

Prepared for:

Federal Communications Commission

CMS Alliance to Modernize Healthcare
Federally Funded Research and Development Center

Contract No. HHSM-5000-2012-000081

Task Order No. FCC15D0002

ACE Connect Lite Proof-of-Concept Release

Final

Version 1.0

October 28, 2016

The views, opinions, and/or findings contained in this report are those of The MITRE Corporation and should not be construed as official government position, policy, or decision unless so designated by other documentation.

Approved for Public Release; Distribution Unlimited. Case Number 16-3501.

© 2016, The MITRE Corporation. All Rights Reserved.

Record of Changes

Version	Date	Author / Owner	Description of Change
1.0	October 28, 2016	CAMH	Version 1.0 for release to Sponsor.

About the CMS Alliance to Modernize Healthcare

The Centers for Medicare & Medicaid Services (CMS) sponsors the CMS Alliance to Modernize Healthcare (CAMH), the first Federally Funded Research and Development Center (FFRDC) dedicated to strengthening our nation's healthcare system.

The CAMH FFRDC enables CMS, the Department of Health and Human Services (HHS), and other government entities to access unbiased research, advice, guidance, and analysis to solve complex business, policy, technology, and operational challenges in health mission areas. The FFRDC objectively analyzes long-term health system problems, addresses complex technical questions, and generates creative and cost-effective solutions in strategic areas such as quality of care, new payment models, and business transformation.

Formally established under Federal Acquisition Regulation (FAR) Part 35.017, FFRDCs meet special, long-term research and development needs integral to the mission of the sponsoring agency—work that existing in-house or commercial contractor resources cannot fulfill as effectively. FFRDCs operate in the public interest, free from conflicts of interest, and are managed and/or administered by not-for-profit organizations, universities, or industrial firms as separate operating units.

The CAMH FFRDC applies a combination of large-scale enterprise systems engineering and specialized health subject matter expertise to achieve the strategic objectives of CMS, HHS, and other government organizations charged with health-related missions. As a trusted, not-for-profit adviser, the CAMH FFRDC has access, beyond what is allowed in normal contractual relationships, to government and supplier data, including sensitive and proprietary data, and to employees and government facilities and equipment that support health missions.

CMS conducted a competitive acquisition in 2012 and awarded the CAMH FFRDC contract to The MITRE Corporation (MITRE). MITRE operates the CAMH FFRDC in partnership with CMS and HHS, and maintains a collaborative alliance of partners from nonprofits, academia, and industry. This alliance provides specialized expertise, health capabilities, and innovative solutions to transform delivery of the nation's healthcare services. Government organizations and other entities have ready access to this network of partners, including RAND Health, the Brookings Institution, and other leading healthcare organizations. This includes select qualified small and disadvantaged business.

The FFRDC is open to all CMS and HHS Operating Divisions and Staff Divisions. In addition, government entities outside of CMS and HHS can use the FFRDC with permission of CMS, CAMH's primary sponsor.

Executive Summary

The Federal Communication Commission's (FCC) Telecommunications Relay Service (TRS) Center of Expertise (COE) Project promotes the Commission's goal to foster innovations that advance functionally equivalent telecommunications. Toward that end, the project ensures that the Telecommunications Relay Service employs improved technology for persons who are deaf, hard of hearing, deaf-blind, and/or have speech disabilities. The FCC has embraced a research-based approach to achieve this goal by engaging the CMS Alliance to Modernize Healthcare (CAMH) federally funded research and development center (FFRDC), operated by The MITRE Corporation (MITRE), to conduct independent engineering assessments that promote and demonstrate TRS's functional equivalence.

CAMH is independently assessing voice telephone services, video access services, and Internet Protocol (IP)-based captioning technology; improvements to TRS efficiency; solutions for direct communication between people with communication disabilities and other telephone users; and the effectiveness, efficiency, and consumer response to current and future approaches for delivering TRS.

This document presents an overview of Accessible Communications for Everyone (ACE) Connect Lite, a proof- of-concept task under the TRS COE Project. ACE Connect Lite builds on ACE Direct, another prototyping effort that successfully demonstrated the feasibility of auto-routing a Voice Relay Service (VRS) user's call into a federal call center to reach a customer service representative with VRS capabilities.

For ACE Connect Lite, the FCC tasked CAMH to develop a proof of concept of the initial version of a server, capable of the interoperability testing related to the Session Initiation Protocol (SIP) Forum Relay User Equipment (RUE) specification. This server provides necessary capabilities to satisfy the testing being conducted by the FCC National Test Lab and the ACE Access Point Platform (APP) vendor.

As part of this proof-of-concept effort, CAMH developed and documented requirements and features, including user stories and associated use cases. CAMH also configured, tested, and integrated provider endpoint video devices with the ACE Connect Lite platform. Detailed configuration and source code files are available for download or reproduction by the public at the [Github.com/MITRE/ACE](https://github.com/MITRE/ACE) website to improve solutions to support the community.

Table of Contents

1. Introduction	1
1.1 Background	1
1.2 Purpose and Scope	1
2. Overview of ACE Connect Lite	2
2.1 Conceptual System Overview	2
2.2 Highlighted User Stories	4
2.3 CA Desktop User Guide.....	5
3. Installation and Configuration	8
3.1 AWS	8
3.2 Asterisk Server	10
3.3 Node.js.....	14
3.4 Management Portal	15
3.4.1 Management Dashboard	16
3.4.2 Call Detail Record Dashboard	21
3.5 VRS User Database (Provider)	26
3.6 STUN	28
3.7 iTRS ENUM Database	29
3.8 VyOS Router for Secure Socket Layer (SSL) Tunnel	29
3.9 SIP Trunking Provider	31
Acronyms.....	34

List of Figures

Figure 1. ACE Connect Lite Notional Diagram	2
Figure 2. ACE Connect Lite CA Desktop	5
Figure 3. CA Status: ACE Connect Lite.....	6
Figure 4. ACE Connect Lite Call Information.....	6
Figure 5. ACE Connect Lite VRS User Information.....	7
Figure 6. Management Dashboard.....	16
Figure 7. ACE Connect Lite Call Detail Record Dashboard.....	21
Figure 8. VyOS Router for SSL Tunnel	30

List of Tables

Table 1. ACE Connect Lite Components	3
Table 2. ACE Connect Lite Highlighted User Stories	4
Table 3. Asterisk Endpoint Extensions	11
Table 4. AMI Actions and Descriptions	18
Table 5. Asterisk AMI Event	19
Table 6. ACE Connect Lite Call Detail Record Column Definition	22

1. Introduction

The Federal Communications Commission (FCC) Telecommunications Relay Service (TRS) Center of Expertise (COE) Project promotes the Commission's goal to foster innovations that advance functionally equivalent telecommunications. Toward that end, the project ensures that the Telecommunications Relay Service employs improved technology for persons who are deaf, hard of hearing, deaf-blind, and/or have speech disabilities.

1.1 Background

The FCC has embraced a research-based approach to achieve this goal by engaging the CMS Alliance to Modernize Healthcare (CAMH) federally funded research and development center (FFRDC), operated by The MITRE Corporation (MITRE), to conduct independent engineering assessments that promote and demonstrate TRS's functional equivalence. CAMH independently assesses voice telephone services, video access services, and Internet Protocol (IP)-based captioning technology; improvements to TRS efficiency; solutions for direct communication between people with communication disabilities and other telephone users; and the effectiveness, efficiency, and consumer response to current and future approaches for delivering TRS. For another prototyping effort, known as ACE Direct, CAMH successfully demonstrated the feasibility of auto-routing a VRS user's call into a federal call center to reach a customer service representative with VRS capabilities.

Recently, the FCC directed CAMH to provide the ACE Connect Lite proof of concept based on the successful ACE Direct effort. ACE Connect Lite consists of the initial server solutions that enable video relay services capable of Session Initiation Protocol (SIP) Forum Relay User Equipment (RUE) interoperability testing. This proof of concept is essentially a SIP proxy/registrar and a configuration server that offers a specific set of features available from the current VRS providers.

CAMH developed and documented requirements and features, including user stories and associated use cases for ACE Connect Lite. CAMH also configured, tested, and integrated provider endpoint video devices using the ACE Connect Lite platform.

1.2 Purpose and Scope

This document presents an overview of the ACE Connect Lite proof of concept that provides the necessary capabilities to satisfy interoperability testing conducted by the FCC National Test Lab and the ACE Access Point Platform (APP) vendor. This overview includes the ACE Connect Lite architecture, user stories, and guidance for installation and configuration.

In addition to this overview document, detailed configuration and source code files are available to the public at the [Github.com/MITRE/ACE](https://github.com/MITRE/ACE) website for download and reproduction of the proof of concept or to improve the solutions to support the deaf and hard-of-hearing community.

2. Overview of ACE Connect Lite

ACE Connect Lite provides a VRS platform for deaf and hard-of-hearing individuals using a videophone and the hearing community. ACE Connect Lite employs a traditional public switched telephone network (PSTN) to communicate through a real-time video connection directly to an American Sign Language (ASL)-trained Communications Assistant (CA) within a call center.

2.1 Conceptual System Overview

CAMH developed this proof of concept with open source-based technologies for implementation in the Amazon Web Services (AWS) cloud environment. Figure 1 presents a notional view of the ACE Connect Lite implementation.

