Shows the agents that belong to a certain team including state information	Supervisors
/finesse/api/SystemInfo	Shows the state of the Finesse server. Used to determine if failover is needed	Agents and Supervisors

Example 1: Use Pidgin to View Finesse XMPP Nodes

Step 1. Download and install the XMPP client Pidgin.

Step 2. Navigate to **Accounts > Modify > Basic** and configure the **Login Options**:

- Protocol: XMPP
- Username: LoginID for any agent
- Domain: FQDN of Finesse server
- Resource: Placeholder - any value can be used, for example, test
- Password: Agent password
- Check the **Remember password** checkbox



Login Options

Protocol:	XMPP
Username:	47483648
Domain:	fin1.ucce.local
Resource:	test
Password:	••••••••••

[✓] Remember password

User Options

Local alias:	

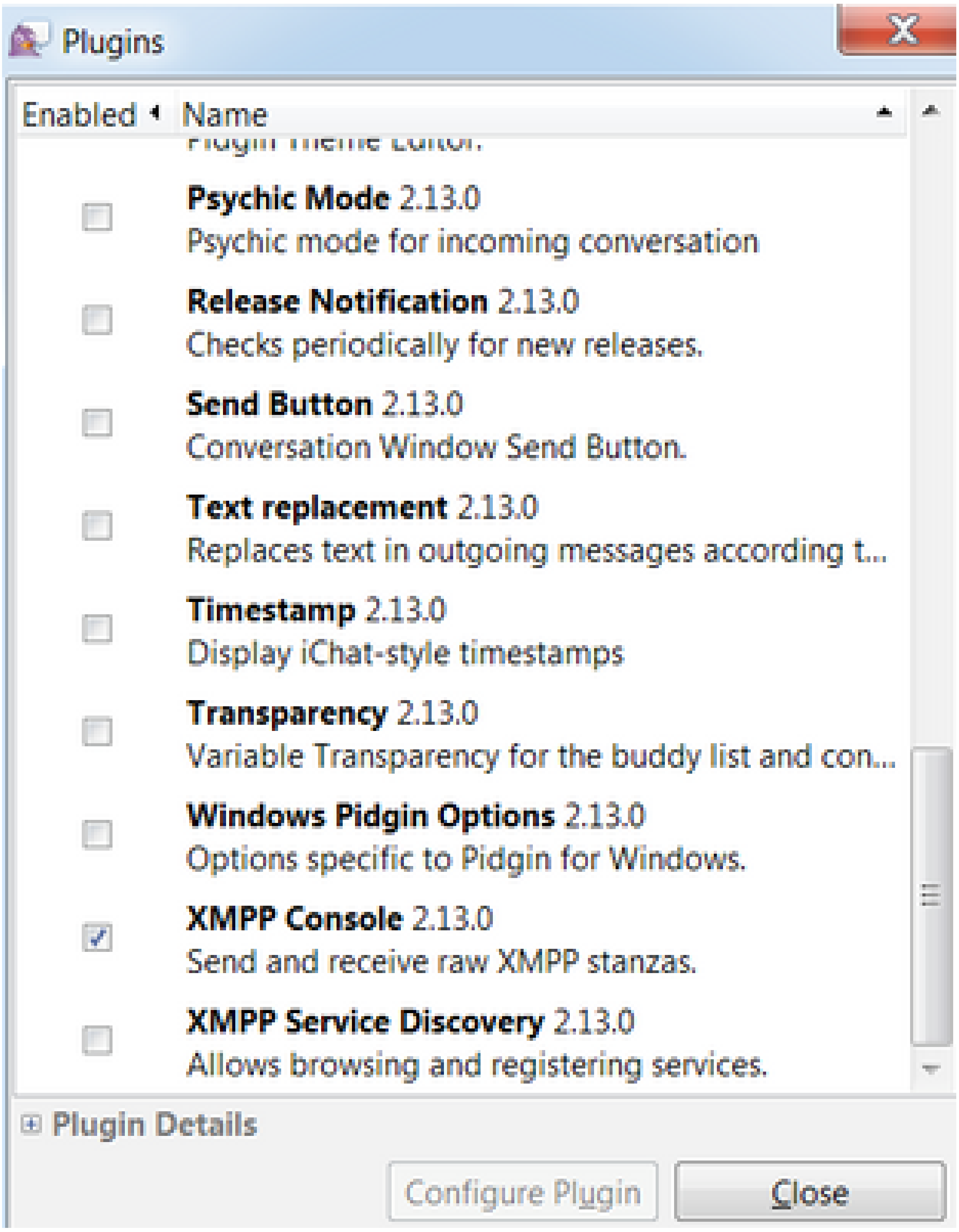
[ ] New mail notifications

[ ] Use this buddy icon for this account:

[ ] Create this new account on the server

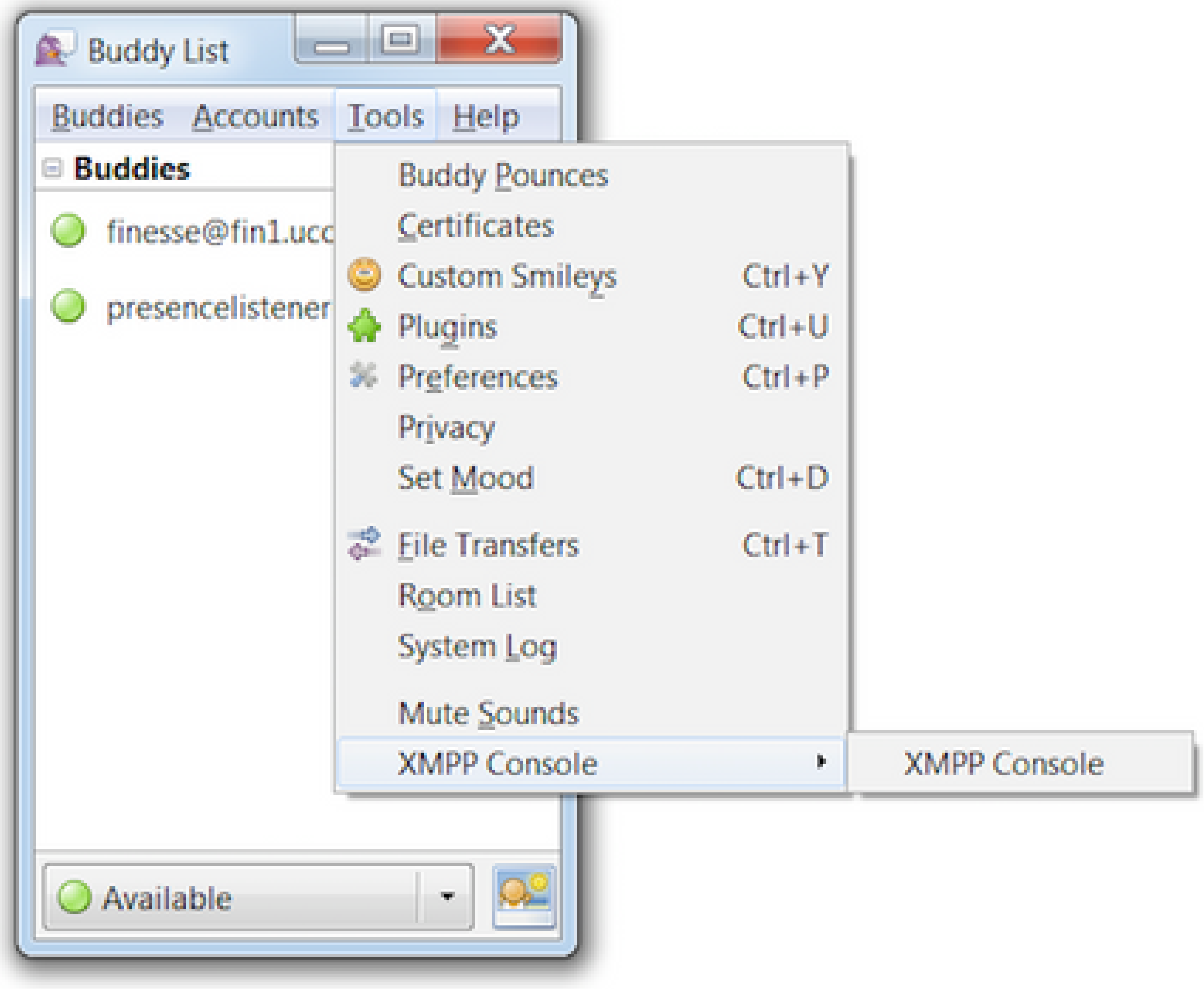
: Port 5222 is used only because Finesse web clients can use port 7443 to connect to the Notification Service.

Step 4. Navigate to **Tools > Plugins** and enable the XMPP Console.



Enabled	Name
☐	**Psychic Mode** 2.13.0 Psychic mode for incoming conversation
☐	**Release Notification** 2.13.0 Checks periodically for new releases.
☐	**Send Button** 2.13.0 Conversation Window Send Button.
☐	**Text replacement** 2.13.0 Replaces text in outgoing messages according t...
☐	**Timestamp** 2.13.0 Display iChat-style timestamps
☐	**Transparency** 2.13.0 Variable Transparency for the buddy list and con...
☐	**Windows Pidgin Options** 2.13.0 Options specific to Pidgin for Windows.
✓	**XMPP Console** 2.13.0 Send and receive raw XMPP stanzas.
☐	**XMPP Service Discovery** 2.13.0 Allows browsing and registering services.

Step 5. Navigate to **Tools** > **XMPP Console** > **XMPP Console** to open the XMPP Console.

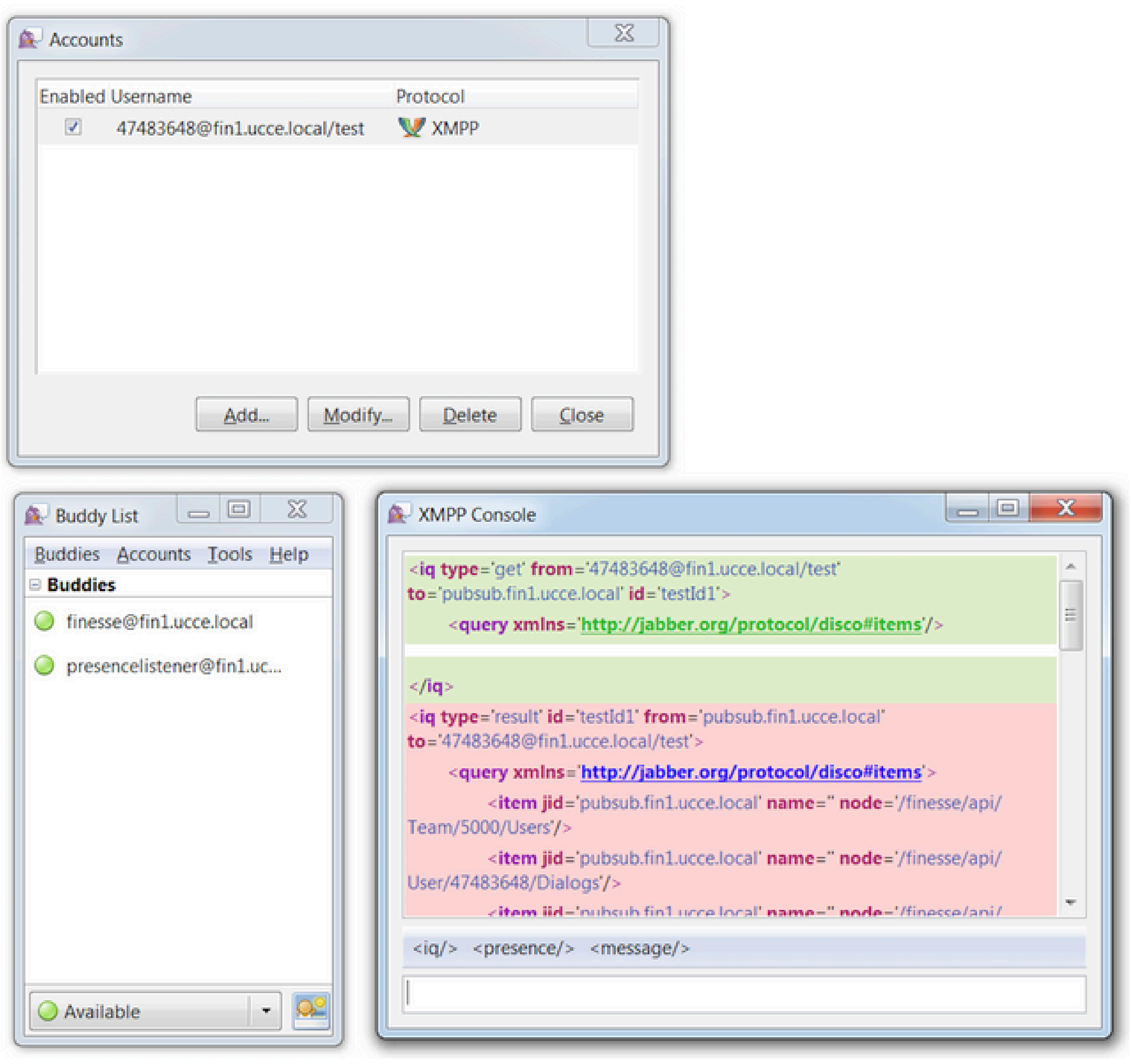


Step 6. Execute this **<iq>** message to see all of the XMPP nodes that exist.

```
<iq type='get' from='<loginID>@<Finesse_FQDN>/test' to='pubsub.<Finesse_FQDN>' id='testId1'
<query xmlns='http://jabber.org/protocol/disco#items'/>
</iq>
```

For example:

```
<iq type='get' from='47483648@fin1.ucce.local/test' to='pubsub.fin1.ucce.local' id='testId1'>
<query xmlns='http://jabber.org/protocol/disco#items'/>
</iq>
```



Enabled	Username	Protocol
✓	47483648@fin1.ucce.local/test	XMPP

finesse@fin1.ucce.local

presencelistener@fin1.uc...

