

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

Version 2.0

November 1, 2017

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 17-3759.

© 2017, 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
1.1	February 17, 2017	CAMH	Version 1.1 for release to Sponsor
2.0	November 1, 2017	CAMH	Version 2.0 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.

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 twenty customer service representatives. This new DVC platform is called ACE Direct.

Implementing the Direct Video Calling platform provides critical benefits toward achieving functionally equivalent telecommunications:

- **Improved Communications** – DVC 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. Executive Order 13548 (July 2010) directed federal agencies to increase employment opportunities for people with disabilities.
- **Simple Implementation** – The technology to implement a DVC 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** – DVC 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 download or clone these files at <https://github.com/FCC/ACEDirect>.

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	3
2.4 Highlighted User Stories	6
2.5 Customer Service Representative Desktop User Guide	7
2.5.1 Logging into ACE Direct.....	7
2.5.2 Side Panels	9
2.5.3 Video and Real-Time Text Communications	12
2.5.4 Agent Desktop Portal Header	13
2.5.5 Video Relay Service and Ticket Information	14
2.6 Consumer Portal	15
2.6.1 Submitting a Complaint	15
2.6.2 Real-Time Text Chat	18
2.6.3 Leaving a Videomail.....	18
2.7 Kuando BusyLight™ Visual Ring Indicator and Agent Status	20
2.7.1 Agent Statuses.....	20
2.7.2 Kuando BusyLight™ Status Configuration.....	20
2.7.3 Lightserver	22
2.8 Management Portal	23
2.8.1 Management Dashboard	23
2.8.2 Call Detail Record Dashboard	25
2.8.3 BusyLight™ Configuration	27
2.9 Identity and Access Management	27
2.9.1 Login Screen	28
2.9.2 Consumer Dashboard.....	31
2.10 NGINX	36
2.11 Redis.....	37
3. Installation and Configuration	38
3.1 Secure Sockets Layer Certificate	38
3.2 Asterisk Installation and Configuration Script.....	38
3.2.1 How It Works.....	38
3.2.2 Web Secure Sockets.....	41
3.2.3 Sample Configuration Files	41
3.3 Node.js.....	43
3.3.1 Node.js Installation	43
3.4 Management Portal	44
3.4.1 Management Portal Installation	44

3.4.2	Management Portal Configuration.....	44
3.4.3	Log Files	44
3.4.4	Management Dashboard	45
3.4.5	Call Detail Record Dashboard	49
3.5	Agent / Agent Database (Provider)	53
3.5.1	MySQL Database Server Configuration.....	53
3.6	Video Relay Service User Database (Provider)	56
3.6.1	MySQL Database Server Configuration.....	56
3.7	STUN Server	57
3.8	iTRS ENUM Database	58
3.9	StrongSWAN for Secure Socket Layer Tunnel	58
3.9.1	Installation	58
3.9.2	AWS-Specific Config.....	62
3.9.3	Troubleshooting.....	62
3.10	Commercial Customer Relationship Management	66
3.11	FenDesk Customer Relationship Management	66
3.12	Enterprise Service Bus	66
3.12.1	Background.....	66
3.12.2	Installation Overview.....	67
3.12.3	Editing blueprint.xml Application File.....	74
3.12.4	Testing the Broker Application.....	75
Acronyms.....		76

List of Figures

Figure 1.	Notional Diagram for ACE Direct Platform	4
Figure 2.	Screenshot of Agent Desktop Login, Agent Desktop Layout	8
Figure 3.	Screenshot of Agent Desktop	9
Figure 4.	Screenshots of the Agent Statuses	10
Figure 5.	Screenshot of Agent Status.....	10
Figure 6.	Screenshot of CSR Mailbox	11
Figure 7.	Screenshot of Videomail Playback.....	11
Figure 8.	Screenshot of Consumer Chat Box.....	12
Figure 9.	Screenshot of Script Box	13
Figure 10.	Screenshot of Call Duration and Assistance Button.....	13
Figure 11.	Screenshot of Agent Profile.....	14
Figure 12.	Screenshot of VRS Information	14

Figure 13. Screenshot of Ticket Information	15
Figure 14. Screenshot of Videophone Number Entry	16
Figure 15. Screenshot of Consumer Portal	16
Figure 16. Screenshot of Complaint Ticket Form	17
Figure 17. Screenshot of Video Chat Window	17
Figure 18. Screenshot of Real-Time Text Chat	18
Figure 19. Screenshot of Ready to Record	19
Figure 20. Screenshot of Videomail Recording in Progress.....	19
Figure 21. Screenshot of Kuando BusyLight™ Configuration Page	21
Figure 22. Screenshot of Kuando BusyLight™ Default Color Scheme	21
Figure 23. Lightserver GUI.....	22
Figure 24. Screenshot of Management Dashboard	23
Figure 25. Screenshot of Resource Status.....	25
Figure 26. Screenshot of Call Detail Record	25
Figure 27. Screenshot of Login Screen.....	28
Figure 28. Screenshot of Retrieve Username	29
Figure 29. Screenshot of Reset Password	30
Figure 30. Screenshot of Register Your Account	31
Figure 31. Screenshot of Consumer Desktop	31
Figure 32. Screenshot of Consumer Basic Info	32
Figure 33. Screenshot of Update Password Tab	33
Figure 34. Screenshot of Add/Update Security Questions	34
Figure 35. Screenshot of Consumer Dashboard	35
Figure 36. Screenshot of Logout Screen.....	36
Figure 37. Internal and External Paths with NGINX.....	36
Figure 38. NGINX and OpenAM Integration.....	37

List of Tables

Table 1. ACE Direct Components	5
Table 2. Highlighted User Stories for ACE Direct.....	7
Table 3. Agent Status	20

Table 4. Lightserver GUI Data Elements	23
Table 5. Call Detail Record Column Definition	26
Table 6. Example Asterisk Endpoint Extensions.....	39
Table 7. AMI Actions and Descriptions	45
Table 8. Asterisk AMI Events and Descriptions	47

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 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. As part of the Accessible Communications for Everyone (ACE) program, 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 that supports Direct Video Calling (DVC) for two to twenty Agents. Implementing ACE Direct in a corporate production environment requires customization to ensure adherence 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 it offers significant gains for providing functionally equivalent telecommunications, including:

- **Improved Communications** – DVC improves privacy and decreases misrepresentation, which enhances efficiency, effectiveness, and productivity.
- **Career Opportunities** – Employing native American Sign Language (ASL) consumers to handle customer service video calls expands hiring opportunities. Executive Order 13548 (July 2010) directed federal agencies to increase employment opportunities for people with disabilities.
- **Simple Implementation** – The technology to implement a DVC 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** – DVC ensures compliance with the Americans with Disabilities Act (ADA) 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 consumer 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 architecture, user stories, and describes how to integrate DVC within an agency's current call center workflow to provide an independent, on-demand service.

In addition to this release documentation, detailed configuration and source code files are available to the public at <https://github.com/FCC/ACEDirect> for download and reproduction of the platform to support and promote future platform enhancements for the deaf, hard-of-hearing, deaf-blind, and speech-disabled community.

2. Overview of Direct Video Calling and ACE Direct

Deaf, hard-of-hearing, deaf-blind, or speech-disabled people use TRS to communicate with hearing people over the phone. Since the early 2000s, video relay service (VRS) calls have been the primary way that ASL consumers 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 ACE 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 consumers. 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 that VRS calls were handled on average 33 percent faster and the number of deaf consumers increased approximately threefold. Most impressive is that the FCC achieved these results without adding staff to handle the increased call volume.

2.2 Open Source Development to Promote Community Involvement

ACE Direct is open source technology that offers one option for implementing DVC. Open source 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 employs a decentralized model of production. A main principle of open source software development is peer production: products such as source code, "blueprints," and documentation are 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 open source ACE Direct platform.

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 architecture of the ACE Direct components from a configuration and programming standpoint.

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

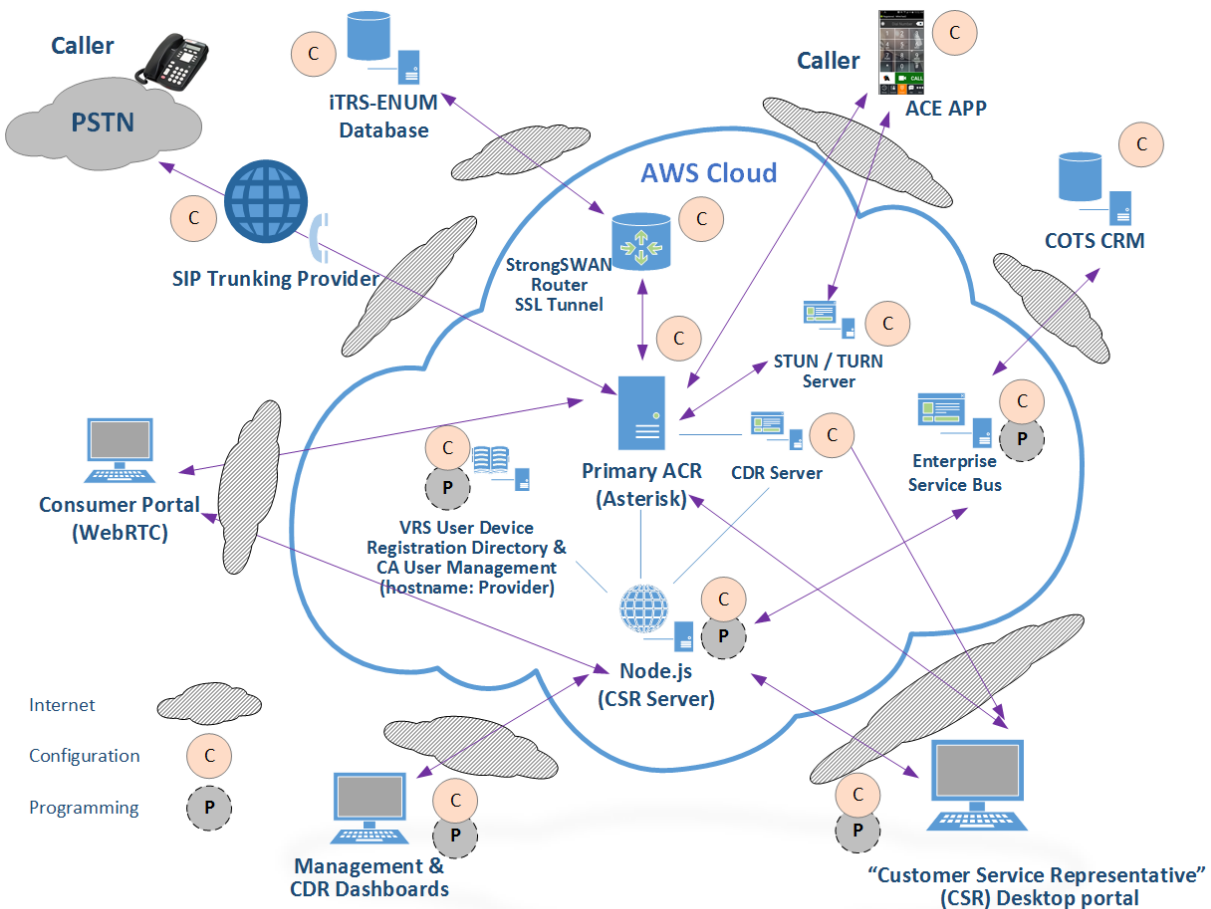


Figure 1. Notional Diagram for ACE Direct Platform

As Figure 1 shows, some ACE Direct components require only configurations (noted as “C”) and other components require both configuration 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 (Private Branch Exchange)	Asterisk is an open source PBX that supports direct video communication via both Public Switched Telephone Network (PSTN) and video calls.	Subsection 3.2
Node.js	Node.js is an open source platform that can be used to develop applications and servers. For ACE Direct, the Node.js server contains several services running on ports to support the Agent 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 Agent 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 combines form submission with real-time audio, video, and text communication to an Agent.	Subsection 2.6
Management and Call Detail Record Dashboards	The ACE Direct Management dashboard provides Key Performance Indicators (KPI) that 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 Agent User Management (hostname: Provider)	The “Provider” server emulates the services for both the Agent 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
Agent Desktop	The Agent desktop provides a user interface to the Agent for login and conducting DVC services to the ACE Direct consumers.	Subsection 2.5
STUN Server	STUN (formerly Simple Traversal of UDP through NAT RFC 3489) 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
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

Component	Description	Additional Details
StrongSwan 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 ESB 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 Agent to document service cases.	Subsection 3.10
Identity and Access Management	ACE Direct uses the OpenAM and OpenIDM components from ForgeRock to provide identity and access management functionality.	Subsection 2.9
External Visual Ring Indicator and Agent Status	The Kuando BusyLight™ is used as an external visual ring indicator and Agent status instrument. ACE Direct supports both of its models, Alpha and Omega.	Subsection 2.7
Reverse Proxy, Load-balancer and HTTP Cache	NGINX is used as a reverse proxy to only expose HTTPS/port 443 and hide internal port number and internal script names.	Subsection 2.10
State Management and Key Information Storage	Redis is an in-memory key-value data store, used as a database to store data previously stored in memory to manage state.	Subsection 2.11

2.4 Highlighted User Stories

The FCC and CAMH partnered with federal agencies to derive typical requests for services and call center workflows. CAMH built ACE Direct to encompass the core functions of a traditional hearing-based call center. ACE Direct focuses on the responsibilities of Agents and their managers. For version 2.0, CAMH added the role of Administrator to provision and deprovision Agents to the system. Table 2 presents a summary of ACE Direct user stories, which demonstrate these functions and capabilities.

Table 2. Highlighted User Stories for ACE Direct

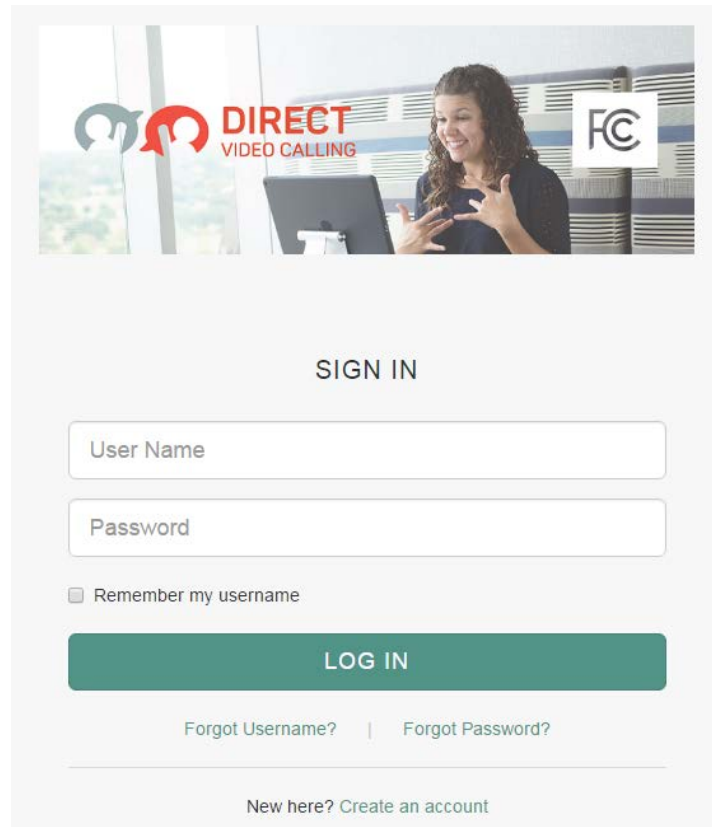
User Story	Description
Direct Video Call with an ASL-fluent Agent	As an ASL user, I want to speak with another ASL user when I contact a call center.
CRM Integration	As an Agent, I want to view, update, and enter new information regarding contact with the consumer from the corporate CRM system.
Call Script Integration	As an Agent, I want to view corporate call scripts based on the needs of the consumer.
Call-handling capabilities	As an ACE Direct Agent, I want to perform “Call on Hold” and “Call Transfer” as needed.
Consumer Portal	(A complaint process illustrates this story.) As a consumer, I want to file a complaint through a web portal on my website. I also want the option of conversing with the Agent through video and Real-Time Text. Please refer to subsection 2.6 for details.
Videomail	As an Agent, I want to retrieve a videomail left by customers.
Management Dashboard	As the ACE Direct manager/operator, I want to access near real-time information on the dashboard.
Call Detail Record	As the ACE Direct Administrator, I want to access the Call Detail Record through a web portal and export CDRs as needed for audit purpose. Please refer to subsection 2.7.2 for details.
Web-based Application	As an Agent, I want the ability to work remotely from the main call center, if necessary.
Multi-CSR Login with Status	As the ACE Direct Agent, I want to log in using the Agent Desktop along with other Agents and I want to change my status between “Ready” and “Away”.
Add, Suspend, or Remove an Agent’s Access	As the Administrator, I want to add, suspend or remove an Agent’s access to ACE Direct.