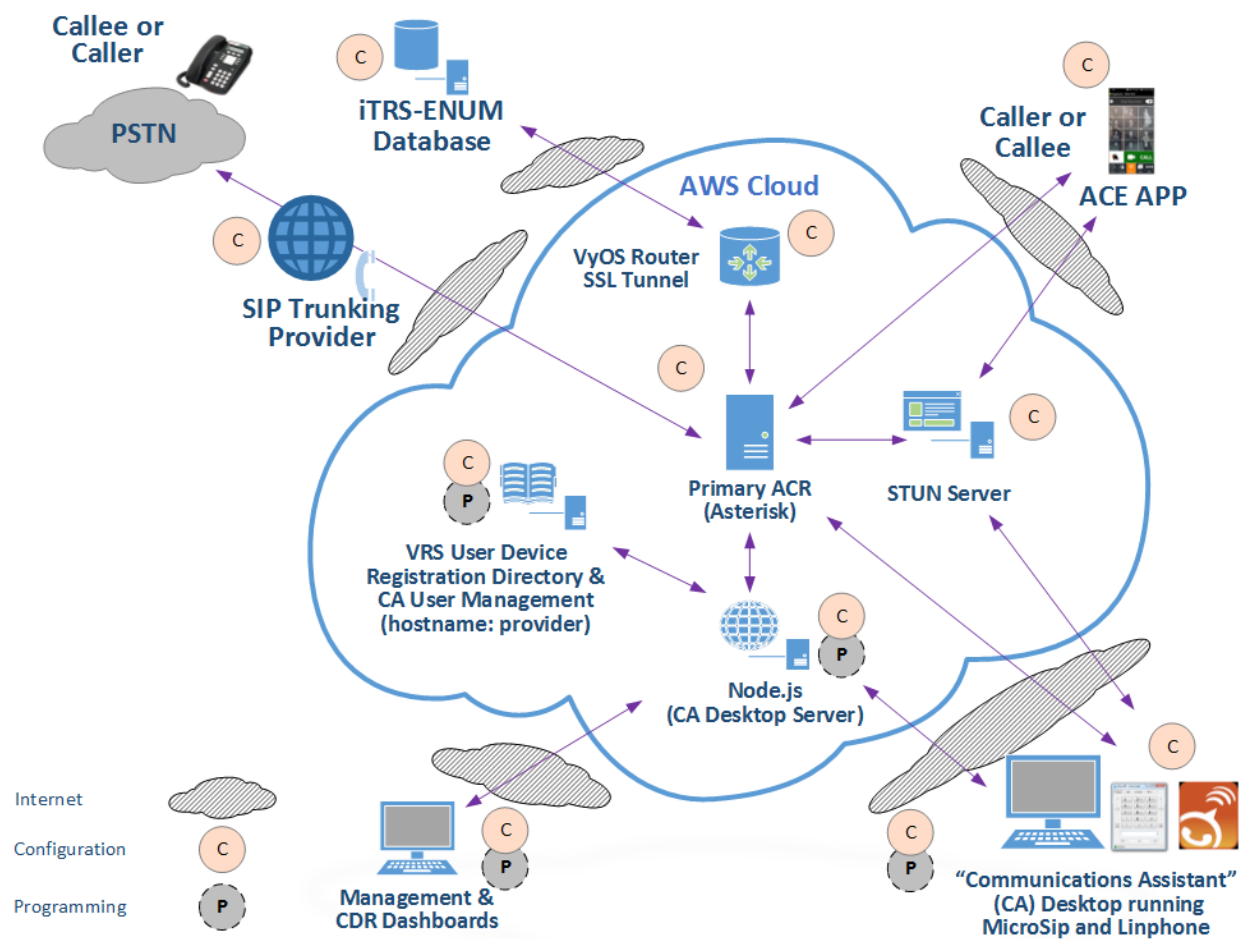


Figure 1. ACE Connect Lite Notional Diagram

The ACE Connect Lite implementation consists of various components. Some components require only configurations (noted as “C”) in the diagram, while others require both configurations and programming (noted as “P”). Table 1 lists and describes the ACE Connect Lite components. Section 3 provides a detailed description of installation and configurations.

Table 1. ACE Connect Lite Components

Component	Description	Additional Details
Asterisk Open Source PBX	Asterisk is an open source private branch exchange (PBX) server that supports direct video communication vis both PSTN and video calls. In the ACE Connect Lite proof of concept, a CA uses two softphones (MicroSIP and Linphone) registered to Asterisk to accept inbound calls (PSTN or Video) and to place outbound calls (Video and PSTN), respectively.	Subsection 3.2
Node.js	Node.js is an open source JavaScript-based web server. The Node.js server contains several services running different ports to support the CA desktop and othe management-related portals, including Dashboard and call detail record (CDR). The Node.js server also provides services for VRS lookup to verify that the phone number is a valid number in the VRS database.	Subsections 3.3 to 3.5
Management and Call Detail Record (CDR) Dashboards	ACE Direct Management Dashboard provides a number of Key Performance Indicators (KPI) the manager can monitor real time and the CDR Dashboard provides the view and export functions of the Asterisk CDRs stored in its MySQL database.	Subsections 3.4.1 and 3.4.2
VRS User Device Registration and CA User Management (hostname: Provider)	The “Provider” server emulates the services for both the CA user authentication and iTRS-URD (Internet Telecommunications Relay Service-User Registration Database) lookup from the Node.js to demonstrate the communications of the Node.js server in real-world scenarios.	N/A
Communications Assistant (CA) Desktop	The CA Desktop provides user interface to the CA for login and conducting VRS services to the ACE Connect Lite users. The desktop is one of the three windows on the CA’s workstation monitor.	Subsections 2.2 and 3.7
STUN server	STUN (formerly Simple Traversal of UDP through NAT rfc3489) is reflexive and identifies if the endpoint is behind a Network Address Translation (NAT) or firewall and determines the public IP address. This helps STUN establish a peer-to-peer connection.	Subsection 3.8

Component	Description	Additional Details
iTRS-ENUM database	The iTRS database maps 10-digit U.S. telephone numbers to IP addresses using the industry-standard ENUM (E.164 Number to URI Mapping) protocol. VRS providers assign these 10-digit telephone numbers to their customers. ACE Connect Lite relies on the iTRS-ENUM lookup to direct calls to another VRS provider's network and vice versa.	Subsection 3.9
SIP Trunking Provider	To support PSTN calls, ACE Connect Lite uses a SIP Trunking service from one of the providers. SIP Trunk can be configured through the provider's account portal.	Subsection 3.10
SIP Softphones for CA and ASL users (Caller and Callee)	ACE Connect Lite uses SIP video softphones, including ACE APP, Lync, and MicroSIP, as end-user devices that are registered to the Asterisk Server.	N/A

2.2 Highlighted User Stories

CAMH built ACE Connect Lite on the core functions of ACE Direct. ACE Connect Lite focuses on the CA capabilities. Table 2 presents a summary of ACE Connect Lite user stories, which demonstrate these functions and capabilities.

Table 2. ACE Connect Lite Highlighted User Stories

Story	Description	Additional Detail
Direct Video Call with an ASL user of another VRS provider	As an ASL user of ACE Connect Lite, I can call/or accept a video call from another ASL user of a VRS provider.	
CA service between the PSTN and ASL users	As an ASL user of ACE Connect Lite, I can place/or accept a PSTN call from a hearing user through the ACE Connect Lite CA service.	
Call handling capabilities	As an ACE Connect Lite CA, I can perform "Call on Hold" and "Call Transfer" as needed.	
Video Mail	As an ACE Connect Lite customer, I can retrieve a video mail.	
Dial Around	As an ACE Connect Lite CA, I can provide CA service to another VRS provider's customer as the result of Dial-Around.	
Management Dashboard	As the ACE Connect Lite manager/operator, I can access near real-time information on the dashboard.	Subsection 3.4
Call Detail Record	As the ACE Connect Lite administrator, I can access the Call Detail Record through a web portal and export CDR as needed for audit purposes.	Subsection 3.5

Story	Description	Additional Detail
Multi-Agent Login with Status	As the ACE Connect Lite CA, I can login using the CA Desktop along with other CAs and I can change my status between “Ready” and “Away”.	

2.3 CA Desktop User Guide

The ACE Connect Lite CA interfaces with three windows on the workstation monitor, including the CA Desktop window, an inbound call softphone window (MicroSIP), and an outbound call softphone window (Linphone).

As Figure 2 shows, the CA Desktop window provides CA status, the “Connect Call” button to place an outbound call, and the VRS user information. A full description of these windows follows.

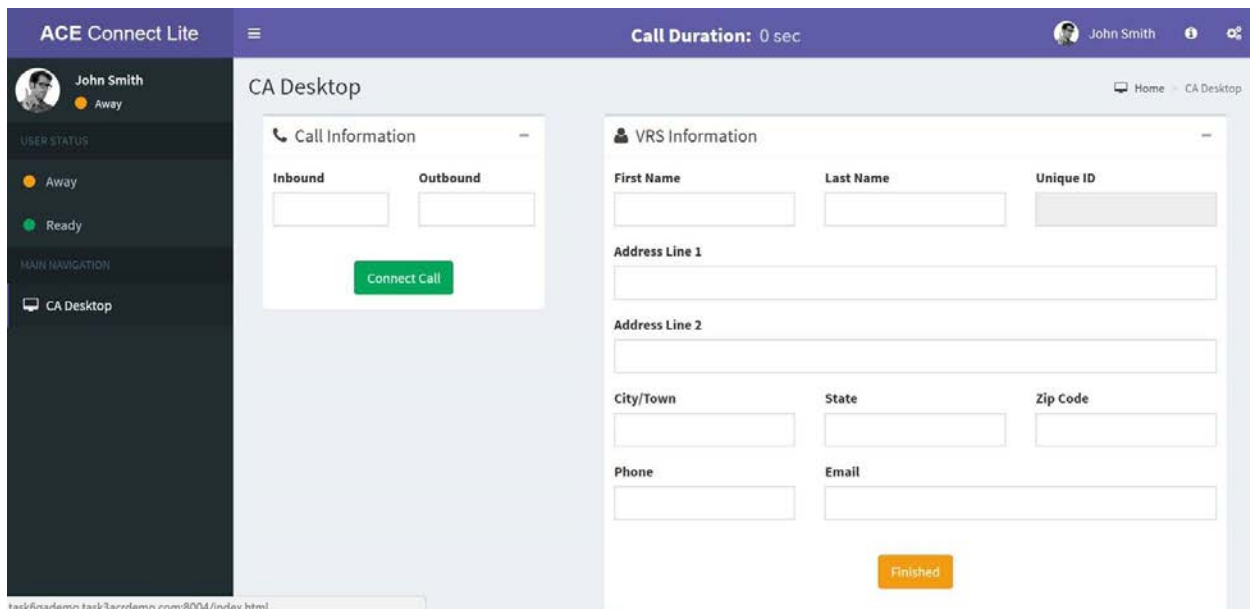


Figure 2. ACE Connect Lite CA Desktop

CA Status

Figure 3 shows views of the CA Desktop left panel. In the first view on the reader’s left, there are the USER STATUS and MAIN NAVIGATION fields. Before and after CAs sign in to the CA Desktop, the default USER STATUS is “Away” as shown in the first image. CAs can use the MAIN NAVIGATION field to select the “Ready” state, as shown in the center image, when the CA is ready to receive calls or to revert the status to the “Away” state when needed. The USER STATUS changes automatically to the “In Call” state as shown in the third image on the reader’s right after the CA receives a call to perform a service. The CA cannot select the “In Call” status.

