

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 Direct Platform Release Documentation

Final

Version 1.0

November 4, 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 16-4058.

© 2016, The MITRE Corporation. All Rights Reserved.

Record of Changes

Version	Date	Author / Owner	Description of Change
1.0	November 4, 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 (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 Centers for Medicare & Medicaid Services (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.

At the FCC's request, CAMH developed a Direct Video Calling (DVC) Auto-Routing Proof of Concept (POC) in support of the FCC's Accessible Communications for Everyone (ACE)¹ program. This DVC auto-routing platform enables direct calling from deaf and hard-of-hearing individuals to an American Sign Language (ASL)-trained agent within the organization's call center. The agent handles the call using a video-capable phone with real-time video connection. To demonstrate the capabilities of DVC, the FCC and CAMH have further advanced the original auto-routing POC into a call center platform for two to ten customer service representatives. This new DVC platform is called ACE Direct.

Implementing the Direct Video Calling platform provides a number of benefits:

- **Improved Communications** – Direct video calling improves privacy and decreases misrepresentation, which improves efficiency, effectiveness, and productivity.
- **Career Opportunities** – Employing native ASL users to handle customer service video calls expands hiring opportunities. President Obama's Executive Order 13548 (July 2010) directed federal agencies to increase employment opportunities for people with disabilities.
- **Simple Implementation** – The technology to implement a direct video communications system is readily obtainable, affordable, and easy to set up.
- **Secure Communications** – With proper configuration, agencies can use high-speed broadband and their own internal networks without compromising security or contending with barriers created by firewalls.
- **Maintain ADA Compliance** – Direct Video Communication ensures compliance with the Americans with Disabilities Act mandates.

¹ <https://www.fcc.gov/ace>

- **Cost Savings** – Replacing three-way interpreted calls with two-way direct communication saves money by minimizing the need for repeat calls due to miscommunication and/or misunderstanding.

As part of this 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 Direct platform. Detailed configuration and source code files are available for download and reproduction to improve solutions to support the community. The public can for download or reproduce these files at the [Github.com/MITRE/ACE](https://github.com/MITRE/ACE) website

Table of Contents

1. Introduction	1
1.1 Background	1
1.2 Purpose and Scope	2
2. Overview of Direct Video Calling and ACE Direct	3
2.1 DVC Is an Alternative to Traditional Relay Calls	3
2.2 Open-Source Development to Promote Community Involvement	3
2.3 Conceptual System Overview	4
2.4 Highlighted User Stories	6
2.5 CSR Desktop User Guide.....	7
2.5.1 Logging into ACE Direct.....	7
2.5.2 CSR Desktop Layout	8
2.5.3 Side Panels	9
2.5.4 Video and Real-Time Text Communications	12
2.5.5 CSR Desktop Portal Header	14
2.5.6 VRS and Ticket Information.....	15
2.6 Consumer Portal	16
2.6.1 Consumer Complaint Form User Guide	16
2.6.2 Submitting a Complaint	17
2.6.3 Contacting the CSR	17
2.6.4 Complaint Ticket	18
2.6.5 Video Chat	18
2.6.6 Real-Time Text Chat	19
2.7 Management Portal	20
2.7.1 Management Dashboard	20
2.7.2 Call Detail Record Dashboard	22
3. Installation and Configuration	25
3.1 AWS Deployment	25
3.1.1 Creating an ACR / Asterisk Instance	25
3.1.2 Creating a Node.js Instance	25
3.1.3 Deploying an EC2 Instance	26
3.2 Asterisk Installation and Configuration Script.....	27
3.2.1 Manual Asterisk Server Installation	28
3.2.2 How It Works.....	29
3.2.3 Secure Calling.....	32
3.2.4 Sample Configurations	32
3.3 Node.js.....	35
3.3.1 Installing	35
3.3.2 Starting.....	36
3.3.3 Status.....	36
3.3.4 Restart	37

3.4	Management Portal	37
3.4.1	Installation	37
3.4.2	Configuration	37
3.4.3	Run	38
3.4.4	Log Files	38
3.4.5	Management Dashboard	38
3.4.6	Call Detail Record Dashboard	42
3.5	VRS User Database (Provider)	45
3.5.1	MySQL Database Server Installation	45
3.5.2	Portal Files Installation	47
3.6	STUN	47
3.7	iTRS ENUM Database	47
3.8	VyOS Router for Secure Socket Layer (SSL) Tunnel	48
3.9	COTS CRM	50
3.10	Enterprise Service Bus	50
3.10.1	Background	51
3.10.2	Installation Overview	51
3.10.3	Testing the Broker Application	57
Acronyms		59

List of Figures

Figure 1.	Notional Diagram for ACE Direct Platform	4
Figure 2.	CSR Desktop Login Screenshot	8
Figure 3.	Screenshot of CSR Desktop	9
Figure 4.	Screenshots of the CSR Ready, Away, and In-Call Status	10
Figure 5.	CSR Scripts Screenshot	10
Figure 6.	CSR Mailbox Screenshot	11
Figure 7.	CSR Information Screenshot	12
Figure 8.	User Chat Box Screenshot	13
Figure 9.	Script Box	13
Figure 10.	Screenshot of Call Duration and Assistance Button	14
Figure 11.	Screenshot of CSR Profile	14
Figure 12.	Screenshot of VRS Information	15
Figure 13.	Screenshot of Ticket Information	16
Figure 14.	Screenshot of Consumer Portal	17

Figure 15. Screenshot of VRS Number Entry	17
Figure 16. Screenshot of Consumer VRS Information	18
Figure 17. Screenshot of Complaint Ticket	18
Figure 18. Screenshot of Video Chat Window	19
Figure 19. Screenshot of Real-Time Text Chat	19
Figure 20. Screenshot of Management Dashboard	20
Figure 21. Screenshot of Resource Status	22
Figure 22. Screenshot of Call Detail Record	22
Figure 23. Shell Scripts	36
Figure 24. <i>pm2 list</i> Command	37
Figure 25. VyOS Router for SSL Tunnel	49

List of Tables

Table 1. ACE Direct Components	5
Table 2. ACE Direct Highlighted User Stories	7
Table 3. Call Detail Record Column Definition	23
Table 4. Asterisk Endpoint Extensions	30
Table 5. AMI Actions and Descriptions	39
Table 6. Asterisk AMI Event	40

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.

In continuing pursuit of the Commission's goal to advance functionally equivalent telecommunications, CAMH developed ACE Direct, an open-source call center platform supporting Direct Video Calling (DVC). The platform, as released, is designed to support a call center with two to ten customer service representatives (CSR). Implementing ACE Direct in a corporate production environment requires a level of customization to adhere to corporate practices and policies related to security, system configurations, cloud services, and availability.

The FCC encourages government agencies and private businesses to make DVC part of their call center strategy because of the significant gains for functionally equivalent telecommunications:

- **Improved Communications** – Direct video calling improves privacy and decreases misrepresentation, which improves efficiency, effectiveness, and productivity.
- **Career Opportunities** – Employing native ASL users to handle customer service video calls expands hiring opportunities. President Obama's Executive Order 13548 (July 2010) directed federal agencies to increase employment opportunities for people with disabilities.
- **Simple Implementation** – The technology to implement a direct video communications system is readily obtainable, affordable, and easy to set up.
- **Secure Communications** – With proper configuration, agencies can use high-speed broadband and their own internal networks without compromising security or contending with barriers created by firewalls.
- **Maintain ADA Compliance** – Direct Video Communication ensures compliance with the Americans with Disabilities Act mandates.
- **Cost Savings** – Replacing three-way interpreted calls with two-way direct communication saves money by minimizing the need for repeat calls due to miscommunication and/or misunderstanding.

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

1.2 Purpose and Scope

This document presents an overview of the ACE Direct platform, including the ACE Direct architecture, user stories, and guidance for installation and configuration. This document describes how to integrate DVC within an agency's current call center workflow to provide an independent, on-demand service.

In addition to this overview document, detailed configuration and source code files are available to the public at <http://mitre.github.io/> for download and reproduction of the platform to support and promote future platform enhancements for the deaf and hard-of-hearing community.

2. Overview of Direct Video Calling and ACE Direct

Deaf, hard-of-hearing, deaf-blind, or speech-disabled people use Telecommunications Relay Services to communicate with hearing people over the phone. Since the early 2000s, video relay service (VRS) calls have been the primary way that American Sign Language (ASL) users access telecommunications. VRS involves the use of third-party communication assistants (CA) as sign language interpreters to place telephone calls. The interpreter translates between ASL and spoken English for the non-signing party. People who communicate in ASL use VRS to place telephone calls to customer assistance divisions of government agencies and businesses in the United States every day, but there are other solutions.

2.1 DVC Is an Alternative to Traditional Relay Calls

The FCC's sponsorship of the Accessible Communications for Everyone program includes creating a direct video calling platform. The ASL Consumer Support Line²—the first of its kind in the federal government—allows ASL users to make video calls directly to an agent fluent in ASL.

English is not the first language of many TRS deaf, hard-of-hearing, deaf-blind, and speech-disabled TRS users. One-to-one communication in ASL is most often preferred.

When comparing calls made to the FCC ASL Consumer Support Line with calls placed through VRS, the FCC found the average call length *decreased 42 percent* and incoming calls *increased 533 percent*. Most impressive is that the FCC achieved these results without adding more staff to handle the increased call volume.

2.2 Open-Source Development to Promote Community Involvement

ACE Direct is open-source technology that is one option for implementing direct video calling. The proof of concept is based on open source as a development model. It promotes universal access via a free license to a product's design/blueprint and universal redistribution of that design/blueprint, including subsequent improvements to it. The open-source model is based on a decentralized model of production. A main principle of open source software development is peer production, with products such as source code, "blueprints", and documentation available to the public at no cost.

The FCC encourages government agencies, educational institutions, and others seeking to enhance the lives of citizens who are deaf, hard of hearing, deaf-blind, and/or have speech disabilities to adopt and improve on the existing code base to provide additional features, improve the workflow, and introduce new technologies to the ACE Direct platform.

² <https://www.fcc.gov/document/fcc-adds-american-sign-language-consumer-support-line-videophone>

2.3 Conceptual System Overview

CAMH developed the open-source based ACE Direct platform for implementation in the Amazon Web Services (AWS) cloud environment. Figure 1 presents a notional view of the ACE Direct implementation.

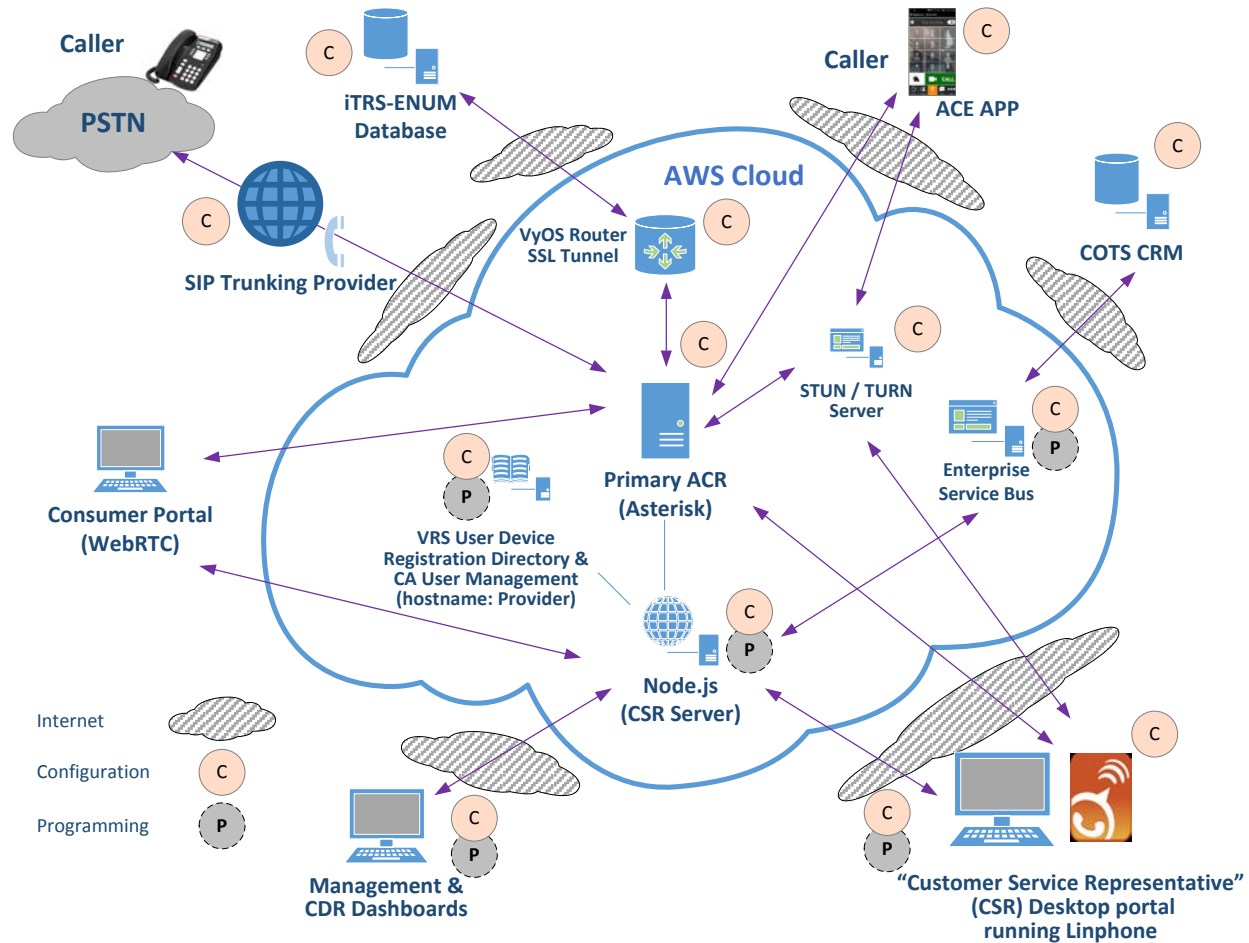


Figure 1. Notional Diagram for ACE Direct Platform

As Figure 1 shows, the ACE Direct implementation consists of various components. Some components require only configurations (noted as “C”) and other components require both configurations and programming (noted as “P”). Table 1 presents an overview of these components. Section 3 provides detailed installation information and configurations.