2.5 Customer Service Representative Desktop User Guide

At the time of this writing, the ACE Direct Agent Portal is compatible with the Chrome web browser. Chrome is a WebRTC-compatible browser. The use of WebRTC technology allows ACE Direct to present video directly through the browser, thus eliminating the need for a second monitor and providing a full omnichannel experience for the Agent. The CSR Desktop User Guide provides a walkthrough of the Agent 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 as shown in Figure 2. To access the portal, Agents must enter their username and password.



DIRECT
VIDEO CALLING

FCC

SIGN IN

User Name

Password

☐ Remember my username

LOG IN

[Forgot Username?](#) | [Forgot Password?](#)

New here? [Create an account](#)

Figure 2. Screenshot of Agent Desktop Login, Agent Desktop Layout

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

- Side panels (left and right) to provide navigation and information to the Agent, including a videomail retrieval panel
- A user chat area for real-time text chats with the VRS consumer
- A header area that displays call duration information and an assistance button
- Profile information displaying the Agent's name and picture and the ability to sign out of the system
- VRS consumer information such as first name, last name, etc.
- Current CRM ticket information provided by the VRS consumer

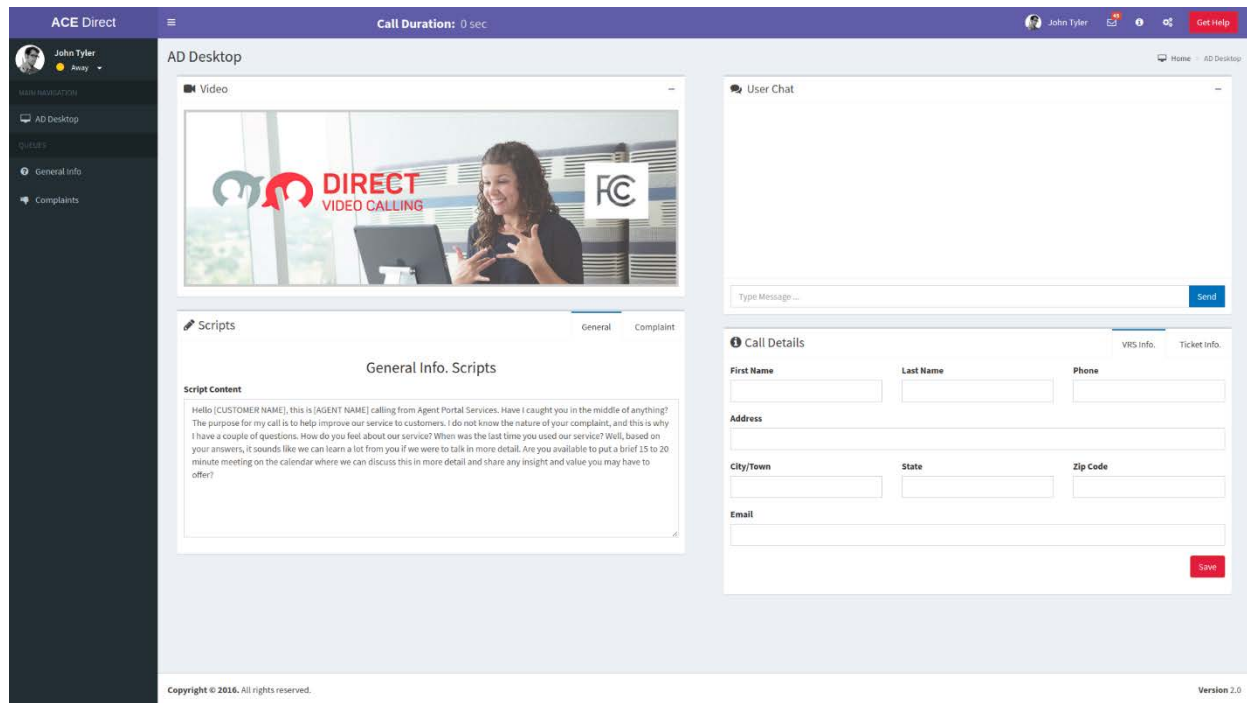


Figure 3. Screenshot of Agent Desktop

2.5.2 Side Panels

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

2.5.2.1 Left Side Panel (Main Navigation)

As shown in Figure 4, the left-side panel of the portal provides the Agent with both Consumer Status and the Main Navigation. Here Agents can select their status as “Ready” or “Away” via the dropdown status change button. When an Agent first signs into the portal, the status defaults to “Away”. When the Agent is ready to receive calls, the Agent selects the “Ready” status. For an incoming call, an intermediate “Incoming Call” status appears, along with a modal alert dialog that takes the foreground. Once the Agent enters a call, the status changes to “In Call”. After the Agent leaves the call, the Agent enters a “Wrap Up” state where the Agent can perform any tasks related to the call before receiving another call. When the Agent is done with wrap up and clicks on the “Save” button, the Agent’s status returns to the “Ready” state. The Agent status will also be reflected on the Management Dashboard and the BusyLight™.

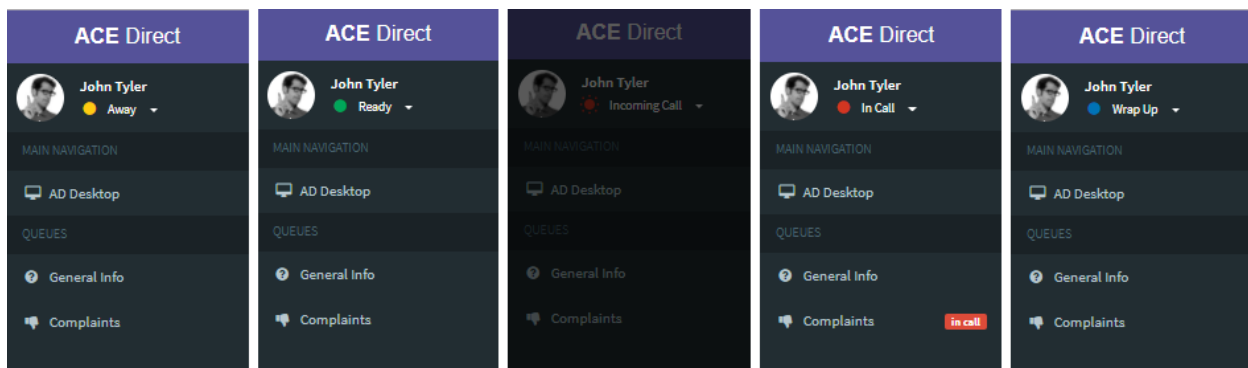


Figure 4. Screenshots of the Agent Statuses

2.5.2.2 Right-side Panel (Agent Status and Videomail mailbox)

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. The Agent can reach the videomail mailbox directly by clicking on the envelope icon in the top right corner of the portal. This section can be collapsed to give the Agent more space for the main content area. Upon opening the right-side panel, the Agent can access two tabbed content areas (Agent Status and Videomail mailbox).

Agents		Video Mail	
Status	Agent	Ext	Queues
● Ready	Millard Fillmore	30004	ComplaintsQueue GeneralQuestionsQueue

Figure 5. Screenshot of Agent Status

2.5.2.2.1 Agent Status

The Agent Status section provides the Agent with a list of Agents logged into ACE Direct. The Agent can view information about each Agent listed, such as their status, extension, and queues.

2.5.2.2.2 Videomail

The Videomail tab, as shown in Figure 6, displays a list of videomails received while the Agent was unavailable to take the call. This list provides the Agent with the time, date, duration and

status of the videomail. The Agent can sort the videomail table by any of the columns in ascending or descending order, and can filter the videomail by status. The status may be “Unread”, “Read”, “In Progress”, or “Closed”. Unread videomails are highlighted in **bold** and an indicator at the top right of the screen provides the Agent with a count of unread videomails.

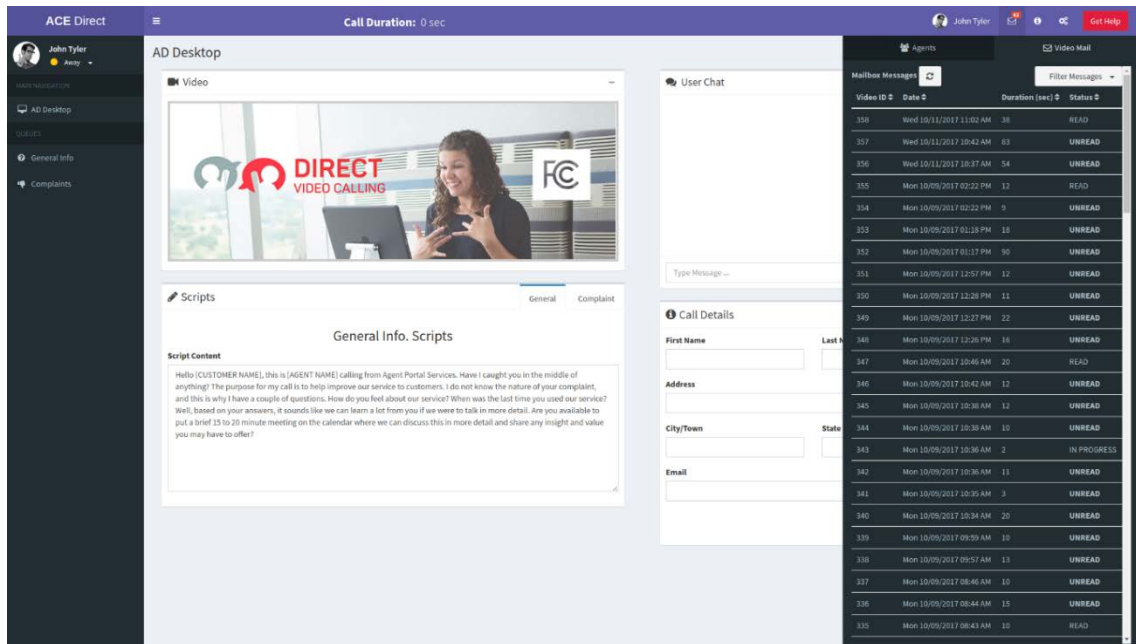


Figure 6. Screenshot of Agent Mailbox

By clicks on a videomail, the Agent can view it and update the status. Figure 7 displays the playback screen.

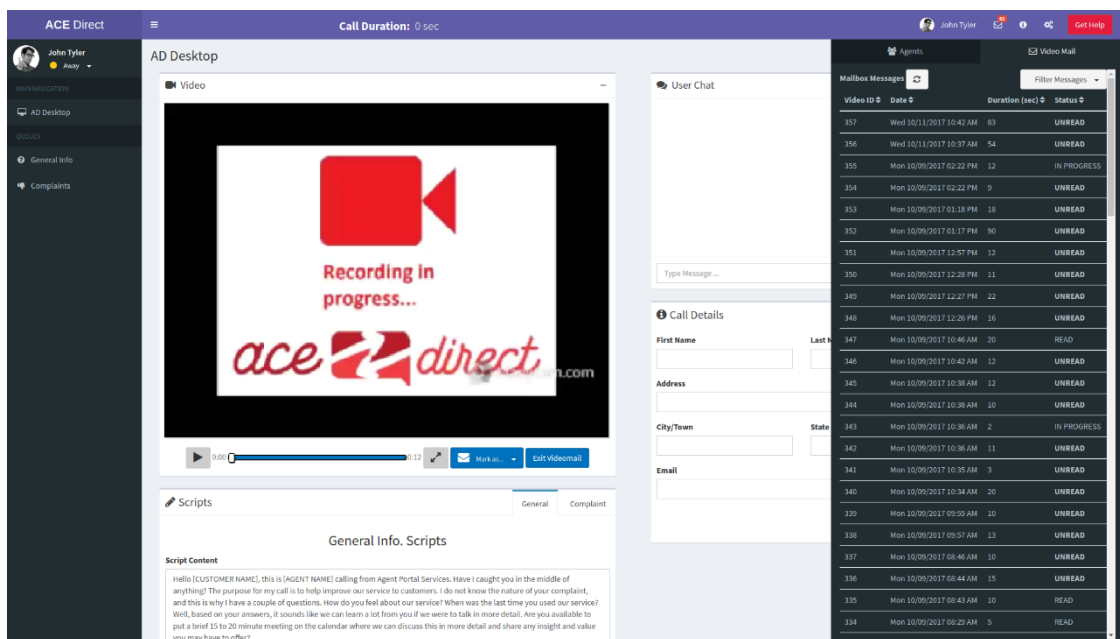


Figure 7. Screenshot of Videomail Playback

The status can be changed to “Unread”, “Read”, “In Progress” or “Closed”. If the Agent deletes the videomail, it is removed from the videomail mailbox.

2.5.3 Video and Real-Time Text Communications

The ACE Direct platform supports two methods of communication; video and Real-Time Text. This subsection describes how each method operates in the platform.

2.5.3.1 Video Chat

Video Chat communications on the platform occur through a browser using WebRTC technology. The consumer must be using a WebRTC-compatible browser to enable this functionality. During a call, the Agent has button options to mute audio, mute video, or view the consumer’s video in full screen mode.

2.5.3.2 Real-Time Text Chat

The User Chat box, as shown in Figure 8, provides the Agent a secondary method of communication with the consumer. As the Agent types a message to the consumer in the input field, the consumer will see the message in real time. The chat history remains visible to the Agent until the Agent closes the ticket. Use of this feature does not require a WebRTC-compatible browser by the consumer.

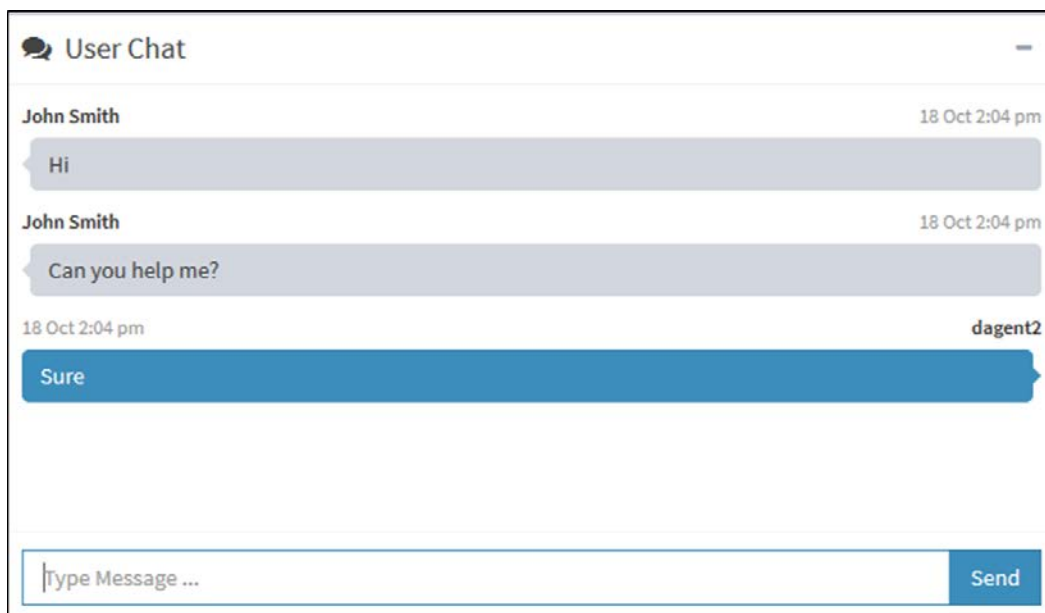


Figure 8. Screenshot of Consumer Chat Box

2.5.3.3 Scripts

The Scripts box, as shown in Figure 9, provides the Agent with two sub-tabbed areas where the Agent can select the appropriate script to apply to the conversation. This area is customizable during integration at a site.