After the CA leaves the call and clicks on the “Finished” button, the USER STATUS returns to the “Ready” state. The CA status is also shown on the Management Dashboard as described in subsection 3.4.1.

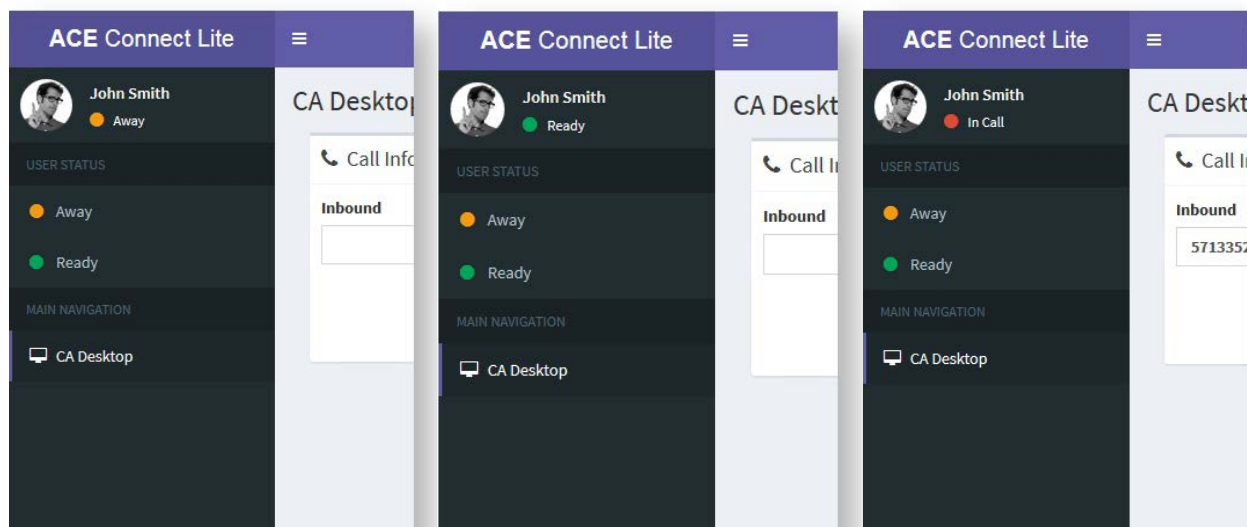


Figure 3. CA Status: ACE Connect Lite

Call Information

The Call Information section, shown in Figure 4, appears in the top left portion of the main content area. Upon receiving a call either from the VRS or PSTN user, the caller's phone number will appear in the "Inbound" field. The "Outbound" field will be populated with the destination (callee) phone number the caller is trying to reach. Depending on which number is a registered VRS user, a deaf/hard-of-hearing icon will appear after the "Inbound" or "Outbound" title. The CA clicks the "Connect Call" button to initiate the outbound call.

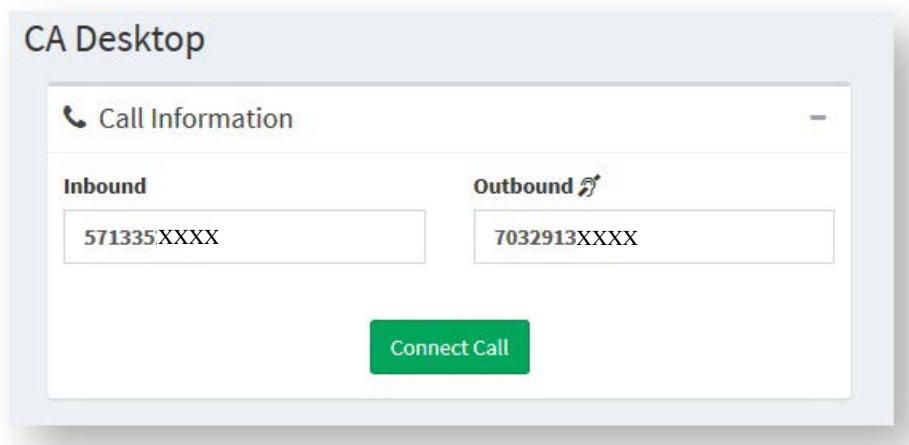


Figure 4. ACE Connect Lite Call Information

VRS User Information

The right section of the main content area presents the VRS User Information. This section will be populated with the VRS caller information once the CA has accepted the incoming call in the ACE APP. Alternatively, when an incoming PSTN call is accepted in order to reach a VSR user, this section will also be populated with the VRS user's information.

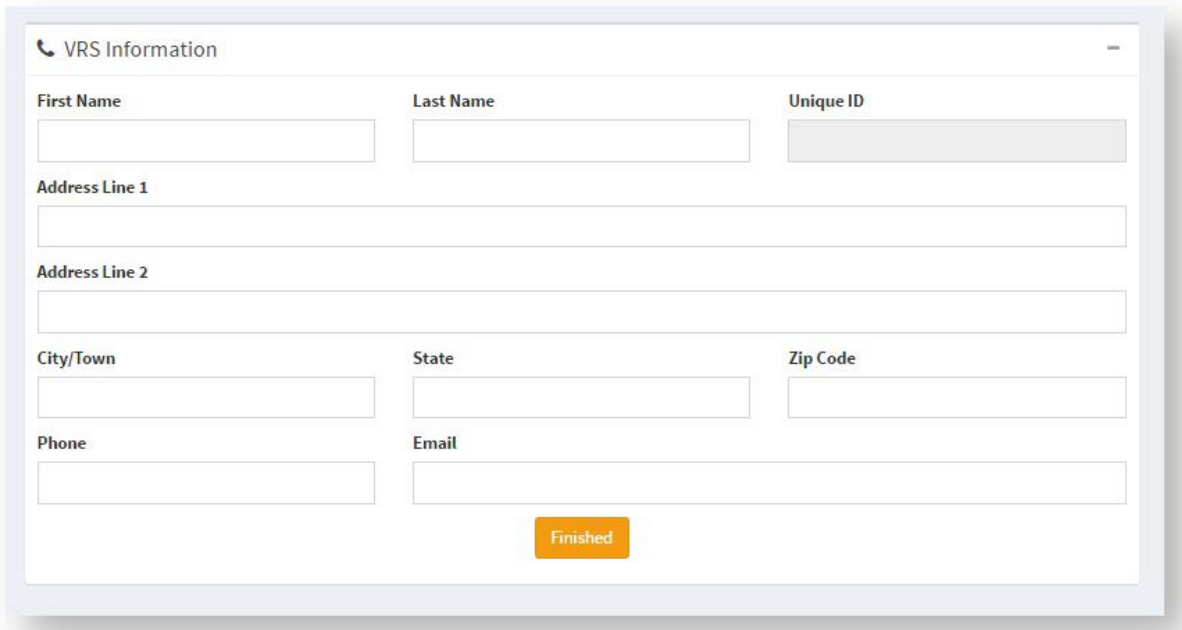
The image shows a web form titled "VRS Information" with a telephone icon. The form contains several input fields: "First Name", "Last Name", and "Unique ID" in the top row; "Address Line 1" and "Address Line 2" in the middle; and "City/Town", "State", "Zip Code", "Phone", and "Email" in the bottom section. A yellow "Finished" button is located at the bottom center of the form.

Figure 5. ACE Connect Lite VRS User Information

As noted in Table 1, the “Provider” server is the source of this VRS User information. The “Provider” server emulates the iTRS-URD where VRS user information resides in the real world. When the CA finishes the call service and is ready to take a new call, the CA clicks the “Finished” button to clear the the VRS information. At that point, the CA’s status will be changed to “Ready”.

3. Installation and Configuration

This section presents guidance for the installation and configuration of the ACE Connect Lite components. To reproduce the proof of concept, download the configuration and source code files from the <http://github.com/MITRE/ACE> website.

3.1 AWS

ACE Connect Lite is implemented in the Amazon Web Service Cloud. The following tasks are required to establish end-to-end communications:

- Creating an ACR/Asterisk instance
 1. In the EC2 console within AWS, click on “AMIs” to the left. Then, select the latest version of the ACR image (AMI ID ami-c643dfd1 as of 8/8/16) and click “Launch”.
 2. Configure your instance, then review and launch. Use <purpose>_ACR_server as the naming convention of the server under the “Tag Instance” step. **Note:** The AMI has a default storage amount configured. Keep this in mind if you desire a different amount of storage.
 3. Once the instance is up and running, click on “Elastic IPs” on the left, then click on the “Allocate New Address” button at the top. Find the IP you just created (it should be the only one without an instance associated with it), right click on it, and select “Associate Address”. Then use the search bar to type in the name of your instance and select it.
 4. Go to GoDaddy and sign into the ACR account. Click on the gear button within the large box to the left, then click on “Manage DNS”. Scroll down to the bottom of the list, then click on “Add”. Select type “A” from the drop-down list, then for host, type in <purpose>sip. In the “Points to” box, copy and paste the Elastic IP address you allocated to the ACR instance, then select your desired TTL (Time to Live, standard option is 1/2 hour). Then click “Save”.
 5. SSH into the ACR instance as the root user. Then open the sip.conf file within the /etc/asterisk directory. Modify the “realm”, “media_address”, and “externaddr” properties to point to your ACR instance’s elastic IP address, then save and quit. Restart the service using the “service asterisk restart” command.
- Creating a Node.js instance
 1. In the EC2 console within AWS, click on “AMIs” to the left. Then, select the latest version of the Apache_v2 image (AMI ID ami-7bbf386c as of 7/18/16) and click “Launch”.
 2. Configure your instance, then review and launch. Use <purpose>_Node_server as the naming convention of the server under the “Tag Instance” step. **Note:** The AMI has a default storage amount configured. Keep this in mind if you desire a different amount of storage.