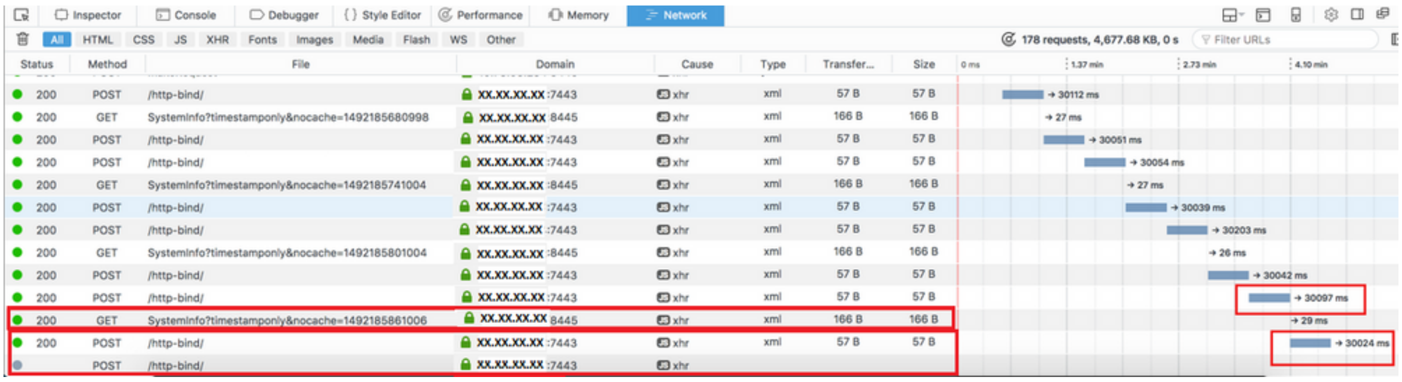
```
<iq type='get' from='47483648@fin1.ucce.local/test'
to='pubsub.fin1.ucce.local' id='testId1'>
      <query xmlns='http://jabber.org/protocol/disco#items'/>

</iq>
<iq type='result' id='testId1' from='pubsub.fin1.ucce.local'
to='47483648@fin1.ucce.local/test'>
      <query xmlns='http://jabber.org/protocol/disco#items'>
            <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/
Team/5000/Users'/>
            <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/
User/47483648/Dialogs'/>
```

In a lab environment with two agents and two CSQs configured, this output is contained in the Finesse response:

```
<iq type='result' id='testId1' from='pubsub.fin1.ucce.local' to='47483648@fin1.ucce.local/test'>
 <query xmlns='http://jabber.org/protocol/disco#items'>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/Team/5000/Users'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483648/Dialogs'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483651'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483651/ClientLog'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483649/Queues'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/Team/5001/Users'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483650/ClientLog'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/Queue/1'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483648/ClientLog'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483648/Queues'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483650'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483650/Queues'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483651/Dialogs'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483648'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/Team/1/Users'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483649'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483651/Queues'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483649/ClientLog'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/Queue/0'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483649/Dialogs'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/User/47483650/Dialogs'/>
 <item jid='pubsub.fin1.ucce.local' name='' node='/finesse/api/SystemInfo'/>
 </query>
</iq>
```

Example 2: Use Browser Developer Tools Network Tab to View HTTP Messages

Each browser has a set of developer tools. The Network tab of the developer tools shows the HTTP messages sent and received by the Finesse web client (browser). For example, this image shows how the Finesse web client sends a SystemInfo request which checks Finesse Tomcat status every minute as a failover check. Additionally, the http-bind messages from the BOSH connection are also displayed. The Finesse server sends back a response within 30 seconds if there are no updates to publish on the XMPP nodes the web client is subscribed to.

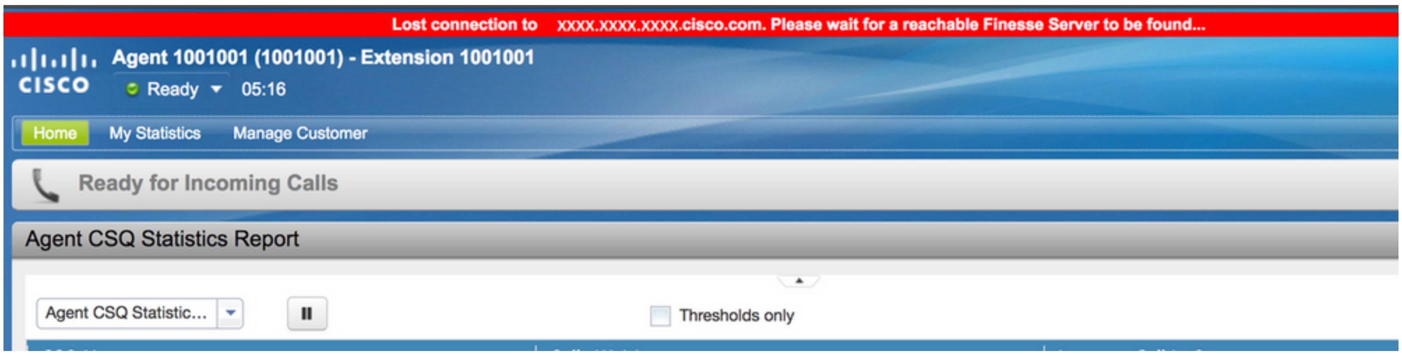


178 requests, 4,677.68 KB, 0 s

Status	Method	File	Domain	Cause	Type	Transfer...	Size	0 ms / 1.37 min / 2.73 min / 4.10 min
200	POST	/http-bind/	XX.XX.XX.XX :7443	xhr	xml	57 B	57 B	→ 30112 ms
200	GET	SystemInfo?timestamponly&nocache=1492185680998	XX.XX.XX.XX 8445	xhr	xml	166 B	166 B	→ 27 ms
200	POST	/http-bind/	XX.XX.XX.XX :7443	xhr	xml	57 B	57 B	→ 30051 ms
200	POST	/http-bind/	XX.XX.XX.XX :7443	xhr	xml	57 B	57 B	→ 30054 ms
200	GET	SystemInfo?timestamponly&nocache=1492185741004	XX.XX.XX.XX :8445	xhr	xml	166 B	166 B	→ 27 ms
200	POST	/http-bind/	XX.XX.XX.XX :7443	xhr	xml	57 B	57 B	→ 30039 ms
200	POST	/http-bind/	XX.XX.XX.XX :7443	xhr	xml	57 B	57 B	→ 30203 ms
200	GET	SystemInfo?timestamponly&nocache=1492185801004	XX.XX.XX.XX :8445	xhr	xml	166 B	166 B	→ 26 ms
200	POST	/http-bind/	XX.XX.XX.XX :7443	xhr	xml	57 B	57 B	→ 30042 ms
200	POST	/http-bind/	XX.XX.XX.XX :7443	xhr	xml	57 B	57 B	→ 30097 ms
200	GET	SystemInfo?timestamponly&nocache=1492185861006	XX.XX.XX.XX 8445	xhr	xml	166 B	166 B	→ 29 ms
200	POST	/http-bind/	XX.XX.XX.XX :7443	xhr	xml	57 B	57 B	→ 30024 ms
	POST	/http-bind/	XX.XX.XX.XX :7443	xhr				

Troubleshoot BOSH Disconnect Error Message

When a BOSH disconnect occurs, the error Lost connection to {Finesse Server FQDN}. Please wait for a reachable Finesse Server to be found... is displayed in a red banner at the top of the Finesse desktop.


Lost connection to xxxx.xxxx.xxxx.cisco.com. Please wait for a reachable Finesse Server to be found...

Agent 1001001 (1001001) - Extension 1001001

Ready 05:16

Ready for Incoming Calls

Agent CSQ Statistics Report

This message is displayed because at this time, no XMPP subscription events can be received from the Cisco Finesse Notification Service. Hence, state information and call details cannot be displayed on the agent desktop.

For UCCX, 60 seconds after the browser disconnects, the agent is put into a Logout state. The agent can be in the Ready or Not Ready state for the logout to happen.

For UCCE, Finesse takes up to 120 seconds to detect when an agent closes the browser or the browser crashes and Finesse waits 60 seconds before sending a forced logout request to the CTI server which causes the CTI server to put the agent in a Not Ready state. Under these conditions, Finesse can take up to 180 seconds to sign out the agent. Unlike in UCCX, the agent moves into a Not Ready state instead of the Logout state.

Note: The CTI disconnect Not Ready vs. Logout state behavior in UCCE is controlled by the PG /LOAD parameter. Per the Release Notes for Unified Contact Center Enterprise & Hosted Release 10.0(1), the /LOAD parameter is deprecated starting in UCCE 10.0.

For more information on UCCE Finesse Desktop behavior, refer to the Desktop Behavior section of the Cisco Finesse Failover Mechanisms chapter in the Cisco Finesse Administration Guide.

Note: Timer values can change in the future as per the product requirement.

Log Analysis

The Finesse and UCCX Notification service logs can be collected via RTMT or via the CLI:

file get activelog /desktop recurs compress

Debug Notification Service Logs

Note: Set debug level logs only while you reproduce an issue. Turn off the debugs after the issue has been reproduced.

Note: Finesse 9.0(1) does not have debug level logging. Debug level logging was introduced in Finesse 9.1(1). The process for enabling the logging is different in 9.1(1) compared to Finesse 10.0(1) - 11.6(1). For this process, consult the Finesse Administration and Serviceability guide.

Enable Notification Service debug logs of Unified Contact Center Express (UCCX), as shown:

<#root>

```
admin:

utils uccx notification-service log enable



WARNING! Enabling Cisco Unified CCX Notification Service logging can affect system performance
and should be disabled when logging is not required.

Do you want to proceed (yes/no)? yes

Cisco Unified CCX Notification Service logging enabled successfully.

NOTE: Logging can be disabled automatically if Cisco Unified CCX Notification Service is restarted.
```

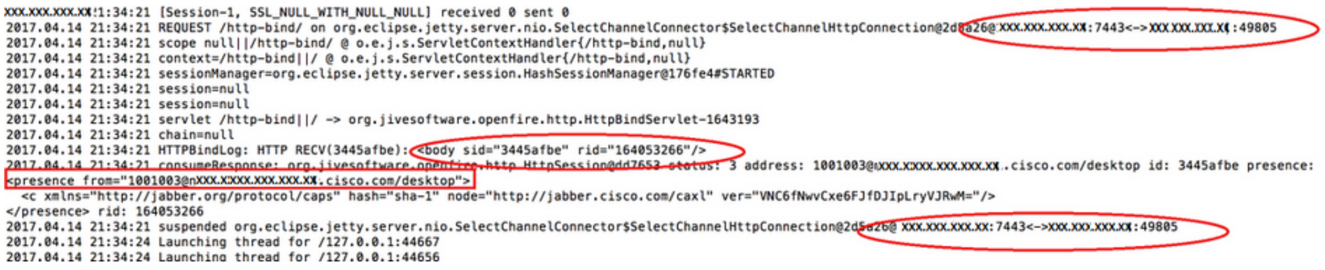
Enable Notification Service debug logs of Unified Contact Center Enterprise (UCCE) (Finesse Standalone), as shown:

<#root>

```
admin:

utils finesse notification logging enable



Checking that the Cisco Finesse Notification Service is started...
The Cisco Finesse Notification Service is started.

Cisco Finesse Notification Service logging is now enabled.

WARNING! Cisco Finesse Notification Service logging can affect system performance
and should be disabled when logging is not required.

Note: Logging can be disabled automatically if you restart the Cisco Finesse Notification Service
```

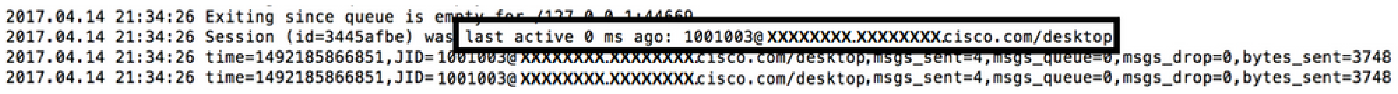
These logs are in the /desktop/logs/openfire folder and are named debug.log.