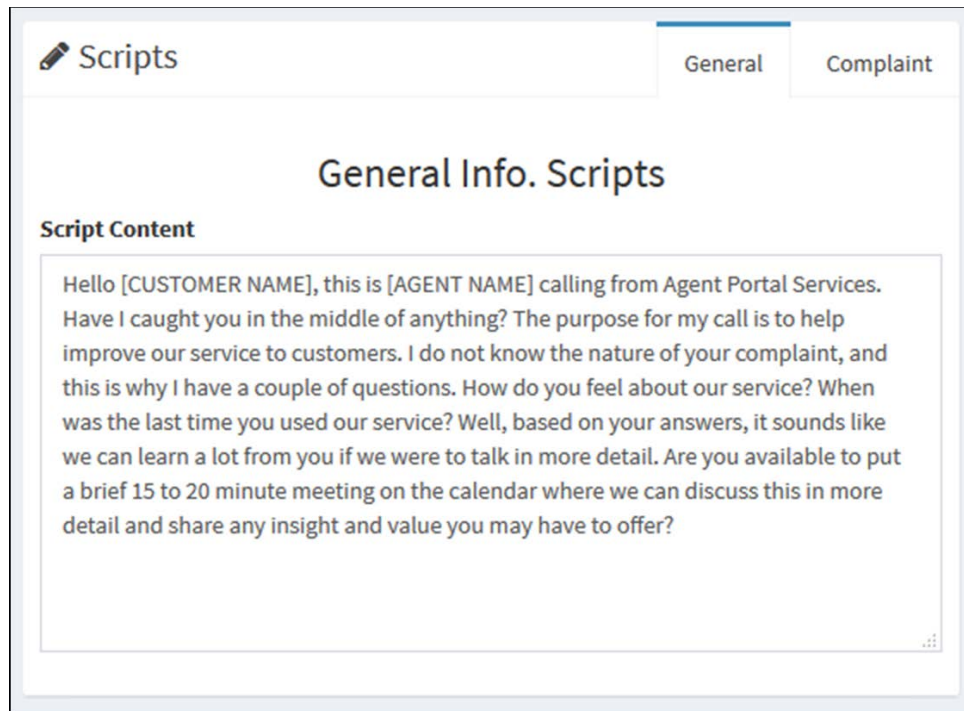


Figure 9. Screenshot of Script Box

2.5.4 Agent Desktop Portal Header

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

2.5.4.1 Call Duration and Get Help Button

The Call Duration shows a running clock of the call length once the Agent accepts the incoming consumer call. As Figure 10 shows, the Get Help button allows the Agent to request help from a Manager during a call. When the Agent clicks the Get Help button, the Agent's name will change color and begin to flash on the Management Dashboard to indicate the Agent needs help.

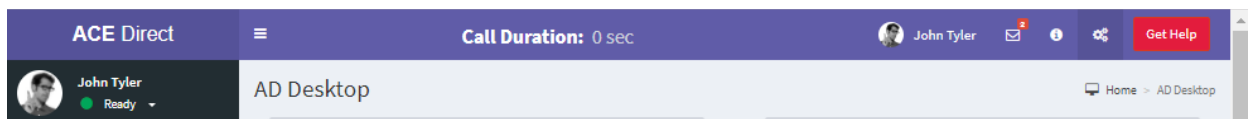


Figure 10. Screenshot of Call Duration and Assistance Button

2.5.4.2 CSR Profile

On log in, the Agent's name and picture (currently all Agents display the same profile picture) will appear in the top right corner of the Agent Desktop portal head as shown in Figure 11. The Agent has the capability to log out of the ACE Direct portal by clicking the "Sign out" button.

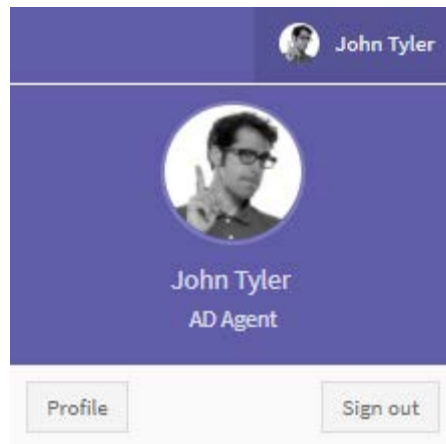


Figure 11. Screenshot of Agent Profile

2.5.5 Video Relay Service and Ticket Information

2.5.5.1 Video Relay Service Information

The VRS Information section displays the information about the consumer currently on file in the CRM system. Figure 12 shows that after the call has ended, the Agent must click on the “Save” button in the VRS Information box to return to the queue and receive new calls.

A screenshot of the 'Call Details' section in a CRM system. The 'Call Details' tab is active, and the 'VRS Info.' sub-tab is selected. The form contains several input fields: 'First Name', 'Last Name', 'Phone', 'Address', 'City/Town', 'State', 'Zip Code', and 'Email'. A red 'Save' button is located at the bottom right of the form.

Figure 12. Screenshot of VRS Information

2.5.5.2 Ticket Information

Figure 13 shows the Ticket Information section that provides the Agent with the consumer's submitted information.

The screenshot shows a web form titled "Call Details" with a sub-tab "Ticket Info." The form contains several input fields and a "Save" button. The fields are organized as follows:

Assignee	Requester	Last Updated
<input type="text"/>	<input type="text"/>	<input type="text"/>

Subject	Ticket ID
<input type="text"/>	<input type="text"/>

Problem Description

Resolution

Save

Figure 13. Screenshot of Ticket Information

2.6 Consumer Portal

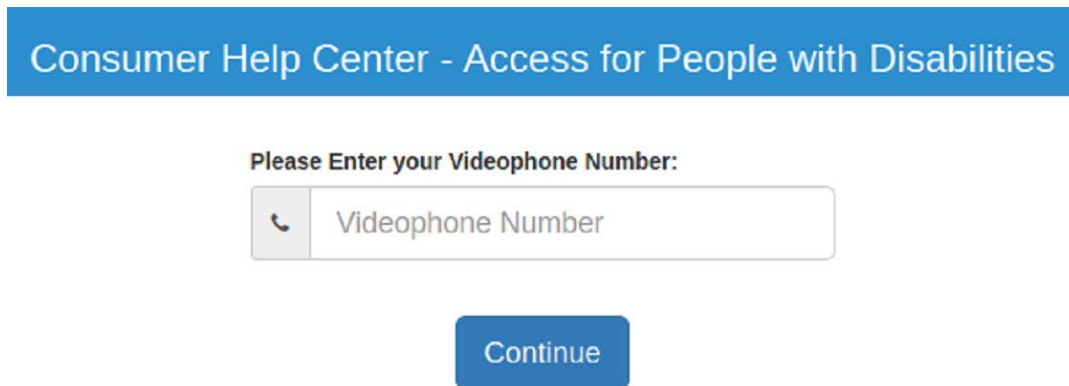
The design of the consumer portal gives consumers the option to submit information prior to a call with an Agent. The consumer uses a web form to submit information to document the complaint.

2.6.1 Submitting a Complaint

The descriptions and web forms presented in Figure 14 through Figure 19 demonstrate how to use and submit consumer complaints in the ACE Direct system.

2.6.1.1 Verify Videophone Number

Figure 14 shows the opening page for the consumer complaint portal. Consumers enter their videophone number here. The ACE Direct system validates the videophone number before allowing the consumer to proceed.



Consumer Help Center - Access for People with Disabilities

Please Enter your Videophone Number:

Videophone Number

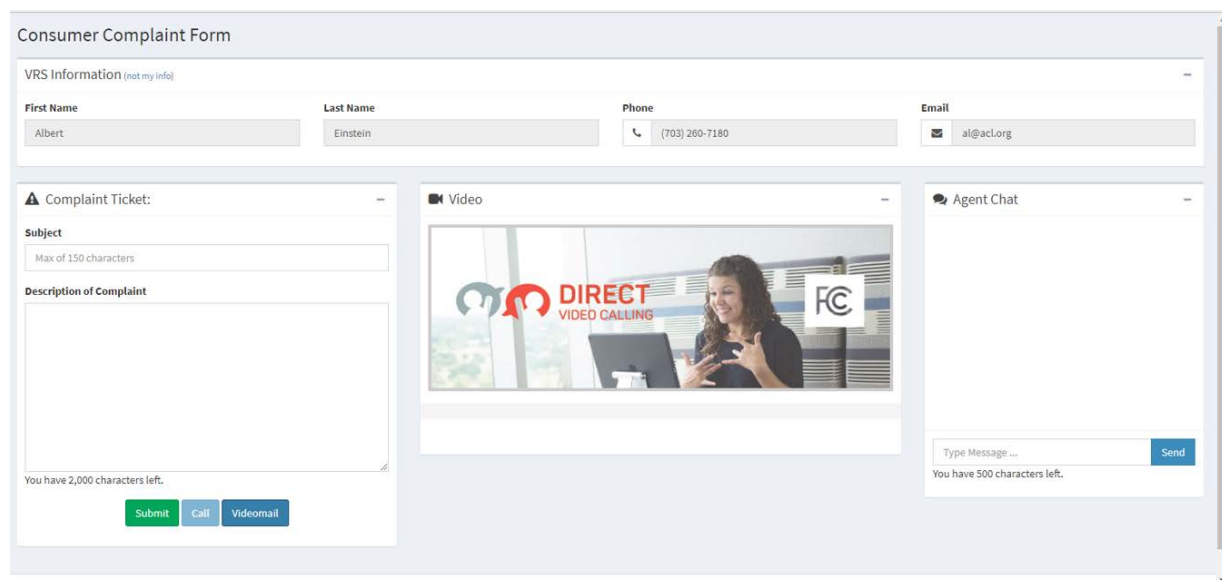
Continue

Figure 14. Screenshot of Videophone Number Entry

2.6.1.2 Consumer Complaint Form

After verifying the videophone number, the consumer complaint form is displayed. Figure 15 presents an example of the consumer complaint form.

If the customer had a prior ticket in the CRM system, the videophone number information provided on the previous form is displayed in the VRS Information section via a ticket lookup. These fields will be empty if this is the consumer's first call.



Consumer Complaint Form

VRS Information (not my info)

First Name: Albert

Last Name: Einstein

Phone: (703) 260-7180

Email: al@ac.org

Complaint Ticket:

Subject: Max of 150 characters

Description of Complaint: You have 2,000 characters left.

Submit Call Videomail

Video

Agent Chat: Type Message ... Send. You have 500 characters left.

Figure 15. Screenshot of Consumer Portal

2.6.1.3 Complaint Ticket Form

The complaint ticket section appears on the left side of the screen. The consumer enters a subject and description of the complaint to provide background information to the Agent. After populating the subject and description fields, the consumer clicks "Submit" to file the complaint. The complaint ticket is created in the ticketing system, which enables the "Call" button. This

allows the consumer to communicate with an Agent via video and chat. Figure 16 shows a screenshot of the complaint ticket form.

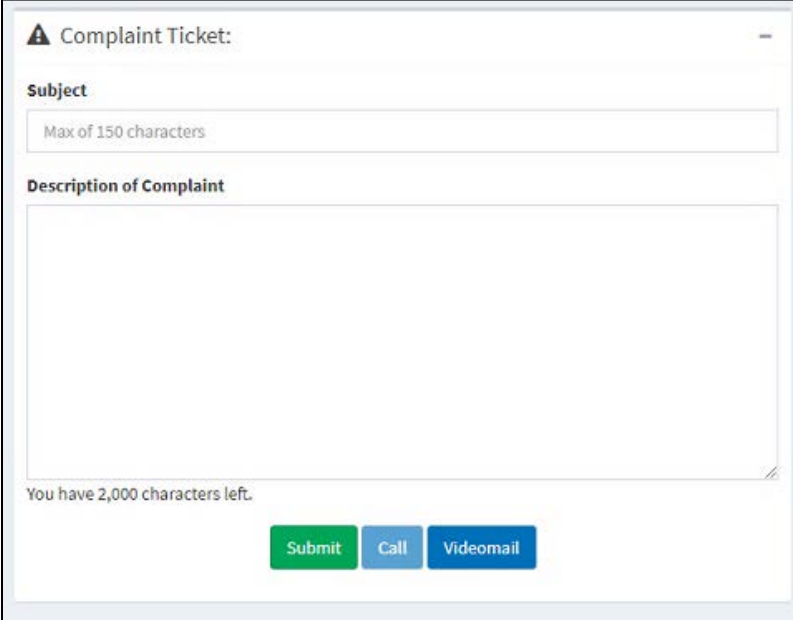
A screenshot of a web form titled "Complaint Ticket:". The form has a "Subject" field with a placeholder "Max of 150 characters". Below it is a "Description of Complaint" text area. At the bottom of the text area, it says "You have 2,000 characters left." At the bottom of the form are three buttons: "Submit" (green), "Call" (blue), and "Videomail" (blue).

Figure 16. Screenshot of Complaint Ticket Form

2.6.1.4 Video Chat

After submitting the complaint ticket and receiving a ticket number, the consumer presses the "Call" button and is connected to an available Agent. Video is the primary form of communication. As shown in Figure 17, the Agent's video is displayed in the video box in the center of the screen. During a call, the consumer has button options to mute audio, mute video, or view the Agent's video in full screen mode.



Figure 17. Screenshot of Video Chat Window

2.6.2 Real-Time Text Chat

The Agent Chat form 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. As Figure 18 shows, a maximum of 500 characters per telephone line are allowed during the agent chat.

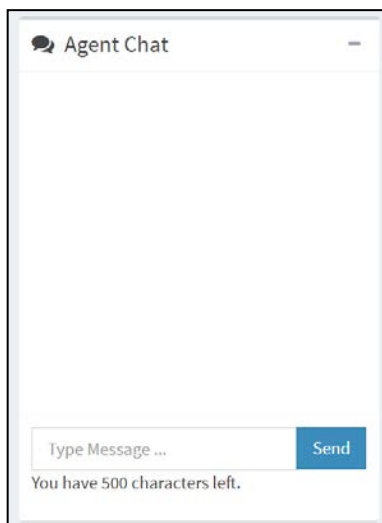


Figure 18. Screenshot of Real-Time Text Chat

When an Agent becomes available, the video chat will begin. After the consumer or the agent hangs up, the consumer will be redirected to a page of his or her choosing. For the initial configuration, the FCC.gov website is used for the redirect.

2.6.3 Leaving a Videomail

A consumer may leave a videomail during the consumer complaint by pressing the “Record” button on the screen as shown in Figure 19. This flow is for illustrative purposes and should be customized to fit your needs as a consumer.

