

MFA Platform / ... / SIP Trunking using Asterisk



Last updated: Mar 12, 2021 • 6 min read • 13 people viewed

DESIGN

- Description
 - Background
- Proposed Implementation
 - Error Codes
 - Current Development State
 - Problems
 - Need a WebSocket protocol implementation.
 - Vox file encoding
 - Assembled Vox files
 - Development on the cloud
 - Asterisk config files
 - extensions.conf
 - modules.conf
 - ari.conf
 - pjsip.conf
 - Useful commands
- Other resources

Description

This is a page to note the developments on SIP trunking for MFA. The ultimate purpose is to replace current telephony framework using SS7 and PRI with telephony cards using the PSTN with a system using Asterisk to place voice calls over the internet.

Background

Asterisk is an Open Source project started in 1999 as the Zanata Telephony project by

MFA Platform / / SIP Trunking using Asterisk

provide telephony services in the world. You can read more about it here.

[Click for a copy of Asterisk The Definitive Guide](#)

Proposed Implementation

The MFA framework will be modified to use Asterisk as the main way to make OOB phone calls. For this purpose we will use the Asterisk ARI APIs. These are all REST APIs which provide a high level control over the Asterisk services, removing the complexity of the SIP protocol and exposing all the services the Asterisk platform can provide. These services have been growing over the year and can be incorporated as modules.

The Asterisk ARI interface has been wrapped in convenient client frameworks for several languages, including Java, JavaScript, Python, C# and C++. See examples [here](#).

As the MFA framework is written in C++ we will be using the excellent Asterisk ARI interface binding for modern C++ library (**aricpp**) to implement the required functionality. This is a high level, modern C++ library that provides a high level interface to manipulate asterisk concepts (channels, bridges, etc.) and a low level interface to send, receive and manipulate ARI commands and Events from asterisk.

aricpp requires a C++ 14 compiler and depends on the Boost libraries. This implies the migration of the MFA framework to use Visual Studio 2019 as the IDE/Compiler and the addition of Boost to the application 3rd party dependencies.

Asterisk 16 and Asterisk 18 are LTS (Long Term Support) versions

Asterisk 17 is a Standard release, meaning it only has one year support

Error Codes

When calls fail, the application uses the following set of error codes to describe the problem. This is the industry standard and it is the same set of values that legacy

NMS code returned.

0031 - ITIL standard

MFA Platform / / SIP Trunking using Asterisk

Current Development State

Sample application using aricpp can originate a call and play sounds pre-configured with Asterisk. Application can play DTMF tones

Problems

Need a WebSocket protocol implementation.

The Asterisk plugin will use a WebSocket to connect to Asterisk and subscribe to events. Some possibilities include:

 [GitHub - zaphoyd/websocketpp: C++ websocket client/server library](#) This is a full implementation of RFC6455 that can be used with our own TCP/UDP sockets.

<https://bitbucket.payfone.com/projects/MFA/repos/aricpp/browse/aricpp/websocket.h> This is the aricpp wrap around over Boost websockets implementation. Means we would need to include Boost asio in the application, but that can be done in C++ 11 with Visual Studio 2015.

Vox file encoding

Currently unable to use vox files from the existing MFA Apps. Sound comes up garbled.

Solution: Convert legacy .vox files to a format Asterisk will recognize. Two steps are needed:

1. Use NMS tool vcecopy to convert current .vox files into .wav files, using Mono (1 channel), 16-bit precision, 16-bit Signed Integer PCM encoding.
2. Convert the .wav file from step 1 into a .gsm file using Asterisk's sox utility:

```
sox -V3 input.wav --channels 1 --rate 8000 --type gsm output.gsm
```

Assembled Vox files

In many Authentify Apps, multiple sounds samples are assembled into a single .vox

MFA Platform / / SIP Trunking using Asterisk

The App will need to create multiple gsm files out of the assembled .vox. We will need scripts that go over the assembled .vox file and create the respective gsm file. The app will use a command to play the appropriate file, based on a naming convention so it can find the right file.

Development on the cloud

Adrian has created an asterisk server, using this repo:

 [GitHub - mlan/docker-asterisk: Docker image providing Asterisk PBX](#)

This is a Docker image providing Asterisk PBX.

You can connect to it as follows.