Table 1. ACE Direct 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 Public Switched Telephone Network (PSTN) and video calls. In the ACE Direct platform, a CSR uses a softphone (Linphone) registered to Asterisk to accept inbound calls (PSTN or Video).	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 CSR desktop portal and other management-related portals, including the management dashboard and call detail record (CDR) dashboard. The Node.js server supports the Real-Time Text (RTT) between the CSR Desktop portal and the consumer portal. It also provides services for VRS lookup to verify that the phone number is a valid number in the VRS database.	Subsection 3.3
Consumer Portal	The Consumer Portal introduces the WebRTC technology to the ACE Direct platform. It combines form submission with video and RTT to a CSR. CAMH elected to use a complaint process to demonstrate this technology.	Subsection 2.6
Management and Call Detail Record (CDR) Dashboards	The ACE Direct Management dashboard provides a number of Key Performance Indicators (KPI) the call center manager can monitor in real time. The CDR dashboard provides the view and export functions of the Asterisk CDRs stored in its MySQL database.	Subsection 2.7.2
VRS User Device Registration and CSR User Management (hostname: Provider)	The “Provider” server emulates the services for both the CSR 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.	Subsection 2.5.6
Customer Service Representative (CSR) Desktop	The CSR desktop provides user interface to the CSR for login and conducting VRS services to the ACE Direct users. The CSR desktop is one of the two windows on the CSR’s workstation monitor.	Subsection 2.5
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 3.6

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.	Subsection 3.7
VyOS (Vyatta Operating System) 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 Elastic IP (EIP) on it that is the encryption domain for the tunnel.	Subsection 3.8
Commercial Off-the-Shelf (COTS) Customer Relationship Management (CRM)	To demonstrate integration with a CRM service, ACE Direct connects to the Zendesk RESTful application programming interface (API) via the Enterprise Service Bus. ACE Direct sends Java Script Object Notation (JSON)-based messages to the RESTful Zendesk API to manage and query customer records.	Subsection 3.9
Enterprise Service Bus (ESB)	The enterprise service bus provides a generic method to update legacy database systems as well the diverse number of databases and unstructured data repositories on the market and in use today. ACE Direct ESB integrates with a COTS CRM service (e.g., Zendesk) as a ticketing system for the CSR to document service cases.	Subsection 3.10
SIP Softphones for CSR and ASL users	ACE Direct uses SIP video softphones, including ACE APP and Linphone, as end-user devices that are registered to the Asterisk Server.	N/A

2.4 Highlighted User Stories

CAMH built ACE Direct to encompass the core functions of a traditional hearing-based call center. ACE Direct focuses on the responsibilities of CSRs and their managers. The FCC and CAMH partnered with federal agencies to derive typical requests for services and call center workflows. Table 2 presents a summary of ACE Direct user stories, which demonstrate these functions and capabilities.

Table 2. ACE Direct Highlighted User Stories

Story	Description
Direct Video Call with an ASL-fluent CSR	As an ASL user, I want to speak with another ASL user when I contact a call center.
CRM Integration	As a CSR, I can view, update, and enter new information regarding contact with the caller from the corporate CRM system.
Call Script Integration	As a CSR, I can view corporate call scripts based on the needs of the caller.
Call-handling capabilities	As an ACE Direct CSR, I can perform “Call on Hold” and “Call Transfer” as needed.
Consumer Portal	For this story, we have elected to use a complaint process to elaborate the story. As a consumer, I wish to file a complaint through a web portal on my website. I will also have the option of conversing with the CSR through video and Real-Time Text (RTT). See Subsection 2.6 for details.
Video Mail	As a CSR, I can retrieve a video mail.
Management Dashboard	As the ACE Direct manager/operator, I can access near real-time information on the dashboard.
Call Detail Record (CDR)	As the ACE Direct administrator, I can access the Call Detail Record through a web portal and export CDRs as needed for audit purpose. See Subsection 2.7.2 for details.
Web-based Application	As a CSR, I have the ability to work remotely from the main call center if necessary.
Multi-CSR Login with Status	As the ACE Direct CSR, I can log in using the CSR Desktop along with other CSRs and I can change my status between “Ready” and “Away”.

2.5 CSR Desktop User Guide

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

This guide provides a walkthrough of the portal, highlighting each of the available functionalities.

2.5.1 Logging into ACE Direct

Upon navigating to the portal host URL, a login screen appears. To access the portal, the agent must enter their username and password.

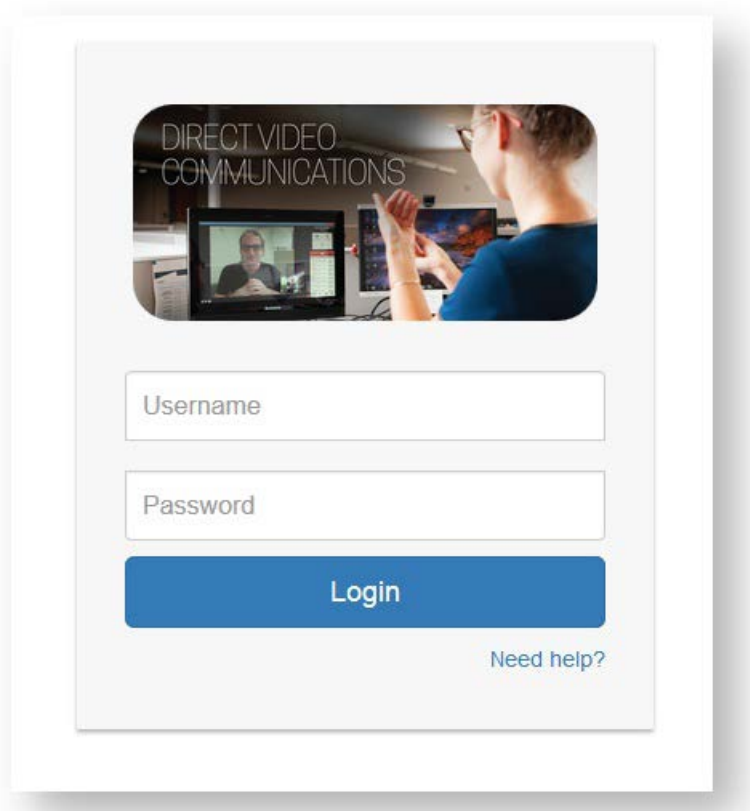


Figure 2. CSR Desktop Login Screenshot

2.5.2 CSR Desktop Layout

Figure 3 presents a screenshot of the CSR Desktop, which consists of the following elements:

- Side panels (left and right) to provide navigation, information, and settings to the CSR
- A user chat area for real-time text chats
- A header area that displays call duration information and an assistance button
- Profile information displaying the CSR's name and picture and the ability to sign out of the system
- VRS caller information such as first name, last name, etc.
- Any prior CRM ticket information by the VRS caller

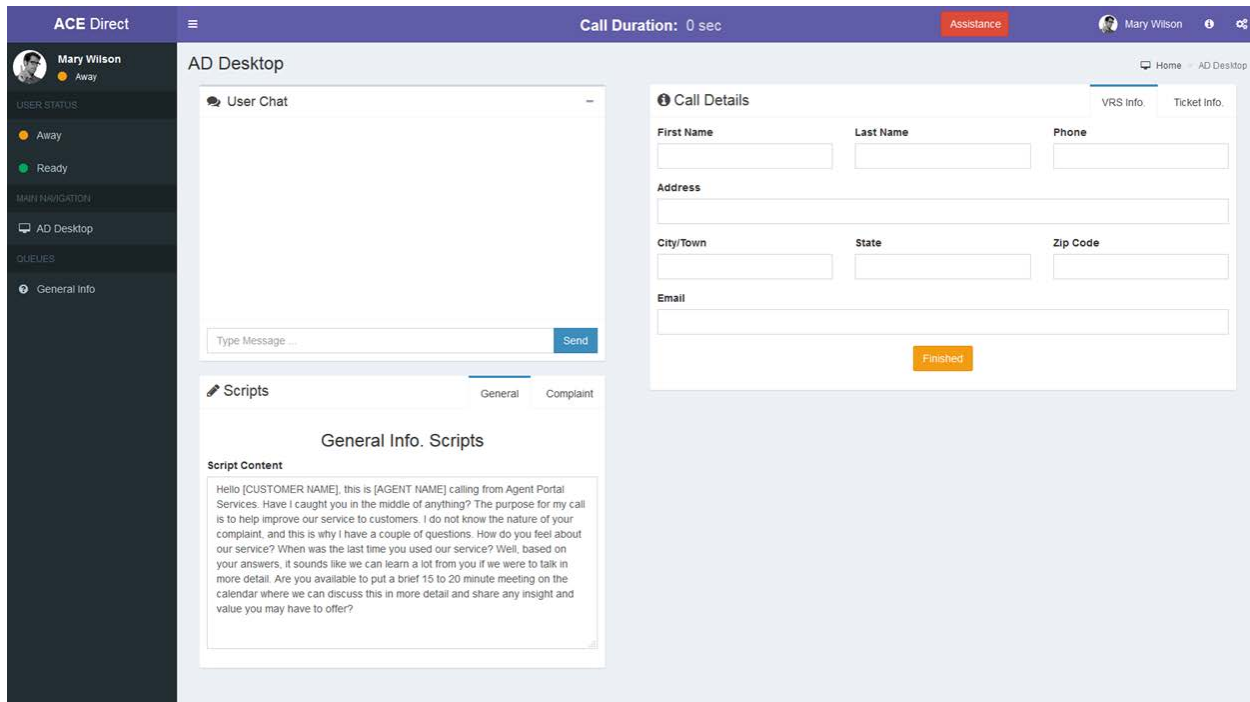


Figure 3. Screenshot of CSR Desktop

2.5.3 Side Panels

The ACE Direct CSR Desktop portal has two side panels that provide navigation, information, and settings to the CSR.

2.5.3.1 Left Side Panel (Main Navigation)

As shown in Figure 4, the left-side panel of the portal provides the CSR with both User Status and the Main Navigation. Here CSRs can select their status as “Ready” or “Away”. When a CSR first signs into the portal, the status defaults to “Away”. When the CSR is ready to receive calls, the CSR selects the “Ready” status. Once the CSR enters a call, the status changes to “In Call”. After the CSR leaves the call and clicks on the “Finished” button, the CSR’s status returns to the “Ready” state. The Agent status will also be reflected on the Management Dashboard.