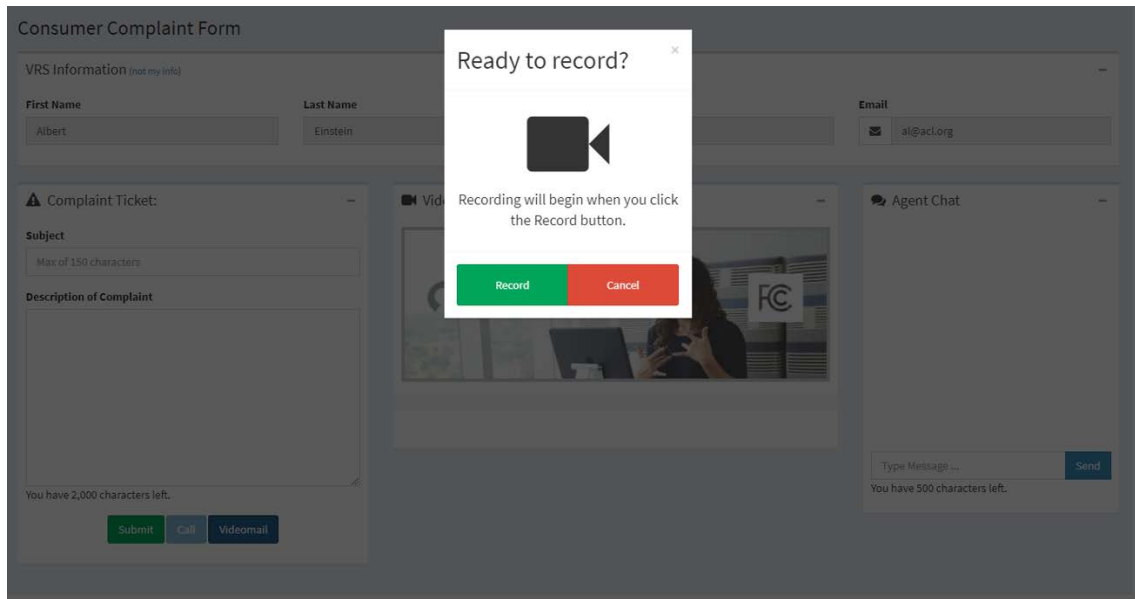


Figure 19. Screenshot of Ready to Record

Consumers see a self-view during recording. A status bar shows the remaining time for the recording (which currently defaults to 90 seconds). The maximum videomail length is a configurable parameter. Figure 20 shows the consumer portal with a videomail recording in progress.

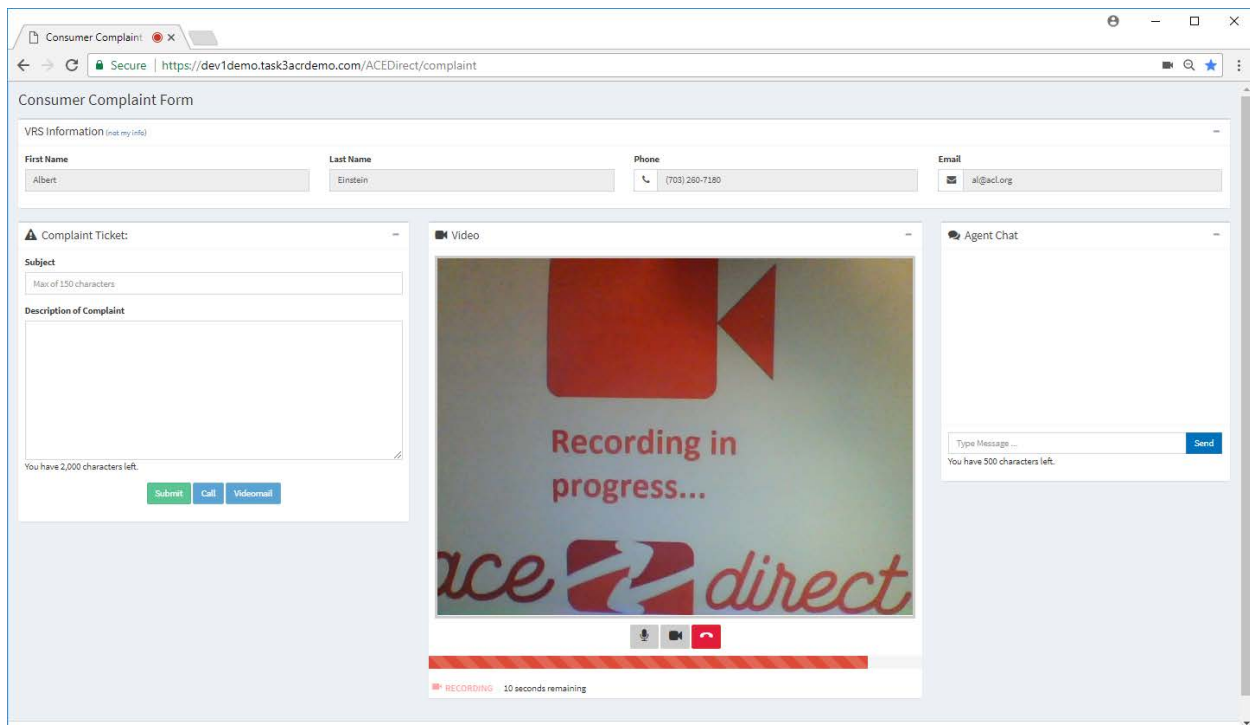


Figure 20. Screenshot of Videomail Recording in Progress

2.7 Kuando BusyLight™ Visual Ring Indicator and Agent Status

ACE Direct incorporates the Kuando BusyLight™ as an integral part of notifying call center personnel of incoming calls and the status of an Agent (such as “Away”, “Ready”, “In call,” etc.). The color of the light is configurable by the Administrator and the configuration is consistent across all Agents. An example of a configuration is depicted in subsection 2.7.2. The Kuando BusyLight™ is not delivered with the ACE Direct platform and must be purchased separately. The Kuando BusyLight™ is available in several different models and from several online vendors.

2.7.1 Agent Statuses

Each Agent has a status based on the Agent’s activity with ACE Direct, as shown in Table 3.

Table 3. Agent Status

Agent Status	Definition
Away	The Agent is not available and no calls will be directed to them.
Ready	The Agent is available to take calls from the queues.
In a Call	The Agent is currently handling a call.
Incoming Call	The Agent is receiving a call, but has not yet answered it.
Wrap Up	The Agent just finished a call, but has not yet hit “finished”. No calls can be directed to the Agent.

The status colors are shown on the Kuando BusyLight™ and in the Agent portal as shown in the image in subsection 2.7.2. Figure 22 shows the default colors. The configurations can be reset to the default values at any time by clicking “Reset to Default”.

2.7.2 Kuando BusyLight™ Status Configuration

Managers can customize the color associated with each possible Agent status through the Light Configuration page in the Management Portal. When a manager saves the form, the color is updated in real time and appears on the Agent’s Kuando BusyLight™ and in the Agent portal as shown in Figure 21. The system defaults are a set of 508-compliant colors to depict status as shown in Figure 22.

The screenshot shows the 'ACE Dashboard' with a sidebar containing 'MAIN NAVIGATION' items: 'Management Dashboard', 'CDR Dashboard', and 'Light Configuration'. The 'Light Configuration' page is active, displaying a table for configuring light status and color. The table has two columns: 'Status' and 'Color'. The rows are: 'Away:' (Yellow - solid), 'Ready:' (Green - solid), 'In a Call:' (Red - solid), 'Incoming Call:' (Red - blinking), and 'Wrap Up:' (Blue - solid). Below the table are 'Reset to Default' and 'Save' buttons. A copyright notice at the bottom reads: 'Copyright © 2016 Gov't Agency. All rights reserved.'

Status	Color
Away:	Yellow - solid
Ready:	Green - solid
In a Call:	Red - solid
Incoming Call:	Red - blinking
Wrap Up:	Blue - solid

Reset to Default

Save

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

Figure 21. Screenshot of Kuando BusyLight™ Configuration Page

Figure 22 shows the light configuration page. Using the “Reset to Default” button, the status and color selections will revert to the default settings.

The screenshot shows the 'ACE Dashboard' with the 'Light Configuration' page. The 'Status' and 'Color' columns are visible. The 'Away:' status is set to 'Off' (represented by a black circle icon). The 'Ready:' status is set to 'White - solid' (represented by a white circle icon). The 'In a Call:' status is set to 'Yellow - solid' (represented by a yellow circle icon). The 'Incoming Call:' status is set to 'White - blinking' (represented by a white circle with a dot icon). The 'Wrap Up:' status is set to 'Yellow - blinking' (represented by a yellow circle with a dot icon). Below the table are 'Reset to Default' and 'Save' buttons. A red text message at the bottom says: 'Click "Save" to submit your changes.'

Status	Color
Away:	Off
Ready:	White - solid
In a Call:	Yellow - solid
Incoming Call:	White - blinking
Wrap Up:	Yellow - blinking

Reset to Default

Save

Click "Save" to submit your changes.

Figure 22. Screenshot of Kuando BusyLight™ Default Color Scheme

2.7.3 Lightserver

The Lightserver program is a graphical user interface (GUI) for integrating the Kuando BusyLight™ with the ACE Direct platform. It provides a RESTful interface via localhost only to the ACE Direct Agent portal. It is only available locally on the Agent's desktop computer. When the Agent status changes, the Agent portal makes RESTful calls to the Lightserver. Lightserver is a standalone Java application that must execute on the desktop computer of the Agent. In addition, a Kuando BusyLight™ device must be connected to a USB port on the same computer.

The ACE Direct portal makes the initial connection to Lightserver. This occurs when the Agent navigates to the ACE Direct Agent portal. This connection enables all requests from the ACE Direct Agent portal to Lightserver. Figure 23 depicts the Lightserver GUI.

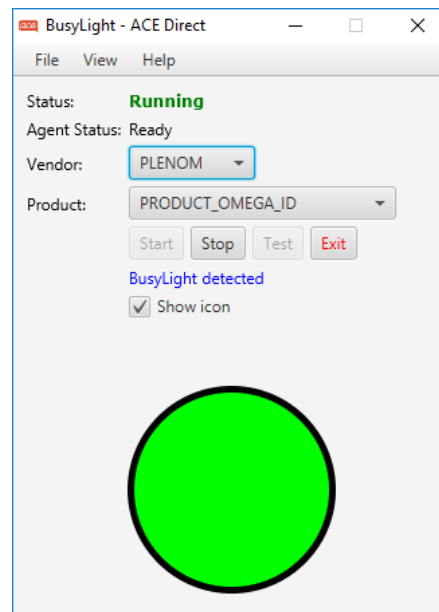


Figure 23. Lightserver GUI

At startup, the Lightserver GUI attempts to detect a connected BusyLight™ device, perform a self-test, and start its server. At this point, an ACE Direct Agent may connect to the Kuando BusyLight™ device from the ACE Direct Agent portal. The Lightserver GUI has the data elements shown in Table 4.

Table 4. Lightserver GUI Data Elements

Data Element	Description
Status	The current status of the Lightserver program (e.g., Running, Stopped, ...)
Agent Status	The status of the connected Agent (e.g., ready, away, in call, ...)
Vendor	The vendor of the light device; currently only PLENOM is supported
Product	The BusyLight™ device model
Start	Starts the local server
Stop	Stops the local server
Test	Perform a self-test of the connected BusyLight™ device
Show icon	Show or hide the graphical BusyLight™

2.8 Management Portal

The Management Portal consists of three main components: the Management Dashboard, the Call Detail Record dashboard, and the Light Configuration page. These pages present the manager with information about the operations of the call center, information about incoming calls, and the ability to customize colors associated with an Agent status on the Kuando BusyLight™.

2.8.1 Management Dashboard

The Management Dashboard, as shown in Figure 24, provides KPIs for monitoring in real time. Follow these two steps to access the Management Dashboard:

- Start a browser on a machine that can access the Management Portal server, Node.js.
- Enter the following URL “https://<hostname>/ManagementPortal” where <hostname> is the host name of the Management Portal server.

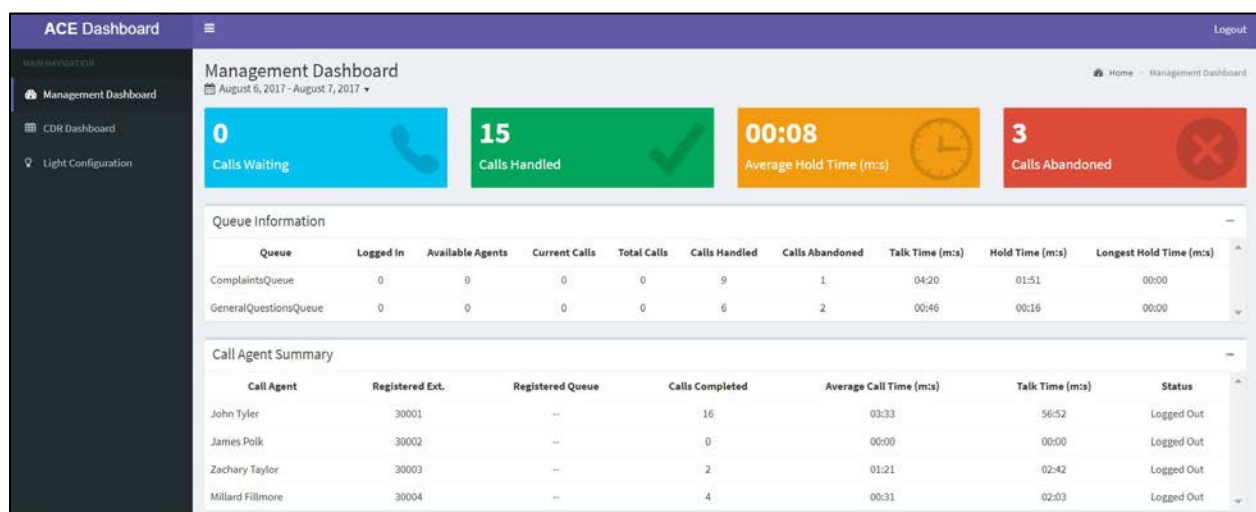
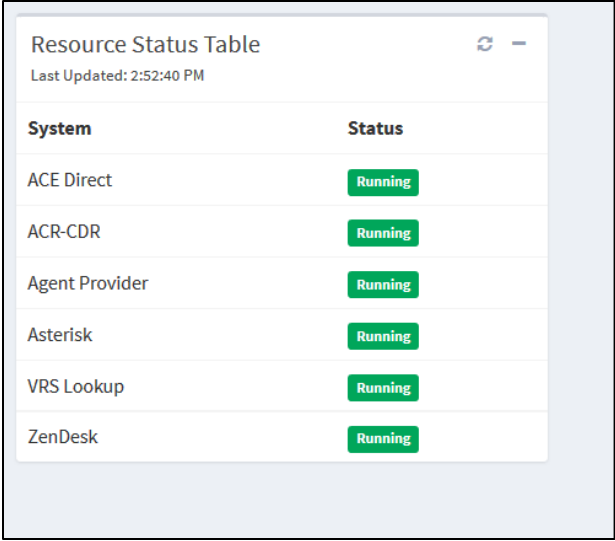


Figure 24. Screenshot of Management Dashboard

Key Performance Indicator Types

The ACE Direct Management Dashboard presents three types of KPIs:

1. **Summary Data** – There are two queues in ACE Direct: ComplaintsQueue and GeneralQuestionQueue. The following KPIs are a summary over both queues combined.
 - a. Calls Waiting – Number of calls waiting in all queues.
 - b. Calls Handled – Number of calls completed in all queues.
 - c. Average Hold Time (minutes:seconds) – Average call holding time in all queues.
 - d. Calls Abandoned – Number of calls not answered in all queues.
2. **Queue-related KPIs** – The following KPIs are displayed per queue (ComplaintsQueue and GeneralQuestionQueue):
 - a. Logged In – Number of Agents currently logged into the system.
 - b. Available Agents – Number of Agents currently in a ready state.
 - c. Current Calls – Number of calls currently in progress.
 - d. Total Calls – Total number of calls made.
 - 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 (minutes:seconds).
 - h. Hold Time – Average hold time (minutes:seconds).
 - i. Longest Hold Time – The longest hold (minutes:seconds).
3. **Agent-related KPIs** – The following KPIs are displayed per Agent. 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. Status – Logged Off, Ready, Away, or In-Call.
4. **Resource Status KPIs** – Figure 25 shows the resource status KPIs:
 - a. Resources – A list of services required for ACE Direct to properly operate. (ACE Direct, ACR-CDR, Agent Provider, Asterisk, VRS Lookup and Zendesk. Zendesk, Asterisk, CDR, Agent Provider and VRS Lookup).
 - b. Status – The current status of each service (Running or Unavailable).