As shown in the image, the Notification Service (Openfire) debug.log shows the http binding with desktop along with the IP address and port of the agent PC.



```
XXX.XXX.XXX.XX!1:34:21 [Session-1, SSL_NULL_WITH_NULL_NULL] received 0 sent 0
2017.04.14 21:34:21 REQUEST /http-bind/ on org.eclipse.jetty.server.nio.SelectChannelConnector$SelectChannelHttpConnection@2d5a26@XXX.XXX.XXX.XX:7443<->XXX.XXX.XXX.XX:49805
2017.04.14 21:34:21 scope null||/http-bind/ @ o.e.j.s.ServletContextHandler{/http-bind,null}
2017.04.14 21:34:21 context=/http-bind||/ @ o.e.j.s.ServletContextHandler{/http-bind,null}
2017.04.14 21:34:21 sessionManager=org.eclipse.jetty.server.session.HashSessionManager@176fe4#STARTED
2017.04.14 21:34:21 session=null
2017.04.14 21:34:21 session=null
2017.04.14 21:34:21 servlet /http-bind||/ -> org.jivesoftware.openfire.http.HttpBindServlet-1643193
2017.04.14 21:34:21 chain=null
2017.04.14 21:34:21 HTTPBindLog: HTTP RECV(3445afbe): <body sid="3445afbe" rid="164053266"/>
2017.04.14 21:34:21 consumeResponse: org.jivesoftware.openfire.http.HttpSession@dd7653 status: 3 address: 1001003@XXX.XXXX.XXX.XXX.XX..cisco.com/desktop id: 3445afbe presence:
<presence from="1001003@nXXX.XXXX.XXX.XXX.XX.cisco.com/desktop">
  <c xmlns="http://jabber.org/protocol/caps" hash="sha-1" node="http://jabber.cisco.com/caxl" ver="VNC6fNwvCxe6FJfDJIpLryVJRwM="/>
</presence> rid: 164053266
2017.04.14 21:34:21 suspended org.eclipse.jetty.server.nio.SelectChannelConnector$SelectChannelHttpConnection@2d5a26@ XXX.XXX.XXX.XX:7443<->XXX.XXX.XXX.XX:49805
2017.04.14 21:34:24 Launching thread for /127.0.0.1:44667
2017.04.14 21:34:24 Launching thread for /127.0.0.1:44656
```

As shown in the image, the last active 0 ms shows that session is still active.



```
2017.04.14 21:34:26 Exiting since queue is empty for /127.0.0.1:44660
2017.04.14 21:34:26 Session (id=3445afbe) was last active 0 ms ago: 1001003@XXXXXXXX.XXXXXXXX.cisco.com/desktop
2017.04.14 21:34:26 time=1492185866851,JID=1001003@XXXXXXXX.XXXXXXXX.cisco.com/desktop,msgs_sent=4,msgs_queue=0,msgs_drop=0,bytes_sent=3748
2017.04.14 21:34:26 time=1492185866851,JID=1001003@XXXXXXXX.XXXXXXXX.cisco.com/desktop,msgs_sent=4,msgs_queue=0,msgs_drop=0,bytes_sent=3748
```

Openfire closing the idle session indicates the agent logout can trigger in 60 seconds where Finesse can send a forced logout with a reason code of 255 to the CTI server. The actual behavior of the desktop under these conditions depends on the setting for Logout on Agent Disconnect (LOAD) in UCCE. In UCCX, this is always the behavior.

If the Fineese client does not send http-bind messages to the Finesse server, the logs can show the session up time and show the session close.

```
2017.06.17 00:14:34 Session (id=f382a015) was last active 0 ms ago: 1001003@xxxxx.xxxx.xxx.cisco.com/de
2017.06.17 00:15:04 Session (id=f382a015) was last active 13230 ms ago: 1001003@xxxxx.xxxx.xxx.cisco.co
2017.06.17 00:15:34 Session (id=f382a015) was last active 43230 ms ago: 1001003@xxxxx.xxxx.xxx.cisco.co
2017.06.17 00:16:04 Session (id=f382a015) was last active 63231 ms ago: 1001003@xxxxx.xxxx.xxx.cisco.co

2017.06.17 00:17:04 Unable to route packet. No session is available so store offline. <message from="pu
```

Info Notification Service Logs

These logs are in the /desktop/logs/openfire folder and are named info.log. If the Fineese client does not send http-bind messages to the Finesse server, the logs can show the the session become inactive.

```
2017.06.17 00:16:04 Closing idle session (id=f382a015): 1001003@xxxxx.xxxx.xxx. cisco.com/desktop
after inactivity for more than threshold value of 60
2017.06.17 00:16:04 A session is closed for 1001003@xxxxx.xxxx.xxx. cisco.com/desktop
```

Webservices Logs

These logs are in the /desktop/logs/webservices folder and are named Desktop-webservices.YYYY-MM-DDTHH-MM-SS.sss.log. If the Fineese client does not send http-bind messages to the Finesse server within the specified amount of time, the logs can show the the agent presense become unavailable and 60 seconds later, a presense driven logout can occur.

```
0000001043: XX.XX.XX.XXX: Jun 17 2017 00:16:04.630 +0530: %CCBU_Smack Listener Processor (1)-6-PRESENCE_
0000000417: XX.XX.XX.XXX: Jun 17 2017 00:16:04.631 +0530: %CCBU_Smack Listener Processor (1)-6-UNSUBSCRI
0000001044: XX.XX.XX.XXX:  Jun 17 2017 00:16:04.631 +0530: %CCBU_Smack Listener Processor (1)-6-AGENT_PI
0000001051: XX.XX.XX.XXX:  Jun 17 2017 00:16:35.384 +0530: %CCBU_pool-8-thread-1-6-AGENT_PRESENCE_MONITO
0000001060: XX.XX.XX.XXX:: Jun 17 2017 00:17:04.632 +0530: %CCBU_CoreImpl-worker12-6-PRESENCE DRIVEN LOG
0000001061: XX.XX.XX.XXX:: Jun 17 2017 00:17:04.633 +0530: %CCBU_CoreImpl-worker12-6-MESSAGE_TO_CTI_SERV
1, workmode : 0, reason code: 255, forceflag :1, agentcapacity: 1, agentext: 1001003, agentid: 1001003,
0000001066:  XX.XX.XX.XXX:: Jun 17 2017 00:17:04.643 +0530: %CCBU_CTIMessageEventExecutor-0-6-DECODED_ME
skillGroupNumber=-1, skillGroupPriority=0, agentState=1 (LOGOUT), eventReasonCode=255, numFltSkillGroups
duration=null, nextAgentState=null, fltSkillGroupNumberList=[], fltSkillGroupIDList=[], fltSkillGroupPri
msgID=30, timeTracker={"id":"AgentStateEvent","CTI_MSG_RECEIVED":1497638824642,"CTI_MSG_DISPATCH":14976
Decoded Message to Finesse from backend cti server
```

Common Reasons for BOSH Disconnect

BOSH connections are setup by the web client, and the Finesse server determines if the agent presence is unavailable. These issues are almost always client side issues relating to the browser, agent computer, or network as the onus of starting up the connection is up to the client.

Problem - Agents Disconnect at Different Times (Client Side Issue)

Recommended Actions

Check for these issues:

1. Network issue:

- Review firewall rules and logs -- TCP port 7443 must not be blocked or throttled
- Use an HTTP web traffic sniffer like Fiddler® or Wireshark® to confirm that the browser sends http-bind requests over TCP port 7443 and receives responses

- Check all network devices/interfaces between the agent computer and the Finesse server for excessive delay or packet drops
  - Traceroute can be useful to determine the path and determine delays
    - On a Microsoft® Windows® PC: tracert {Finesse Server IP | Finesse Server FQDN}
    - On a Mac®: traceroute {Finesse Sever IP | Finesse Server FQDN}
    - On Cisco IOS® software, the interface statistics can be checked: show interfaces
      - Refer Troubleshooting Input Queue Drops and Output Queue Drops

- Collect Finesse Client logs for a test agent. Client logs can be collected in three ways:
  1. Browser web console logs
     - Firefox Web Console
     - Microsoft Edge Web Console
     - Chrome Web Console
  2. Press the Send Error Report button on the Finesse page and collect the Finesse server logs. The logs are located in /desktop/logs/clientlogs.
  3. Log in via https://<Finesse-FQDN>/desktop/locallog and collect the logs after the issue occurs.

Every minute, the client connects to the Finesse server to calculate drift and network latency:

```
<PC date-time with GMT offset>: : <Finesse FQDN>: <Finesse server date-time with offset>:
Header : Client: <date-time>, Server: <date-time>, Drift: <drift> ms, Network Latency (round trip): <RT

2019-01-11T12:24:14.586 -05:00: : fin1.ucce.local: Jan 11 2019 11:24:14.577 -0600: Header : Client: 201
```

In case of any log collection issues, refer to Troubleshoot Cisco Finesse Desktop Persistent Logging Problem

2. Unsupported browser and/or version:

Use supported browser/version and settings as per the compatibility matrices:

UCCE Compatibility Matrix

UCCX Compatibility Matrix

3. Browser stuck condition due to content/processing of other tab/window:

Check the agent workflow to see if they:

- Commonly have other tabs or windows up that are constantly running other real-time applications such as music/video streaming, WebSocket connections, custom Customer Relationship Management (CRM) web clients, etc
- Have a very large number of tabs or windows open
- Have disabled browser caching
- Have kept their browser running for a long time and do not close the browser at the end of the workday

4. Computer put to sleep:

Check to see if the agent puts their computer to sleep before logging out of Finesse or if their computer sleep setting timer is very low.

5. High CPU or high memory issue on client computer:

- If the agent browser runs in a shared environment such as Microsoft Windows Remote Desktop Services, Citrix® XenApp®, Citrix XenDesktop®, determine if the browser performance depends on the number of users running the browser at the same time
  - Ensure that the proper memory and CPU resources is configured based on the number of users

- Check computer resource utilization issues:
  - Windows:
    - Windows PowerShell Get-Counter command that checks % of CPU time, Megabytes of memory available, and % of memory in use every 2 seconds: Get-Counter -Counter "\Processor(_Total)\% Processor Time","\Memory\Available MBytes","\Memory\% Committed Bytes In Use" -SampleInterval 2 -Continuous
    - Alternative to using PowerShell to view the Windows' performance counters, Windows Perfomance Monitor can be used
    - Task Manager can be used to view live CPU and memory statistics globally and on a process-by-process basis
  - Mac:
    - Terminal Top command that checks live total CPU and memory: top
      - Check processes and sort by CPU utilization: top -o CPU
      - Check processes and sort by memory utilization: top -o MEM
    - Activity Monitor can be used to view live CPU and memory statistics globally and on a process-by-process basis

6. 3rd party gadgets performing unexpected, problematic activity in background:

Test the Finesse desktop behavior with all 3rd party gadgets removed.