3. Once the instance is up and running, click on “Elastic IPs” on the left, then click on the “Allocate New Address” button at the top. Find the IP you just created (it should be the only one without an instance associated with it), right click on it, and select “Associate Address”. Then use the search bar to type in the name of your instance and select it. Add the new elastic IP address to the “Provider Instance EIP” security group, following the same rule format as the other IP addresses in the “Inbound” tab.
 4. Go to GoDaddy and sign into the ACR account. Click on the gear button within the large box to the left, then click on “Manage DNS”. Scroll down to the bottom of the list, then click on “Add”. Select type “A” from the drop-down list, then for host, type in “<purpose>demo”. In the “Points to” box, copy and paste the Elastic IP address you allocated to the ACR instance, then select your desired TTL (Time to Live, standard option is 1/2 hour). Then click “Save”.
 5. In the EC2 console of AWS, click on “Security Groups” on the left-hand side. Create a new security group titled “<purpose> Asterisk Incoming” with the same description. Add an inbound rule to allow all traffic from the Node instance you just created, using its elastic IP address as the source for the rule. Then under “Instances” on the left-hand side, right click on your Asterisk instance and click on “Change Security Groups” under “Networking” and add the newly created security group.
 6. SSH into the ACR instance as the centos user (modify the connection string given by AWS when you right click on the instance and select “Connect”). Change to the root user, then move to the /root/.ssh directory and use vim to open the authorized_keys file. Copy the public key with the name of the private key you used to create the instance. Then move to the /home/ec2-user/.ssh directory and append the public key to the end of the authorized_keys file in that directory.
 7. Log out of the instance and log back in as the ec2-user using the private key. Then move to the /home/ec2-user/acrdemo-all directory and open the config-aws.json file with vim. Under “asterisk”:“sip”, modify the “host” and “realm” properties to point to the DNS name of the associated Asterisk instance, and modify the “host-o” and “realm-o” properties to point to the elastic IP address of the Asterisk instance. Then save and exit.
 8. Execute the following three scripts: start.sh, start-user.sh, start-db.sh. Your environment should now be set up. To validate your environment, run a quick sanity check.
- Deploying an EC2 Instance
 1. In the EC2 dashboard of the AWS console, click on “Launch Instance”.
 2. Choose an AMI for launching the instance. Choose an instance type. For dev/test instances, use the standard type (t2.micro).a. For a good CentOS image, search for the AMI with the following ID: ami-6d1c2007.
 3. Configure the instance details. The standards can be left alone on this page; however, you may want to select “Enable” for the “Auto-assign IP Role” option to ensure your instance is easily accessible.

4. Select the amount of storage you want available on the instance. The default is 8 GB. If you do not want a different amount, you can skip this step.
5. Tag the instance with a unique name. Use “firstName_purpose” as the naming convention. Add any other tags you would like your instance to have.
6. Click on “Select an existing security group”, then select the “Company IPs” and “Home IPs” security groups. The “Company IPs” security group contains all the IPs from the Company networks.
7. Review your details then launch the instance. You will be prompted to select a key-pair to use with this instance. If you have not done so already, create a new key-pair and download the private key to your local machine. You will need this later when you want to SSH into the instance.
8. Once the instance has been launched, click on “View Instances” to check the status of your new instance. Once the instance is in status “running”, right-click on your instance and select “Connect” to view instructions for connecting to your instance via SSH or a web browser.
 - a. You will need the private key you generated in the previous step to connect to your instance. Be sure not to lose this key or you may not be able to connect to your instance in the future!
 - b. If you are using an open source terminal (such as Cygwin), you may need to install the openssh and openssl package in order to SSH into the instance.

3.2 Asterisk Server

ACE Connect Lite Asterisk Server is built with the following software versions:

- OS: CentOS 6.7
- Asterisk: 13.8.0

The version 13.8.0 can be downloaded from

<http://downloads.asterisk.org/pub/telephony/asterisk/releases/> (scroll down to “asterisk-13.8.0-rc1.tar.gz 22-Mar-2016 16:25 31M”).

Details of the Asterisk installation can be found at

<https://wiki.asterisk.org/wiki/display/AST/Installing+Asterisk+From+Source>

Download and install the required prerequisite software. This will vary by OS. The following listing is for CentOS:

```
Sudo yum -y install gcc gcc-c++ php-xml php php-mysql php-pear php-mbstring sqlite-devel lynx bison gmime-devel psmisc tftp-server httpd make ncurses-devel libtermcap-devel sendmail sendmail-cf caching-nameserver sox newt-devel libxml2-devel libtiff-devel audiofile-devel gtk2-devel uuid-devel libtool libuuid-devel subversion kernel-devel kernel-devel-$(uname -r) git subversion kernel-devel
```



```
php-process crontabs cronie cronie-anacron wget vim
jansson-devel libsrtp-devel zip unzip
```

After starting your preferred server instance, update the OS base packages by running:

```
sudo update -y
```

Once you have downloaded Asterisk, navigate to the directory and unzip the file. Run:

```
./configure
make menuselect
```

This option loads a graphic interface from which you may select add-ons and resource modules for the asterisk installation. The resource module `res_srtp` should be selected. If the option is not available, you may be lacking the `srtp` packages. Ensure that the “`libsrtp-devel`” package installed by the above `yum` command was installed successfully. If you need to reinstall a package, you will have to rerun the “`configure`” script.

```
make && make install
```

Generate the sample config files from Asterisk (by running the following command) and move them to the `/etc/asterisk` directory, removing the “.sample” suffix from the files. Once the files have been created, make sure the asterisk instance is running by running ‘`service asterisk status`’. If the server is not running, ‘`service asterisk start`’ may be used.

```
make samples
```

Then restart the asterisk service with the command:

```
service asterisk restart
```

Dial Plan

The dial plan is defined in the `extensions.conf` and `sip.conf` files under `/etc/asterisk`. Table 3 shows an example of endpoint devices and the associated extension along with the required codecs for the sample dial plan configuration in this guide.

Table 3. Asterisk Endpoint Extensions

Extension	Purpose	Device	Codec
6001	ACE APP user 1	softphone	h264/ulaw
6002	ACE APP user 2	softphone	h264/ulaw
6003	ACE APP user 3	softphone	h264/ulaw
6004	ACE APP user 4	softphone	h264/ulaw

There are two steps to set up a dial plan:

Step 1 (extensions.conf file configuration)

Define the dial plan in extensions.conf for specific phone numbers to route, as well as number schemes for unknown numbers, using the wildcard “X”. In our example and as shown below, in the section [from-internal] of the extensions.conf file, all extensions ended with 6xxx are configured. Nothing else needs to be changed in the file.

```
[from-internal]

exten => _6XXX,1,Dial(SIP/${EXTEN})
same => n,DumpChan( )
same => n,HangUp( )
```

The “[from-internal]” label must be defined as the same for the “Context” attribute of the sip.conf file (see Step 2 below). For the Dial() function call, the extensions/numbers defined within the function call must be defined in sip.conf, as shown in Step 2.

Step 2 (sip.conf file configuration)

Once the dial plan has been configured for the devices (e.g., ACE APP) to be registered with the Asterisk server, the profile for those devices can be configured in sip.conf under the “[h264_phone_test](!)” section. The “h264_phone_test” profile name can be changed according to the user’s preference.

```
-----top of sip.conf file-----
[general]
icesupport=true
stunaddr=<STUN_server_DNS_name>:3478
context=from-internal
allowoverlap=no
videosupport=yes
realm=<IP_address>      ---- External IP of Asterisk AWS Instance
tcpenable=yes
tcpbindaddr=0.0.0.0
transport=udp,tcp,ws,wss
srvlookup=yes
tlsenable=yes
tlsbindaddr=0.0.0.0
tlscertfile=/etc/asterisk/keys/asterisk.pem
tlscipher=ALL
tlsclientmethod=tlsv1
websocket_enabled=yes
media_address=<IP_address> ---- External IP of Asterisk AWS Instance
localnet=172.31.16.0/255.255.240.0
externaddr=<IP_address>  ---- External IP of Asterisk AWS Instance
nat=force_rport,comedia

[h264_phone_test](!)
```

```
type=friend
transport=tcp,udp,tls
port=5060
secret=1qaz!qaz           ; login password for the registered device
host=dynamic               ; This device needs to register
directmedia=no             ; Typically set to NO if behind NAT
disallow=all
allow=ulaw
allow=h264
registertrying=yes         ; Send a 100 Trying when the device registers.
context=from-internal      ; the context to be used with this profile
callcounter = yes          ; Enable call counters on devices. This can be set per
qualify=yes
busydetect=yes
videosupport=yes           ; Turn on support for SIP video. You need to turn this
dtmfmode=rfc2833
insecure = invite,port
accept_outofcall_message=yes
outofcall_message_context=IM
auth_message_requests=yes
```

```
[6001](h264_phone_test)
regexten=6001
callerid=<6001>
```

```
[6002](h264_phone_test)
regexten=6002
callerid=<6002>
```

```
[6003](h264_phone_test)
regexten=6003
callerid=<6003>
```

```
[6004](h264_phone_test)
regexten=6004
callerid=<6004>
```

-----end of sip.conf file-----

As shown above, the profile starts with “disallow=all” first and then uses “allow” to explicitly allow certain codec formats such as ulaw and h264.

Once the profile has been defined, extensions and numbers can be associated with the profile, which will be referred to back in the extensions.conf file to help route calls to their proper destination.

