Display SIP Message flow using Sipviewer

Diagnosing complex SIP problems often requires looking at the SIP message flow between the components of sipXcom, as well as to and from phones and external gateways. This page tells the adventurous how to use the tools that are available for sipXcom to display SIP message flows. Sipviewer is a very powerful tool used to diagnose problems.

Installation

Before you can display the messages, you'll need to run sipx-trace to generate XML files of a SIP call you're interested in and you'll need to install sipviewer on a computer that has a GUI. You can do that either on the sipXcom system itself (which requires a few prerequisites not installed by the ISO image, but most of us don't want to run a GUI on the communications server), or standalone on your regular system (Windows, Mac, or Linux). Once you've gotten one of these done, you're ready to collect trace data for viewing.

You'll want sipxtools installed on your sipXcom server and sipviewer installed on some computer with a GUI and Java (works on Linux, Mac and Windows).

```
yum install sipxtools
```

SIPViewer Installation

The installer for the sipviewer tool is available here: Download the sipviewer installer

The above installer is a java .jar file - execute it on the command line of your system as follows:

```
java -jar sipviewer-install.jar
```

The installer should work on any system (Linux, Windows, Mac) that has java installed.

Getting SIP Messages to display

The SIP messages are logged by sipXcom services when their logging is at the INFO or DEBUG logging level; this is a more verbose level than the default NOTICE level. The log level can be changed in the Web UI by going to each service needed in the System menu (Proxy, Registrar, etc.). Logging level changes do not require service restarts (they used to in older versions of code).

To get trace information, at least the proxy and registrar *must* be set at INFO or DEBUG (INFO is sufficient for traces, and makes for much smaller log files). You should also have detailed logging for any other component you suspect to be involved in the problem.

If you can get the time a problem call is made, a good way to find it is to look in the proxy log file by searching the list of dialogs with the sipx-dialog-count command:

```
sipx-dialog-count /var/log/sipxpbx/sipXproxy.log
```

it will print a list like this:

```
Messages Method Time Call-Id
-------------- ---------- -------------------- ------------------------------------------------
50 INVITE 2009-12-11T22:22:20 nekyxpqiedpefov@example.org
36 INVITE 2009-12-11T22:17:19 zbwmxexjtxvlwer@example.org
```

search that list for the call that starts at the right time (log times are in UTC), and then use the call-id from that call as the <token> argument to sipx-trace - you'll get just the one call you're after.

Alternatively, if you download CDR's from Diagnostic -> Call Detail Records -> Historic (download is on the right, just above the call table). The CSV's include the call id of each call.

Next, use 'sipx-trace' to create an xml file that contains trace data for messages on your system:

```
sipx-trace --all-components --output <filename> <token>...
```

where <filename> is the output file and <token> is some token that will be in the call (call-id values are best for this, but even the calling number followed by '@' will find it; it will just find lots of other stuff too).

You can copy that xml file to any system where you have installed the sipviewer tool to display the trace, and you can post that file to a mailing list to ask for help interpreting it (be sure to also post a description of your configuration, including the IP addresses for all the components that show in the trace).

Using Sipviewer over an ssh connection

If you have an account on the sipXecs system that you can log into using ssh, and your workstation is a Linux (or compatible) system you can set up remote use of sipx-trace from another system:

- Install your ssh public key in your account on the sipXcom system. You should then be able to execute a command remotely on that system without entering a password:

```
> ssh sipxecs.example.com echo ok
ok
```

- Configure that account so that it can read the sipXecs log files.