7. NTP issue on server or client:

- Check **utils ntp status** on the Finesse publisher server to ensure the NTP server stratum is 4 or lower
- In the client logs, check drift and network latency

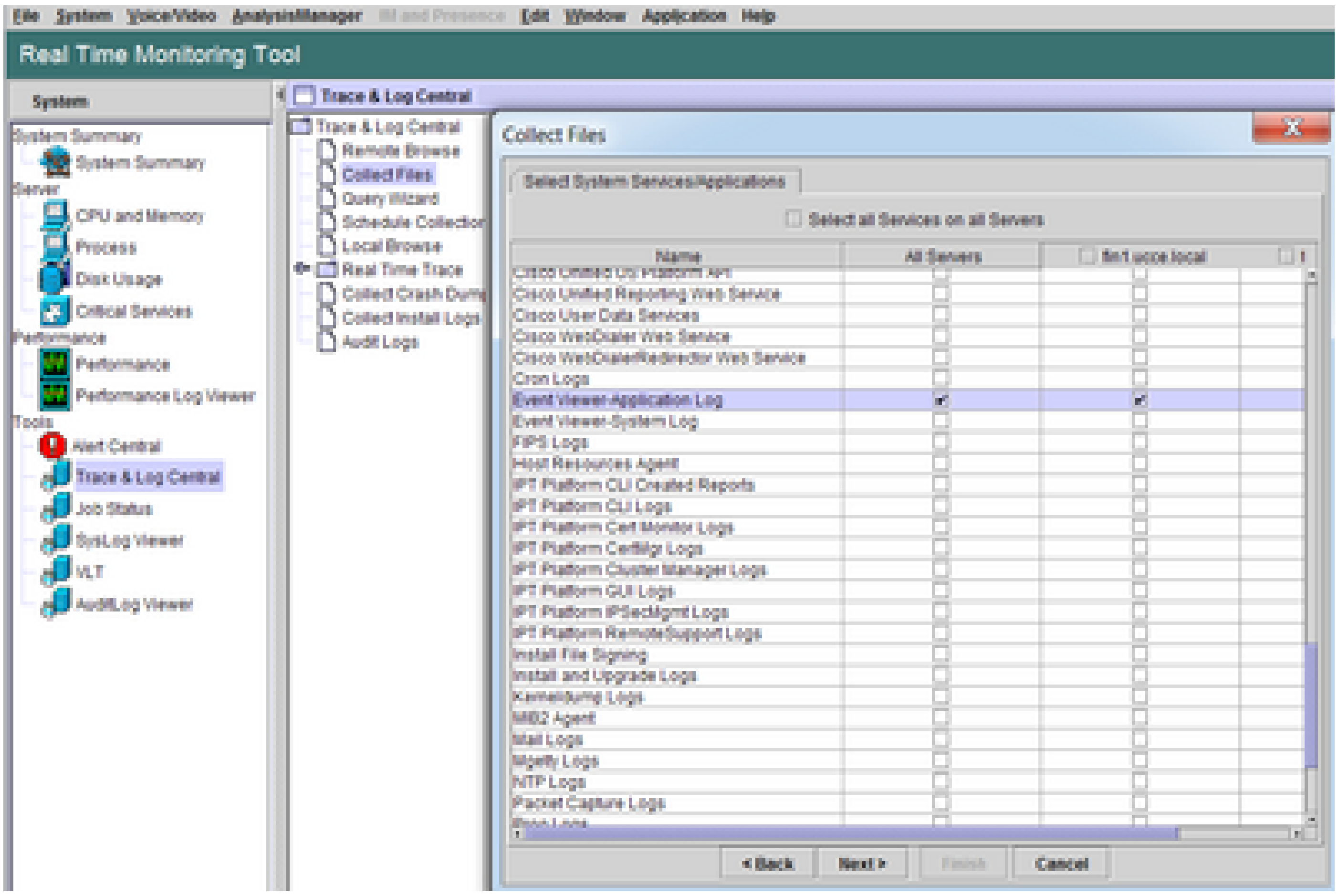
Problem - All Agents Disconnect at the Same Time (Server Side Issue)

Recommended Actions

Check for these issues:

1. Cisco Unified Communications Manager CTIManager service disconnect. If all CTIManager providers for UCCX are shutdown or crash, UCCX agents see the red banner error. UCCE agents do not see the red banner if this happens, but calls fail to route propertly to the agents.

- Check to see if the Cisco CTIManager service is started on the CUCM servers used as CTI providers
- Check to see if the Cisco CTIManager service crashed via the Event Viewer - Application logs on RTMT to see if the Cisco CTIManager service crashed
  - To collect event viewer logs on RTMT, navigte to **System > Tools > Trace and Log Central > Collect Files > Select System Services/Applications > Event Viewer-Application Log**.



Name	All Servers	fin1.uccx.local
Cisco Unified OS Platform API	☐	☐
Cisco Unified Reporting Web Service	☐	☐
Cisco User Data Services	☐	☐
Cisco WebDialer Web Service	☐	☐
Cisco WebDialerRedirector Web Service	☐	☐
Cron Logs	☐	☐
Event Viewer-Application Log	✓	✓
Event Viewer-System Log	☐	☐
FIPS Logs	☐	☐
Host Resources Agent	☐	☐
IPT Platform CLI Created Reports	☐	☐
IPT Platform CLI Logs	☐	☐
IPT Platform Cert Monitor Logs	☐	☐
IPT Platform CertMgr Logs	☐	☐
IPT Platform Cluster Manager Logs	☐	☐
IPT Platform GUI Logs	☐	☐
IPT Platform IPSecMgmt Logs	☐	☐
IPT Platform RemoteSupport Logs	☐	☐
Install File Signing	☐	☐
Install and Upgrade Logs	☐	☐
Kerneldump Logs	☐	☐
MIB2 Agent	☐	☐
Mail Logs	☐	☐
Mgetty Logs	☐	☐
NTP Logs	☐	☐
Packet Capture Logs	☐	☐
Prog Logs	☐	☐

- To collect the Event Viewer-Application logs on CLI: file get activelog /syslog/CiscoSyslog* abstime hh:mm:MM/DD/YY hh:mm:MM/DD/YY
- To view core dumps on the CLI: utils core active list

Note: Core dumps file names use the format: core.<ProcessID>.<SignalNumber>.<ProcessName>.<EpochTime>.
Example: core.24587.6.CTIManager.1533441238
Hence, the time of the crash can be determined from the epoch time.

---

2. Finesse/UCCX Notification Service stopped or crashed:

- Check the Event Viewer-Application Logs for Notification Service errors or to see if the service was stopped
- Check to see if the Notification service is up: utils service list
- Check the times the Notification service shut down: file search activelog /desktop/logs/openfire "Openfire stopped"
- Check the times the Notification service started: file search activelog /desktop/logs/openfire "HTTP bind service started"
- Check for Notification service memory dumps that resulted from a crash: file list activelog /desktop/logs/openfire/*.hprof
- Check to see if the Notification service is listening for traffic on TCP port 7443: show open ports regexp 7443.*LISTEN
- Check to see if these defects are applicable (these defects would cause login failure for agents logging in and for agents already logged in, those agents would see the red banner Finesse disconnect message):
  - Cisco bug ID CSCva72280 - Finesse Tomcat and Openfire Crash for invalid XML characters

  - Cisco bug ID CSCva72325 - UCCX: Finesse Tomcat and Openfire Crash for invalid XML characters

Restart Cisco Finesse Tomcat and Notification Service if a crash is suspected. This is only recommended in a network down situation, otherwise, these restarts disconnect agents from the Finesse server.

Steps for UCCE:

- utils service stop Cisco Finesse Tomcat
- utils service stop Cisco Finesse Notification Service
- utils service start Cisco Finesse Tomcat
- utils service start Cisco Finesse Notification Service

Steps for UCCX:

- utils service stop Cisco Finesse Tomcat
- utils service stop Cisco Unified CCX Notification Service
- utils service start Cisco Finesse Tomcat
- utils service start Cisco Unified CCX Notification Service

Use Fiddler

Configuring Fiddler can be a somewhat challenging task without understanding the steps needed and understanding how Fiddler works. Fiddler is a man-in-the-middle web proxy that stands between the Finesse client (web browser) and the Finesse server. Due to the connections secured between the Finesse client and Finesse server, this adds a layer of complexity to the Fiddler configuration in order to view secured messages.

Common Fiddler Issue

Since Fiddler stands in between the Finesse client and Finesse server, the Fiddler application needs to create signed certificates for all Finesse TCP ports that require certificates:

Cisco Finesse Tomcat service certificates

1. Finesse publisher server TCP 8445 (and/or 443 for UCCE)
2. Finesse subscriber server TCP 8445 (and/or 443 for UCCE)

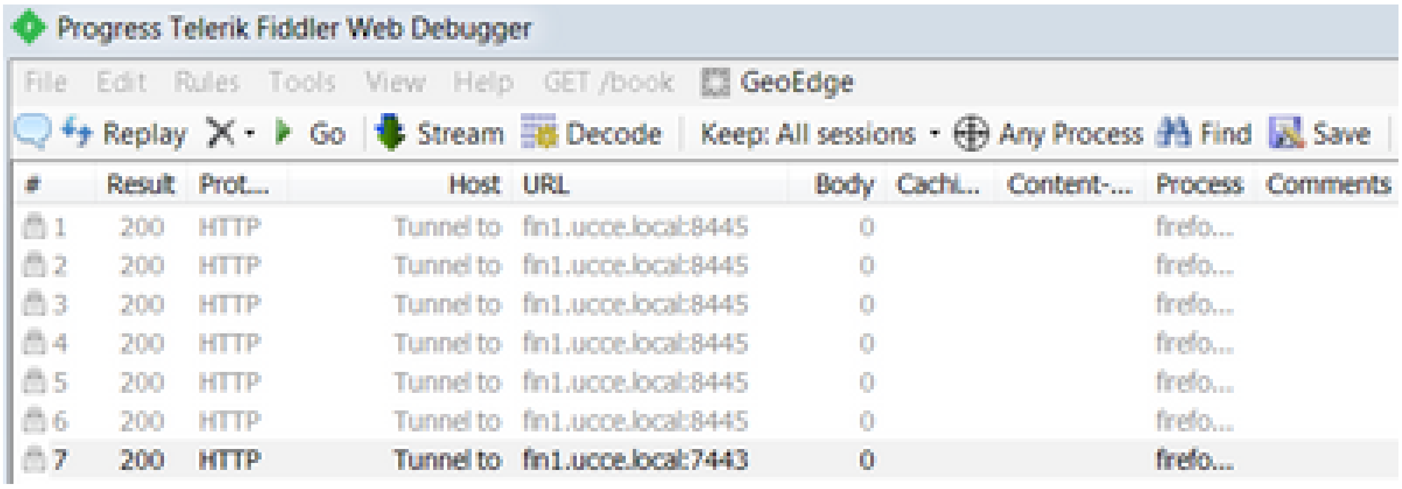
Cisco Finesse (Unified CCX) Notification Service certificates

1. Finesse publisher server TCP 7443
2. Finesse subscriber server TCP 7443

HTTPS decryption must be enabled for Fiddler to dynamically generate certificates on behalf of the Finesse server. This is not enabled by default.

If HTTPS decryption is not configured, the initial tunnel connection to the Notification service is seen, but the http-bind traffic is not. Fiddler only shows:

```
Tunnel to <Finesse server FQDN>:7443
```

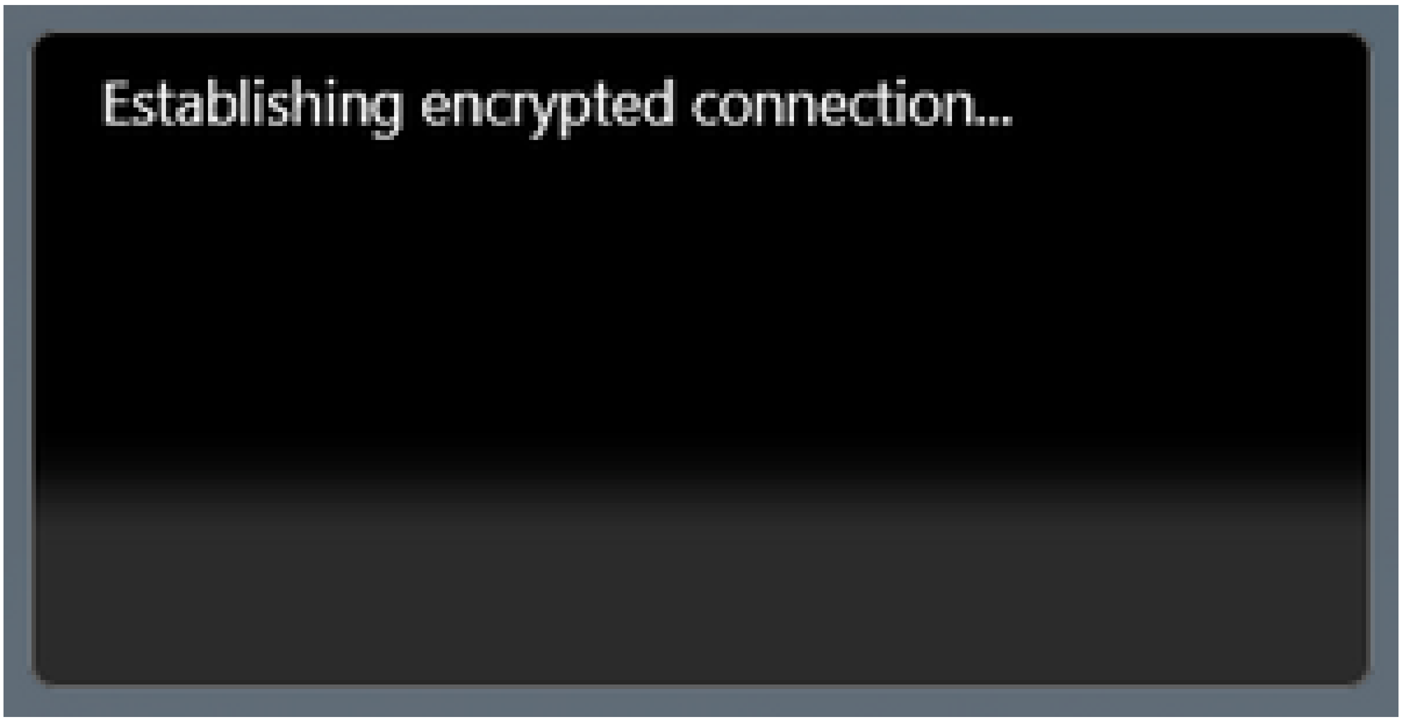

Progress Telerik Fiddler Web Debugger

File Edit Rules Tools View Help GET /book GeoEdge

Replay X Go Stream Decode Keep: All sessions Any Process Find Save

#	Result	Prot...	Host	URL	Body	Cachi...	Content-...	Process	Comments
1	200	HTTP	Tunnel to	fin1.ucce.local:8445	0			firefo...	
2	200	HTTP	Tunnel to	fin1.ucce.local:8445	0			firefo...	
3	200	HTTP	Tunnel to	fin1.ucce.local:8445	0			firefo...	
4	200	HTTP	Tunnel to	fin1.ucce.local:8445	0			firefo...	
5	200	HTTP	Tunnel to	fin1.ucce.local:8445	0			firefo...	
6	200	HTTP	Tunnel to	fin1.ucce.local:8445	0			firefo...	
7	200	HTTP	Tunnel to	fin1.ucce.local:7443	0			firefo...	