3.3 Node.js

The Node.js is a platform built on Chrome's V8 JavaScript run-time engine developed to build fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. The Node.js server hosts the CA desktop, Dashboard, and CDR. Instructions for downloading and installing node.js can be found on the official Node.js website (<https://nodejs.org/en/>). For this proof of concept, the Node.js server is built with the following software versions:

- OS: CentOS 6.7
- Node.js: v0.10.41

NPM

The npm is the Node.js package manager. As its name implies, npm is used to install and uninstall node packages. Also, it makes it easier to specify and link dependencies. A standard Node.js project contains a package.json file that lists the metadata about the Node.js application and a list of its dependencies. To install the Node.js dependencies, run the following command within the applications folder:

Code Block 1 npm

```
C:\Projects\aceconnectlite> npm install
```

BOWER

Bower is a dependency management tool for managing front-end components such as HTML, CSS, and JavaScript files. It ensures that the correct version of any dependent package is installed. Because Bower is not included as part of a standard Node.js installation, install it separately. To install Bower, first install Node.js, and then run the following command:

Code Block 2 Bower Global Install

```
C:\Projects\aceconnectlite> npm install -g bower
```

After Bower is installed, packages can be added to the project by running the following command:

Code Block 3 Bower Install Packages

```
C:\Projects\aceconnectlite> bower install <package-name> --  
save
```

These packages are defined in a manifest file, bower.json. Once the project has the manifest file defined, all of the project's dependencies can be installed by running the following command:

Code Block 4 Bower Dependency Install

```
C:\Projects\aceconnectlite> bower install
```

PROXY CONFIGURATION

Some networks may require a proxy to be configured in order to use Bower. To configure the proxy locate (or create if not present) the `.bowerrc` file within the project folder and include the following:

Code Block 5 .bowerrc

```
{
  "directory": "bower_components",
  "proxy": "<proxy>",
  "https-proxy": "<proxy>"
}
```

3.4 Management Portal

ACE Connect Lite Management Portal provides a number of Key Performance Indicators (KPI) for real-time monitoring by the manager. This information may be used to improve user experience, increase user satisfaction, and overall call center performance.

Installation

- Use WinSCP (or an equivalent tool) to securely copy the management portal folder to `/tmp` on the server hosting the management portal components
- Copy contents of `/tmp/managementportal` to `/var/www/html`
- `cd /var/www/html/managementportal`
- Run `npm install` to install the required Node.js modules

Configuration

Update the following parameters in `config.json` if necessary:

- `port-dashboard` – This is the port number used by the dashboard; the default port is 9081
- `Asterisk/sip/host` – Fully qualified domain name (e.g., `mySIP.example.com`)
- `dashboard/queues` – Queue names configured for ACE Connect Lite. Currently, ACE Connect Lite has one queue named: `InboundQueue`

Run

Enter the following commands at the command prompt to launch the Node.js server.

```
cd /var/www/html/managementportal
```

```
Run "./start-db.sh" to start the management portal Node.js
server
```

Log Files

Log files are located in `/var/www/html/managementportal/logs`. The debug level is defined by the `debugLevel` field in the configuration file and the valid options are: `all`, `trace`, `debug`, `info`, `warn`, `error` and `fatal`.

3.4.1 Management Dashboard

The Management Portal Dashboard, as shown in Figure 6, provides a number of KPIs the manager can monitor in real time. To access the Management Portal Dashboard:

- Start a browser on a machine that can access the Management Portal server, Node.js.
- Enter the following URL “http://<hostname>:<port>/dashboard.html” where <hostname> is the host name of the Management Portal server, and the port is the port-dashboard parameter defined in config.json, e.g., “http://myhost.example.com:9081/dashboard.html”.

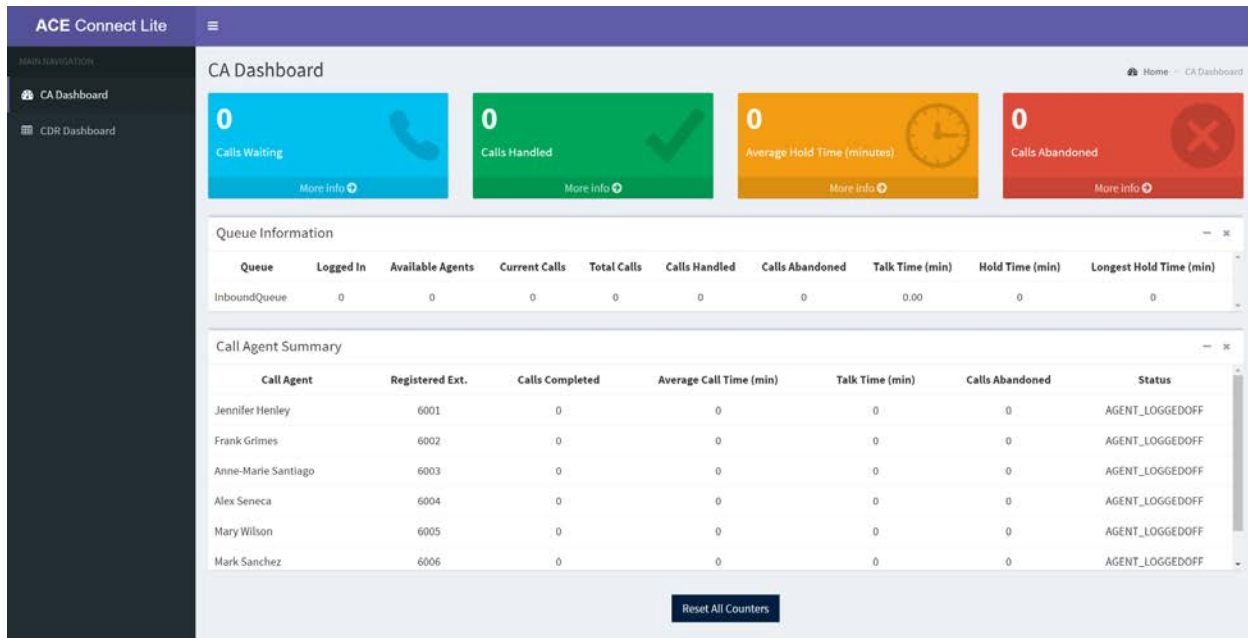


Figure 6. Management Dashboard

KPI Types

ACE Connect Lite Management Dashboard presents three types of KPIs:

1. **Summary Data**
 - a. Calls Waiting – Number of calls waiting in the queue
 - b. Calls Handled – Number of calls completed
 - c. Average Hold Time (in min) – Average call holding time
 - d. Calls Abandoned – Number of calls not answered
2. **Queue-related KPIs**
 - a. Logged In – Number of agents currently logged into the system
 - b. Available Agents – Number of agents currently in a ready state
 - c. Current Calls – Number of calls currently in progress
 - d. Total Calls – Total number of calls made for the queue

- e. Calls Handled – Total number of calls answered by an agent
 - f. Calls Abandoned – Total number of calls abandoned
 - g. Talk Time – Average talk time
 - h. Hold Time – Average hold time
 - i. Longest Hold Time – The longest hold for a call
3. **Agent-related KPIs** – The following KPIs are displayed per agent. The agent name and registered extensions are displayed along with the KPI:
- a. Calls Completed – Number of calls handled (answered and completed) by the agent
 - b. Average Call Time – Talk Time divided by number of calls
 - c. Talk Time – The cumulative time the agent has spent on calls
 - d. Calls Abandoned – Calls not answered by the agent
 - e. Agent status – Logged Off, Ready, Away, or In-Call

Web Server and Client Configuration

The management dashboard functionality relies on the Asterisk server to collect reports on call agents and queue status. The management dashboard (dashboard.html) is hosted on the Node.js server. The operational data flow works as follows.

Server Side

server-db.js – Communicates with the Asterisk Server through the Asterisk Management Interface (AMI) protocol. AMI events provide the Management Portal with data on queue and agent status.

Client Side

dashboard.html – Controls the UI for the CA Dashboard

dashboard.js – Uses JavaScript data structures to store information to be displayed on the dashboard.html page

Asterisk Management Interface

The AMI allows a client application to connect to an Asterisk instance and issue actions (requests) and receive events (responses). The Management Portal performs five unique AMI action calls to collect data on queue and agent status. Table 4 shows the AMI actions and descriptions.

Table 4. AMI Actions and Descriptions

AMI Action	Description	Syntax	Arguments	Triggered Events
Agents	Queries for information about all agents.	Action: Agents ActionID: <value>	<ul style="list-style-type: none"> ActionID – ActionID for this transaction. Will be returned. 	Agents AgentsComplete
QueueReset	Resets the running statistics for a queue.	Action: QueueReset ActionID: <value> Queue: <value>	<ul style="list-style-type: none"> ActionID – ActionID for this transaction. Will be returned. Queue – The name of the queue on which to reset statistics. 	
Queues	Queries for queues information.	Action: Queues		Queues
QueueStatus	Check the status of one or more queues.	Action: QueueStatus ActionID: <value> Queue: <value> Member: <value>	<ul style="list-style-type: none"> ActionID – ActionID for this transaction. Will be returned. Queue – Limit the response to the status of the specified queue. Member – Limit the response to the status of the specified member. 	QueueStatus QueueStatusComplete
QueueSummary	Requests Asterisk to send a QueueSummary event.	Action: QueueSummary ActionID: <value> Queue: <value>	<ul style="list-style-type: none"> ActionID – ActionID for this transaction. Will be returned. Queue – Queue for which the summary is requested. 	QueueSummary QueueSummaryComplete

The dashboard server listens for the Asterisk AMI events shown in Table 5.