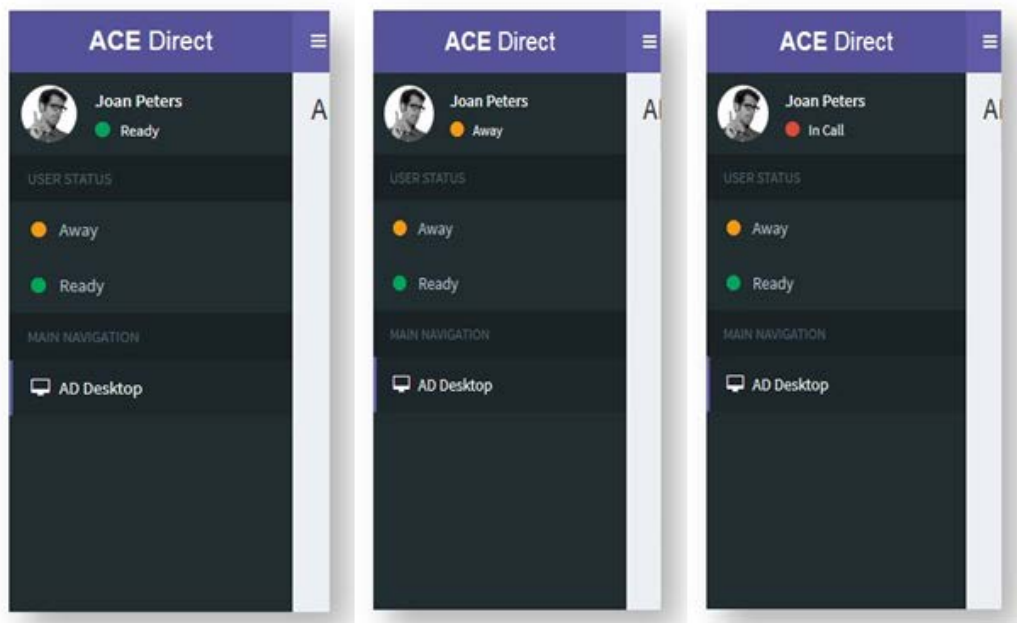


Figure 4. Screenshots of the CSR Ready, Away, and In-Call Status

2.5.3.2 Right-side Panel (Scripts, Mailbox, and Settings)

The right-side panel as shown in Figure 5 is accessible by clicking on the gears icon in the top right corner of the portal. This section can be collapsed to give the CSR more space for the main content area. Upon opening the right-side panel, the CSR will be presented with three tabbed content areas (Scripts, Mailbox, and Settings).

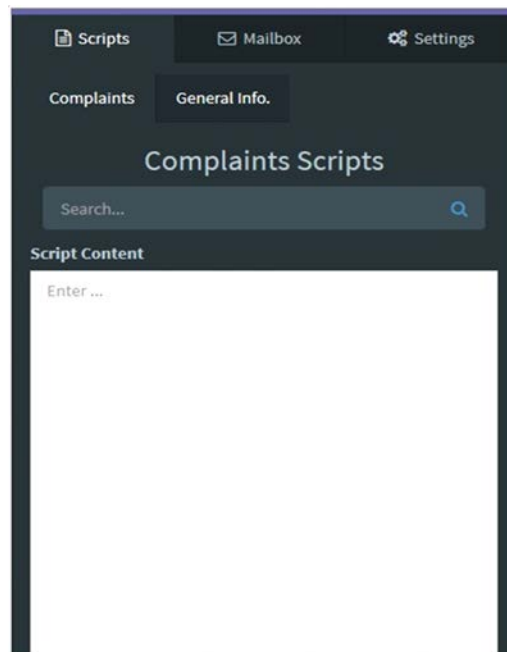


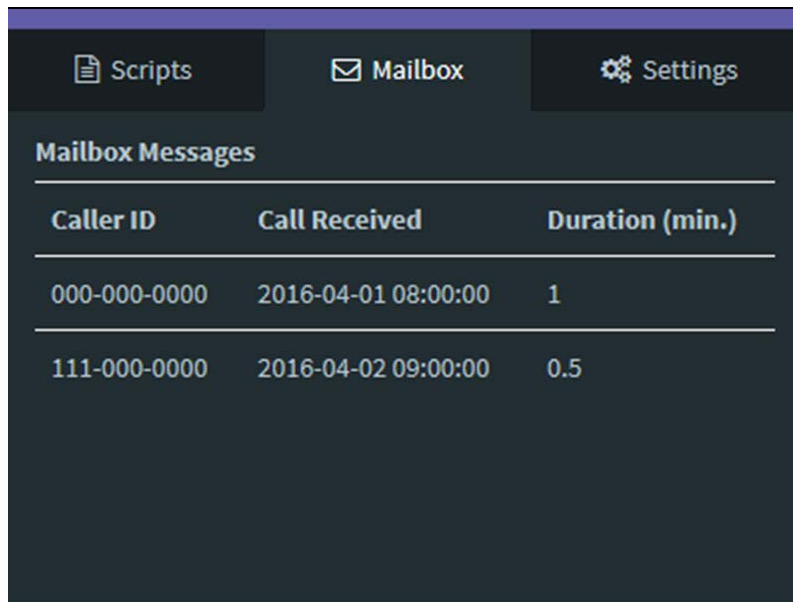
Figure 5. CSR Scripts Screenshot

2.5.3.2.1 Scripts

The scripts section provides the CSR with two sub-tabbed areas where the CSR can select the appropriate script to apply to the conversation. The CSR can also search for another script depending on the direction of the conversation. (**Note:** The search bar is a placeholder, and the search function is not available at this time.)

2.5.3.2.2 Mailbox

The Mailbox tab, as shown in Figure 6, displays a list of messages received while the CSR was not available to take the call. This list provides the CSR with the phone number, time and date of the call, and the length of the message.



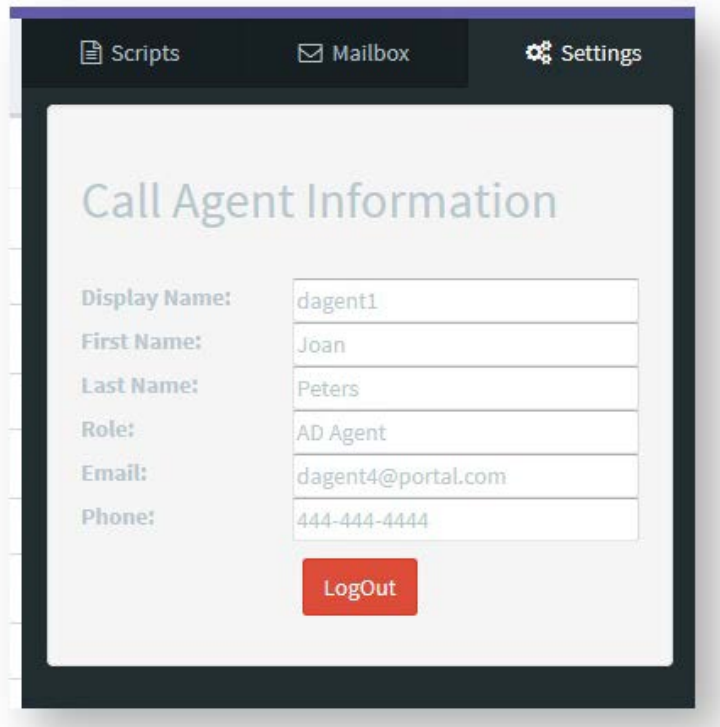
The screenshot shows a web interface with three tabs: 'Scripts', 'Mailbox' (selected), and 'Settings'. Below the tabs is a section titled 'Mailbox Messages' containing a table with three columns: 'Caller ID', 'Call Received', and 'Duration (min.)'. The table lists two messages.

Mailbox Messages		
Caller ID	Call Received	Duration (min.)
000-000-0000	2016-04-01 08:00:00	1
111-000-0000	2016-04-02 09:00:00	0.5

Figure 6. CSR Mailbox Screenshot

2.5.3.2.3 Settings

Figure 7 shows the Settings tab that provides the CSR with the ability to log out of Asterisk. This section is used primarily for testing and ideally would be hidden from the primary user.



Call Agent Information	
Display Name:	dagent1
First Name:	Joan
Last Name:	Peters
Role:	AD Agent
Email:	dagent4@portal.com
Phone:	444-444-4444
LogOut	

Figure 7. CSR Information Screenshot

2.5.4 Video and Real-Time Text Communications

The ACE Direct platform supports multiple forms of communications with the CSR. Two methods are through video and RTT. This subsection describes how each of these methods operates in the platform.

2.5.4.1 Video Chat

Video Chat communications between the CSR and callers occurs through a softphone (i.e., Linphone, Microsip, etc.). After the CSR logs into ACE Direct, the CSR would log into their softphone and register their extension in Asterisk.

2.5.4.2 RTT Chat

The User Chat box, as shown in Figure 8, provides the CSR a secondary method of communication with the caller. As the CSR types a message to the caller in the input field, the message will appear in real time to the user. The chat history will remain visible to the CSR until the CSR closes the ticket.

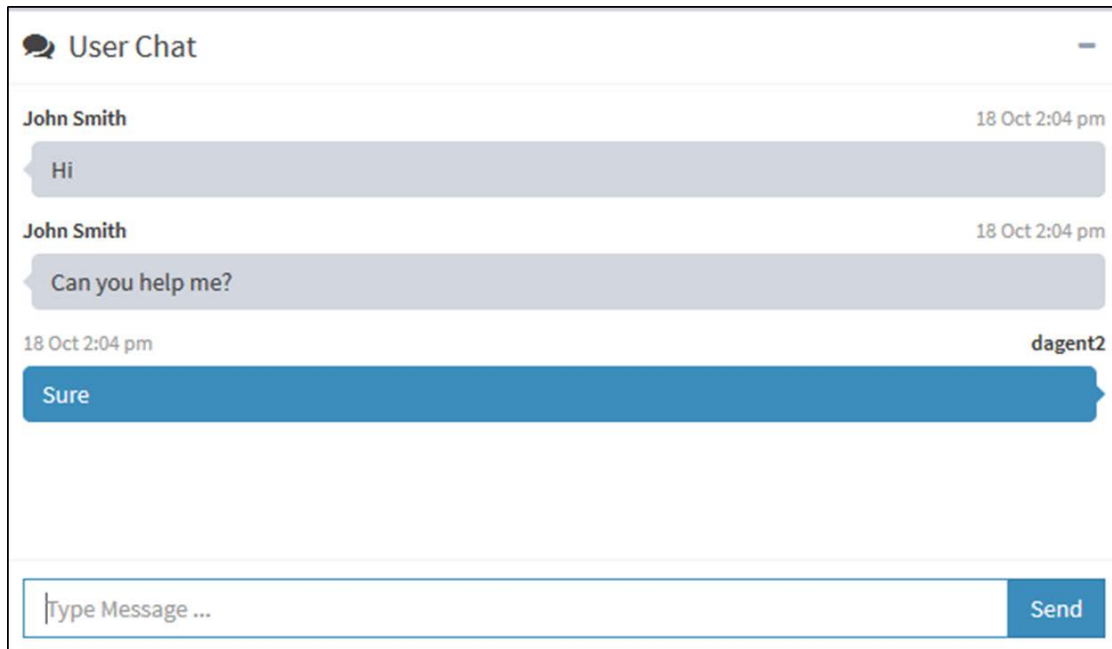


Figure 8. User Chat Box Screenshot

2.5.4.3 Scripts

The Scripts box, as shown in Figure 9, provides the CSR with two sub-tabbed areas where the CSR can select the appropriate script to apply to the conversation.

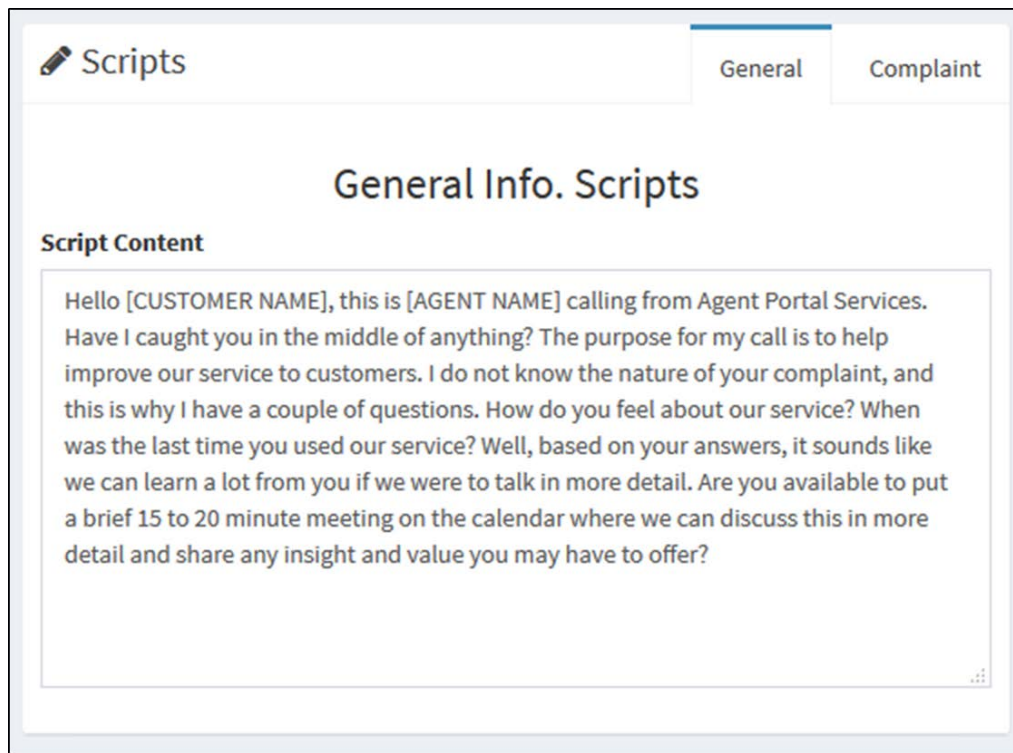


Figure 9. Script Box

2.5.5 CSR Desktop Portal Header

The CSR Desktop portal header provides the CSR with the call duration information, an assistance button, and profile information about the CSR along with the ability to sign out of the system.