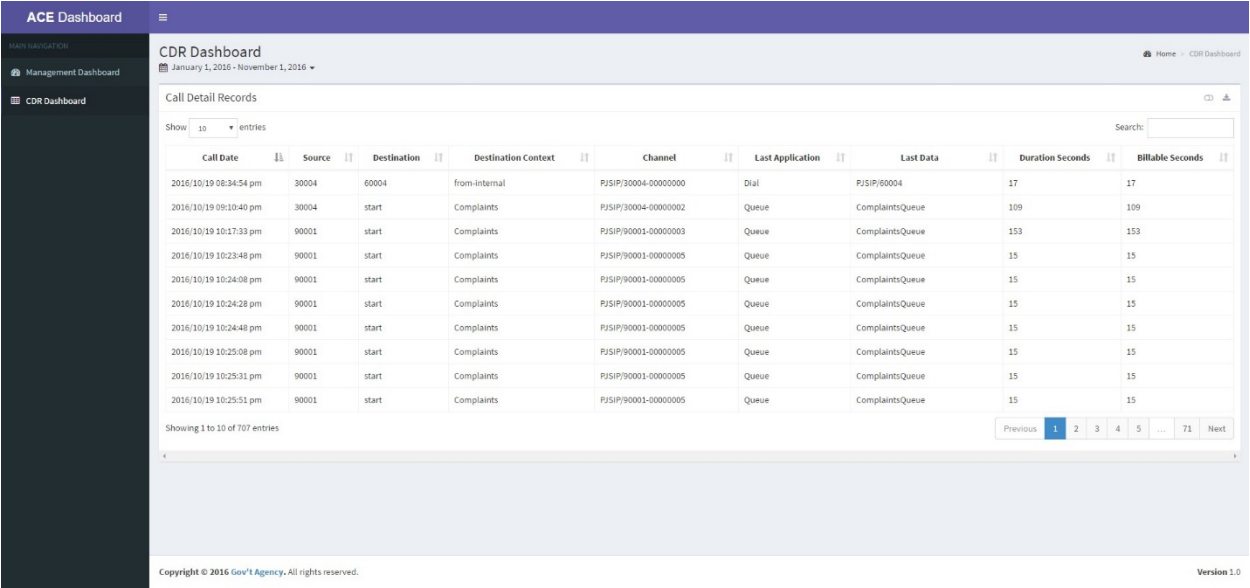
Resource Status Table
Last Updated: 2:52:40 PM

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

Figure 25. Screenshot of Resource Status

2.8.2 Call Detail Record Dashboard

Asterisk generates an Agent event when a call is completed. A CDR contains metadata that describes each call, such as the time of the call, call source, and the call destination. The CDR dashboard provides a means of auditing call activity, tracking a call Agent's activity, and creating a report of both incoming and outgoing calls. The ACE Direct Call Detail Record Dashboard, as shown in Figure 26, provides a method to view and export the Asterisk CDRs stored in the MySQL database.



ACE Dashboard

MAIN NAVIGATION

- Management Dashboard
- CDR Dashboard

CDR Dashboard

January 1, 2016 - November 1, 2016

Home - CDR Dashboard

Call Detail Records

Show 10 entries

Search:

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/30004-00000000	Dial	PJSIP/60004	17	17
2016/10/19 09:10:40 pm	30004	start	Complaints	PJSIP/30004-00000002	Queue	ComplaintsQueue	109	109
2016/10/19 10:17:33 pm	90001	start	Complaints	PJSIP/90001-00000003	Queue	ComplaintsQueue	153	153
2016/10/19 10:23:48 pm	90001	start	Complaints	PJSIP/90001-00000005	Queue	ComplaintsQueue	15	15
2016/10/19 10:24:08 pm	90001	start	Complaints	PJSIP/90001-00000005	Queue	ComplaintsQueue	15	15
2016/10/19 10:24:28 pm	90001	start	Complaints	PJSIP/90001-00000005	Queue	ComplaintsQueue	15	15
2016/10/19 10:24:48 pm	90001	start	Complaints	PJSIP/90001-00000005	Queue	ComplaintsQueue	15	15
2016/10/19 10:25:08 pm	90001	start	Complaints	PJSIP/90001-00000005	Queue	ComplaintsQueue	15	15
2016/10/19 10:25:31 pm	90001	start	Complaints	PJSIP/90001-00000005	Queue	ComplaintsQueue	15	15
2016/10/19 10:25:51 pm	90001	start	Complaints	PJSIP/90001-00000005	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 26. Screenshot of Call Detail Record

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

- **Select Date Range** – The consumer 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 “Last 7 Days”.
- **Sort Column** – The consumer can sort on any column by clicking the sort icon located next to each column name. To multi-sort columns, the consumer 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 5 presents the column definitions in the CDR table.

Table 5. 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 consumer 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.

Display Name	Database Column	Description
Billable Seconds	Billsec	The number of seconds between the answer and end times for the call. This field is set automatically and is read-only.
Disposition	Disposition	An indication of what happened to the call. This may be NO ANSWER, FAILED, BUSY, ANSWERED, or UNKNOWN.
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

2.8.3 BusyLight™ Configuration

For information on how to customize the Agent status colors on the Kuando BusyLight™ in the Agent Portal, please refer to subsections 2.7.1 and 2.7.2

2.9 Identity and Access Management

ACE Direct integrates with ForgeRock OpenAM, an open source identity and access management enterprise solution that provides users management and access control capabilities. The ForgeRock OpenAM and embedded OpenDJ are the only ForgeRock packages used in ACE Direct.

ACE Direct consumer, such as agent, manager, or supervisor, must be provisioned in OpenAM before the consumer accesses the Agent, Management, or CDR portals. Provisioning instructions to create new roles and consumers can be found in Provision.md under <https://github.com/FCC/ACEDirect> in the 'iam' section.

There are two ways a consumer authenticates with the Agent or Management portals:

1. **ACD Direct URL** – Access ACE Direct directly by entering the ACE Direct URL. If the consumer is already authenticated with ACE Direct and the session is still valid, the Agent Portal will be displayed. Otherwise, the consumer is re-directed to the Main Login page to authenticate. The Agent Portal is displayed after a successful login.

2. **An OpenAM login URL (Main login page)** – This is the main login page for a consumer to authenticate, reset the password, or create a new account. The consumer selects the application to access as listed on the dashboard after logging in. The following subsections describe the identity and access management capabilities in detail.

2.9.1 Login Screen

Figure 27 shows the login screen.

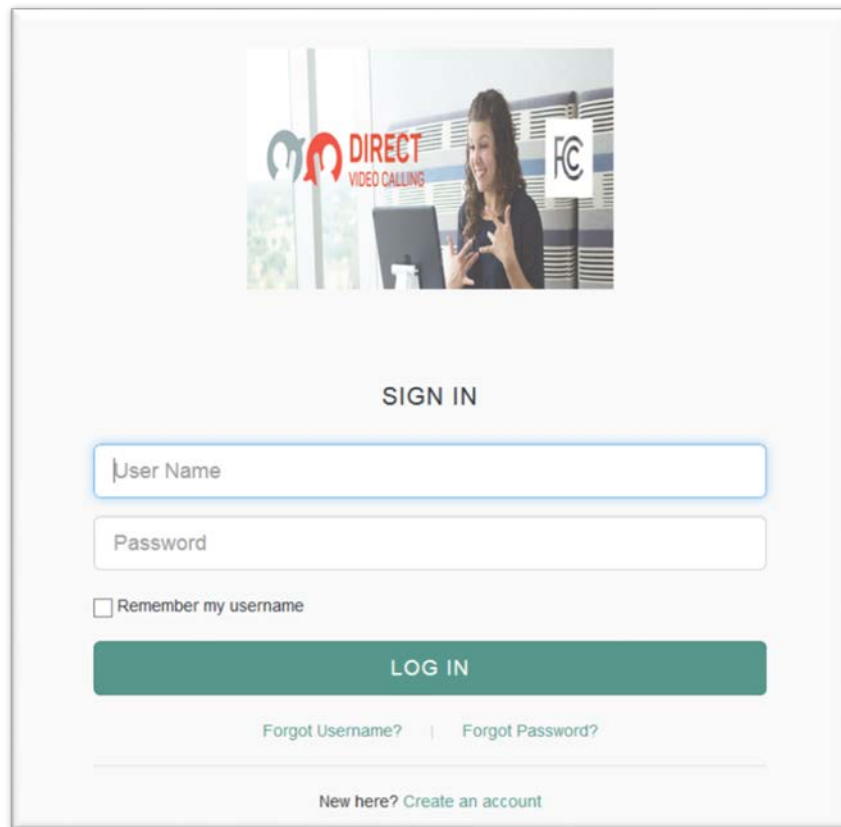


Figure 27. Screenshot of Login Screen

The page consists of the following elements:

1. Login – Consumer provides login credentials (username, password) to authenticate to ACE Direct.
2. Forgot Username – Redirects the consumer to the “Forgot Username” page to retrieve username.
3. Forgot Password – Redirects the consumer to the “Forgot Password” page to reset password.
4. Self Registration – Allows the consumer to self-register and create an account.

2.9.1.1 Forgot Username

When the consumer clicks the “Forgot Username” link to retrieve a forgotten username, the following screen is displayed as shown in Figure 28.

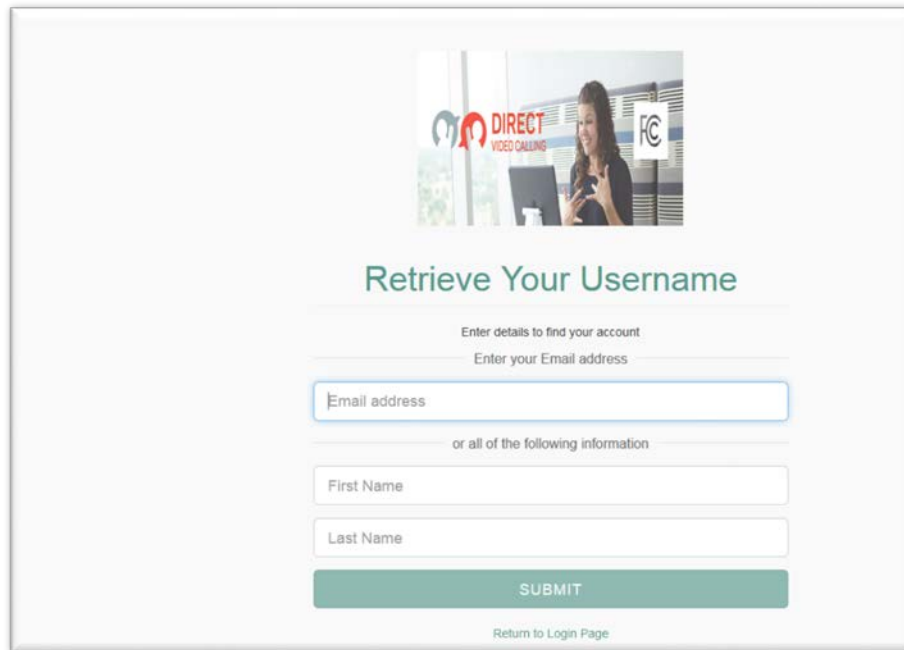
The screenshot shows a web form titled "Retrieve Your Username" in green text. At the top, there is a banner image featuring a woman on a video call, with the "DIRECT VIDEO CALLING" logo and the FCC logo. Below the banner, the text "Enter details to find your account" is displayed. The form includes a text input field labeled "Enter your Email address" with a placeholder "Email address". Below this, it says "or all of the following information". There are two more text input fields, one labeled "First Name" and one labeled "Last Name". At the bottom of the form is a green "SUBMIT" button. A link "Return to Login Page" is located at the very bottom of the form area.

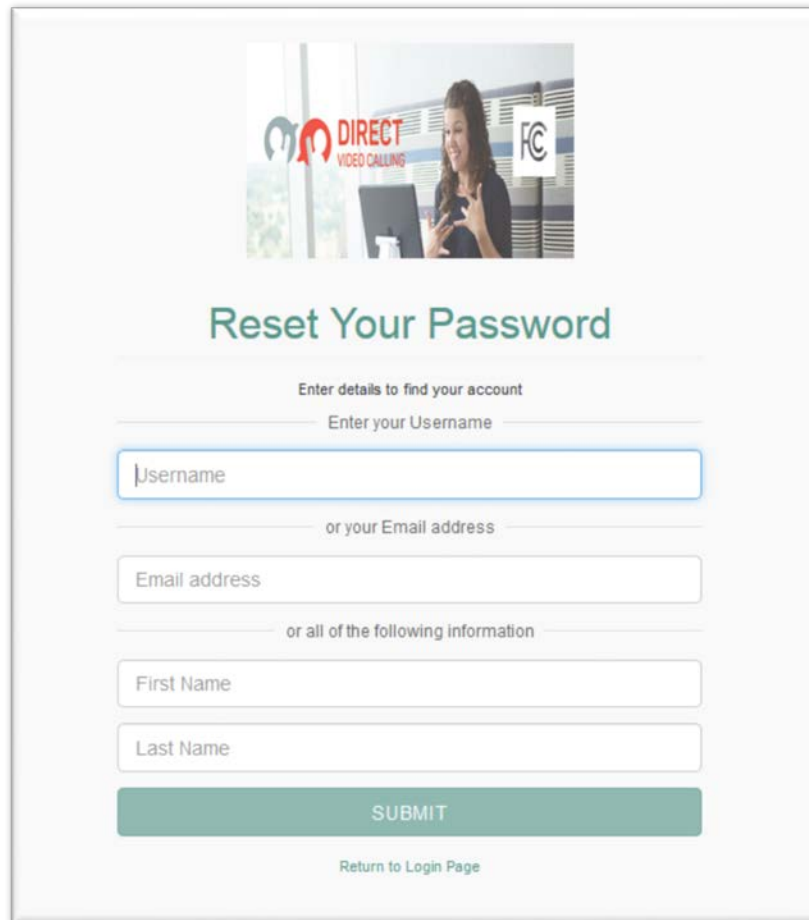
Figure 28. Screenshot of Retrieve Username

A consumer may retrieve their username via:

- Email – If the consumer entered his/her email on the reset username screen.
- Answering security questions – Consumer is prompted with security questions after he/she submits First Name/Last name on the retrieve username screen. The consumer's dashboard is displayed after the security questions are answered correctly.

2.9.1.2 Forgot Password

Figure 29 shows the screen displayed when the consumer clicks the “Forgot Password” link to reset password.