Table 5. Asterisk AMI Event

AMI Action	Description	Syntax	Fields
AgentsComplete	Generated when an agent has finishes servicing a member in the queue.	Event: AgentComplete Queue: <value> Member: <value> MemberName: <value> HoldTime: <value> [Variable:] <value> TalkTime: <value> Reason: <value> Queue: <value> Uniqueid: <value> Channel: <value> Member: <value> MemberName: <value> HoldTime: <value>	<ul style="list-style-type: none"> • Queue – The name of the queue. • Member – The queue member's channel technology or location. • MemberName – The name of the queue member. • HoldTime – The time the channel was in the queue, expressed in seconds since 00:00, Jan 1, 1970 UTC. • Variable – Optional channel variables from the ChannelCalling channel • TalkTime – The time the agent talked with the member in the queue, expressed in seconds since 00:00, Jan 1, 1970 UTC. • Reason <ul style="list-style-type: none"> – caller – agent – transfer • Queue • Uniqueid • Channel • Member • MemberName • HoldTime
AgentLogin	Generated when an agent logs in.	Event: AgentLogin Agent: <value> Channel: <value> Uniqueid: <value>	<ul style="list-style-type: none"> • Agent – The name of the agent. • Channel • Uniqueid
AgentLogoff	Generated when an agent logs off.	Event: AgentLogoff Agent: <value> Agent: <value> Logintime: <value> Uniqueid: <value>	<ul style="list-style-type: none"> • Agent – The name of the agent. • Agent • Logintime • Uniqueid
FullyBooted	Generated when all Asterisk initialization procedures complete.	Event: FullyBooted Status: <value>	<ul style="list-style-type: none"> • Status

AMI Action	Description	Syntax	Fields
QueueMemberAdded	Generated when a new member is added to the queue.	Event: QueueMemberAdded Queue: <value> Location: <value> MemberName: <value> StateInterface: <value> Membership: <value> Penalty: <value> CallsTaken: <value> LastCall: <value> Status: <value> Paused: <value> Queue: <value> Location: <value> MemberName: <value> StateInterface: <value> Membership: <value> Penalty: <value> CallsTaken: <value> LastCall: <value> Status: <value> Paused: <value>	<ul style="list-style-type: none"> • Queue – The name of the queue. • Location – The queue member's channel technology or location. • MemberName – The name of the queue member. • StateInterface – Channel technology or location from which to read device state changes. • Membership <ul style="list-style-type: none"> – dynamic – realtime – static • Penalty – The penalty associated with the queue member. • CallsTaken – The number of calls this queue member has serviced. • LastCall – The time this member last took call, expressed in seconds since 00:00, Jan 1, 1970 UTC. • Status – The numeric device state status of the queue member. <ul style="list-style-type: none"> – 0 - AST_DEVICE_UNKNOWN – 1 - AST_DEVICE_NOT_INUSE – 2 - AST_DEVICE_INUSE – 3 - AST_DEVICE_BUSY – 4 - AST_DEVICE_INVALID – 5 - AST_DEVICE_UNAVAILABLE – 6 - AST_DEVICE_RINGING – 7 - AST_DEVICE_RINGINUSE – 8 - AST_DEVICE_ONHOLD • Paused <ul style="list-style-type: none"> – 0 – 1 • Queue • Location • MemberName • StateInterface • Membership • Penalty • CallsTaken • LastCall • Status • Paused

AMI Action	Description	Syntax	Fields
QueueMemberRemoved	Generated when a queue member is removed from the queue.	Event: QueueMemberRemoved Queue: <value> Location: <value> MemberName: <value> Queue: <value> Location: <value> MemberName: <value>	<ul style="list-style-type: none"> Queue – The name of the queue. Location – The queue member's channel technology or location. MemberName – The name of the queue member. Queue Location MemberName

3.4.2 Call Detail Record Dashboard

A CDR is an event generated by Asterisk when a call is completed. A CDR contains metadata that describes each call, such as call source, destination, and timestamp. The CDR dashboard provides a means of auditing call activity, tracking a call agent's activity, and creates a listing of both incoming and outgoing calls. The ACE Connect Lite Call Detail Record Dashboard, as shown in Figure 7 provides a method to view and export the Asterisk CDRs stored in the MySQL database.

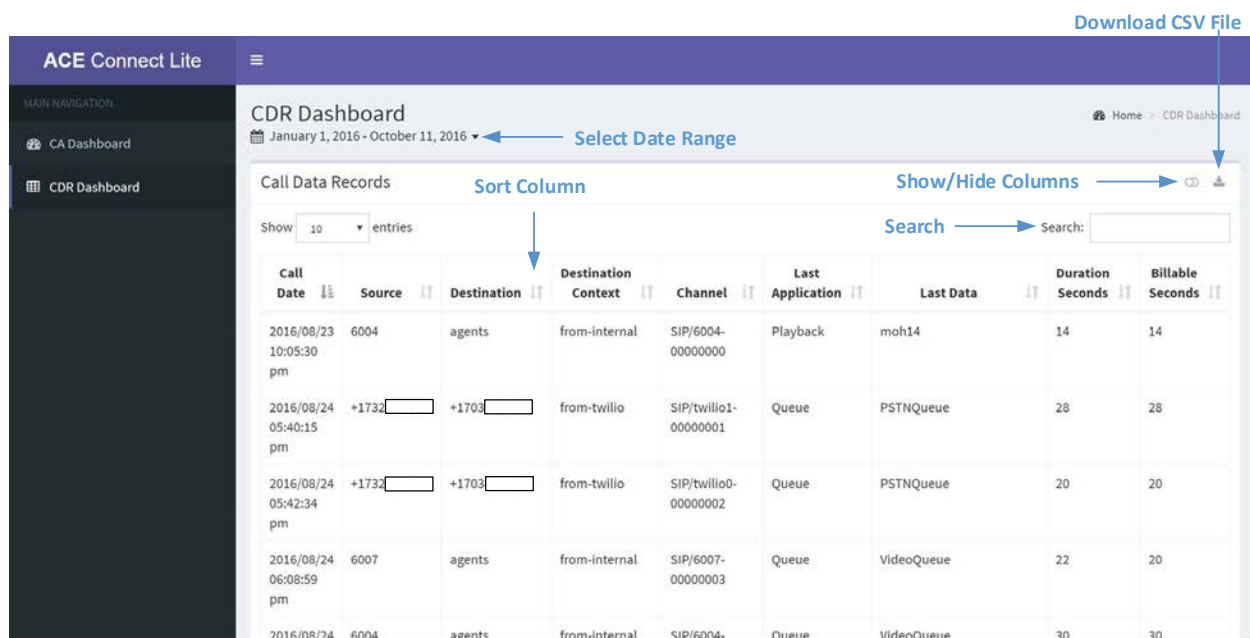


Figure 7. ACE Connect Lite Call Detail Record Dashboard

The CDR Dashboard allows the user to perform the following actions:

- Select Date Range:** The user can select a date range for the report. Predefined values are Today, Yesterday, Last 7 days, Last 30 days, This Month, Last Month, All Time, and Custom Range. Default selection is “All Time” (January 1st 2016 to Today).

- **Sort Column:** The user can sort any column by clicking the sort icon located next to each column name. To multi-sort columns, the user depresses the shift key when sorting columns.
- **Show/Hide Columns:** This action expands the table to include the following columns: Caller ID Text, Destination Channel, Disposition, AMA Flags, Account Code, User Field, Unique ID, Linked ID, Sequence, and Peer Account.
- **Download CSV File:** This action downloads the table as a Comma Separated Values (CSV) file. The CSV file contains only data within the date range.
- **Search:** The user can search the entire table. Search results are displayed in near real time.

Table 6 presents the column definitions in the CDR table.

Table 6. ACE Connect Lite Call Detail Record Column Definition

Display Name	Database Column	Description
Call Date	calldate	The start datetime of the call. Default format: 2016-09-07T09:35:41Z dashboard formats the date to 2016/09/07 09:35:41 pm (adjusted for timezone)
Caller ID Text	clid	The full caller ID, including the name, of the calling party. This field is set automatically and is read-only.
Source	src	The calling party's caller ID number. It is set automatically and is read-only.
Destination	dst	The destination extension for the call. This field is set automatically and is read-only.
Destination Context	dcontext	The destination context for the call. This field is set automatically and is read-only.
Channel	channel	The calling party's channel. This field is set automatically and is read-only.
Destination Channel	dstchannel	The called party's channel. This field is set automatically and is read-only.
Last Application	lastapp	The last dialplan application that was executed. This field is set automatically and is read-only.
Last Data	lastdata	The arguments passed to the lastapp. This field is set automatically and is read-only.
Duration Seconds	duration	The number of seconds between the start and end times for the call. This field is set automatically and is read-only.
Billable Seconds	billsec	The number of seconds between the answer and end times for the call. This field is set automatically and is read-only.
Disposition	disposition	An indication of what happened to the call. This may be NO ANSWER, FAILED, BUSY, ANSWERED, or UNKNOWN.

Display Name	Database Column	Description
AMA Flags	amaflags	The Automatic Message Accounting (AMA) flag associated with this call. This may be one of the following: OMIT, BILLING, DOCUMENTATION, or Unknown.
Account Code	accountcode	An account ID. This field is user-defined and is empty by default.
User Field	userfield	A general-purpose user field. This field is empty by default and can be set to a user-defined string.
Unique ID	uniqueid	The unique ID for the src channel. This field is set automatically and is read-only.
Linked ID	linkedid	A unique identifier that unites multiple CDR records.
Sequence	sequence	A numeric value that, combined with uniqueid and linkedid, can be used to uniquely identify a single CDR record.
Peer Account	peeraccount	The account code of the called party's channel

Configuration

By default, the Asterisk server stores CDRs in a file called master.csv located in /var/log/cdr-csv. CDR records can be collected in real time, or near real time, in various ways. The Asterisk server allows different types of database connections. One method is to utilize asterisks on an internal application for connecting to a database. Another is to configure an Open Database Connectivity (ODBC) connection for Asterisk. There is no restriction on the type of database that can be used. MySQL, MariaDB, and others are all viable options. For this proof of concept, CAMH decided to use a MySQL server with Asterisk connecting through an ODBC.

To start, it is necessary to install additional packages for the MySQL server and the ODBC connector, as follows:

```
sudo yum install mysql-server unixODBC unixODBC-devel
libtool-ltdl libtool-ltdl-devel
sudo yum install mysql-connector-odbc
```

Once the packages are installed, Asterisk must recognize these new packages. To do so, migrate to the Asterisk directory located in /usr/src/asterisk-13.8.0/. From this directory, run:

```
./configure
make menuselect
```

Once the graphic window is displayed, check to make sure that the res_odbc module is selected. Save and exit the graphic window. Run the following:

```
make && make install
```

This will configure the Asterisk installation for use with the ODBC connection. Once the installation script successfully finishes, you can start the SQL server.