2.5.5.1 Call Duration and Assistance Button

The Call Duration shows a running clock of the call length once the CSR accepts the incoming consumer call. As Figure 10 shows, the Assistance button allows the CSR to ask for help during a call. Once the Assistance button is clicked, the CSR's name will change color and begin to flash on the Management Dashboard to indicate the CSR needs assistance.

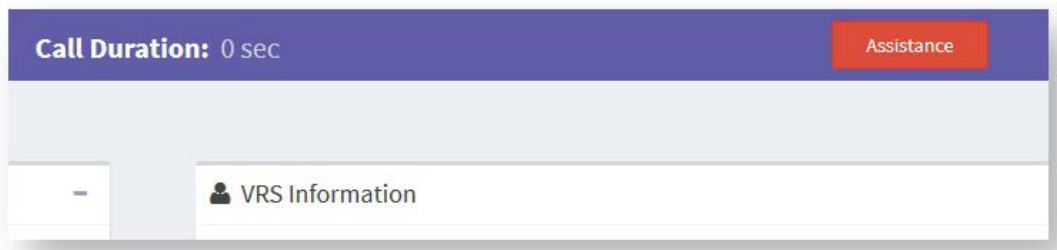


Figure 10. Screenshot of Call Duration and Assistance Button

2.5.5.2 CSR Profile

On log in, the CSR's name and picture will appear in the top right corner of the CSR Desktop portal head as shown in Figure 11. The CSR clicks on the Sign out button to log out of the ACE Direct portal.

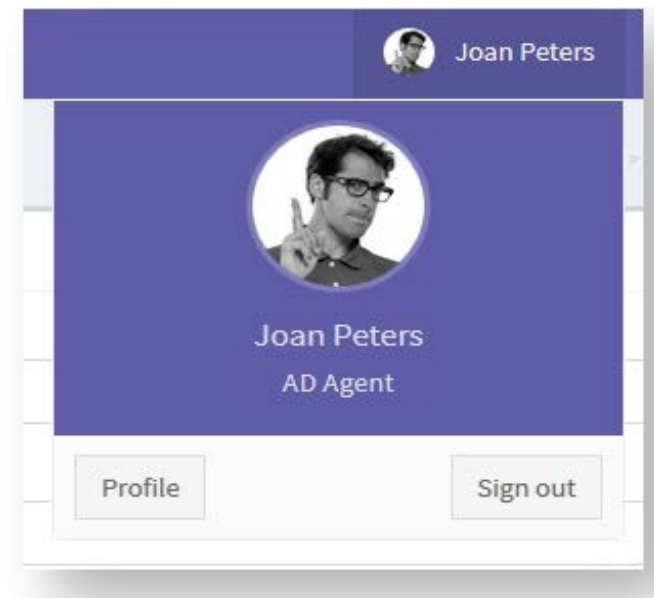
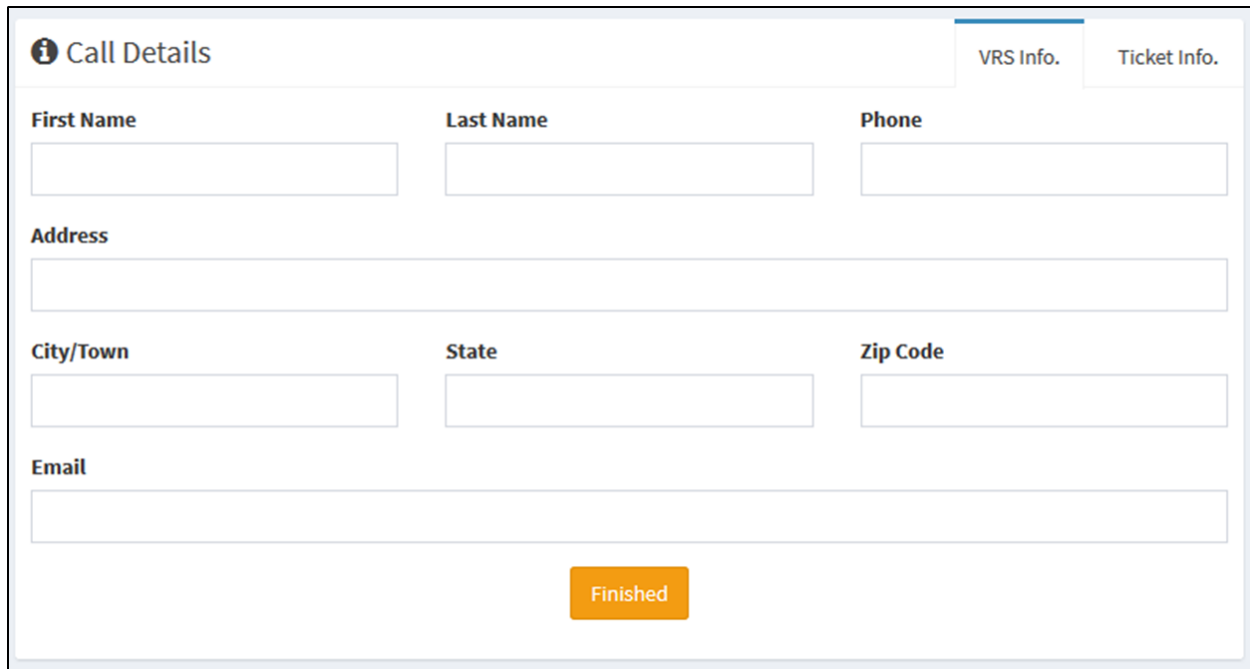


Figure 11. Screenshot of CSR Profile

2.5.6 VRS and Ticket Information

2.5.6.1 VRS Information

The VRS Information section displays the information about the caller that is currently on file. Figure 12 shows that, after the call has ended, the CSR must click on the Finished button in the VRS Information box to return to the queue and receive new calls.



The screenshot shows a web interface titled "Call Details" with an information icon. It features two tabs: "VRS Info." (which is active) and "Ticket Info.". The "VRS Info." tab contains several input fields: "First Name", "Last Name", and "Phone" are in a row; "Address" is a single-line field; "City/Town", "State", and "Zip Code" are in a row; and "Email" is a single-line field. At the bottom center of the form is an orange button labeled "Finished".

Figure 12. Screenshot of VRS Information

2.5.6.2 Ticket Information

Figure 13 shows the Ticket Information section that provides the CSR with the ticket information submitted by the user.

The screenshot shows a web form titled "Call Details" with a sub-tab "Ticket Info." The form contains several input fields: "Assignee", "Requester", "Last Updated", "Subject", and "Ticket ID". Below these are two large text areas for "Problem Description" and "Resolution". A "Save" button is located at the bottom right of the form.

Call Details		
		Ticket Info.
Assignee	Requester	Last Updated
<input type="text"/>	<input type="text"/>	<input type="text"/>
Subject	Ticket ID	
<input type="text"/>	<input type="text"/>	
Problem Description		
<input type="text"/>		
Resolution		
<input type="text"/>		
<input type="button" value="Save"/>		

Figure 13. Screenshot of Ticket Information

2.6 Consumer Portal

The consumer portal is designed as a web site to give consumers an option for submitting information without having to wait on the phone for a CSR. In this option, the consumer submits information through a web form. To illustrate this option, the web forms in Figures 14–19 are designed as a consumer complaint form.

2.6.1 Consumer Complaint Form User Guide

This Consumer Complaint User Guide provides a walkthrough of the consumer portal, as shown in Figure 14, highlighting each of the available functionalities.

The screenshot shows the 'Consumer Complaint Form' interface. At the top, there's a 'VRS Information' section with fields for 'First Name' (John), 'Last Name' (Smith), 'Phone' ((727) 551-4865), and 'Email' (jsmith@gmail.com). Below this is a 'Complaint Ticket' section with a 'Subject' field (Max of 80 characters) and a 'Description of Complaint' text area (You have 500 characters left). To the right of the 'Description of Complaint' is a 'Video' window showing a woman on a laptop with 'DIRECT VIDEO CALLING' and 'FC' logos. Further right is an 'Agent Chat' section with a 'Type Message ...' input field (You have 140 characters left) and a 'Send' button. At the bottom of the 'Description of Complaint' section are 'Submit' and 'Call' buttons.

Figure 14. Screenshot of Consumer Portal

2.6.2 Submitting a Complaint

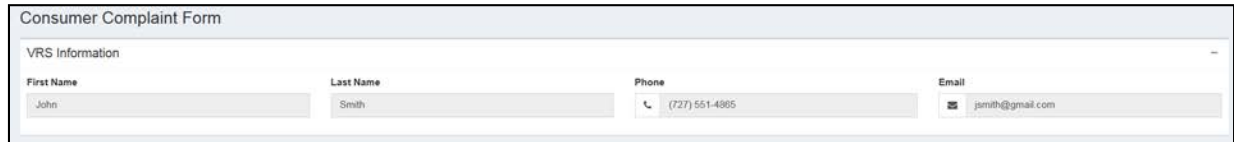
Upon navigating to the portal host URL, the consumer will be prompted to enter their VRS number as shown in Figure 15.

The screenshot shows the 'Consumer Help Center - Access for People with Disabilities' header. Below the header, there's a prompt 'Please Enter your VRS Number:' followed by a text input field with a phone icon and the placeholder text 'VRS Number'. Below the input field is a blue 'Continue' button.

Figure 15. Screenshot of VRS Number Entry

2.6.3 Contacting the CSR

Upon entering the Complaint Form page, the consumer will be presented their VRS information on file as shown in Figure 16.

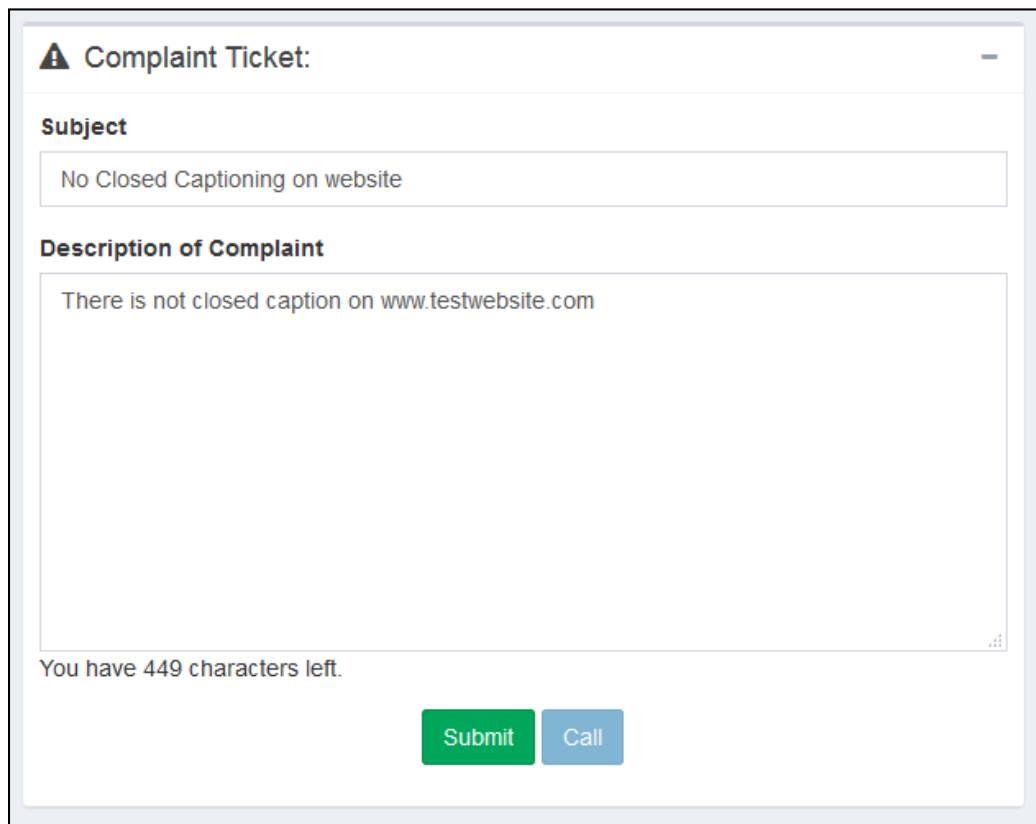


The screenshot shows a web form titled "Consumer Complaint Form". Under the "VRS Information" section, there are four input fields: "First Name" (containing "John"), "Last Name" (containing "Smith"), "Phone" (containing "(727) 551-4365"), and "Email" (containing "jsmith@gmail.com"). Each field has a small icon to its right: a person for First Name, a person for Last Name, a telephone for Phone, and an envelope for Email.