Reset Your Password

Enter details to find your account

Enter your Username

Username

or your Email address

Email address

or all of the following information

First Name

Last Name

SUBMIT

[Return to Login Page](#)

Figure 29. Screenshot of Reset Password

Consumers may reset the password via:

- Email – If the consumer entered his/her email on the reset password screen.
- Answering security questions – Consumer is prompted with security questions after he/she submits username or First Name/Last name on the reset password screen. The consumer's dashboard is displayed after the security questions are answered correctly.

2.9.1.3 Self Registration

An Agent can self register and create an account; however, the account is not activated until the ACE Direct Administrator activates the consumer account. Figure 30 shows the screen displayed when a consumer clicks the “Create an account” link on the login page.

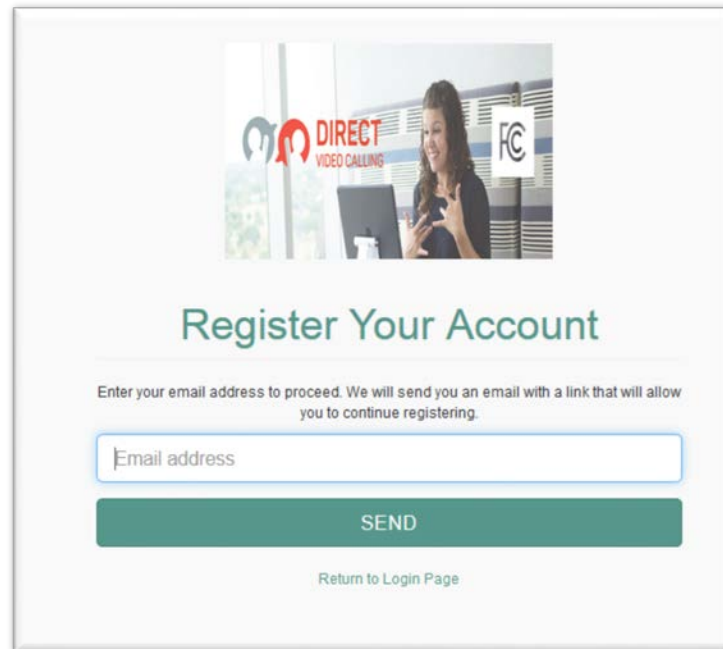


Figure 30. Screenshot of Register Your Account

An email will be sent to the Agent's email address to proceed to create an account.

2.9.2 Consumer Dashboard

Figure 31 shows the screen displayed when the consumer is authenticated with the Main login page.

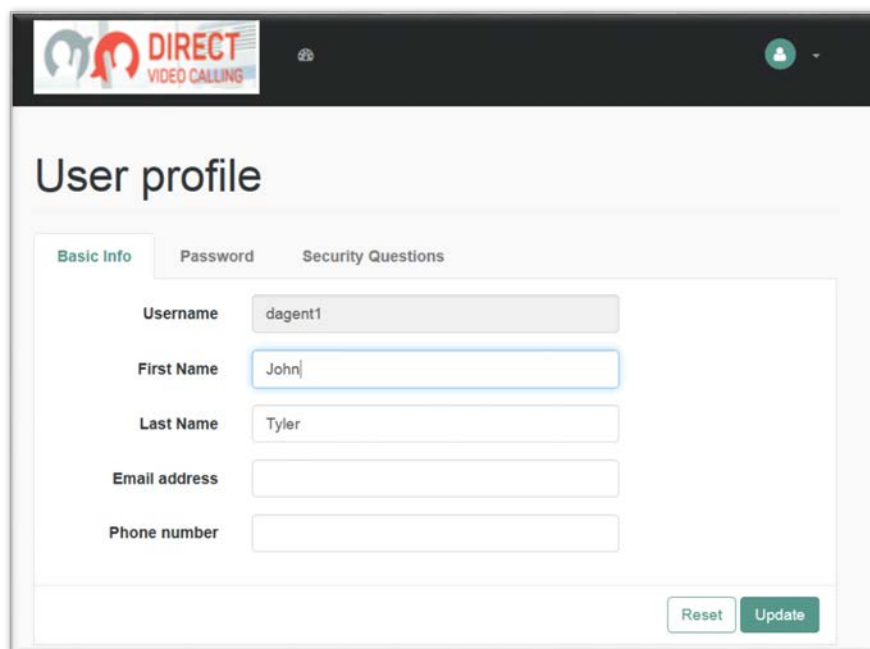


Figure 31. Screenshot of Consumer Desktop

The page consists of the following sections:

- Consumer profile
 - Basic Info
 - Password
 - Security Questions
- Dashboard
- Logout

2.9.2.1 Consumer Profile

2.9.2.1.1 Basic Info

Figure 32 shows the Basic Info tab on the Consumer Profile screen.

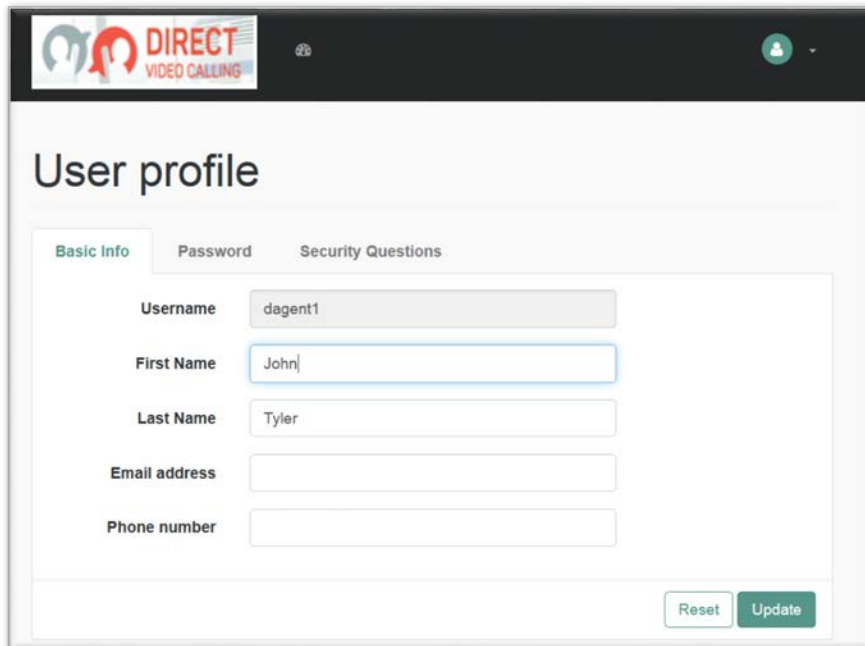
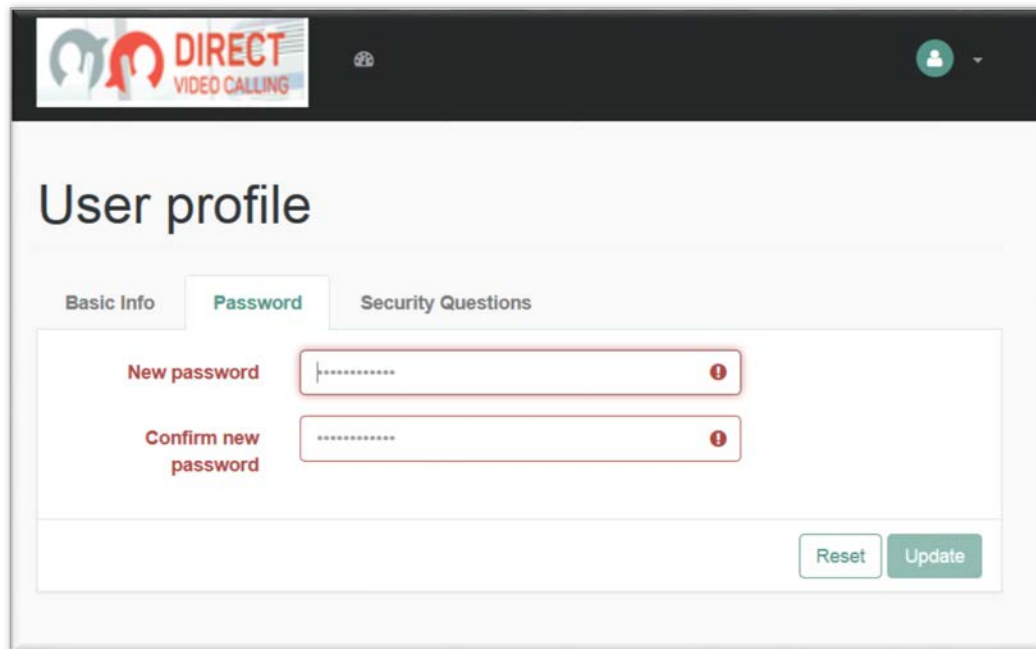
The screenshot shows a web interface for a 'User profile'. At the top, there is a dark header bar with the 'DIRECT VIDEO CALLING' logo on the left and a user profile icon on the right. Below the header, the main content area is titled 'User profile'. Under this title, there are three tabs: 'Basic Info' (which is selected and highlighted in green), 'Password', and 'Security Questions'. The 'Basic Info' tab contains several input fields: 'Username' (with the value 'dagent1'), 'First Name' (with the value 'John'), 'Last Name' (with the value 'Tyler'), 'Email address', and 'Phone number'. At the bottom right of the form, there are two buttons: 'Reset' and 'Update'.

Figure 32. Screenshot of Consumer Basic Info

The consumer views or updates his/her first name, last name, email address or phone number on this page, and click “Update” to save the changes.

2.9.2.1.2 Password

Figure 33 shows the Password tab on the Consumer profile screen. This screen allows the consumer to update their password.



The screenshot shows the 'User profile' page with three tabs: 'Basic Info', 'Password', and 'Security Questions'. The 'Password' tab is active. It contains two input fields: 'New password' and 'Confirm new password'. Both fields are currently empty and have a red border with a red exclamation mark icon, indicating a validation error. At the bottom right of the form are two buttons: 'Reset' and 'Update'.

Figure 33. Screenshot of Update Password Tab

The consumer clicks on the Password tab to change password, and clicks “Update” to save changes.

2.9.2.1.3 Security Questions

Figure 34 shows the Security Questions tab on the Consumer profile screen. A consumer may retrieve his/her password via email by answering security questions. This screen allows the consumer to add security questions to retrieve a forgotten password by email.




Figure 34. Screenshot of Add/Update Security Questions

The consumer must provide at least two security questions for this purpose. To accomplish this, the consumer must:

- Select the “Security Questions” tab.
- Select a pre-defined security question or create a new security question.
- Provide an answer to the question.
- Repeat for at least one additional security question.
- Click “Update” to save the changes.

2.9.2.2 Dashboard

The dashboard is an area where the consumer views the list of applications he/she is approved to access. Figure 35 shows an example of an Agent’s dashboard and a list of approved applications under the My Application Drop Down.

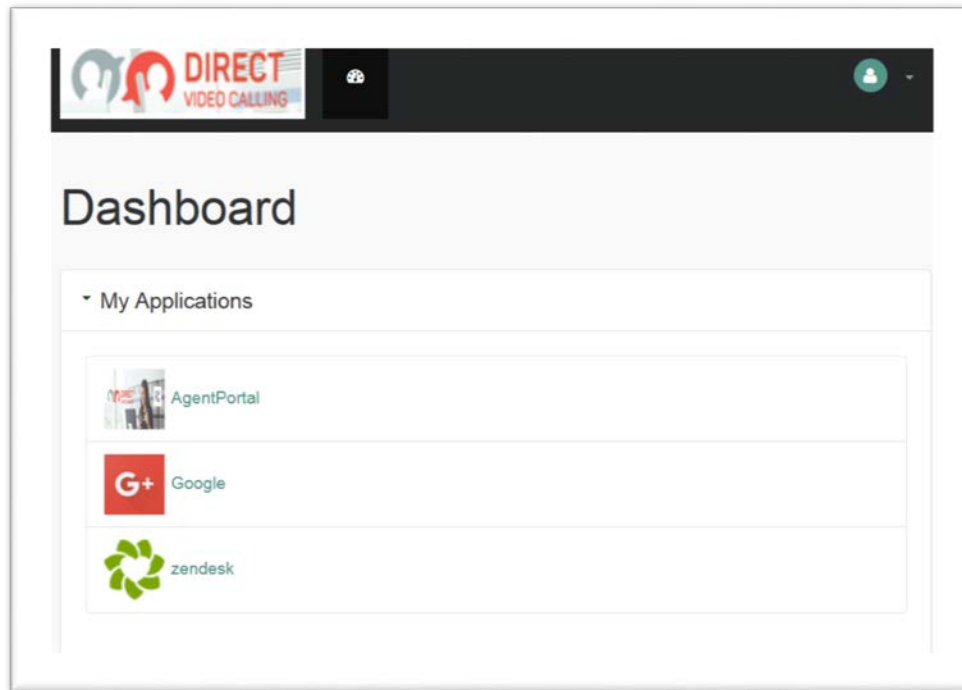


Figure 35. Screenshot of Consumer Dashboard

2.9.2.3 Logout

To logout from the Main Login page and all open sessions to Agent portal and/or Management portal, select "LOG OUT" from the dropdown list on the top right corner of the page as shown in Figure 36.

USER PROFILE

SIGNED IN AS
JOHN TYLER

PROFILE
LOG OUT

Basic Info Password Security Questions

Username dagent1

First Name John

Last Name Tyler

Email address

Phone number

Reset Update

Figure 36. Screenshot of Logout Screen

2.10 NGINX

NGINX is an open source HTTP and reverse proxy server. For ACE Direct, only the reverse proxy component is used. A reverse proxy allows ACE Direct to hide internal topology (ports and script names) from users.

Figure 37 shows the mapping between public-facing and internal URLs performed by NGINX. The left side of the figure shows the URLs available to the public via HTTPS. Each URL maps to another path and port internally on the Node.js server. In this case, NGINX is mapping port 443 on the public-facing interface to ports 8005 and 8081 on the internal side.

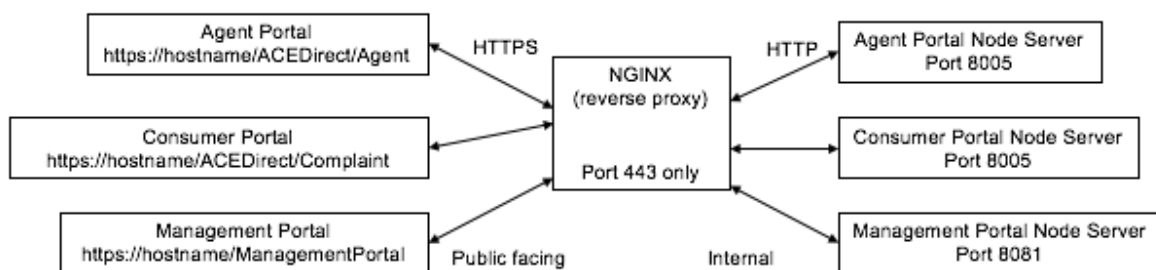


Figure 37. Internal and External Paths with NGINX

In ACE Direct, NGINX is configured to work with OpenAM, directing incoming connections to the correct location based on URL and authentication level. Figure 38 shows the relationship between NGINX and OpenAM and the mapping between public-facing URLs (to the left or outside of the firewall) and the internal processes.