Once MySQL has been installed, run the basic hardening script for MySQL to remove anonymous connections, the test database, and establish the root password. To do this, run the following script and follow the prompts:

```
sudo /usr/bin/mysql_secure_installation
```

After installing MySQL and completing the initial configuration, log in to the MySQL using:

```
MySQL -u root -p
```

Once logged in, create the database and the table needed to store the CDRs. To do so, run the following commands:

```
CREATE DATABASE asterisk;
GRANT INSERT ON asterisk.* TO asterisk@localhost
IDENTIFIED BY 'yourpassword';USE asterisk;
CREATE TABLE `bit_cdr` ( `calldate` datetime NOT NULL
default '0000-00-00 00:00:00', `clid` varchar(80) NOT NULL
default '', `src` varchar(80) NOT NULL default '', `dst`
varchar(80) NOT NULL default '', `dcontext` varchar(80) NOT
NULL default '', `channel` varchar(80) NOT NULL default
'', `dstchannel` varchar(80) NOT NULL default '', `lastapp`
varchar(80) NOT NULL default '', `lastdata` varchar(80) NOT
NULL default '', `duration` int(11) NOT NULL default '0',
`billsec` int(11) NOT NULL default '0', `disposition`
varchar(45) NOT NULL default '', `amaflags` int(11) NOT
NULL default '0', `accountcode` varchar(20) NOT NULL
default '', `userfield` varchar(255) NOT NULL default '',
`uniqueid` VARCHAR(32) NOT NULL default '', `linkedid`
VARCHAR(32) NOT NULL default '', `sequence` VARCHAR(32) NOT
NULL default '', `peeraccount` VARCHAR(32) NOT NULL default
'' );
ALTER TABLE `bit_cdr` ADD INDEX ( `calldate` );
ALTER TABLE `bit_cdr` ADD INDEX ( `dst` );
ALTER TABLE `bit_cdr` ADD INDEX ( `accountcode` );
```

To create a user account for remote connections, use the following syntax:

```
CREATE USER 'username'@'localhost' IDENTIFIED BY
'password';GRANT ALL PRIVILEGES ON *.* TO
'username'@'localhost' WITH GRANT OPTION;CREATE USER
'username'@'%' IDENTIFIED BY 'password';GRANT ALL
PRIVILEGES ON *.* TO 'username'@'%' WITH GRANT OPTION;FLUSH
PRIVILEGES;
```

Now that the database is up and running, connect to Asterisk by using the ODBC connector, as follows:

```
vi /etc/odbcinst.ini
[MySQL]
Description      = ODBC for MySQL
Driver           = /usr/lib/libmyodbc5.so
Setup            = /usr/lib/libodbcmyS.so
Driver64         = /usr/lib64/libmyodbc5.so
Setup64          = /usr/lib64/libodbcmyS.so
FileUsage        = 1

vi /etc/odbc.ini
[asterisk-connector]
Description = MySQL connection to 'asterisk' database
Driver = MySQL
Database = asterisk
Server = localhost
User = root
Password = password
Port = 3306
Socket = /var/lib/mysql/mysql.sock
```

To test your configuration, run the following command from the CLI:

```
echo "select 1" | isql -v asterisk-connector
```

You should see a message of Connected! returned.

```
| Connected!          |
SQLRowCount returns 1 1 rows fetched
```

The final component is to point Asterisk to your new database. To do so, modify the `/etc/asterisk/res_odbc.conf` file with the database name, username, password, and port of the odbc connection you have set up:

```
[asterisk]
enabled => yes
dsn => asterisk-connector
username => username
password => password
pooling => no
limit => 99999
pre-connect => yes
```

Additionally, apply the following three Asterisk config files from the <http://github.com/MITRE/ACE> website:

- Cdr.conf
- Cdr_odbc.conf
- Cdr_manager.conf

The CDR reporting functionality relies on two Node.js servers: the management portal (server-db.js) and the acr-cdr (app.js). These servers provide the following capabilities:

- server-db.js –
 - Receives GET call for /cdrinfo.
 - Performs GET call to app.js /getallcdrrecs.
 - Acts as a middle man for the cdr.html page to the app.js node server. This node server can be bypassed by changing the cdr.html GET call from /cdrinfo to `http://<app.js location>/getallcdrrecs` if the CDR Dashboard needs to be separated from the management portal.
- app.js –
 - Receives GET call for /getallcdrrecs.
 - Performs query to the Asterisk CDR database table. Returns a JSON object of the data.

3.5 VRS User Database (Provider)

The Video Relay Service user database was developed to emulate a VRS user lookup in the iTRS-URD from the CA desktop. This lookup verifies that the user-provided phone number is a registered VRS number. A MySQL database server is required and a RESTful Application Programming Interface (API) developed using Node.js provides access to the data for user verification. For this proof-of-concept effort, the Node.js server is built with the following software versions:

- Linux: Ubuntu 14.04.3 LTS
- Node.js: v0.10.25
- PHP: v5.5.9

MySQL Database Server Installation

The CAMH team developed a MySQL database containing a single table to store the emulated VRS lookup data. Create a database named portaldb and use the following MySQL script to create the table (user_data) that will store the VRS lookup data:

```
-- MySQL dump 10.13 Distrib 5.5.47, for debian-linux-gnu  
(x86_64)  
-- Host: localhost Database: portaldb  
-- Server version 5.5.47-0ubuntu0.14.04.1
```



```
/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET
@OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
--
-- Table structure for table `user_data`
--
DROP TABLE IF EXISTS `user_data`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `user_data` (
  `vrs` bigint(20) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `first_name` varchar(255) NOT NULL,
  `last_name` varchar(255) NOT NULL,
  `address` varchar(255) NOT NULL,
  `city` varchar(255) NOT NULL,
  `state` varchar(255) NOT NULL,
  `zip_code` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `isAdmin` tinyint(1) NOT NULL,
  PRIMARY KEY (`vrs`),
  UNIQUE KEY `username` (`username`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=7275514879 DEFAULT
CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT
*/;
```

```
/*!40101 SET
CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION
*/;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

Verify that the MySQL database is up, running, and, accepting connections. A tool like MySQL Workbench can quickly verify that the configuration is correct.

Portal Files Installation

Install the portal files by following these steps:

- Use WinSCP (or similar tool) to copy the acrdemo-provider folder to /tmp on provider-host.example.com
- Copy contents of /tmp/acrdemo-provider/ to /var/www/html
- Start MySQL (if down): `sudo /etc/init.d/mysql start`
- Create a database named portaldb, create a username and password with read/write permissions
- Run the user_data.sql script to create the user_data table
- Restart Apache: `sudo service apache2 restart`
- Test `http://provider-host.example.com`
 - Default login: root/root
 - Default login: admin/admin

3.6 STUN

If a host is located behind a NAT, then in certain situations it can be impossible for that host to communicate directly with other hosts (peers). In these situations, the host must use the services of an intermediate node as a communication relay. This specification defines a protocol, called TURN (Traversal Using Relays around NAT), which allows the host to control the operation of the relay and to exchange packets with its peers using the relay. TURN differs from other relay control protocols because it allows a client to communicate with multiple peers using a single relay address. A TURN server, which is an implementation of the STUN protocol, uses a relay to provide an alternate method for NAT discovery and traversal (STUN). TURN is able to traverse symmetric NAT instances.

For this project, reTurn server (from reSIProcate) is used for STUN/TURN services. reTurn is a C++ implementation of STUN RFC5389 and TURN RFC5766. To install STUN services on a Ubuntu installation, start by installing the TURN server package:

```
sudo apt-get install resiprocate-turn-server
```

Edit the `etc/reTurn/reTurnServer.config` file, and modify the “AuthenticationRealm” property to point to the DNS of the server. Then restart the TURN service:

```
sudo service resiprocate-turn-server restart
```

Once the service is running, TURN will be listening on port 3478. In Asterisk, you can modify the “stunaddr” attribute in sip.conf to point to the DNS name of the server and port 3478.

3.7 iTRS ENUM Database

For deaf and hard of hearing phone numbers, iTRS is the authoritative database managed by Neustar. The lookup of the iTRS ENUM database performs DNS queries to determine a Uniform Resource Identifier (URI) for a 10-digit telephone number. There is a GUI interface as well as a programmatic interface. For this proof of concept, the CAMH team used the GUI to provision telephone numbers.

Querying of the production iTRS database requires establishing an IPsec tunnel with Neustar. For the proof of concept, the team added the IP address first in the DNS settings (/etc/resolv.conf) for the ENUM lookup to work. Note that with the default AWS instance settings, any changes made to the resolv.conf file will be overwritten on restart of the network service as new values are queries from DNS. This behavior needs to be disabled by adding the following to /etc/sysconfig/network-scripts/ifcfg-eth0:

```
ONBOOT=yes
DEVICE=eth0
BOOTPROTO=dhcp
PeerDNS=no
DNS1=(your DNS IP_address here)
DNS2=8.8.8.8
```

The following example snippet of code, which is part of an Asterisk Dial Plan, comes from an Extensions.conf file:

```
exten => _9.[1-9]XXXXXXXXXX, 1,
Set(sipuri=${ENUMLOOKUP(+${EXTEN:1},sip,,1,itrs.us)})
same => n,NoOp("Outbound Direct Video Call to: ${EXTEN:1}")
; just for informational purposes
same => s,n,SipAddHeader(P-Asserted-Identity:
<sip:7034369339>) ;set the callerID number
same => n,NoOp("sipuri: ${sipuri:1}")
same => n,Dial(SIP/${sipuri:1},30)
```

3.8 VyOS Router for Secure Socket Layer (SSL) Tunnel

To support the iTRS-ENUM database lookup, a VPN tunnel is established to Neustar from a VyOS router instance running in AWS. The router instance has a loopback interface with an EIP on it that is the encryption domain for the tunnel. Neustar requires public IP addresses for the encryption domain and do not set aside a /30 network as is typical for an ipsec tunnel.

There is a NAT (really PAT) rule created on the VyOS router to allow traffic from the AWS instances to route out over the ipsec tunnel. A routing rule is added to the VPC to route traffic destined to Neustar Encryption domain IP address to the router's private IP address. Note source/destination checking needs to be disabled on the router instance elastic network interface.

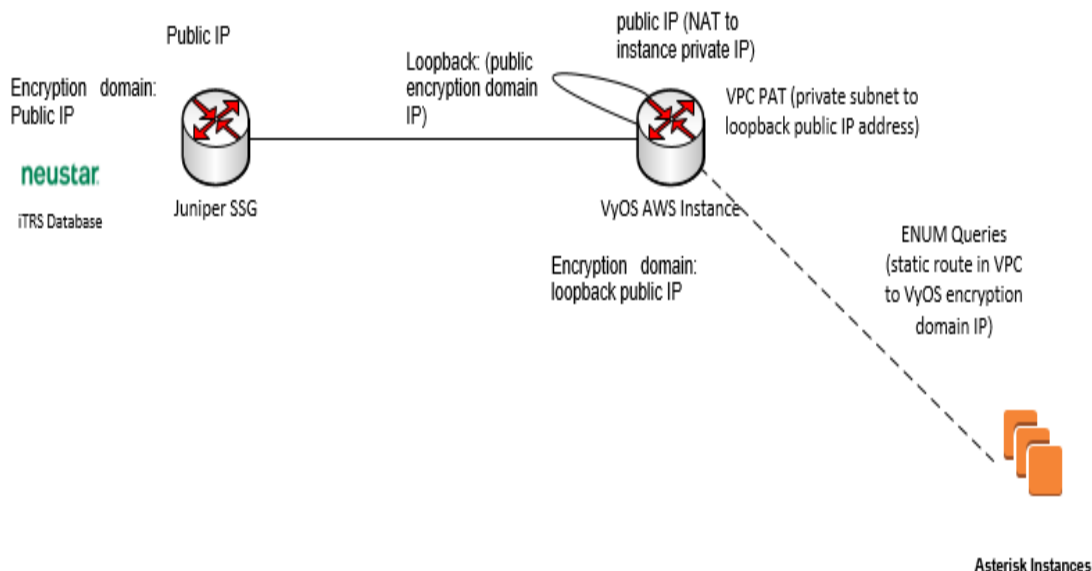


Figure 8. VyOS Router for SSL Tunnel

The VyOS router should be configured as follows.

```
set vpn ipsec esp-group trs compression disable
set vpn ipsec esp-group trs lifetime 28800
set vpn ipsec esp-group trs mode tunnel
set vpn ipsec esp-group trs pfs dh-group2
set vpn ipsec esp-group trs proposal 1 encryption aes128
set vpn ipsec esp-group trs proposal 1 hash sha1
set vpn ipsec ike-group trs lifetime 86400
set vpn ipsec ike-group trs proposal 1 dh-group 2
set vpn ipsec ike-group trs proposal 1 encryption aes128
set vpn ipsec ike-group trs proposal 1 hash sha1
set vpn ipsec ipsec-interfaces interface eth0
set vpn ipsec site-to-site peer "Neustar IP" authentication
id "VyOS public IP"
set vpn ipsec site-to-site peer "Neustar IP" authentication
mode pre-shared-secret
set vpn ipsec site-to-site peer "Neustar IP" authentication
pre-shared-secret secret_code_goes_here
set vpn ipsec site-to-site peer "Neustar IP" connection-
type initiate
set vpn ipsec site-to-site peer "Neustar IP" default-esp-
group trs
set vpn ipsec site-to-site peer "Neustar IP" ike-group trs
set vpn ipsec site-to-site peer "Neustar IP" local-address
"VyOS private address"
set vpn ipsec site-to-site peer "Neustar IP" tunnel 1 local
prefix "your Encryption-Domain"/32
```

```

set vpn ipsec site-to-site peer "Neustar IP" tunnel 1
protocol
set vpn ipsec site-to-site peer "Neustar IP" tunnel 1
remote prefix "Neustar Encryption-Domain"/32
set vpn ipsec site-to-site peer "Neustar IP" tunnel 1
allow-public-networks enable
set interfaces dummy dum0 address "your Encryption-
Domain"/32
set protocols static route "Neustar Encryption-Domain"/32
next-hop "your Encryption-Domain"
set nat source rule 100 outbound-interface dum0
set nat source rule 100 source address "your private
subnet"
set nat source rule 100 translation address masquerade
set nat source rule 100 translation address "your
Encryption-Domain"

```

In the AWS Cloud, a route must be created in the VPC route table so that traffic to the iTRS database is routed to the VyOS instance (which in turn then routes it out over the VPN tunnel). By default instances drop packets when source and destination information does not match that of an instance. This behavior can be disabled from the AWS console by selecting an instance and going to network options.

3.9 SIP Trunking Provider

To support calls from and to the PSTN, ACE Connect Lite subscribes to a SIP Trunking service to provision a SIP trunk and telephone numbers used for this proof of concept.

Setting up a PSTN SIP Trunk in Asterisk requires modifications to two files (which are found in `/etc/asterisk`): `extensions.conf` and `sip.conf`. The configuration in both files will reference back to each other, which will be demonstrated in the following examples. The Asterisk documentation site provides detailed information regarding the code snippets, and there are other examples available. The four steps for completing this task follow.

1. Set up the dial plan

The first step is to define the dial plan for the SIP trunk, which is configured in the `extensions.conf` file. This will tell Asterisk what incoming traffic to forward to the trunk. You can define specific phone numbers to route, as well as number schemes for unknown numbers (using the wildcard "X"), as shown in the following example:

```

[from-internal]

exten => _<10_digit_number>,1,Dial(SIP/<10_digit_number>)
        same => n,DumpChan()
        same => n,HangUp()

exten => agents,1,Dial(SIP/extension)
same => n,DumpChan()

```

```

same => n, HangUp()

exten => _XXXXXXXXXX,1,NoOp("Caller ID:
${CALLERID(number)}")
    same =>
n,Set(sipuri=${ENUMLOOKUP(+1${EXTEN},sip,,1,itrs.us)})
    same => n,NoOp("number: ${EXTEN} has sipuri:
${sipuri}")
    ;If a blank URI this is a hearing user, direct to a
CA => n,GotoIf("${sipuri}" = "" ]?from-internal,agents,1)
    same => n,GotoIf("${sipuri}" = "" ]?from-
internal,agents,1)
    ;else do a direct dial
    same => n,Dial(SIP/${sipuri:1},30)
    same => n,DumpChan()
    same => n, HangUp()

```

For the Dial() functions, the extensions/numbers defined within the function call must be defined in sip.conf, as we will show below.

2. Set up profile

Once the dial plan has been configured, you can begin to define the profile for the SIP trunk within sip.conf. This will typically look like this:

```

[h264_phone](!)
    type=friend
    transport=tcp,udp
    port=5060
    secret=<passphrase>
    host=dynamic
    directmedia=no
    disallow=all
    allow=ulaw
    allow=h264
    registertrying=yes
    context=from-internal
    callcounter = yes
    qualify=yes
    busydetect=yes
    videosupport=yes
    dtmfmode=rfc2833
    accept_outofcall_message=yes
    outofcall_message_context=IM
    auth_message_requests=yes

```

Although the Asterisk documentation provides definitions for all of the different attributes, the following definitions are notable:

- a. “disallow/allow”: By default, it is a good idea to start with “disallow=all” first, then use allow” to explicitly allow certain formats.
- b. secret: The passphrase used to login to an extension/number extended from this generic profile
- c. context: The context to be used with this profile, defined in extensions.conf

3. Add extensions/numbers

Once the profile has been defined, extensions and numbers can be associated with the profile in sip.conf, which will be referred to back in the extensions.conf file to help route calls to their proper destination.

```
[6111](h264_phone)
regexten=6111
callerid=<6111>

[<10_digit_number>](h264_phone)
regexten=<10_digit_number>
callerid=<10_digit_number>
```

4. External PSTN SIP Trunk Profile

If you need to configure an external PSTN SIP trunk to work with Asterisk, you can define a simplified profile and extensions as follows:

```
[providerX-trunk](!)
    type=peer
    context=from-ProviderX
    dtmfmode=rfc4733
    canreininvite=no
    insecure=port,invite

[ProviderX0](ProviderXtrunk)
host=<somedomain>.pstn.ProviderX.com

[ProviderX1](ProviderXtrunk)
host=<some_ip_address>
```

Acronyms

ACE	Accessible Communications for Everyone
ALI	Automatic Location Identification
AMA	Automatic Message Accounting
AMI	Asterisk Management Interface
API	Application Programming Interface
APP	Access Point Platform
CA	Communications Assistant
CAMH	CMS Alliance to Modernize Healthcare
COE	Center of Excellence
CSS	Cascading Style Sheets
CSV	Comma Separated Values
DNS	Domain Name Service
EIP	Elastic Internet Protocol
ENUM	E.164 Number to URI Mapping
FAR	Federal Acquisition Regulation
FCC	Federal Communications Commission
FFRDC	Federally Funded Research and Development Center
GUI	Graphical User Interface
HHS	Department of Health and Human Services
HTML	Hypertext Markup Language
iTRS	Internet Telecommunications Relay Service
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
NAT	Network Address Translation
ODBC	Open Database Connectivity
OS	Operating System
PBX	Private Branch Exchange
PSTN	Public Switched Telephone Network
RESTful	Representation State Transfer-based web services

RFC	Request for Comment
RUE	Relay User Equipment
SIP	Session Initiation Protocol
STUN	Session Traversal Utilities for NAT
TTL	Time to Live
TURN	Traversal Using Relay NAT
UI	User Interface
URD	User Registration Database
URI	Uniform Resource Identifier
URL	Universal Resource Locator
VRS	Video Relay Service
VyOS	Vyatta Operating System (open source)