Figure 16. Screenshot of Consumer VRS Information

2.6.4 Complaint Ticket

In the complaint ticket box on the left side of the screen, as shown in Figure 17, the consumer enters a subject and description of the complaint to provide information to the agent. After writing a subject and description, the consumer clicks “Submit” to file the complaint. The complaint ticket is generated, which enables the “Call” button. This allows the consumer to communicate with an agent via video and chat.



The screenshot shows a "Complaint Ticket" form. It has a title bar with a warning icon and the text "Complaint Ticket:". Below the title bar, there are two sections: "Subject" and "Description of Complaint". The "Subject" section has a text input field containing "No Closed Captioning on website". The "Description of Complaint" section has a larger text area containing "There is not closed caption on www.testwebsite.com". Below the text area, it says "You have 449 characters left." At the bottom of the form, there are two buttons: a green "Submit" button and a blue "Call" button.

Figure 17. Screenshot of Complaint Ticket

2.6.5 Video Chat

After submitting and calling through the complaint ticket, the consumer will be connected with an agent. The primary form of communication is video. As Figure 18 shows, the agent’s video will be displayed in the video box in the center of the screen.



Figure 18. Screenshot of Video Chat Window

2.6.6 Real-Time Text Chat

As Figure 19 shows, the Agent Chat box on the right side of the screen provides the consumer a secondary method of communication with the agent. Notifications appear while the consumer is typing a message to the agent and vice versa. The messages will show up in real time, and the chat history will remain visible until the agent closes the ticket.

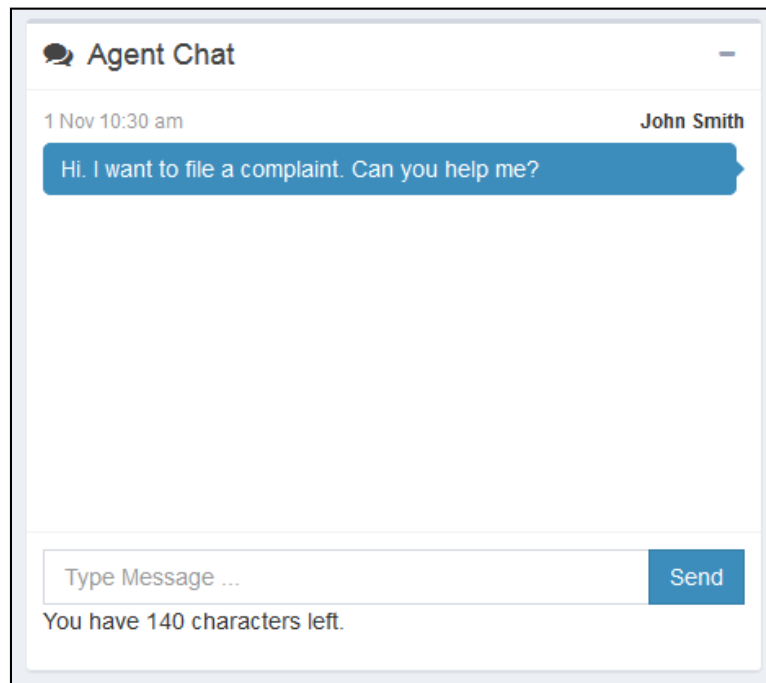


Figure 19. Screenshot of Real-Time Text Chat

When a CSR becomes available, the video chat will begin. After the call ends, the consumer is redirected to FCC.gov.

2.7 Management Portal

The Management Portal consists of two components, the Management and CDR dashboards. These dashboards present the manager with information about the operations of the call center and information about incoming calls.

2.7.1 Management Dashboard

The Management Dashboard, as shown in Figure 20, provides a number of KPIs the manager can monitor in real time. To access the Management 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>”.

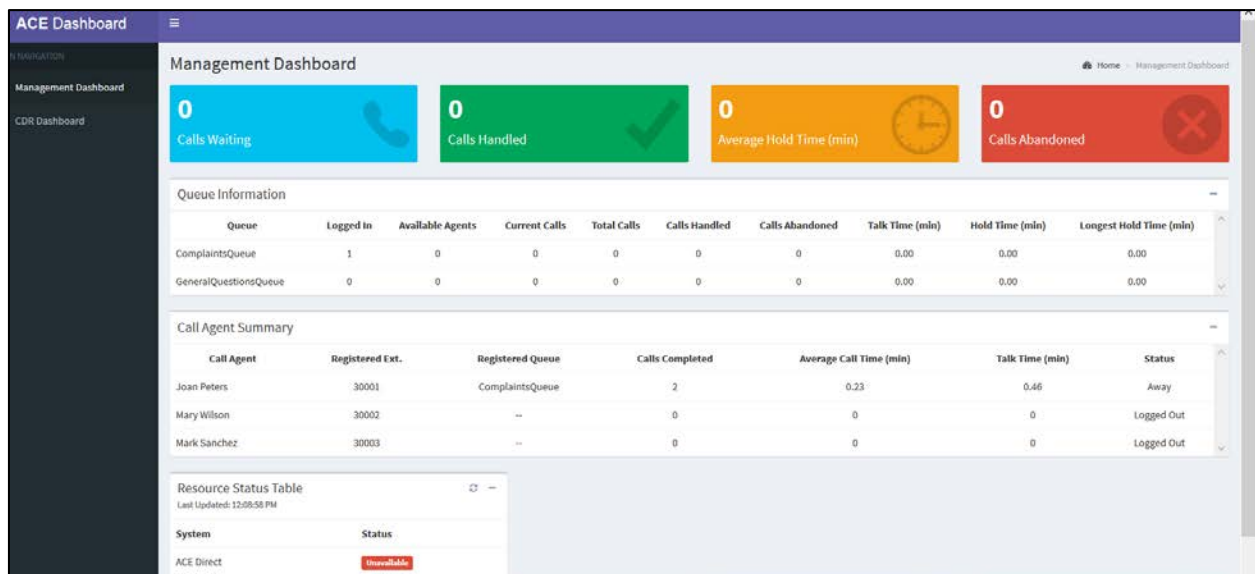


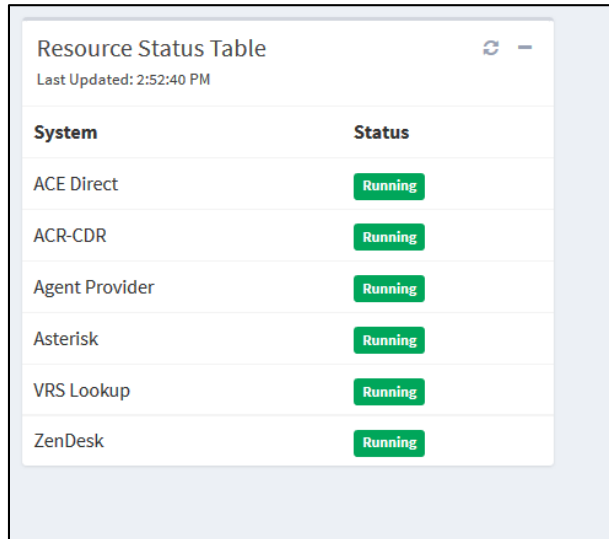
Figure 20. Screenshot of Management Dashboard

KPI Types

The ACE Direct 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 in the queue

- d. Calls Abandoned – Number of calls not answered
- 2. **Queue-related KPIs** – There are two queues in ACE Direct—ComplaintsQueue and GeneralQuestionQueue. The following KPIs are displayed per queue:
 - a. Logged In – Number of agents (CSR) 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
 - 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
- 3. **Agent-related KPIs** – The following KPIs are displayed per agent (CSR). The agent name, extension, and registered queues are displayed along with the KPI:
 - a. Agent name – Name of the agent
 - b. Registered extension – Extension assigned to the agent
 - c. Registered queues – Asterisk queues assigned to the agent. All queue names are displayed if an agent is assigned to more than one queue.
 - d. Calls Completed – Number of calls handled (answered and completed) by the agent
 - e. Average Call Time – Talk Time divided by number of calls
 - f. Talk Time – The cumulative time the agent has spent on calls
 - g. Agent status – Logged Off, Ready, Away, or In-Call
- 4. **Resource Status KPIs** – Figure 21 shows the resource status KPIs:
 - a. Resources – A list of services required (ACE Direct, Zendesk, Asterisk, CDR, Agent Provider and VRS Lookup)
 - b. Status – The current status of each service (Running or Unavailable)

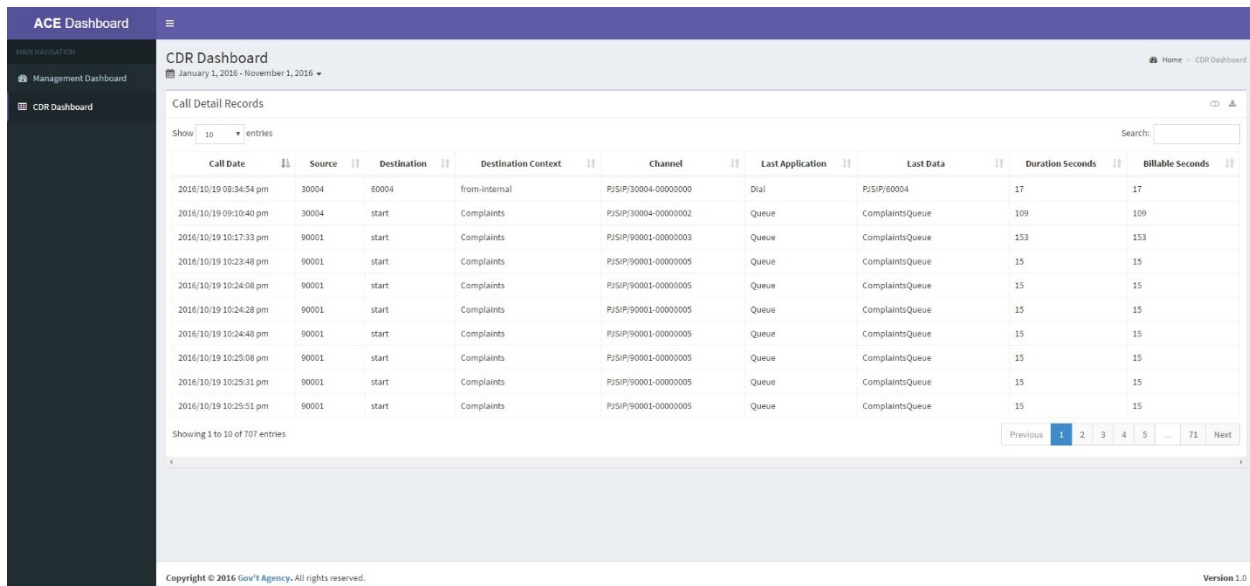


System	Status
ACE Direct	Running
ACR-CDR	Running
Agent Provider	Running
Asterisk	Running
VRS Lookup	Running
ZenDesk	Running

Figure 21. Screenshot of Resource Status

2.7.2 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 CSR's activity, and creating a listing of both incoming and outgoing calls. The ACE Direct Call Detail Record Dashboard, as shown in Figure 22, provides a method to view and export the Asterisk CDRs stored in the MySQL database.



Call Date	Source	Destination	Destination Context	Channel	Last Application	Last Data	Duration Seconds	Billable Seconds
2016/10/19 08:34:54 pm	30004	60004	from-internal	PJSIP/30004-0000000	Dial	PJSIP/60004	17	17
2016/10/19 09:10:40 pm	30004	start	Complaints	PJSIP/30004-0000000	Queue	ComplaintsQueue	109	109
2016/10/19 10:17:33 pm	90001	start	Complaints	PJSIP/90001-0000000	Queue	ComplaintsQueue	153	153
2016/10/19 10:23:48 pm	90001	start	Complaints	PJSIP/90001-0000000	Queue	ComplaintsQueue	15	15
2016/10/19 10:24:08 pm	90001	start	Complaints	PJSIP/90001-0000000	Queue	ComplaintsQueue	15	15
2016/10/19 10:24:28 pm	90001	start	Complaints	PJSIP/90001-0000000	Queue	ComplaintsQueue	15	15
2016/10/19 10:24:48 pm	90001	start	Complaints	PJSIP/90001-0000000	Queue	ComplaintsQueue	15	15
2016/10/19 10:25:08 pm	90001	start	Complaints	PJSIP/90001-0000000	Queue	ComplaintsQueue	15	15
2016/10/19 10:25:31 pm	90001	start	Complaints	PJSIP/90001-0000000	Queue	ComplaintsQueue	15	15
2016/10/19 10:25:51 pm	90001	start	Complaints	PJSIP/90001-0000000	Queue	ComplaintsQueue	15	15

Showing 1 to 10 of 707 entries

Previous 1 2 3 4 5 ... 71 Next

Copyright © 2016 Gov't Agency. All rights reserved. Version 1.0

Figure 22. Screenshot of Call Detail Record

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 (January 1st 2016 to Today), and Custom Range. Default selection is “All Time”.
- **Sort Column:** The user can sort on any column by clicking the sort icon located next to each column name. To multi-sort columns, the user depresses the shift key when selecting 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 3 presents the column definitions in the CDR table.