- Add this to your ~/.ssh/config file:

```
1 Host bastion.poc.mfaHostname 54.204.175.227
2 User ubuntu
3 IdentityFile ~/.ssh/poc-mfa-bastion.pem
4 Host framos.sip.poc.mfa
5 Hostname 10.48.3.93
6 User dev
7 IdentityFile ~/.ssh/framos_dev_mfa.pem
8 ProxyJump bastion.poc.mfa
```

- ssh into the instance:

```
ssh framos.sip.poc.mfa
dev@ip-10-48-3-93:~$ cd docker-asterisk/demo
docker-compose exec tele asterisk -rvvvddd
Seeding global EID '02:42:ac:11:00:02' from 'eth0' using 'siocgifhwaddr'
```

```
Parsing /etc/asterisk/asterisk.conf
```

```
Asterisk 16.7.0, Copyright (C) 1999 - 2018, Digium, Inc. and others.
```

MFA Platform / / SIP Trunking using Asterisk

```
This is free software, with components licensed under the GNU General Public  
License version 2 and other licenses; you are welcome to redistribute it  
under
```

```
certain conditions. Type 'core show license' for details.
```

```
Connected to Asterisk 16.7.0 currently running on 502cafa07b57 (pid = 192)
```

```
Core debug was OFF and is now 3.
```

```
502cafa07b57*CLI>
```

You will need two ssh keys (bastion and your personal key) to connect using ssh.
Download them and store them in ~/.ssh/ directory

If you want to play with the server locally, you can forward ports to your computer.
The docker container is currently exposing these ports:

```
0.0.0.0:5060->5060/udp, 0.0.0.0:5060-5061->5060-5061/tcp, 0.0.0.0:10000-  
10099->10000-10099/udp, 0.0.0.0:8080->80/tcp
```

In order to use the AEI interface from your laptop, you need forward port 8088 to
your local computer:

```
ssh -NL 8088:localhost:8088 framos.sip.poc.mfa
```

Asterisk config files

When customizing Asterisk you modify the config files at
/sip_server/docker_sip_volume/etc/asterisk/. The container will use that volume when
it starts, so modification to those files are available in the container.

extensions.conf

```
[default]
```

```

exten => 1000,1,NoOp()
same =>      n,Answer()

```

MFA Platform / / SIP Trunking using Asterisk

, these blocks allow for forwarding calls to callcentric

```

[voice-otp]
exten => _X.,1,Stasis(voice-otp)
exten => 100,1,Stasis(voice-otp)
exten => 101,1,Stasis(voice-otp)

[voice-otp-out]
exten => _X.,1,Set(CALLERID(num)=+17731234123)
exten => _X.,n,Set(CALLERID(name)=Prove Voice OTP)
exten => _X.,n,Dial(PJSIP/${EXTEN}@callcentric)
same = n,Stasis(voice-otp)

;The following blocks are recommendation from call-centric

[to-callcentric]
exten => s,1,Dial(PJSIP/100,20)
exten => s,n,Hangup

[from-callcentric]
exten => 17778990266,1,Set(Var_TO=${CUT(CUT(PJSIP_HEADER(read,To),@,1),:,2)})
exten => 17778990266,2,GotoIf("${Var_TO}" = "17778990266")?to-100,s,1:3)
exten => 17778990266,3,Hangup

[to-100]
exten => s,1,Dial(PJSIP/100,20)
exten => s,n,Hangup()

```

Add to modules.conf to enable VOX files and the ARI interface which is disabled by default. These modules are C dynamically loaded libraries. The container does have them in the image, but not all of them are configured to load.

modules.conf

```
; Codecs
```

```
load = codec_gsm.so  
load = codec_resample.so
```

MFA Platform / / SIP Trunking using Asterisk

```
; Formats
```

```
load = format_gsm.so  
load = format_pcm.so  
load = format_wav_gsm.so  
load = format_wav.so  
load = format_vox.so          ← Add this line for vox support
```

```
; ARI Support
```

```
load = res_http_websocket.so ; HTTP WebSocket Support, needed for  
res_ari_events  
load = res_ari.so ; Asterisk RESTful Interface  
load = res_ari_applications  
load = res_ari_asterisk.so  
load = res_ari_bridges  
load = res_ari_channels.so  
load = res_ari_device_states.so ; RESTful API module - Device state resources  
load = res_ari_endpoints.so  
load = res_ari_events.so  
load = res_ari_model.so ; ARI Model validators  
load = res_ari_playbacks.so  
load = res_ari_recordings.so  
load = res_ari_sounds  
load = res_stasis.so ; Stasis application support  
load = app_stasis.so ; Stasis dialplan application  
load = res_stasis_answer.so  
load = res_stasis_device_state.so ; Stasis application device state support  
load = res_stasis_playback.so ; Stasis application playback support  
load = res_stasis_recording.so ; Stasis application recording support  
load = res_stasis_snoop.so ; Stasis application snoop support
```