Then, the Finesse certificates signed by Fiddler must be trusted by the client. If these certificates are not trusted, moving past the Establishing encrypted connection... stage of Finesse login is not possible.


Establishing encrypted connection...

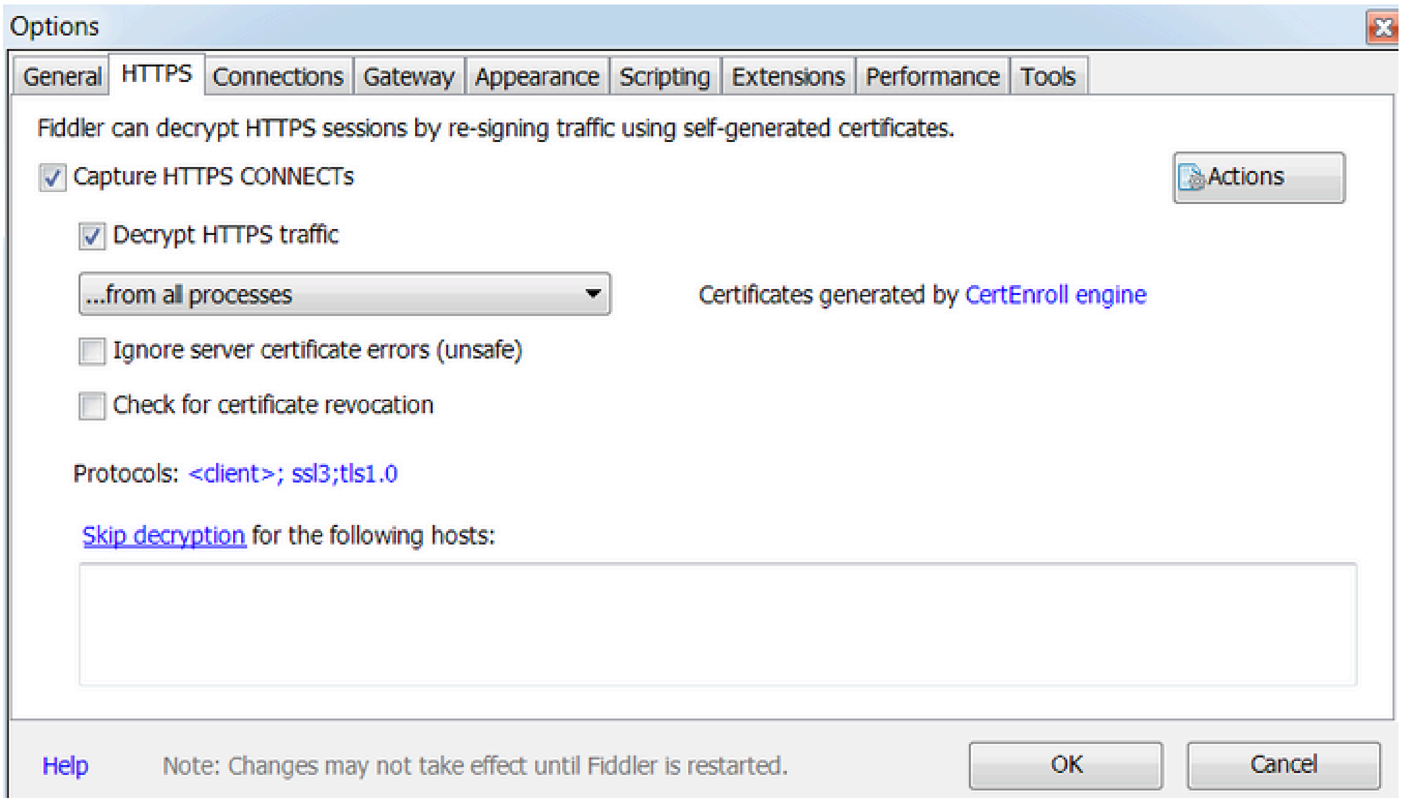
In some cases, accepting the certificate exceptions from the login does not work, and the certificates need to be trusted by the browser manually.

Example Configuration Steps

Caution: The example configuration provided is for Fiddler v5.0.20182.28034 for .NET 4.5and Mozilla Firefox 64.0.2 (32-bit) on Windows 7 x64 in a lab environment. These procedures can not generalize to all versions of Fiddler, all browsers, or all computer operating systems. If your network is live, ensure that you understand the potential impact of any configuration. Reference the official Fiddler documentation for more information.

Step 1. Download Fiddler

Step 2. Enable HTTPS decryption. Navigate to **Tools** > **Options** > **HTTPS** and check the **Decrypt HTTPS traffic** checkbox.



Fiddler can decrypt HTTPS sessions by re-signing traffic using self-generated certificates.

[✓] Capture HTTPS CONNECTs

[✓] Decrypt HTTPS traffic

...from all processes

Certificates generated by CertEnroll engine

[ ] Ignore server certificate errors (unsafe)

[ ] Check for certificate revocation

Protocols: <client>; ssl3;tls1.0

Skip decryption for the following hosts:

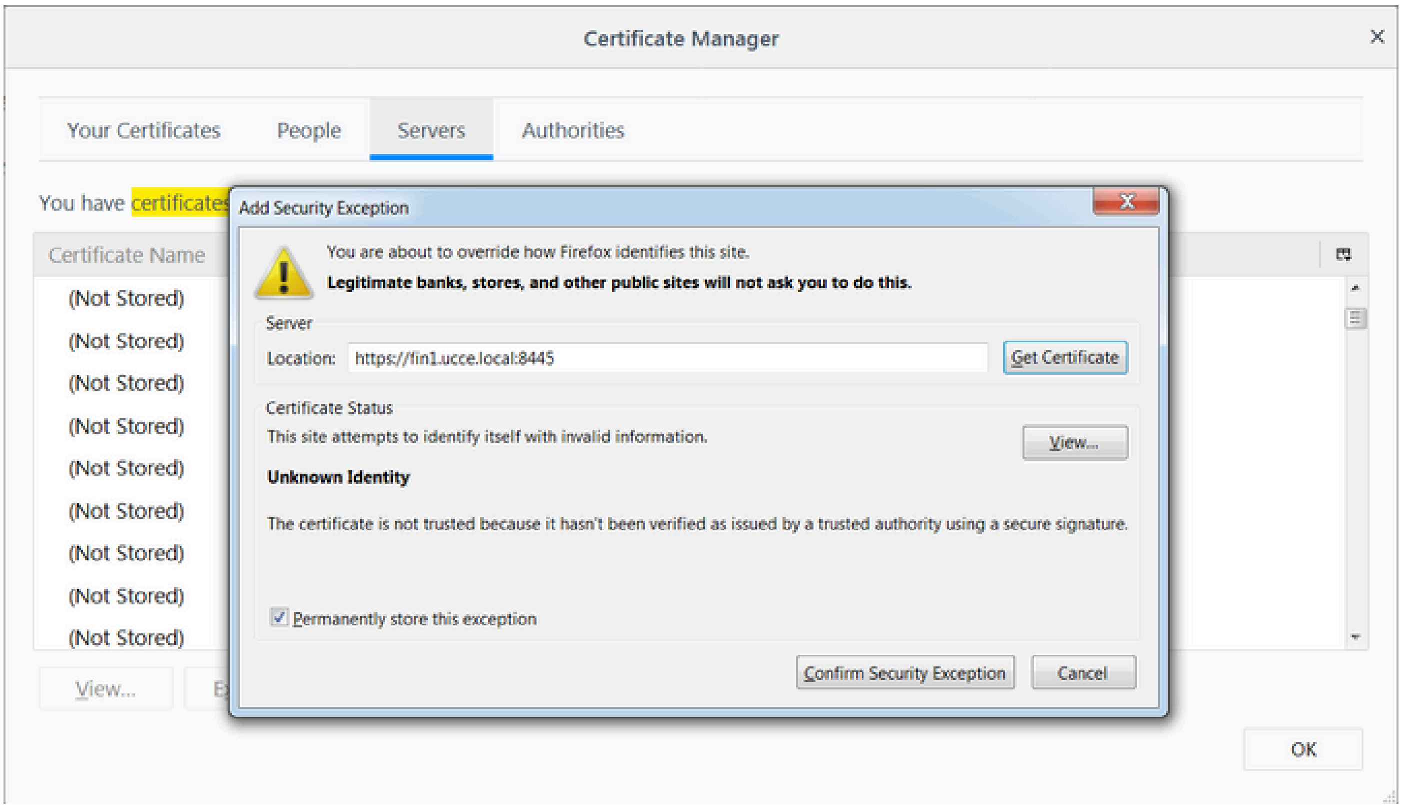
Note: Changes may not take effect until Fiddler is restarted.

Step 3. A warning message box opens to ask to trust the Fiddler Root Certificate. Select **Yes**.

Step 4. A warning message box opens with the message "You are about to install a certificate from a certification authority (CA) claiming to represent: DO_NOT_TRUST_FiddlerRoot... Do you want to install this certificate?". Select **Yes**.

Step 5. Manually add the Finesse publisher and subscriber certificates to the computer or browser certificate trust store. Ensure ports 8445, 7443, and (only for UCCE) 443. For example, on Firefox, this can be done simply without downloading certificates from the Finesse Operating System Administration page:

Options > **Find in Options** (search) > **Certificates** > **Servers** > **Add Exception** > **Location** > Enter https://<Finesse server>:port for the relevant ports for both Finesse servers.


Certificate Manager

Your Certificates | People | Servers | Authorities

You have certificates

Certificate Name

(Not Stored)
(Not Stored)
(Not Stored)
(Not Stored)
(Not Stored)
(Not Stored)
(Not Stored)
(Not Stored)
(Not Stored)

Add Security Exception

You are about to override how Firefox identifies this site.
Legitimate banks, stores, and other public sites will not ask you to do this.

Server
Location: https://fin1.ucce.local:8445

Certificate Status
This site attempts to identify itself with invalid information.