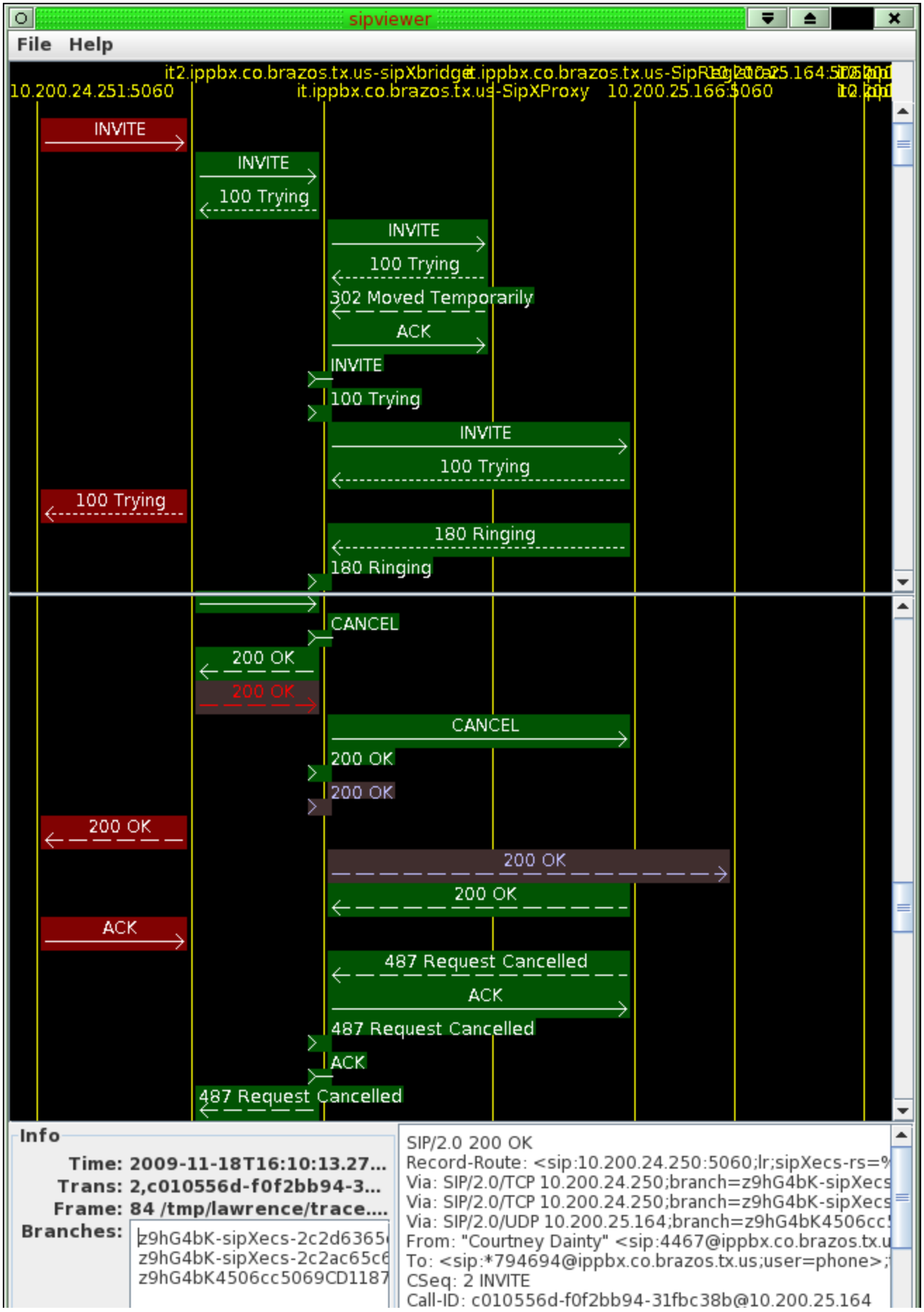
Now you can execute sipx-trace remotely by adding the `--system` (`-s`) argument:

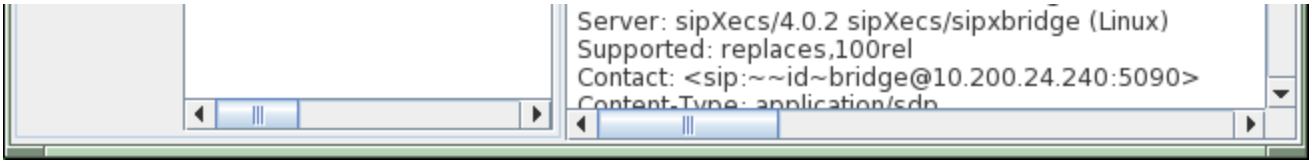
```
sipx-trace --system sipxecs.example.com --all-components --output <filename> <token>...
```

If you have an HA sipXecs, you must query all the systems at the same time to get the trace data:

```
sipx-trace -s sipxecs-master.example.com -s sipxecs-dist1.example.com --all-components --output <filename> <token>...
```

The sipviewer display


sipviewer
File Help
10.200.24.251:5060
it2.ippbx.co.brazos.tx.us-sipXbridge
it.ippbx.co.brazos.tx.us-SipXProxy
10.200.25.166:5060
INVITE
INVITE
100 Trying
INVITE
100 Trying
302 Moved Temporarily
ACK
INVITE
100 Trying
INVITE
100 Trying
100 Trying
180 Ringing
180 Ringing
CANCEL
200 OK
200 OK
CANCEL
200 OK
200 OK
200 OK
200 OK
200 OK
ACK
487 Request Cancelled
ACK
487 Request Cancelled
ACK
487 Request Cancelled
Info
Time: 2009-11-18T16:10:13.27...
Trans: 2,c010556d-f0f2bb94-3...
Frame: 84 /tmp/lawrence/trace....
Branches:
z9hG4bK-sipXecs-2c2d6365
z9hG4bK-sipXecs-2c2ac65c6
z9hG4bK4506cc5069CD1187
SIP/2.0 200 OK
Record-Route: <sip:10.200.24.250:5060;lr;sipXecs-rs=%
Via: SIP/2.0/TCP 10.200.24.250;branch=z9hG4bK-sipXecs
Via: SIP/2.0/TCP 10.200.24.250;branch=z9hG4bK-sipXecs
Via: SIP/2.0/UDP 10.200.25.164;branch=z9hG4bK4506cc
From: "Courtney Dainty" <sip:4467@ippbx.co.brazos.tx.u
To: <sip:*794694@ippbx.co.brazos.tx.us;user=phone>;
CSeq: 2 INVITE
Call-ID: c010556d-f0f2bb94-31fbc38b@10.200.25.164



sipviewer-install.jar