Table 3. 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.

Display Name	Database Column	Description
Disposition	disposition	An indication of what happened to the call. This may be NO ANSWER, FAILED, BUSY, ANSWERED, or UNKNOWN.
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

3. Installation and Configuration

This section presents guidance for the installation and configuration of the ACE Direct components. To reproduce this version of the platform, download the configuration and source code files from the <http://github.com/MITRE/ACE> website.

3.1 AWS Deployment

ACE Direct is implemented in the Amazon Web Service Cloud; however, it can be implemented in any cloud environment or an organization's corporate servers. The following tasks are required to establish end-to-end communications in an AWS environment.

3.1.1 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, and 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 pjsip.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.

3.1.2 Creating a Node.js Instance

1. In the EC2 console within AWS, click on "AMIs" to the left, select the 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.

3.1.3 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 Installation and Configuration Script

An installation script has been developed to assist in the deployment and configuration of the Asterisk server. The Asterisk install script will install and configure Asterisk for ACE Direct on a CentOS 7 server. This script can be utilized to quickly stand up an Asterisk instance. The script will perform the following tasks automatically:

- Update current packages and install required packages for Asterisk
- Install PJSIP
- Install Asterisk
- Retrieve and configure ACE Direct configuration files from git repository
- Start Asterisk

If you do not want to update the server’s packages, modify the script by commenting out or deleting the following line:

```
yum -y update
```

There are a few pre requisites to the script that must be satisfied before running it:

- A STUN server is available. (You may use Google’s free STUN server; however, it is recommended you set up a dedicated server because many users rely on Google’s STUN server, which may result in high latency.)
- The server must have a public IP address and a private IP address associated with it.
- A Dial-in number must be set up and configured in Twilio (or any other phone-number provisioning service) and point to the Asterisk server.
- A public/private key-pair must be created for and are located on the server.

The script is executed with six parameters:

```
./asterisk_install.sh <public_ip> <private_ip> <dialin>  
<stun_server> <public_key> <private_key>
```

The six parameters are defined as follows:

- <public_ip> - The public IP address of the Asterisk server
- <local_ip> - The private/internal IP address of the Asterisk server
- <dialin> - The ten-digit dialin phone number associated with the server
- <stun_server> - The DNS name and port of the dedicated STUN server
- <public_key> - The absolute path of the server's public key
- <private_key> - The absolute path of the server's private key

3.2.1 Manual Asterisk Server Installation

If you prefer to install Asterisk manually the following procedures may be used. ACE Direct Asterisk Server is built with the following software versions:

- OS: CentOS 6.7
- Node.js 0.10.41
- Asterisk: 13.8.0

The Asterisk 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 libsrtplib-  
devel zip unzip
```

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

```
sudo update -y
```

Once you have downloaded asterisk*.*.tar.gz, unzip the file and change directory to the newly created asterisk folder:

- `cd ./asterisk*version/`
- `./configure` - The configure command will look at the libraries you installed earlier as the prerequisites and compile Asterisk accordingly. If you are missing options in the next step, you may be missing libraries. The packages required to enable the res_srtp resource module may need to be installed separately.
- `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.

This option loads a graphical 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 “libstrp-devel” package installed by the above yum command completed successfully. If you need to reinstall a package, you will have to rerun the “configure” script.

```
make && make install
```

A successful installation will display on the console window. Following a successful installation, it is recommended to create the sample configuration files. These sample configuration files will provide a template from which the Asterisk server can be configured. To make the sample configuration files, simply run the script by executing the following command:

```
make samples
```

The configuration files are automatically placed in the correct `/etc/asterisk/*` directory. Once the files have been created, make sure the Asterisk instance is running using:

```
service asterisk status
```

If the server is not running, ‘service asterisk start’ may be used.

You may now restart the Asterisk service with the command:

```
service asterisk restart
```

3.2.2 How It Works

The Asterisk PBX system relies on the interconnected workings of three key files. The `pjpsip.conf`, `extensions.conf`, and `http.conf`. The `pjpsip.conf` file contains the connection endpoints and parameters while the `extensions` file contains the syntax and programming for the extensions assigned to those endpoint connections. The `http.conf` file is the server configuration for Asterisk. The certificates Asterisk uses for secure communications are defined here.

3.2.2.1 Dial Plan

The dial plan is defined by the `extensions.conf` and `pjpsip.conf` files under `/etc/asterisk` working together to establish the connection between devices and route the calls to those devices. An endpoint device is any device that places or receives the call. Table 4 shows an example of

common endpoint devices and the associated extension along with the required codecs for the sample dial plan configuration in this guide.

Table 4. 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

As displayed in Table 4, the extension is the number assigned to the endpoint. The ability of that endpoint to connect to the Asterisk instance is configured in the `pjpsip.conf` file. Its ability to dial to another endpoint is configured within the `extensions.conf` file. The device can be any such phone able to connect to the Asterisk instance. This could be a softphone, a desktop phone, a Cisco video communication phone, or other such device. The codec refers to the preferred media codec, or in some cases, the only available codec, used by the endpoint. This is also configured in the `pjpsip.conf` file.

3.2.2.2 Extensions.conf

The `extensions.conf` file is the configuration of your PBX. In this file you will need to define how the endpoint extensions communicate with each other, or with the Asterisk server itself. The Asterisk server consists of many different applications working together to perform different functions. These functions can be called in the `extensions.conf` dial plan to route an endpoint device to a desired outcome. For example, a caller wants to dial their voicemail. In `extensions.conf`, that configuration would need to state that the caller's phone number is to load the voicemail application. This would be done by the following syntax:

```
exten => _6XXX,1,Answer( )
      same => n,VoiceMail(1234@ourpbx)
```

In this example, any extension matching the pattern `_6XXX` will be answered to be placed immediately into the Voicemail application.

For specific phone numbers to route, as well as number schemes for unknown numbers, use the wildcard "X". In our example and as shown in the context `[from-internal]` of the `extensions.conf` file, all extensions ended with `6xxx` are configured. The file needs no other changes.

```
[from-internal]
exten => _50XX,1,Dial(PJSIP/${EXTEN})
same => n,DumpChan( )
same => n,HangUp( )

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

```
exten => _70XX,1,Dial(PJSIP/${EXTEN})  
same => n,DumpChan()  
same => n,HangUp()
```

The “[from-internal]” label refers to the context in which those extensions reside. A context is an organizational container that can host a group of extensions and extension patterns that can route and dial to other extensions within the same context. Extension patterns use wildcards to encompass a greater range of possible matches to the same string. To dial to an extension in another context, that context must be specifically identified in the coding. The context defined must also be the same for the “Context” attribute of the endpoints defined in the `pjpsip.conf` file (shown in subsection 3.2.4.1). For the `Dial()` function call, the extensions/numbers defined within the function call must be defined in `pjpsip.conf`, as shown in subsection 3.4.2.1.

3.2.2.3 Pjpsip.conf

The `pjpsip.conf` file defines the endpoint connection parameters. Some key elements must be configured in this file. The extension itself must be identified, which serves as its SIP identity and subsequently the phone extension. In addition, the file must define the authentication method for this extension as well as the protocol Asterisk must use. The extensions configuration for network connectivity, the allowed codecs, and Asterisk-specific configuration settings must also be defined to make the endpoint a viable connection.

The `pjpsip.conf` file tells the Asterisk PBX about the connection points from which your endpoint devices will acquire their information and connect. This means that this file is responsible for the endpoint connection, handling the contact that connects to that endpoint, routing the call, and passing information. The following is a list of critical attributes defined in a profile within `pjsip.conf`:

- **Transport** – The transport defined within the `pjpsip.conf` file specifies the configuration for TCP, UDP, WS, or WSS connections. These transport settings require configuration as to the external IP, internal IP, and in the case of web sockets, certificate information to be used for secure communication.
- **Endpoint** – To the `pjpsip.conf` file, an endpoint is a profile that matches to an endpoint device. The endpoint can be defined in `pjsip` independently or as a template that numerous extensions can call. This template helps to keep a condensed file because otherwise this file can grow quite large. Endpoint templates are identified by (!) after the identifying name.
- **Register** – A registration is a type of authentication from an endpoint device. The device will send authentication information to Asterisk that will then ‘Register’ the device to the PBX with the assigned endpoint profile. A registration is a temporary assignment of configuration options and network settings to identify the endpoint device and route calls.
- **Contact** – A contact is the network identifying information of a device that has registered and its corresponding extension. Viewing the contact information within the Asterisk console verifies that an endpoint device has authenticated and been assigned its configuration correctly.

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 pjsip.conf under the “[h264_phone_test](!)” section. The “h264_phone_test” profile name can be changed according to the user’s preference.

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.

3.2.3 Secure Calling

This web application makes use of a technology called WebRTC. In order for Asterisk to successfully communicate with WebRTC, it must use web sockets and, for most browsers, secure web sockets. Previously, the Asterisk server certificates could be used with self-signed certificates; however, this is no longer the case and Asterisk must be used with a valid certificate authority.

3.2.4 Sample Configurations

3.2.4.1 Pjpsip.conf

-----top of pjpsip.conf file-----

[transport-wss] ;name of transport configuration

```
type=transport           ; type being configured is of type transport
protocol=wss             ;protocol is web secure socket, can be tcp,udp,ws,wss
bind=0.0.0.0:443         ;bound ip/port
external_media_address=52.45.109.230 ;external ip of Asterisk
cert_file=/etc/asterisk/keys/star.pem ;certificate pem
priv_key_file=/etc/asterisk/keys/star.key ;certificate key
```

[endpoint-basic](!) ;Defines the name of the endpoint, the (!) designates it as a template

```
type=endpoint           ;defines the type
transport=transport-wss ;defines the transport to be used
context=from-internal   ;defines the context, must correspond to context in extensions.conf
disallow=all            ;explicitly deny all media unless specified
allow=ulaw              ;specify allow ulaw audio codec
allow=vp8               ;specify allow vp8 video
allow=h264              ;specify h264 video
allow=t140              ;specify allow t140 text protocol
force_rport=yes         ;network configuration, reflexive port
direct_media=no         ;network configuration, do not establish direct media link (NAT)
rewrite_contact=yes
media_address=52.45.109.230
rtp_symmetric=yes
ice_support=yes
message_context=internal-im ;context for internal messenger
```



```
[30001](endpoint-basic) ;Extension information, extension username is 30001
auth=auth30001 ;The auth profile to use
aors=30001 ;The aors profile to use
```

```
[auth30001](auth-userpass)
password=changeit! ;password to be changed
username=30001 ;username for authentication, typically extension number
```

```
[30001](aor-single-reg)
remove_existing=yes ; Remove pre-existing contact
max_contacts=1 ;Number of simultaneous contacts registered to an AOR
qualify_frequency=5 ;Interval in seconds of qualifying a registered contact
authenticate_qualify=yes ;Send authentication request on qualify if required
-----end of pjpsip.conf file-----
```

3.2.4.2 WebRTC Sample

The following is a sample WebRTC-enabled endpoint template in pjpsip.conf:

```
[endpoint-webrtc](!)
type=endpoint
transport=transport-wss
context=from-internal
disallow=all
allow=vp8
allow=ulaw
allow=t140
media_encryption=dtls ; Determines whether res_pjsip will use and enforce
dtls_verify=fingerprint ; Verify that the provided peer certificate is valid (default:
dtls_fingerprint=SHA-1
dtmf_mode=auto
dtls_rekey=0 ; Interval at which to renegotiate the TLS session and rekey
dtls_cert_file=/etc/asterisk/keys/asterisk.pem
dtls_ca_file=/etc/asterisk/keys/ca.crt
dtls_setup=actpass
ice_support=yes ;This is specific to clients that support NAT traversal
```