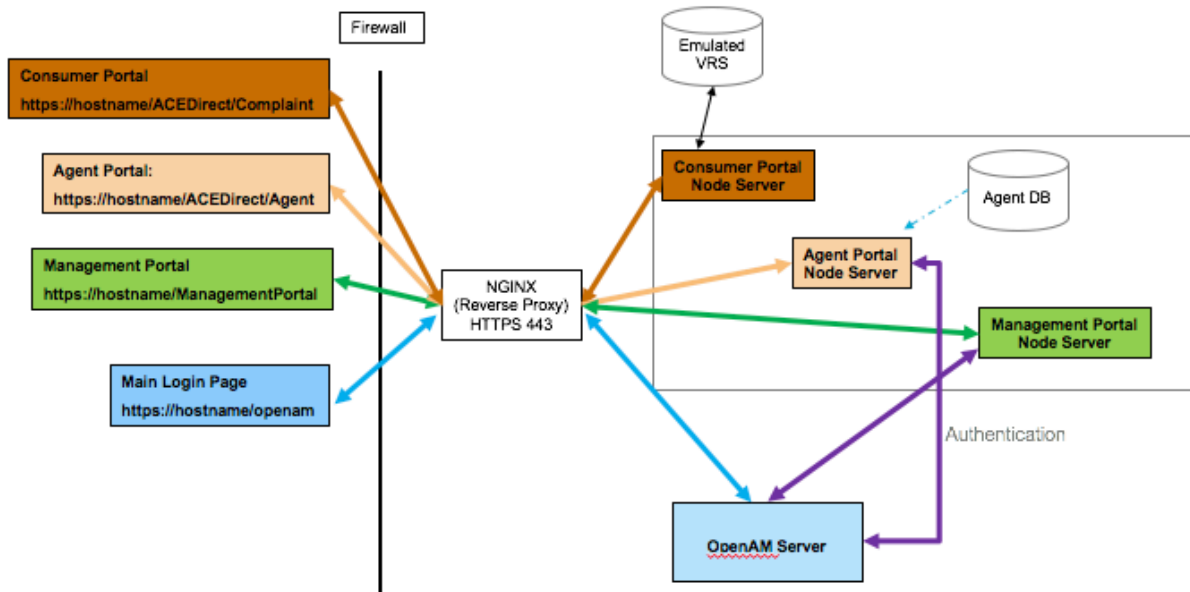


Figure 38. NGINX and OpenAM Integration

2.11 Redis

Redis is an open source, in-memory key-value data store. In ACE Direct, Redis is used to store state data that were previously stored in memory in the Node.js server. Some of the data stored in Redis include gent status (active or away), maps of agent extensions, and agent information.

Redis offers advantages over in-memory storage. For example, Redis allows several applications to share the same data. This ensures that all applications have the same view of system state. Another advantage is that Redis supports different data formats, which means it can store more than just typical key-value pairs. The most widely used data types are strings; however, Redis also supports lists, sets, sorted sets, and hashes. Because Redis can also write data to disk, it can maintain state after a system restart.

3. Installation and Configuration

This section presents guidance for the installation and configuration of the ACE Direct components except for OpenAM, the ESB (Apache ServiceMix), MySQL databases, and the Kuando BusyLight™. To reproduce this version of the platform, review the README.md file in the autoinstall folder at <https://github.com/FCC/ACEDirect>.

3.1 Secure Sockets Layer Certificate

The applications within ACE Direct use the Hypertext Transport Protocol Secure (HTTPS) to provide security during web transmissions. To prevent major web browsers from potentially flagging the application as untrusted, it is recommended to acquire and implement an SSL certificate from a trusted certificate authority (CA), such as LetsEncrypt (<https://letsencrypt.org/>). After obtaining a certificate, follow the instructions in the respective README files in the Node and Asterisk GitHub repositories to install the SSL certificate.

3.2 Asterisk Installation and Configuration Script

An automated installation script has been developed to assist in the deployment and configuration of the Asterisk server. The previous manual installation procedures have been deprecated and are no longer supported. The Asterisk install script will install and configure Asterisk for ACE Direct on a CentOS 7 server. This script, which is available at <https://github.com/FCC/ACEDirect> in the ‘scripts’ section of the Asterisk repository, 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 Asterisk configuration files from the Git repository
- Start Asterisk

It is necessary to satisfy several prerequisites to the script before installing Asterisk:

- A STUN server is available. (Although you may use Google’s free STUN server, 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 both a public and private IPv4 address associated with it.

Review the README.md file in the Asterisk repository before running the script.

3.2.1 How It Works

The Asterisk PBX system relies on the configuration of three key files: pjsip.conf, extensions.conf, and http.conf. The pjsip.conf file contains the connection endpoints and parameters, and the extensions.conf file contains the syntax and programming for the extensions

assigned to those endpoint connections. The `http.conf` file is the server configuration for Asterisk and also defines the certificates Asterisk uses for secure communications.

3.2.1.1 Dial Plan Configuration

The dial plan is defined by the `extensions.conf` and `pjsip.conf` files in the `/etc/asterisk` directory, which work 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 6 shows an example of common endpoint devices, the associated extension and the required codecs for the sample dial plan configuration in this guide.

Table 6. Example Asterisk Endpoint Extensions

Extension	Purpose	Device	Codec
30001	ACE Direct Agent 1	softphone	h264/vp8/ulaw
30002	ACE Direct agent 2	softphone	h264/vp8/ulaw
30003	ACE Direct agent 3	softphone	h264/vp8/ulaw
30004	ACE Direct agent 4	softphone	h264/vp8/ulaw

As displayed in Table 6, the extension is the number assigned to the endpoint. The ability of that endpoint to connect to the Asterisk instance is configured in the `pjsip.conf` file and the ability to dial to another endpoint is configured within the `extensions.conf` file. The device can be any phone capable of connecting to the Asterisk instance—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. The codec is also configured in the `pjsip.conf` file.

3.2.1.2 Extensions.conf Configuration File

The `extensions.conf` file is the configuration of your PBX. This file defines 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 consumer wants to dial their voicemail. In `extensions.conf`, that configuration must state that the consumer's phone number is to load the voicemail application, which is 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 for immediate placement into the Voicemail application.

For specific phone numbers to route, as well as number schemes for unknown numbers, use the wildcard “X”. In the following example as shown in the context “[from-internal]” of the `extensions.conf` file, all extensions ended with `6xxx` are configured. 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. As shown in this example, 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()
```

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.

3.2.1.3 Pjsip.conf Configuration File

The `pjsip.conf` file defines the endpoint connection parameters. Key elements must be configured in this file. The transport protocol settings and extension must be identified. The extension profile serves as its SIP identity and phone extension. In addition, the file must define the authentication method for this extension. PJSIP does not require the specification of a protocol and will dynamically select the best available option to use. The extension profile must specify the configuration settings for the endpoint, the network connection, the allowed codecs, and NAT traversal settings. The extension profile may also require that particular settings be specified to not be used or disabled. It is highly recommended that you visit the Digium Asterisk website to thoroughly understand the function of the `PJSIP.conf` file and its associated settings.

The Asterisk PBX uses the `pjsip.conf` file to load the connection points from which your endpoint devices will acquire their information and connect. This means that this file is responsible for (1) the endpoint connection, (2) handling the contact that connects to that endpoint, (3) routing the call, and (4) 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** – An endpoint to the `pjpsip.conf` file 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 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. Use the pre-defined profiles in the Asterisk repository, because they are already configured specifically for use with ACE Direct.

3.2.2 Web Secure Sockets

This consumer portal uses a technology called WebRTC. Asterisk must use web sockets and, for most browsers, secure web sockets to successfully communicate with WebRTC. It is highly recommended that Asterisk use a certificate from a valid certificate authority when using WebRTC.

3.2.3 Sample Configuration Files

3.2.3.1 Pjsip.conf Configuration

```
-----top of pjpjsip.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
rtsp_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 pjpjsip.conf file-----
-----

```

3.2.3.2 WebRTC Sample Configuration

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

[endpoint-webrtc](!)

```

type=endpoint
transport=transport-wss
context=from-internal
disallow=all
allow=ulaw
allow=h264

```

```
allow=vp8
allow=t140
force_avp=yes
use_avpf=yes
media_encryption=dtls
dtls_verify=fingerprint
dtls_fingerprint=SHA-1
dtmf_mode=auto
dtls_rekey=0
dtls_cert_file=/etc/asterisk/keys/asterisk.pem
dtls_ca_file=/etc/asterisk/keys/ca.crt
dtls_setup=actpass
ice_support=yes
media_use_received_transport=yes
rtp_symmetric=yes
force_rport=yes
rewrite_contact=yes
message_context=internal-im
rtcp_mux=yes
```

3.3 Node.js

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 input/output (I/O) model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js servers provide the functionality of ACE Direct. More specifically, the servers host the Agent, Consumer, Management, CDR, and Virtual Agent Videomail portals. Node.js also provides Agent Data and Provider Data RESTful APIs to ACE Direct.

Examples of these RESTful services are VRS lookup and Agent 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:

- Operating System: CentOS 7.x
- Node.js: Version 7.2.1

3.3.1 Node.js Installation

The manual installation procedures for the Node.js components of ACE Direct have been deprecated and are no longer supported. To install the Node.js components, follow the README.md file in the autoinstall folder located at <https://github.com/FCC/ACEDirect>.

3.4 Management Portal

The ACE Direct Management Portal provides a number of key performance indicators for real-time monitoring by the call center manager. This information may be used to improve user experience, increase user satisfaction, and monitor overall call center performance.

3.4.1 Management Portal Installation

The CAMH team developed a Node.js-based management dashboard to show system status. To install and launch the service, follow the autoinstall instructions at <https://github.com/FCC/ACEDirect>.

3.4.2 Management Portal 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 8081.
- AsteriskAD/sip/host – Fully-qualified domain name of the associated Asterisk server (e.g., mySIP.example.com).
- dashboard/queuesAD – Queue names configured for ACE Direct. Currently, ACE Direct has two queues—GeneralQuestionsQueue and ComplaintsQueue.
- Resource section – URL of the monitored resources (Zendesk, VRS lookup, Agent Provider, ACE Direct, CDR).

If necessary, update the following parameters in /etc/asterisk/queues.conf:

```
[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 Log Files

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

3.4.4 Management Dashboard

3.4.4.1 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.ejs) is hosted on the Node.js server (server-db.js). The following subsections describe the operational data flow.

3.4.4.1.1 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.

3.4.4.1.2 Client Side

- **dashboard.ejs**– Controls the UI for the Mangement Portal Dashboard.
- **dashboard.js** – Uses JavaScript data structures to store information to be displayed on the dashboard.ejs page.

3.4.4.2 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 7 shows the AMI actions and descriptions.

Table 7. 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

AMI Action	Description	Syntax	Arguments	Triggered Events
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 8.

Table 8. Asterisk AMI Events and Descriptions

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"> – consumer – 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

3.4.5 Call Detail Record Dashboard

The Asterisk PBX system is capable of collecting and storing call data for each call that traverses the system. These data records, referred to as CDRs, are collected and stored for use by business intelligence (BI) tools, statistical analysis, and even compensation and reimbursement. The information collected is invaluable and can be retrieved and stored using various methods.

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 also supports different database connections. One method is to use 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 used. For example, MySQL, MariaDB, and others are all viable options. CAMH elected to use a MySQL database server with Asterisk connecting through an ODBC connection.

3.4.5.1 MySQL Installation

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

```
sudo yum install unixODBC unixODBC-devel libtool-ltdl libtool-ltdl-devel mysql-connector-odbc
wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
sudo rpm -ivh mysql-community-release-el7-5.noarch.rpm
sudo yum install mysql-server
sudo systemctl start mysqld
sudo yum update
```

Once the packages are installed, it is necessary that Asterisk recognizes these new packages. To do so, migrate to the Asterisk directory, which in our build is 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 finishes successfully, you can proceed to 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
```

3.4.5.2 Database Configuration

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

3.4.5.3 Asterisk Integration

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

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

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

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

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

You should see a message of Connected! returned.

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

Now you will need 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. The dsn variable will point to the dsn identified:

```
vi /etc/asterisk/res_odbc.conf

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

Next, edit /etc/asterisk/cdr_odbc.conf to include the following:

```
vi /etc/asterisk/cdr_odbc.conf

[global]
dsn=asterisk
loguniqueid=yes
table=bit_cdr
```

Note: the dsn variable will point to the DSN identified in res_odbc.conf, NOT /etc/odbc.ini

Next, Edit /etc/ cdr_adaptive_odbc.conf to include the following:

```
vi /etc/asterisk/cdr_adaptive_odbc.conf
[asteriskcdr]
connection=asterisk
table=cdr
alias start=calldate
```

Next, ensure CDR writing is enabled in cdr_manager.conf:

```
vi /etc/asterisk/cdr_manager.conf
[general]
enabled = yes
```

Finally, restart Asterisk:

```
sudo service asterisk restart
```

3.4.5.4 Node.js Integration

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 middleman 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 Agent / Agent Database (Provider)

The ACE Direct uses a database to store Agent information and to verify Agent identity when an Agent logs into the system.

To host the agent data, a MySQL database server is required. A RESTful API developed using Node.js provides access to the data for Agent verification. For this effort, CAMH built the Node.js server with the following software versions:

- Linux: CentOS 7.x
- Node.js: v7.2.1

3.5.1 MySQL Database Server Configuration

The CAMH team developed a MySQL database containing several tables to store the agent-related data. Create a database named agentdb_new and use the following MySQL script to create the tables that will store the agent-related data:

```
-- MySQL dump 10.13  Distrib 5.5.47, for debian-linux-gnu (x86_64)
--
-- Host: localhost      Database: agentdb_new
--
-- 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 `agent_data`
--

DROP TABLE IF EXISTS `agent_data`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `agent_data` (
  `agent_id` int(10) NOT NULL AUTO_INCREMENT,
  `username` varchar(10) NOT NULL,
  `password` varchar(30) NOT NULL,
  `first_name` varchar(20) NOT NULL,
  `last_name` varchar(20) NOT NULL,
  `role` varchar(50) NOT NULL,
  `phone` varchar(12) NOT NULL,
  `email` varchar(40) NOT NULL,
  `organization` varchar(50) NOT NULL,
  `is_approved` tinyint(1) NOT NULL,
  `is_active` tinyint(1) NOT NULL,
  `extension_id` int(10) DEFAULT NULL,
  `queue_id` int(10) DEFAULT NULL,
  `queue2_id` int(10) DEFAULT NULL,
  PRIMARY KEY (`agent_id`),
  UNIQUE KEY `username` (`username`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=26 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `asterisk_extensions`
--

DROP TABLE IF EXISTS `asterisk_extensions`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `asterisk_extensions` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `extension` int(4) DEFAULT NULL,
  `extension_secret` varchar(30) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=25 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
```

```
-- Table structure for table `asterisk_queues`
--

DROP TABLE IF EXISTS `asterisk_queues`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client  = utf8 */;

CREATE TABLE `asterisk_queues` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `queue_name` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client  = @saved_cs_client */;

--
-- Table structure for table `outgoing_channels`
--

DROP TABLE IF EXISTS `outgoing_channels`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client  = utf8 */;
CREATE TABLE `outgoing_channels` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `channel` varchar(30) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client  = @saved_cs_client */;

--
-- Table structure for table `scripts`
--

DROP TABLE IF EXISTS `scripts`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client  = utf8 */;
CREATE TABLE `scripts` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `queue_id` int(10) NOT NULL,
  `text` varchar(10000) NOT NULL,
  `date` date NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
/*!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. Populate the table with the simulated VRS data, again using a tool such as MySQL Workbench.

3.6 Video Relay Service User Database (Provider)

The VRS user database was developed to emulate a VRS user lookup in the iTRS-URD from the Agent desktop until full access to the iTRS-URD is obtained. This lookup verifies that the consumer-provided phone number is a registered VRS number. A MySQL database server is required. A RESTful API developed using Node.js provides access to the data for consumer verification. For this effort, CAMH built the Node.js server with the following software versions:

- Operating System: CentOS 7.x
- Node.js: Version 7.2.1

3.6.1 MySQL Database Server Configuration

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 (named 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. Populate the table with the simulated VRS data, again using a tool such as MySQL Workbench.

3.7 STUN Server

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 can traverse symmetric NAT instances.

For this project, reTurn server (from reSIProcate) is used for STUN/TURN services. The reTurn server 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 yum install resiprocate-turn-server
```

Edit the `etc/reTurn/reTurnServer.config` file, and modify the “AuthenticationRealm” property to point to the Domain Name Service (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.