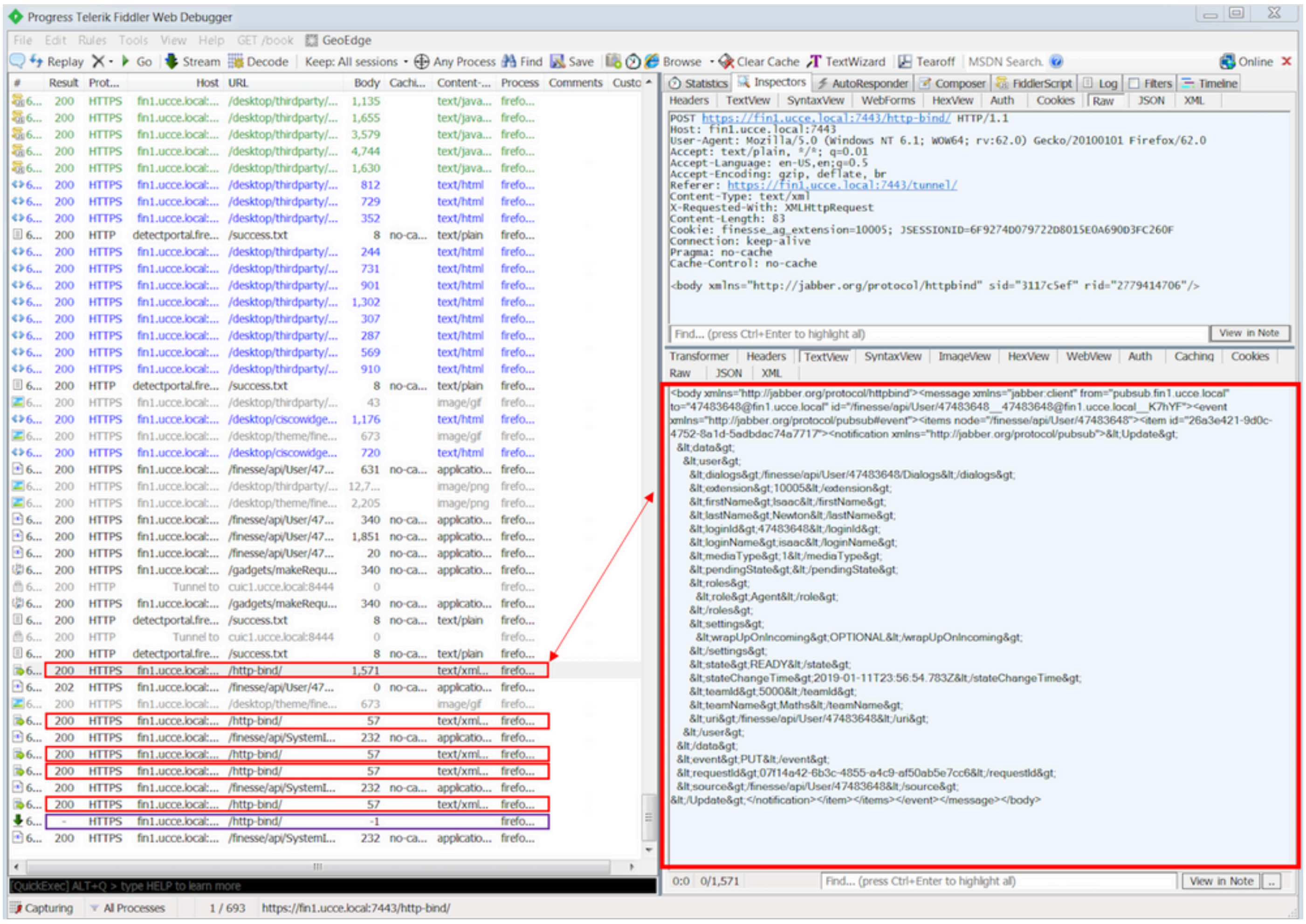
Unknown Identity

The certificate is not trusted because it hasn't been verified as issued by a trusted authority using a secure signature.

☑ Permanently store this exception

Step 6. Log into Finesse and see the http-bind messages leave the Finesse client to the Finesse Server via Fiddler.

In the example provided, the first 5 messages show http-bind messages that were responded to by the Finesse server. The first message contains 1571 bytes of data returned in the message body. The body contains an XMPP update regarding an agent event. The final http-bind message has been sent by the Finesse client, but has not gotten a response from the Finesse server. This can be determined when you see that the HTTP result is null (-) and the number of bytes in the response body is null (-1).


Progress Telerik Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search. Online
Statistics Inspectors AutoResponder Composer FiddlerScript Log Filters Timeline
Headers TextView SyntaxView WebForms HexView Auth Cookies Raw JSON XML
Transformer Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML
Find... (press Ctrl+Enter to highlight all) View in Note

```
#     Result  Prot...  Host                  URL                       Body    Cachi...  Content-...   Process
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...   1,135             text/java...  firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...   1,655             text/java...  firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...   3,579             text/java...  firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...   4,744             text/java...  firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...   1,630             text/java...  firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...     812             text/html     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...     729             text/html     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...     352             text/html     firefo...
6...  200     HTTP     detectportal.fire...  /success.txt                  8   no-ca...  text/plain    firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...     244             text/html     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...     731             text/html     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...     901             text/html     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...   1,302             text/html     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...     307             text/html     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...     287             text/html     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...     569             text/html     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...     910             text/html     firefo...
6...  200     HTTP     detectportal.fire...  /success.txt                  8   no-ca...  text/plain    firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...      43             image/gif     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/ciscowidge...    1,176             text/html     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/theme/fine...      673             image/gif     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/ciscowidge...      720             text/html     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /finesse/api/User/47...     631   no-ca...  applicatio... firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/thirdparty/...   12,7...           image/png     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/theme/fine...    2,205             image/png     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /finesse/api/User/47...     340   no-ca...  applicatio... firefo...
6...  200     HTTPS    fin1.ucce.local:...   /finesse/api/User/47...   1,851   no-ca...  applicatio... firefo...
6...  200     HTTPS    fin1.ucce.local:...   /finesse/api/User/47...      20   no-ca...  applicatio... firefo...
6...  200     HTTPS    fin1.ucce.local:...   /gadgets/makeRequ...        340   no-ca...  applicatio... firefo...
6...  200     HTTP              Tunnel to    cuic1.ucce.local:8444         0                           firefo...
6...  200     HTTPS    fin1.ucce.local:...   /gadgets/makeRequ...        340   no-ca...  applicatio... firefo...
6...  200     HTTP     detectportal.fire...  /success.txt                  8   no-ca...  text/plain    firefo...
6...  200     HTTP              Tunnel to    cuic1.ucce.local:8444         0                           firefo...
6...  200     HTTP     detectportal.fire...  /success.txt                  8   no-ca...  text/plain    firefo...
6...  200     HTTPS    fin1.ucce.local:...   /http-bind/               1,571             text/xml...   firefo...
6...  202     HTTPS    fin1.ucce.local:...   /finesse/api/User/47...       0   no-ca...  applicatio... firefo...
6...  200     HTTPS    fin1.ucce.local:...   /desktop/theme/fine...      673             image/gif     firefo...
6...  200     HTTPS    fin1.ucce.local:...   /http-bind/                  57             text/xml...   firefo...
6...  200     HTTPS    fin1.ucce.local:...   /finesse/api/SystemI...     232   no-ca...  applicatio... firefo...
6...  200     HTTPS    fin1.ucce.local:...   /http-bind/                  57             text/xml...   firefo...
6...  200     HTTPS    fin1.ucce.local:...   /http-bind/                  57             text/xml...   firefo...
6...  200     HTTPS    fin1.ucce.local:...   /finesse/api/SystemI...     232   no-ca...  applicatio... firefo...
6...  200     HTTPS    fin1.ucce.local:...   /http-bind/                  57             text/xml...   firefo...
6...  -       HTTPS    fin1.ucce.local:...   /http-bind/                  -1                           firefo...
6...  200     HTTPS    fin1.ucce.local:...   /finesse/api/SystemI...     232   no-ca...  applicatio... firefo...
```

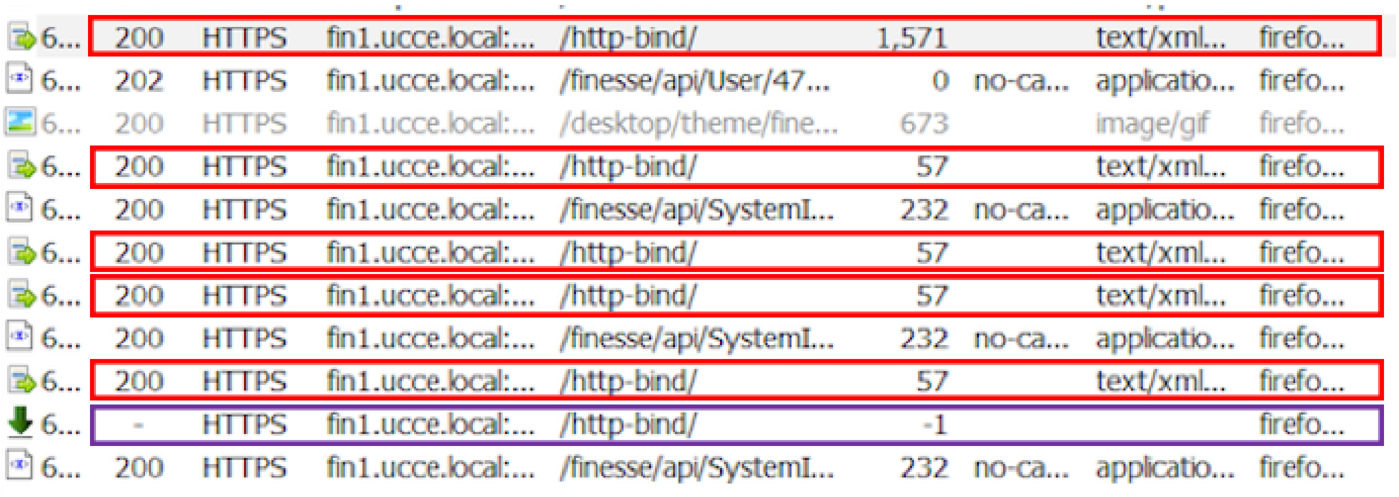
```
POST https://fin1.ucce.local:7443/http-bind/ HTTP/1.1
Host: fin1.ucce.local:7443
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:62.0) Gecko/20100101 Firefox/62.0
Accept: text/plain, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://fin1.ucce.local:7443/tunnel/
Content-Type: text/xml
X-Requested-With: XMLHttpRequest
Content-Length: 83
Cookie: finesse_ag_extension=10005; JSESSIONID=6F9274D079722D8015E0A690D3FC260F
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

<body xmlns="http://jabber.org/protocol/httpbind" sid="3117c5ef" rid="2779414706"/>
```

```
<body xmlns="http://jabber.org/protocol/httpbind"><message xmlns="jabber:client" from="pubsub.fin1.ucce.local"
to="47483648@fin1.ucce.local" id="/finesse/api/User/47483648__47483648@fin1.ucce.local__K7hYF"><event
xmlns="http://jabber.org/protocol/pubsub#event"><items node="/finesse/api/User/47483648"><item id="26a3e421-9d0c-
4752-8a1d-5adbdac74a7717"><notification xmlns="http://jabber.org/protocol/pubsub">&lt;Update&gt;
  &lt;data&gt;
   &lt;user&gt;
     &lt;dialogs&gt;/finesse/api/User/47483648/Dialogs&lt;/dialogs&gt;
     &lt;extension&gt;10005&lt;/extension&gt;
     &lt;firstName&gt;Isaac&lt;/firstName&gt;
     &lt;lastName&gt;Newton&lt;/lastName&gt;
     &lt;loginId&gt;47483648&lt;/loginId&gt;
     &lt;loginName&gt;isaac&lt;/loginName&gt;
     &lt;mediaType&gt;1&lt;/mediaType&gt;
     &lt;pendingState&gt;&lt;/pendingState&gt;
     &lt;roles&gt;
       &lt;role&gt;Agent&lt;/role&gt;
     &lt;/roles&gt;
     &lt;settings&gt;
       &lt;wrapUpOnIncoming&gt;OPTIONAL&lt;/wrapUpOnIncoming&gt;
     &lt;/settings&gt;
     &lt;state&gt;READY&lt;/state&gt;
     &lt;stateChangeTime&gt;2019-01-11T23:56:54.783Z&lt;/stateChangeTime&gt;
     &lt;teamId&gt;5000&lt;/teamId&gt;
     &lt;teamName&gt;Maths&lt;/teamName&gt;
     &lt;uri&gt;/finesse/api/User/47483648&lt;/uri&gt;
   &lt;/user&gt;
  &lt;/data&gt;
  &lt;event&gt;PUT&lt;/event&gt;
  &lt;requestId&gt;07f14a42-6b3c-4855-a4c9-af50ab5e7cc6&lt;/requestId&gt;
  &lt;source&gt;/finesse/api/User/47483648&lt;/source&gt;
&lt;/Update&gt;</notification></item></items></event></message></body>
```

```
0:0  0/1,571
```