3.2.4.3 Installation Script

The following is an Asterisk installation script hosted at www.GitHub/MITRE/ACE:

```
#####
#####
#!/bin/bash
yum -y update
```

```
yum -y install -y epel-release bzip2 dmidecode gcc-c++
ncurses-devel libxml2-devel make wget openssl-devel newt-
devel kernel-devel sqlite-devel libuuid-devel gtk2-devel
jansson-devel binutils-devel git

#install PJSIP
cd /usr/src
wget http://www.pjsip.org/release/2.3/pjproject-2.3.tar.bz2
tar -jxf pjproject-2.3.tar.bz2
cd pjproject-2.3
./configure CFLAGS="-DNDEBUG -DPJ_HAS_IPV6=1" --prefix=/usr
--libdir=/usr/lib64 --enable-shared --disable-video --
disable-sound --disable-opencore-amr
make dep
make
make install
ldconfig
ldconfig -p | grep pj
#install asterisk
cd /usr/src
wget http://downloads.asterisk.org/pub/telephony/certified-
asterisk/asterisk-certified-13.8-current.tar.gz
tar -zxf asterisk-certified-13.8-current.tar.gz
cd asterisk-certified-13.8-cert2
./configure --libdir=/usr/lib64
make && make install && make samples
make config
service asterisk start
#add git repo domain name to known_hosts so RSA fingerprint
prompt does not pause script
ssh-keyscan git.codev.mitre.org >> ~/.ssh/known_hosts
# pull down config files and add to /etc/asterisk
# replace with public facing Git site
git clone ssh://git@git.codev.mitre.org/~smasoud/asterisk-
config.git
cd asterisk-config
yes | cp -rf * /etc/asterisk
cd /etc/asterisk
sed -i -e "s/<public_ip>/$1/g" pjsip.conf
sed -i -e "s/<local_ip>/$2/g" pjsip.conf
sed -i -e "s/<dialin>/$3/g" extensions.conf pjsip.conf
sed -i -e "s/<stun_server>/$4/g" pjsip.conf
sed -i -e "s/<public_key>/$5/g" http.conf
sed -i -e "s/<private_key>/$6/g" http.conf
service asterisk restart
```

3.3 Node.js

The Node.js is a platform built on Chrome's V8 JavaScript run-time engine, which was 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.

Node.js servers host the functionality of ACE Direct. More specifically, the servers host the Customer Service Representative, Consumer, Management, and Call Data Record portals. Node.js also provides CSR Data and Provider Data RESTful APIs to ACE Direct.

Examples of these RESTful services are VRS lookup and CSR verify functions. An Asterisk instance should already be up and running to support most of the Node.js instances.

Instructions for downloading and installing node.js can be found on the official Node.js website (<https://nodejs.org/en/>). For this version of ACE Direct, the Node.js server is built with the following software versions:

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

3.3.1 Installing

Start with a fresh CentOS 7 installation. Then update the current commands on the instance and install the following packages that are used to support Node.js:

```
sudo yum -y update
sudo yum -y install xterm 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 sendmailcf 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 zip unzip
```

Then, use wget to download and execute the nvm install script, then reload bashrc (you may need to restart your terminal for this to take effect):

```
wget -qO-
https://raw.githubusercontent.com/creationix/nvm/v0.23.2/in
stall.sh | bash
source ~/.bashrc
```

Validate that nvm has been successfully installed, then install Node.js. The first command should print out the version of node installed, which should coincide with the URL used to download the npm install script:

```
nvm -version
nvm install stable
```

Clone the Git appropriate repository to the instance:

```
git clone ssh://<repo_URL>.gitcd <repo>
```

If needed, generate a public/private key-pair for a non-root user and add the public key to your Git profile. You can use the following command to create an RSA key:

```
ssh-keygen -t rsa
```

Afterwards, move to the directory of the Git repo you cloned earlier and check out the desired release/branch:

```
git checkout <release>
```

In all the folders (except the CSR Data and Provider Data portals), there are JSON configuration file templates that must be configured prior to running the applications. For example, the `config.json_TEMPLATE` file must be renamed to `config.json` and configured with the appropriate values; 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. Refer to the additional configuration instructions in the *README.md* files of each source code repository.

Find the directory with the npm command, then link it so that it may be executed from any location. If npm is already linked to /usr/bin, skip this step:

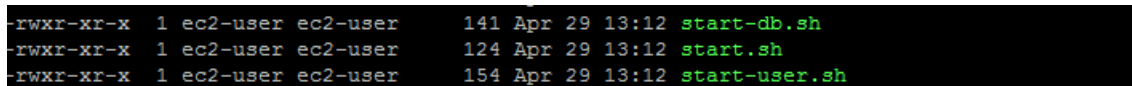
```
sudo ln -s <npm_dir> /usr/bin/npm
```

Finally, install the Node.js process manger (NPM) package.

```
npm install -g
npm install pm2 -g
```

3.3.2 Starting

In the root folder of each application, there are Shell scripts (`start*.sh`) for starting the node servers as shown in Figure 23. If they are not executable, use the UNIX command “`chmod +x`” to make them executable, and then execute the scripts to start up the Node.js servers:



```
rwxr-xr-x 1 ec2-user ec2-user 141 Apr 29 13:12 start-db.sh
rwxr-xr-x 1 ec2-user ec2-user 124 Apr 29 13:12 start.sh
rwxr-xr-x 1 ec2-user ec2-user 154 Apr 29 13:12 start-user.sh
```

Figure 23. Shell Scripts

```
chmod +x start*.sh./start.sh; ./start-db.sh; ./start-
user.sh
```

3.3.3 Status

The `pm2 list` command in Figure 24 shows the running node server(s). Run this command as the `ec2-user`:

```
ec2-user@ip-172-31-49-228 ~]$ pm2 list
```

App name	id	mode	pid	status	restart	uptime	memory	watching
server	0	fork	6938	online	1	23h	73.828 MB	disabled
server-user	1	fork	4413	online	0	2D	77.633 MB	disabled
server-db	2	fork	6861	online	1	24h	65.359 MB	disabled

Figure 24. *pm2 list* Command

3.3.4 Restart

The UNIX command “pm2 restart <node server>” will restart a node server if the server has been updated. For example, “pm2 restart server-user” will restart the server-user node server for the consumer complaint portal. You can also use “pm2 restart all” to restart all node processes currently running if necessary.

3.4 Management Portal

The ACE Direct 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.

3.4.1 Installation

For installation of Node.js:

- 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

3.4.2 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.
- AsteriskAD/sip/host – Fully qualified domain name (e.g., mySIP.example.com).
- dashboard/queuesAD – Queue names configured for ACE Direct. Currently, ACE Direct has two queue named: GeneralQuestionsQueue and ComplaintsQueue.
- Resource section – URL of the monitored resources (Zendesk, VRS lookup, Agent Provider, ACE Diorect, CDR).

Update the following parameters in Asterisk queues.conf if necessary:

```
[StandardQueue](!)
autofill=no ; calls come from Asterisk serially
```

```
strategy=rrmemory
joinempty=yes
leavewhenempty=no
ringinuse=no
wrapuptime=25 ; When a member hangs up, how many seconds
does queue wait before assigning another caller. Default
is 0.
setqueuevar=yes
eventwhencalled=yes
```

3.4.3 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.
```

3.4.4 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 as follows: all, trace, debug, info, warn, error and fatal.

3.4.5 Management Dashboard

3.4.5.1 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.js and dashboard.html) is hosted on the Node.js server (server-db.js). The operational data flow works as follows.

3.4.5.1.1 *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.

3.4.5.1.2 *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

3.4.5.2 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 5 shows the AMI actions and descriptions.

Table 5. 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 6.

Table 6. 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

3.4.6 Call Detail Record Dashboard

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 Asterisk 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/mysqld/mysqld.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 CSR desktop. This lookup verifies that the user-provided phone number is a registered VRS number. A MySQL database server is required, and a RESTful 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

3.5.1 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.

3.5.2 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 an 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 phone numbers of deaf and hard-of-hearing users, 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

graphical user interface (GUI) as well as a programmatic interface. For this proof of concept, the CAMH team used the GUI to provision telephone numbers.

To query the production iTRS database requires establishing an IPsec tunnel with Neustar. For the proof of concept, the CAMH team added the IP address 156.154.59.67 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 because new values are queries from DNS. This behavior must be disabled.

The following example snippet of code, which is part of an Asterisk Dial Plan, comes from an Extensions.conf file. Subsection 3.2.2.1 provides more information on the Asterisk Dial Plan.

```
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:nnnnnnnnnn>) ;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 (Figure 25). 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 does not set aside a /30 network as is typical for an IPSEC tunnel.

There is a NAT (actually a Port Address Translation) 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 the Neustar Encryption domain IP address to the router's private IP address. **Note:** Source/destination checking must be disabled on the router instance's elastic network interface.

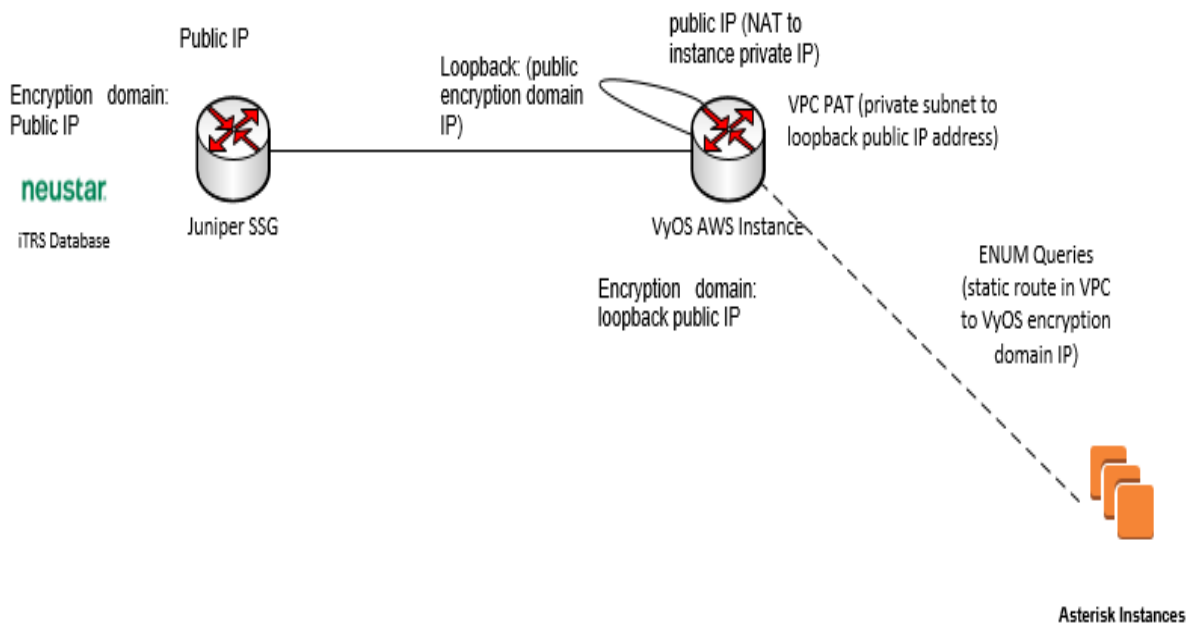


Figure 25. VyOS Router for SSL Tunnel

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 to ensure 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 do 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 COTS CRM

To demonstrate integration with a commercial Customer Relationship Management (CRM) service, ACE Direct connects to the Zendesk portal via the Enterprise Service Bus. ACE Direct sends JSON-based messages to the RESTful Zendesk API to manage and query customer records. Zendesk is a suite of web-based products that help companies provide better customer service. ACE Direct uses Zendesk to capture consumer complaints. When a consumer files a complaint, the ACE Direct software uses the Zendesk web service API to create and store a complaint ticket. This API also allows querying and updating of created tickets. When an ACE Direct CSR answers a call from a consumer, the application requests the ticket information from Zendesk and displays it on the ACE Direct CSR Desktop portal.

3.10 Enterprise Service Bus

The ESB provides a generic method to integrate with legacy database systems as well as the diverse number of databases and unstructured data repositories on the market and in use today. ACE Direct ESB integrates with a COTS CRM service (e.g., Zendesk) as a ticketing system for the CSR to document service cases.

3.10.1 Background

Apache ServiceMix 6.1.2 is used as the service broker. Apache ServiceMix is an enterprise-class, open-source distributed ESB based on the service-oriented architecture (SOA) model. It is a project of the Apache Software Foundation and was built on the semantics and application programming interfaces of the Java Business Integration (JBI) specification JSR 208. The software is distributed under the Apache License.

The productized and supported release of ServiceMix 4 is from Red Hat JBoss and called JBoss Fuse ESB. Fabric8 is a free Apache 2.0 Licensed upstream community for the JBoss Fuse product from Red Hat.