Add the application users to ari.conf. These are the user/password combinations to

MFA Platform / / SIP Trunking using Asterisk

```
[margayUser]
type=user
read_only=no
password_format=plain
password=1234

[17778990266101]
type=user
read_only=no
password_format=plain
password=testPasswd
```

In pjsip.conf, you will set up extensions and the connection to the trunk provider – the trunk provider will give you this data. You can have multiple trunks set up at once. You will also need to set up at least one local extension to act as the caller for the SIP trunk. The following is the configuration to use Callcentric as the SIP trunk.

pjsip.conf

```
[global]
type = global
user_agent = Prove Dev Asterisk

;-----
; TRANSPORTS
;-----

[transport-udp]
type=transport
protocol=udp
bind=0.0.0.0:5060

; -----
; CALLCENTRIC TRUNK
```

```
; -----
```

```
[callcentric]
```

MFA Platform / / SIP Trunking using Asterisk

```
outbound_auth=callcentric_auth
```

```
retry_interval=60
```

```
expiration=3600
```

```
auth_rejection_permanent=yes
```

```
contact_user=17778990266
```

```
server_uri=sip:callcentric.com
```

```
client_uri=sip:17778990266@callcentric.com
```

```
[callcentric_auth]
```

```
type=auth
```

```
auth_type=userpass
```

```
password=xxxxxxxx ; use the password set for the SIP trunk
```

```
username=17778990266
```

```
[callcentric]
```

```
type=endpoint
```

```
transport=transport-udp
```

```
context=from-callcentric
```

```
disallow=all
```

```
allow=ulaw
```

```
outbound_auth=callcentric_auth
```

```
aors=callcentric
```

```
from_domain=callcentric.com
```

```
from_user=17778990266
```

```
sdp_owner=17778990266
```

```
direct_media=no
```

```
ice_support=no
```

```
send_rpid=yes
```

```
rtp_symmetric=yes
```

```
force_rport=yes
```

```
timers=no
```

```
[callcentric]
```

```
type=aor
```

MFA Platform / / SIP Trunking using Asterisk

```
type=identity
```

```
endpoint=callcentric
```

```
match=callcentric.com
```

```
; =====
```

```
; EXTENSIONS
```

```
; -----
```

```
; Extension 100
```

```
; -----
```

```
[100]
```

```
type=endpoint
```

```
aors=100
```

```
auth=100-auth
```

```
allow=ulaw
```

```
context=to-callcentric
```

```
callerid="MFA Extension" <100>
```

```
ice_support=no
```

```
rtp_symmetric=yes
```

```
force_rport=yes
```

```
rewrite_contact=yes
```

```
dtmf_mode=rfc4733
```

```
[100-auth]
```

```
type=auth
```

```
auth_type=userpass
```

```
password=xxxxxxxxxx
```

```
username=17778990266
```

```
[100]
```

```
type=aor
```

```
max_contacts=1
```

```
remove_existing=yes  
qualify_frequency=60
```

MFA Platform / / SIP Trunking using Asterisk

```
, -----  
  
[101]  
type=endpoint  
aors=101  
auth=101-auth  
allow=ulaw  
context=to-callcentric  
callerid="MFA Extension" <101>  
ice_support=no  
rtp_symmetric=yes  
force_rport=yes  
rewrite_contact=yes  
dtmf_mode=rfc4733  
  
[101-auth]  
type=auth  
auth_type=userpass  
password=testPasswd  
username=17778990266101  
  
[101]  
type=aor  
max_contacts=1  
remove_existing=yes  
qualify_frequency=60
```

Useful commands

docker ps	list containers running
-----------	-------------------------

docker logs -f xxx	tail docker logs on the screen
--------------------	--------------------------------

MFA Platform / / SIP Trunking using Asterisk

	Asterisk CLI using config reload
cd ~/docker-asterisk/demo make destroy make up	make destroy shuts-down the container make up restarts the container
vim ~/docker-asterisk/demo/docker-compose.yml	Edit the docker settings. Ports, environment variables, persistent storage

Other resources

CyberMegaPhone - A soft phone that uses Asterisk WebRTC API.

<https://www.mizu-voip.com/Software/Softphones.aspx> - Simple to use SIP phone applications

Asterisk community - A place to ask questions about Asterisk

Getting Started with ARI - Basic configuration and example of using the ARI APIs

llvl × + Add label

 Be the first to add a reaction