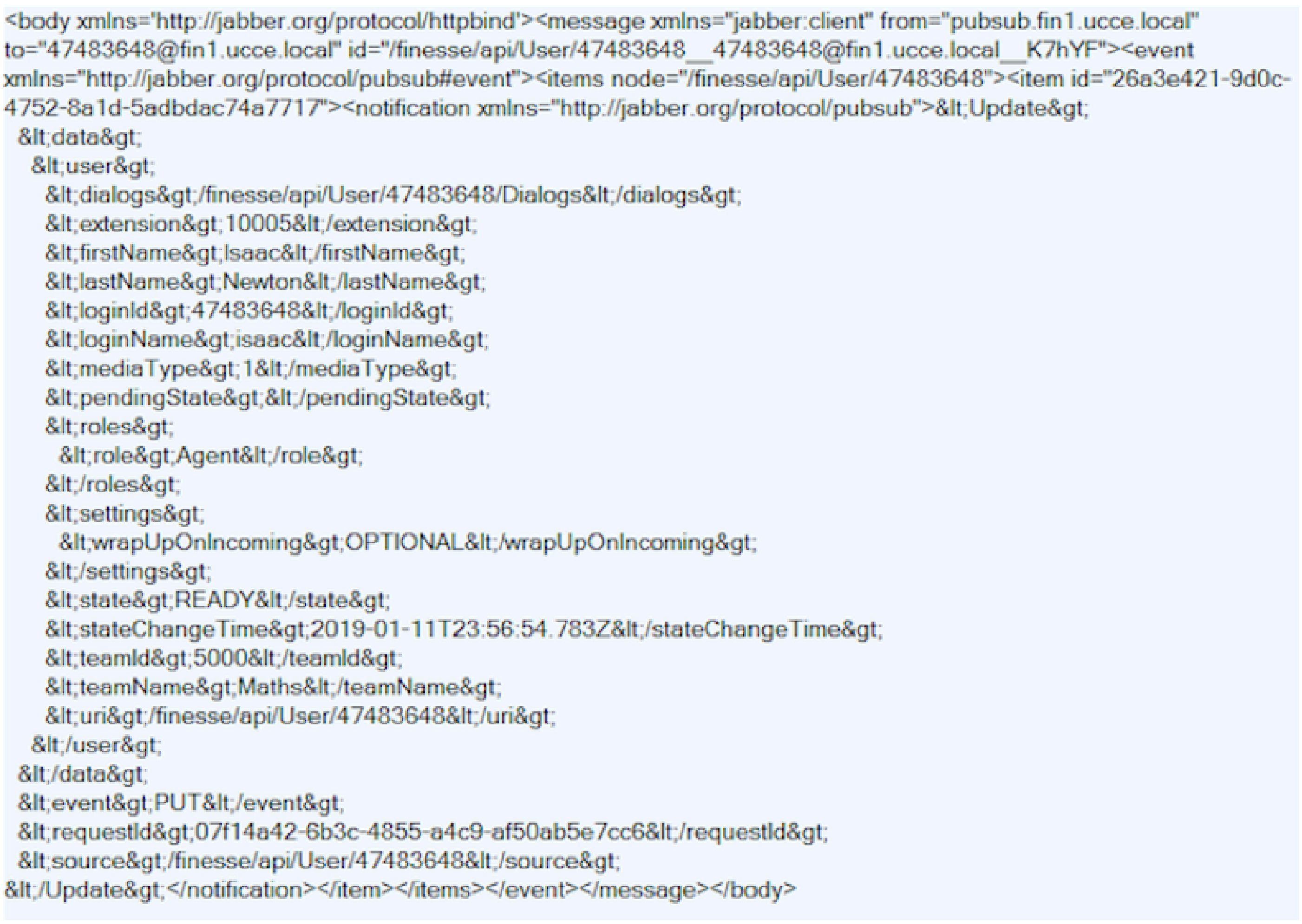
```
[QuickExec] ALT+Q > type HELP to learn more
Capturing   All Processes   1 / 693   https://fin1.ucce.local:7443/http-bind/
```

Closer view of the data:



```
6...  200  HTTPS  fin1.ucce.local:...  /http-bind/               1,571           text/xml...   firefo...
6...  202  HTTPS  fin1.ucce.local:...  /finesse/api/User/47...       0  no-ca... applicatio... firefo...
6...  200  HTTPS  fin1.ucce.local:...  /desktop/theme/fine...      673           image/gif     firefo...
6...  200  HTTPS  fin1.ucce.local:...  /http-bind/                  57           text/xml...   firefo...
6...  200  HTTPS  fin1.ucce.local:...  /finesse/api/SystemI...     232  no-ca... applicatio... firefo...
6...  200  HTTPS  fin1.ucce.local:...  /http-bind/                  57           text/xml...   firefo...
6...  200  HTTPS  fin1.ucce.local:...  /http-bind/                  57           text/xml...   firefo...
6...  200  HTTPS  fin1.ucce.local:...  /finesse/api/SystemI...     232  no-ca... applicatio... firefo...
6...  200  HTTPS  fin1.ucce.local:...  /http-bind/                  57           text/xml...   firefo...
6...  -    HTTPS  fin1.ucce.local:...  /http-bind/                  -1                         firefo...
6...  200  HTTPS  fin1.ucce.local:...  /finesse/api/SystemI...     232  no-ca... applicatio... firefo...
```

Response body for XMPP message:



```
<body xmlns='http://jabber.org/protocol/httpbind'><message xmlns="jabber:client" from="pubsub.fin1.ucce.local"
to="47483648@fin1.ucce.local" id="/finesse/api/User/47483648__47483648@fin1.ucce.local__K7hYF"><event
xmlns="http://jabber.org/protocol/pubsub#event"><items node="/finesse/api/User/47483648"><item id="26a3e421-9d0c-
4752-8a1d-5adbdac74a7717"><notification xmlns="http://jabber.org/protocol/pubsub">&lt;Update&gt;
  &lt;data&gt;
    &lt;user&gt;
      &lt;dialogs&gt;/finesse/api/User/47483648/Dialogs&lt;/dialogs&gt;
      &lt;extension&gt;10005&lt;/extension&gt;
      &lt;firstName&gt;Isaac&lt;/firstName&gt;
      &lt;lastName&gt;Newton&lt;/lastName&gt;
      &lt;loginId&gt;47483648&lt;/loginId&gt;
      &lt;loginName&gt;isaac&lt;/loginName&gt;
      &lt;mediaType&gt;1&lt;/mediaType&gt;
      &lt;pendingState&gt;&lt;/pendingState&gt;
      &lt;roles&gt;
        &lt;role&gt;Agent&lt;/role&gt;
      &lt;/roles&gt;
      &lt;settings&gt;
        &lt;wrapUpOnIncoming&gt;OPTIONAL&lt;/wrapUpOnIncoming&gt;
      &lt;/settings&gt;
      &lt;state&gt;READY&lt;/state&gt;
      &lt;stateChangeTime&gt;2019-01-11T23:56:54.783Z&lt;/stateChangeTime&gt;
      &lt;teamId&gt;5000&lt;/teamId&gt;
      &lt;teamName&gt;Maths&lt;/teamName&gt;
      &lt;uri&gt;/finesse/api/User/47483648&lt;/uri&gt;
    &lt;/user&gt;
  &lt;/data&gt;
  &lt;event&gt;PUT&lt;/event&gt;
  &lt;requestId&gt;07f14a42-6b3c-4855-a4c9-af50ab5e7cc6&lt;/requestId&gt;
  &lt;source&gt;/finesse/api/User/47483648&lt;/source&gt;
&lt;/Update&gt;</notification></item></items></event></message></body>
```

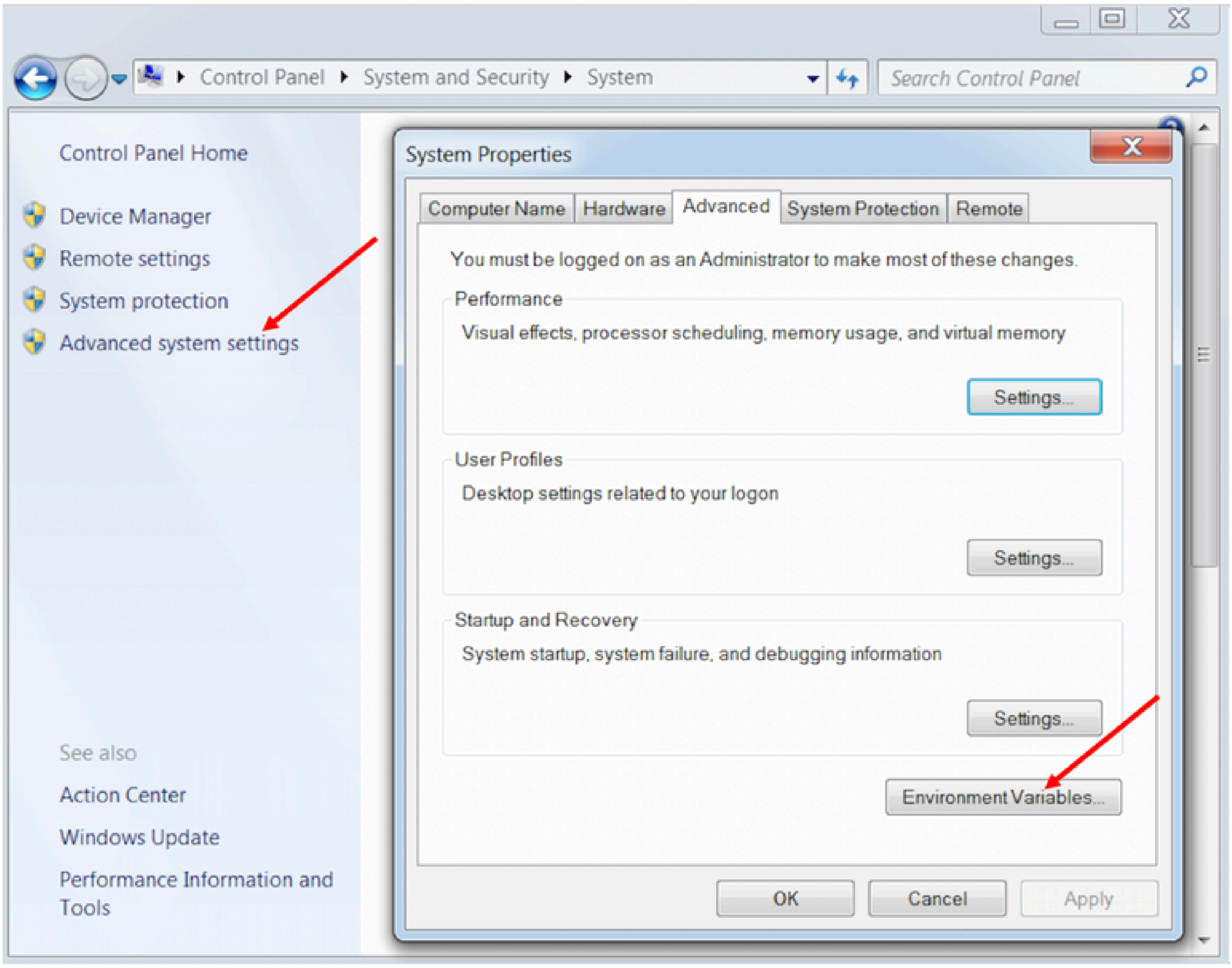
Use Wireshark

Wireshark is a commonly used packet sniffing tool that can be used to sniff and decode HTTPS traffic. HTTPS traffic is HTTP traffic secured over Transport Layer Security (TLS). TLS provides integrity, authentication and confidentiality between to hosts. It is used commonly in web applications, but it can be used with any protocol that uses TCP as the transport layer protocol. Secure Sockets Layer (SSL) is the former version of the TLS protocol, which is no longer used as it is insecure. These names are often used interchangeably, and the Wireshark filter used for SSL or TLS traffic is ssl.

Caution: The example configuration provided is for Wireshark 2.6.6 (v2.6.6-0-gdf942cd8)and Mozilla Firefox 64.0.2 (32-bit) on Windows7 x64 in a lab environment. These procedures cannot generalize to all versions of Fiddler, all browsers, or all computer operating systems. If your network is live, ensure that you understand the potential impact of any configuration. Reference the Official Wireshark SSL documentation for more information. Wireshark 1.6 or greater is required.

Note: This method can only work for Firefox and Chrome. This method does not work for Microsoft Edge.

Step 1. On the agent's Windows PC navigate to **Control Panel > System and Security > System > Advanced system settings Environmental Variables...**



Control Panel ▸ System and Security ▸ System

Control Panel Home

Device Manager

Remote settings

System protection

Advanced system settings

See also

Action Center

Windows Update

Performance Information and Tools

System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

Settings...

User Profiles

Desktop settings related to your logon

Settings...

Startup and Recovery

System startup, system failure, and debugging information

Settings...

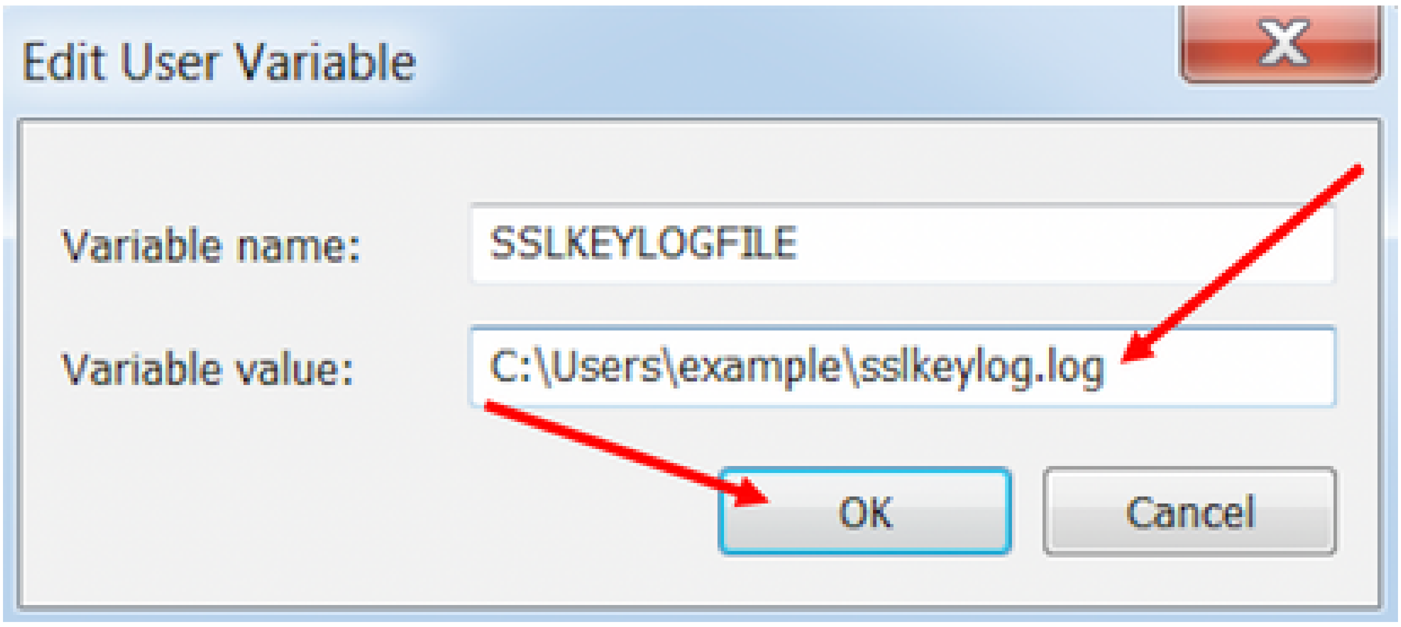
Environment Variables...