The current version of ServiceMix fully supports the OSGi framework. ServiceMix is lightweight and easily embeddable, and has integrated Spring Framework support. It can be run at the edge of the network (inside a client or server), as a standalone ESB provider, or as a service within another ESB. ServiceMix is compatible with Java SE or a Java EE application server. ServiceMix uses ActiveMQ to provide remoting, clustering, reliability, and distributed failover. The basic frameworks used by ServiceMix are Spring and XBean.

ServiceMix comprises the latest versions of Apache ActiveMQ, Apache Camel, Apache CXF, and Apache Karaf.

Additional installation features include:

- BPM engine via Activiti
- JPA support via Apache OpenJPA
- XA transaction management via JTA via Apache Aries

The ServiceMix ESB provides:

- Federation, clustering, and container-provided failover
- Hot deployment and life-cycle management of business objects
- Vendor independence from vendor-licensed products
- Compliance with the JBI specification JSR 208
- Compliance with the OSGi 4.2 specification through Apache Felix
- Support for OSGi Enterprise through Apache Aries

3.10.2 Installation Overview

Before you can start working with Apache ServiceMix, first install and get ServiceMix and its prerequisites running on the host machine.

3.10.2.1 ServiceMix system requirements

To run Apache ServiceMix itself, you will need Java Runtime Environment (JRE) 1.7.x (Java 7) or Java Runtime Environment (JRE) 1.8.x (Java 8), and about 100 MB of free disk space for the default assembly.

If you are developing your own integration applications and OSGi bundles, you will also need:

- Java Developer Kit (JDK) 1.7.x (Java 7) or
- Java Developer Kit (JDK) 1.8.x (Java 8)
- MySQL
- Apache Maven 3.0.4 or higher

The ACRDEMO broker application depends on MySQL for a database and Maven for building the application.

3.10.2.2 Installing the JDK

Issue the following commands to install the Java 8 JDK:

```
sudo apt-add-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

Set the JAVA_HOME environment variable in the bash startup:

```
vi ~/.bashrc
```

Add the following lines to the end of the .bashrc script:

```
JAVA_HOME=/usr/lib/jvm/java-8-oracle/
export JAVA_HOME
PATH=$PATH:$JAVA_HOME/bin
export PATH
```

3.10.2.3 Installing Apache Maven

To install Apache Maven, issue the following command:

```
sudo apt-get install maven
```

3.10.2.4 Installing MySQL

You will be able to set the password for the root account. **Note:** There is a current issue with the ESB that requires also setting the privileges for anonymous local users; accordingly, do not disable access for anonymous users. To install MySQL, issue the following commands:

```
sudo apt-get update
sudo apt-get install mysql-server
sudo mysql_secure_installation
sudo mysql_install_db
```

3.10.2.5 Configuring MySQL

The ServiceMix broker application for the ACR demo connects to the MySQL database; therefore, first create and configure the demo database.

Login to the MySQL command-line tool using the root account with the password you set earlier:

```
mysql --user=root --password=somepassword
```

Create a database named “broker” for the ACR demo:

```
mysql> CREATE DATABASE broker;  
mysql> USE broker;
```

Create the database user named “broker” for the ACR demo and set the password:

```
mysql> CREATE USER 'broker'@'localhost' IDENTIFIED BY  
      'somepassword';
```

Set the permissions for the database user “broker”. **Note:** This should be tuned to only grant the necessary privileges:

```
mysql> GRANT ALL PRIVILEGES ON broker.* TO 'broker'@'%'  
      WITH GRANT OPTION;
```

Note: There is a current issue with the ESB that requires also setting the privileges for anonymous local users. Privileges must also be set for anonymous users:

```
mysql> GRANT ALL PRIVILEGES ON broker.* TO ''@'localhost' WITH  
      GRANT OPTION;
```

Create the “users” table:

```
mysql> CREATE TABLE `users` ( `user_id` bigint(20) NOT  
      NULL, `user_name` varchar(50) DEFAULT NULL,  
      `user_description` varchar(45) DEFAULT NULL, `user_phone`  
      varchar(20) DEFAULT NULL, `user_address` varchar(50)  
      DEFAULT NULL, `user_account` varchar(50) DEFAULT NULL,  
      PRIMARY KEY (`user_id`));
```

Populate the 'users' table with test records. **Note:** The user_id values need to correspond to IDs of Zendesk users:

```
mysql> INSERT INTO `users` VALUES (3770168798,'John Doe  
      ','Some Details','555-555-  
      1111',NULL,'121212'),(4060741111,'Jane Doe','No  
      Description','222-111-1111','12341 Main  
      Street','12345671'),(4758821111,'Tim','No  
      Description','555-666-7777','','5656565');
```

3.10.2.6 Downloading and Building Broker Application

Set up SSH keys to access the git repository according to these instructions:

- [Adding a new SSH key to your GitHub account](#)

Create a folder for cloning the broker source code and navigate to that folder:

```
mkdir ~/code && cd ~/code
```

Clone the broker git repository:

```
git clone ssh://git@github.com:mitrefccace/acrdemo-broker.git
```

Navigate to the broker code folder:

```
cd camel-rest-proxy-blueprint/
```

Modify the applications blueprint file for your environment.

Build the broker application with Maven:

```
mvn clean install
```

3.10.2.7 Downloading and Installing Apache ServiceMix

Apache ServiceMix 6.1.2 is available under the Apache License v2 and can be downloaded from <http://servicemix.apache.org/downloads/servicemix-6.1.2.html>.

Create and navigate to a folder where the downloaded zip file will be placed:

```
mkdir ~/dev-tools && cd ~/dev-tools
```

Download and uncompress the zip file. For example:

```
wget  
http://mirror.cc.columbia.edu/pub/software/apache/servicemix/servicemix-6/6.1.2/apache-servicemix-6.1.2.zip  
unzip apache-servicemix-6.1.2.zip
```

3.10.2.8 Running and Configuring ServiceMix

In a command shell, navigate to the ServiceMix bin directory (e.g., ~/dev-tools/apache-servicemix-6.1.2):

```
cd ~/dev-tools/apache-servicemix-6.1.2/bin
```

Start ServiceMix:

```
./servicemix
```

Install the following features:

```
karaf@root>feature:install jdbc
karaf@root>feature:install pax-jdbc-mysql
karaf@root>feature:install camel-jsonpath
karaf@root>feature:install camel-jetty
karaf@root>feature:install camel-jdbc
```

You may need to download the pax-jdbc artifact from the Maven repository if the pax-jdbc-mysql install does not work:

```
karaf@root>feature:repo-add pax-jdbc 0.6.0
```

Create the JDBC connection to the MySQL database:

Note: Depending on your version of jdbc, you will need either the command “jdbc:ds-create” or “jdbc:create”. Type <tab> to print a list of available commands and find the one you need:

```
karaf@root>jdbc:ds-create -dn mysql -url
jdbc:mysql://localhost:3306/demo?user=broker&password=somep
assword mySqlDataSource
karaf@root>jdbc:create -d mysql -t MySQL -url
jdbc:mysql://localhost:3306/broker -u broker -p
somepassword mySqlDataSource
```

Check the JDBC connection was created:

```
karaf@root>jdbc:datasources
```

Another way to check the database connection is to issue a query:

```
karaf@root>jdbc:query jdbc/mySqlDataSource "select * from
users"
```

Install the broker application. The application will be installed as an OSGI bundle:

```
karaf@root>bundle:install -s mvn:org.apache.camel/camel-
rest-proxy-blueprint/2.16.3
```

Check that the broker was installed. The bundle should be the last bundle in the list and its status should be ACTIVE:

```
karaf@root>bundle:list
```

3.10.2.9 Install and Start ServiceMix as a Service

Start the ServiceMix if is not already started. Issue the following commands:

```
karaf@root>feature:install wrapper
karaf@root>wrapper:install -s AUTO_START -n KARAF -d Karaf
-D "Karaf Service"
```

A message similar to the following will be displayed:

Setup complete. You may wish to tweak the JVM properties in the wrapper configuration file:

```
        /home/ubuntu/dev-tools/apache-servicemix-  
6.1.2/etc/KARAF-wrapper.conf
```

before installing and starting the service.

Ubuntu/Debian Linux system detected (SystemV):

To install the service:

```
$ ln -s /home/ubuntu/dev-tools/apache-servicemix-  
6.1.2/bin/KARAF-service /etc/init.d/
```

To start the service when the machine is rebooted:

```
$ update-rc.d KARAF-service defaults
```

To disable starting the service when the machine is rebooted:

```
$ update-rc.d -f KARAF-service remove
```

To start the service:

```
$ /etc/init.d/KARAF-service start
```

To stop the service:

```
$ /etc/init.d/KARAF-service stop
```

To uninstall the service :

```
$ rm /etc/init.d/KARAF-service
```

For systemd compliant Linux:

To install the service (and enable at system boot):

```
$ systemctl enable /home/ubuntu/dev-tools/apache-  
servicemix-6.1.2/bin/KARAF.service
```

To start the service:

```
$ systemctl start KARAF
```

To stop the service:

```
$ systemctl stop KARAF
```

To check the current service status:

```
$ systemctl status KARAF
```

To see service activity journal:

```
$ journalctl -u KARAF
```

To uninstall the service (and disable at system boot):

```
$ systemctl disable KARAF
```


Exit the ServiceMix/Karaf shell, by shutting down ServiceMix:

```
karaf@root>shutdown
```

Install the ServiceMix service:

```
sudo ln -s /home/ubuntu/dev-tools/apache-servicemix-6.1.2/bin/KARAF-service /etc/init.d/
```

Set the service to start when the machine is rebooted:

```
sudo update-rc.d KARAF-service defaults
```

Start the ServiceMix service:

```
sudo /etc/init.d/KARAF-service start
```

To log back into ServiceMix once the service is started, issue the following commands:

```
cd ~/dev-tools/apache-servicemix-6.1.2/bin
./client
```

To exit the ServiceMix shell without shutting down the service, type ^D (i.e., Ctrl-D). **Note:** If you type shutdown in ServiceMix shell, the entire service will be shut down.

3.10.3 Testing the Broker Application

At this point, all of the code for the Broker application is contained in the OSGI Blueprint file (i.e., blueprint.xml) that can be viewed here: <https://github.com/mitrefccace/acrdemo-provider.git>

If the Broker application is running in ServiceMix running, you can check the application by using curl at the command line. For example:

```
$ curl -u
wkchang@mitre.org/token:hLLUnPzJtpvMZ5WnntN3wCneKHkl20kP0Hh
n5NrD http://localhost:9090/api/v2/users/me.json --
insecure
```

You can check the status and statics of the main Broker route in the ServiceMix/Karaf shell:

```
karaf@root>camel:route-info rest-http-zendesk-mysql-demo
```

Here is example output from the camel:route-info command:

```
Camel Route rest-http-zendesk-mysql-demo
  Camel Context: camel-1
  State: Started
  State: Started

Statistics
Exchanges Total: 2
```

Exchanges Completed: 2
Exchanges Failed: 0
Exchanges Inflight: 0
Min Processing Time: 240 ms
Max Processing Time: 494 ms
Mean Processing Time: 367 ms
Total Processing Time: 734 ms
Last Processing Time: 240 ms
Delta Processing Time: -254 ms
Start Statistics Date: 2016-07-19 14:43:44
Reset Statistics Date: 2016-07-19 14:43:44
First Exchange Date: 2016-07-19 15:12:15
Last Exchange Date: 2016-07-19 15:12:3

Acronyms

ACE	Accessible Communications for Everyone
ALI	Automatic Location Identification
AMA	Automatic Message Accounting
API	Application Programming Interface
APP	Access Point Platform
AWS	Amazon Web Services
CAMH	CMS Alliance to Modernize Healthcare
CFR	Code of Federal Regulations
CRM	Customer Relationship Management
CSR	Customer Service Representative
CSV	Comma Separated Value
DVC	Direct Video Calling
FCC	Federal Communications Commission
FFRDC	Federally Funded Research and Development Center
iTRS	Internet Telecommunications Relay Service
JB	Java Business Integration
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
NAT	Network Address Translation
OS	Operating System
OSGi	OSGi Alliance (formerly Open Systems Group Initiative)
P2P	Point to Point
PAT	Port Address Translation
PBX	Private Branch Exchange
PSTN	Public Switch Telephone Network
QoS	Quality of Service
RFC	Request for Comment
RTCP	RTP (Real-time Transport Protocol) Control Protocol
RTM	Requirements Traceability Matrix

RTT	Real-Time Text
RUE	Relay User Equipment
SIP	Session Initiation Protocol
SOA	Service-Oriented Architecture
SRTP	Security Real-Time Transport Protocol
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TURN	Traversal Using Relay NAT
UDID	Unique Device Identifier
UDP	User Datagram Protocol
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)
WS	Web Services
WSS	Web Services Security?