OK | Cancel | Apply

Step 2. Navigate to **User variables for user <username> > New...**

Create a variable named **SSLKEYLOGFILE.**

Create a file to store the SSL premaster secret in a private directory: SSLKEYLOGFILE=</path/to/private/directory/with/logfile>



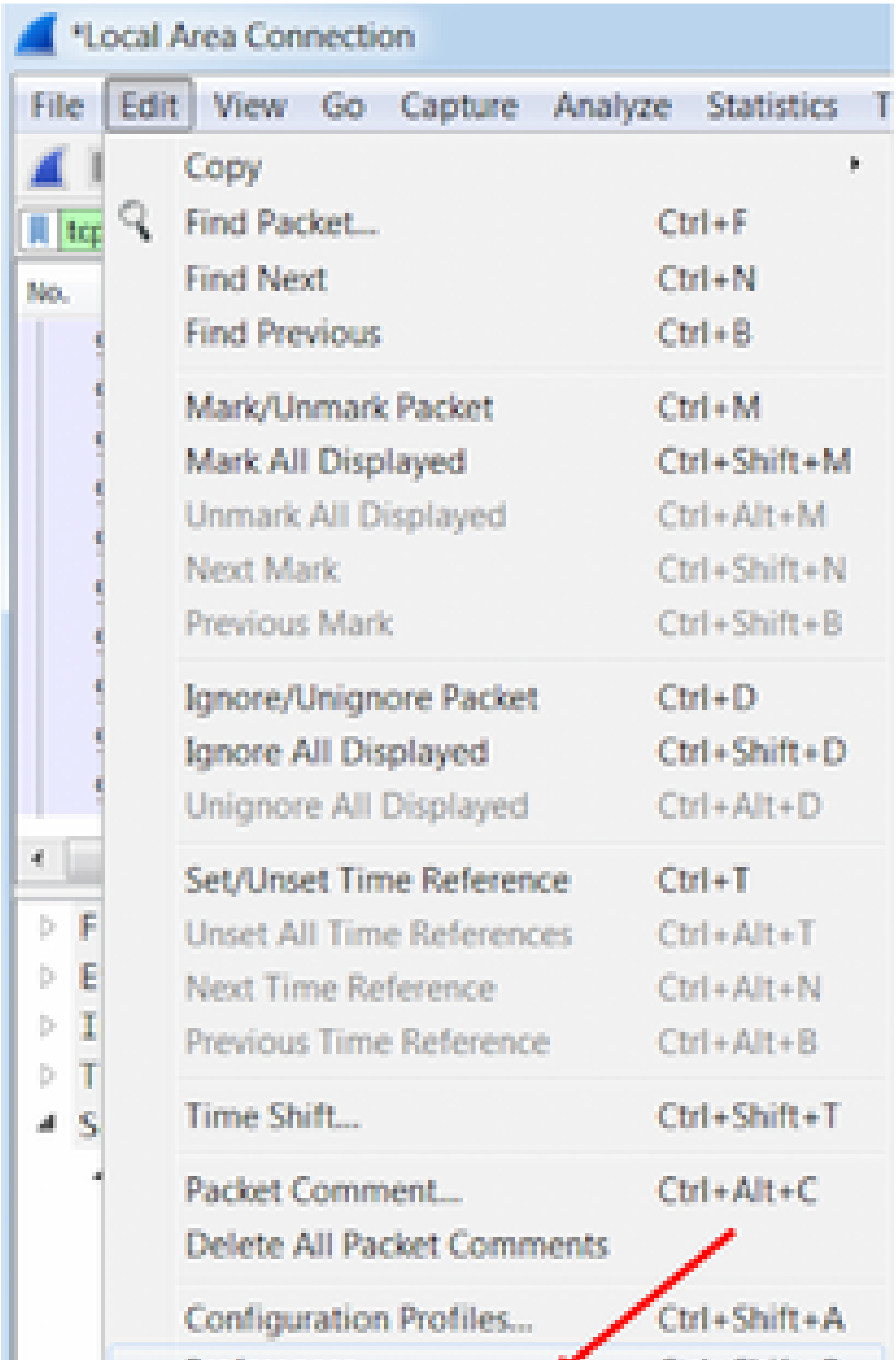
Variable name: SSLKEYLOGFILE

Variable value: C:\Users\example\sslkeylog.log

Note: Create a system variable instead of a user variable and/or store the file in a non-private directory, but then all users on the system can access the premaster secret, which is less secure.

Step 3. If Firefox or Chrome are open, close the applications. After they are reopened, they can start writing to the SSLKEYLOGFILE.

Step 4. On Wireshark, navigate to **Edit** > **Preferences...**

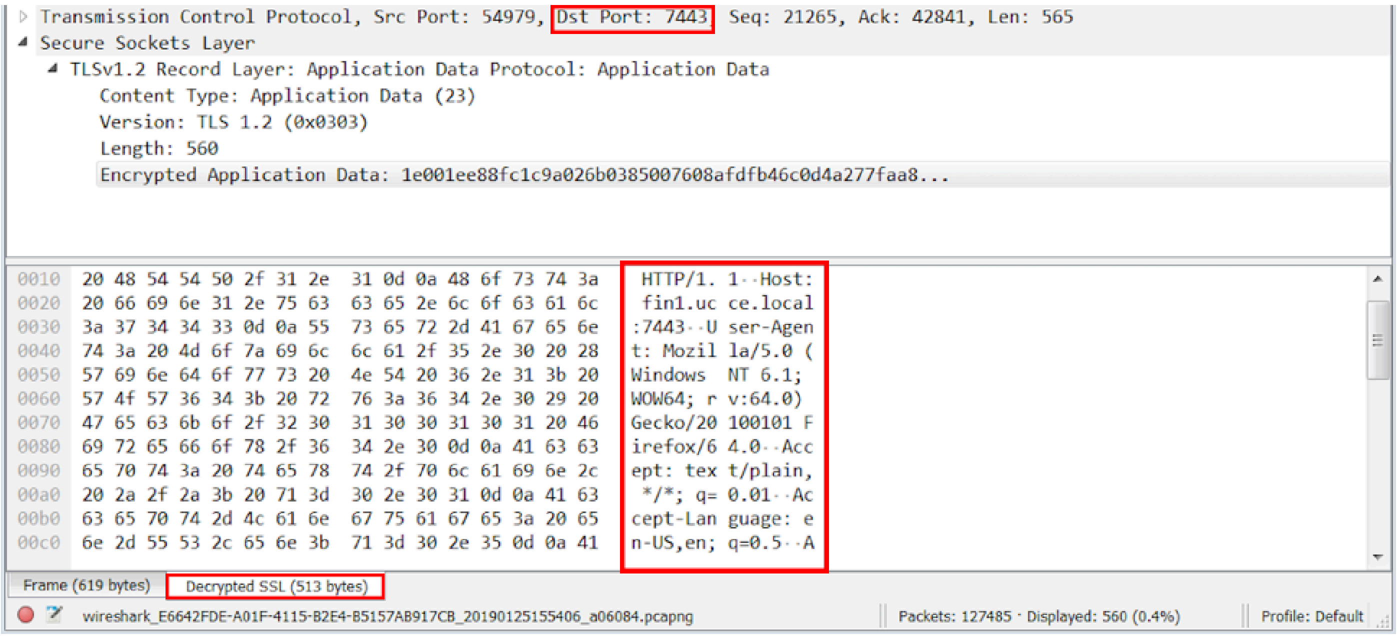


*Local Area Connection

File Edit View Go Capture Analyze Statistics T

Copy	
Find Packet...	Ctrl+F
Find Next	Ctrl+N
Find Previous	Ctrl+B
Mark/Unmark Packet	Ctrl+M
Mark All Displayed	Ctrl+Shift+M
Unmark All Displayed	Ctrl+Alt+M
Next Mark	Ctrl+Shift+N
Previous Mark	Ctrl+Shift+B
Ignore/Unignore Packet	Ctrl+D
Ignore All Displayed	Ctrl+Shift+D
Unignore All Displayed	Ctrl+Alt+D
Set/Unset Time Reference	Ctrl+T
Unset All Time References	Ctrl+Alt+T
Next Time Reference	Ctrl+Alt+N
Previous Time Reference	Ctrl+Alt+B
Time Shift...	Ctrl+Shift+T
Packet Comment...	Ctrl+Alt+C
Delete All Packet Comments	
Configuration Profiles...	Ctrl+Shift+A

, the secured HTTP communication between the Finesse client and Finesse server (Notification Service) is seen decrypted.



```
▷ Transmission Control Protocol, Src Port: 54979, Dst Port: 7443, Seq: 21265, Ack: 42841, Len: 565
◢ Secure Sockets Layer
  ◢ TLSv1.2 Record Layer: Application Data Protocol: Application Data
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 560
      Encrypted Application Data: 1e001ee88fc1c9a026b0385007608afdfb46c0d4a277faa8...

0010  20 48 54 54 50 2f 31 2e  31 0d 0a 48 6f 73 74 3a   HTTP/1. 1··Host:
0020  20 66 69 6e 31 2e 75 63  63 65 2e 6c 6f 63 61 6c   fin1.uc ce.local
0030  3a 37 34 34 33 0d 0a 55  73 65 72 2d 41 67 65 6e  :7443··U ser-Agen
0040  74 3a 20 4d 6f 7a 69 6c  6c 61 2f 35 2e 30 20 28  t: Mozil la/5.0 (
0050  57 69 6e 64 6f 77 73 20  4e 54 20 36 2e 31 3b 20  Windows  NT 6.1;
0060  57 4f 57 36 34 3b 20 72  76 3a 36 34 2e 30 29 20  WOW64; r v:64.0)
0070  47 65 63 6b 6f 2f 32 30  31 30 30 31 30 31 20 46  Gecko/20 100101 F
0080  69 72 65 66 6f 78 2f 36  34 2e 30 0d 0a 41 63 63  irefox/6 4.0··Acc
0090  65 70 74 3a 20 74 65 78  74 2f 70 6c 61 69 6e 2c  ept: tex t/plain,
00a0  20 2a 2f 2a 3b 20 71 3d  30 2e 30 31 0d 0a 41 63   */*; q= 0.01··Ac
00b0  63 65 70 74 2d 4c 61 6e  67 75 61 67 65 3a 20 65  cept-Lan guage: e
00c0  6e 2d 55 53 2c 65 6e 3b  71 3d 30 2e 35 0d 0a 41  n-US,en; q=0.5··A

Frame (619 bytes)   Decrypted SSL (513 bytes)
wireshark_E6642FDE-A01F-4115-B2E4-B5157AB917CB_20190125155406_a06084.pcapng   Packets: 127485 · Displayed: 560 (0.4%)   Profile: Default
```

Related Defects

- Cisco bug ID CSCva72280 - Finesse Tomcat and Openfire Crash for invalid XML characters
- Cisco bug ID CSCva72325 - UCCX: Finesse Tomcat and Openfire Crash for invalid XML characters

Related Information

- XMPP Specifications
- XEP-0124: BOSH
- XEP-0060: Publish-Subscribe
- Firefox Web Console
- Microsoft Edge Web Console
- Chrome Web Console
- Windows PowerShell
- Windows Perfomance Monitor
- Troubleshooting Input Queue Drops and Output Queue Drops
- Windows Task Manager
- Mac Terminal
- Mac Activity Monitor
- Fiddler Download
- Fiddler Configuration
- Wireshark Download
- Wireshark SSL Decryption
- Technical Support & Documentation - Cisco Systems