3.8 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 GUI as well as a programmatic interface. **Note: Access to the iTRS database requires permission from the FCC.**

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)
```

3.9 StrongSWAN for Secure Socket Layer Tunnel

StrongSWAN is an open-source VPN software that is widely popular within the IPSec industry. StrongSWAN can be installed on both CentOS and Amazon Linux servers; the following instructions were implemented on an Amazon Linux EC2 instance.

In the following installation scenario, the local side of the VPN is running within AWS. The remote side provider requires both a public IP for the VPN endpoint and for the tunneled traffic. The implementation requires two public IP addresses on the local side, and NATs all local traffic to one of those address for the VPN tunnel. Also, on the remote end there is only one accessible IP address. Minor adjustments would be required to allow access to more than one destination IP address or to allow a range of IP addresses directly (with or without NAT) through the VPN.

All commands below need to be run as root (or via sudo).

3.9.1 Installation

First, use yum to install StrongSWAN:

```
yum install -y strongswan
```

Then, modify the /etc/strongswan/ipsec.conf table to reflect the following:

```
config setup
```

```

        #charondebug="ike 2, net 3, knl 2, cfg 2"      #useful debugs
conn %default
    ikelifetime=1440m
    keylife=60m
    rekeymargin=3m
    keyingtries=1
    keyexchange=ikev1
    authby=secret
conn PeerProvider
    auto=start
    ike=aes128-sha1-modp1024          #P1: modp1536 = DH group 2
    esp=aes128-sha1-modp1024          #P2

    left=<StrongSWAN local address>    #Local outside address
    leftsubnet=<dummy address>/32      #network behind Local
    leftid=<StrongSWAN public IP>      #IKEID sent by Local
    leftfirewall=yes

    right=<remote peer address.        #PeerProvider outside address
    rightsubnet=<remote server address>/32 #network behind PeerProvider
    rightid=<remote peer address.      #IKEID sent by PeerProvider

```

Agree with your peer on a shared key and then modify the `/etc/strongswan/ipsec.secrets` file to reflect the following:

```

<StrongSWAN public IP> <remote server address> : PSK "<Key
obtained from PeerProvider in quotes>"

```

IPTables must be configured to perform NAT between the source subnet and the public IP identified for tunnel traffic. Create the `/etc/iptables.conf` file and add the following:

```

*nat
:PREROUTING ACCEPT [5:436]
:INPUT ACCEPT [1:92]
:OUTPUT ACCEPT [34:9996]
:POSTROUTING ACCEPT [34:9996]
-A POSTROUTING -s <VPC CIDR block>/24 -d <remote server
address>/32 -o eth0 -j SNAT --to-source <dummy address>

```

COMMIT

```
*filter
:INPUT ACCEPT [1063:95316]
:FORWARD ACCEPT [12:1032]
:OUTPUT ACCEPT [1018:375057]
COMMIT
```

The server needs a dummy interface associated with the public IP for tunnel traffic, and the IPTables configuration must be loaded at startup. Append the following lines to `/etc/rc.d/rc.local`:

```
/sbin/modprobe dummy
/sbin/ifconfig dummy0 <dummy address> netmask 255.255.255.0
/sbin/iptables-restore < /etc/iptables.conf
```

Move into the `/etc/rc3.d` directory and create the following symbolic links to ensure that StrongSWAN gracefully starts/stops when the EC2 instance is rebooted:

```
cd /etc/rc3.d
ln -s ../init.d/strongswan S48strongswan
ln -s ../init.d/strongswan K52strongswan
```

Routing must be enabled on the server. Modify the variable within the `/etc/sysctl.conf` file to the following:

```
net.ipv4.ip_forward = 1
```

Create the following script in `/root/scripts` to automatically restart StrongSWAN if ITRS queries fail:

```
#!/bin/bash

DEBUG=true
LOG=/var/log/itrsmon.log

if ! dig @<remote server address> in naptr
9.6.8.4.1.5.5.7.2.7.1.itrs.us>/dev/null 2>&1
then
    echo $(/bin/date) " - ITRS Lookup failed" >> $LOG
```

```
    /etc/init.d/strongswan restart
else
    $DEBUG && echo $(/bin/date) " - ITRS Lookup success" >> $LOG
fi
```

Add the following line to the crontab to monitor StrongSWAN by running the script once per minute:

```
$ crontab -e
```

```
* * * * * /root/scripts/itrs_mon.sh
```

Finally, reboot the instance to verify that the StrongSWAN service is running (using the 'strongswan status' command). Once it is, you can move onto the following "AWS Specific Config" section if you are in AWS. Once those steps are complete, your DNS queries to the iTRS database from Asterisk servers should be successful. An example of a successful iTRS query is as follows:

```
$ dig @<remote server address> in naptr
1.1.1.1.1.1.1.1.1.1.1.1.1.1.itrs.us
; <<>> DiG 9.9.4-RedHat-9.9.4-50.el7_3.1 <<>> @<remote server
address> in naptr 1.1.1.1.1.1.1.1.1.1.1.1.1.1.itrs.us
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32364
;; flags: qr aa rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 5120
;; QUESTION SECTION:
;1.1.1.1.1.1.1.1.1.1.1.1.1.1.itrs.us. IN NAPTR

;; ANSWER SECTION:
```

```
1.1.1.1.1.1.1.1.1.1.1.1.1.itrs.us. 900 IN NAPTR 10 1 "u" "E2U+h323"
"!^(.*)$!h323:\\1@192.168.1.1!" .
```

```
;; Query time: 15 msec
;; SERVER: <remote server address>#53(<remote server address>)
;; WHEN: Fri Aug 04 14:40:59 UTC 2017
;; MSG SIZE rcvd: 115
```

3.9.2 AWS-Specific Config

3.9.2.1 VPC Route Table

A route must be created in the VPC route table to ensure that traffic to the iTRS database is routed to the VPN instance (which in turn routes it out over the VPN tunnel).

3.9.2.2 Source/Destination Checking

By default, instances drop packets when source and destination information does not match that of an instance. This behavior can be disabled from the AWS console by selecting an instance and going to network options.

3.9.3 Troubleshooting

You can run 'strongswan statusall' to view the status of the VPN tunnel. The expected command output will look like the following:

```
$ strongswan statusall

Status of IKE charon daemon (strongSwan 5.4.0, Linux 4.9.38-
16.33.amzn1.x86_64, x86_64):
  uptime: 116 minutes, since Aug 04 12:46:06 2017
  malloc: sbrk 1622016, mmap 0, used 502608, free 1119408
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0,
scheduled: 2
  loaded plugins: charon aes des rc2 sha2 sha1 md4 md5 random nonce x509
revocation constraints acert pubkey pkcs1 pkcs8 pkcs12 pgp dnskey sshkey
pem openssl gcrypt fips-prf gmp xcbc hmac ctr ccm gcm curl attr
kernel-netlink resolve socket-default farp stroke vici updown eap-identity
eap-md5 eap-gtc eap-mschapv2 eap-tls eap-ttls eap-peap xauth-generic
xauth-eap xauth-pam xauth-noauth dhcp
Listening IP addresses:
  <StrongSWAN local address>
  <dummy address> ← both IP addresses should be listed
Connections:
  PeerProvider: <StrongSWAN local address>...<remote peer address. IKEv1
  PeerProvider: local: [<StrongSWAN public address>] uses pre-shared key
authentication
  PeerProvider: remote: [<remote peer address.>] uses pre-shared key
authentication
  PeerProvider: child: <dummy address>/32 === <remote server address>/32
TUNNEL
Security Associations (1 up, 0 connecting):
  PeerProvider[1]: ESTABLISHED 116 minutes ago, <StrongSWAN local
```

```

address>[<StrongSWAN public address>]...<remote peer address.[<remote peer
address.>]
PeerProvider[1]: IKEv1 SPIs: <<SPI>>, pre-shared key reauthentication in
21 hours
PeerProvider[1]: IKE proposal:
AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024
PeerProvider{3}: INSTALLED, TUNNEL, reqid 1, ESP SPIs: <<SPI>>
PeerProvider{3}: AES_CBC_128/HMAC_SHA1_96/MODP_1024, 1494 bytes_i (8
pkts, 2s ago), 611 bytes_o (8 pkts, 2s ago), rekeying in 47 minutes
PeerProvider{3}: <dummy address>/32 === <remote server address>/32

```

To find system messages related to StrongSWAN, use the following grep command:

```

$ grep charon /var/log/messages

#Shutdown
Jul 27 00:27:58 ServerName charon: 00[DMN] signal of type SIGINT received.
Shutting down
Jul 27 00:27:58 ServerName charon: 00[IKE] closing CHILD_SA
PeerProvider{1} with SPIs <<SPI>> (0 bytes) 5401197d_o (0 bytes) and TS
<Dummy Address>/32 === <remote server address>/32
Jul 27 00:27:58 ServerName charon: 00[IKE] sending DELETE for ESP CHILD_SA
with SPI cc95ab50
Jul 27 00:27:58 ServerName charon: 00[ENC] generating INFORMATIONAL_V1
request 4051018568 [ HASH D ]
Jul 27 00:27:58 ServerName charon: 00[NET] sending packet: from
<StrongSWAN public IP>[500] to <remote peer address.[500] (76 bytes)
Jul 27 00:27:58 ServerName charon: 00[IKE] deleting IKE_SA PeerProvider[1]
between <StrongSWAN public IP>[<local server address>]...<remote peer
address.[<remote peer address.>]
Jul 27 00:27:58 ServerName charon: 00[IKE] sending DELETE for IKE_SA
PeerProvider[1]
Jul 27 00:27:58 ServerName charon: 00[ENC] generating INFORMATIONAL_V1
request 3332107879 [ HASH D ]
Jul 27 00:27:58 ServerName charon: 00[NET] sending packet: from
<StrongSWAN public IP>[500] to <remote peer address.[500] (92 bytes)
Jul 27 00:27:58 ServerName charon: 00[KNL] received netlink error: Address
family not supported by protocol (97)

#Startup
Jul 27 00:27:58 ServerName charon: 00[DMN] Starting IKE charon daemon
(strongSwan 5.4.0, Linux 4.9.32-15.41.amzn1.x86_64, x86_64)
Jul 27 00:27:58 ServerName charon: 00[LIB] openssl FIPS mode(2) - enabled

#No IPv6 - that's OK
Jul 27 00:27:58 ServerName charon: 00[NET] could not open socket: Address
family not supported by protocol
Jul 27 00:27:58 ServerName charon: 00[NET] could not open IPv6 socket,
IPv6 disabled
Jul 27 00:27:58 ServerName charon: 00[KNL] received netlink error: Address
family not supported by protocol (97)
Jul 27 00:27:58 ServerName charon: 00[KNL] unable to create IPv6 routing
table rule

#No certs - that's OK
Jul 27 00:27:58 ServerName charon: 00[CFG] loading ca certificates from
'/etc/strongswan/ipsec.d/cacerts'
Jul 27 00:27:58 ServerName charon: 00[LIB] opening directory
'/etc/strongswan/ipsec.d/cacerts' failed: No such file or directory
Jul 27 00:27:58 ServerName charon: 00[CFG] reading directory failed
Jul 27 00:27:58 ServerName charon: 00[CFG] loading aa certificates from

```



```
'/etc/strongswan/ipsec.d/aacerts'
Jul 27 00:27:58 ServerName charon: 00[LIB] opening directory
'/etc/strongswan/ipsec.d/aacerts' failed: No such file or directory
Jul 27 00:27:58 ServerName charon: 00[CFG] reading directory failed
Jul 27 00:27:58 ServerName charon: 00[CFG] loading ocspsigner
certificates from '/etc/strongswan/ipsec.d/ocspcerts'
Jul 27 00:27:58 ServerName charon: 00[LIB] opening directory
'/etc/strongswan/ipsec.d/ocspcerts' failed: No such file or directory
Jul 27 00:27:58 ServerName charon: 00[CFG] reading directory failed
Jul 27 00:27:58 ServerName charon: 00[CFG] loading attribute certificates
from '/etc/strongswan/ipsec.d/acerts'
Jul 27 00:27:58 ServerName charon: 00[LIB] opening directory
'/etc/strongswan/ipsec.d/acerts' failed: No such file or directory
Jul 27 00:27:58 ServerName charon: 00[CFG] reading directory failed
Jul 27 00:27:58 ServerName charon: 00[CFG] loading crls from
'/etc/strongswan/ipsec.d/crls'
Jul 27 00:27:58 ServerName charon: 00[LIB] opening directory
'/etc/strongswan/ipsec.d/crls' failed: No such file or directory
Jul 27 00:27:58 ServerName charon: 00[CFG] reading directory failed
```

#Interesting stuff starts here - loading secrets

```
Jul 27 00:27:58 ServerName charon: 00[CFG] loading secrets from
'/etc/strongswan/ipsec.secrets'
Jul 27 00:27:58 ServerName charon: 00[CFG] loaded IKE secret for <local
server address> <remote server address>
Jul 27 00:27:58 ServerName charon: 00[LIB] loaded plugins: charon aes des
rc2 sha2 sha1 md4 md5 random nonce x509 revocation constraints acert
pubkey pkcs1 pkcs8 pkcs12 pgp dnskey sshkey pem openssl gcrypt fips-prf
gmp xcbc cmac hmac ctr ccm gcm curl attr kernel-netlink resolve socket-
default farp stroke vici updown eap-identity eap-md5 eap-gtc eap-mschapv2
eap-tls eap-ttls eap-peap xauth-generic xauth-eap xauth-pam xauth-noauth
dhcpc
Jul 27 00:27:58 ServerName charon: 00[JOB] spawning 16 worker threads
Jul 27 00:27:58 ServerName charon: 05[CFG] received stroke: add connection
'PeerProvider'
Jul 27 00:27:58 ServerName charon: 05[CFG] added configuration
'PeerProvider'
Jul 27 00:27:58 ServerName charon: 09[CFG] received stroke: initiate
'PeerProvider'
Jul 27 00:27:58 ServerName charon: 09[IKE] initiating Main Mode IKE_SA
PeerProvider[1] to <remote peer address>.
Jul 27 00:27:58 ServerName charon: 09[ENC] generating ID_PROT request 0 [
SA V V V V ]
```

#Sending and RECEIVING packets - good sign

```
Jul 27 00:27:58 ServerName charon: 09[NET] sending packet: from
<StrongSWAN public IP>[500] to <remote peer address>.[500] (228 bytes)
Jul 27 00:27:58 ServerName charon: 15[NET] received packet: from <remote
peer address>.[500] to <StrongSWAN public IP>[500] (164 bytes)
Jul 27 00:27:58 ServerName charon: 15[ENC] parsed ID_PROT response 0 [ SA
V V V ]
Jul 27 00:27:58 ServerName charon: 15[ENC] received unknown vendor ID:
05:16:dc:8a:88:2c:54:a5:66:90:dc:05:bd:da:3b:9e:c8:05:e5:86:12:00:00:00:1e
:06:00:00
Jul 27 00:27:58 ServerName charon: 15[IKE] received DPD vendor ID
Jul 27 00:27:58 ServerName charon: 15[ENC] received unknown vendor ID:
48:65:61:72:74:42:65:61:74:5f:4e:6f:74:69:66:79:38:6b:01:00
Jul 27 00:27:58 ServerName charon: 15[ENC] generating ID_PROT request 0 [
KE No ]
Jul 27 00:27:58 ServerName charon: 15[NET] sending packet: from
<StrongSWAN public IP>[500] to <remote peer address>.[500] (196 bytes)
```

```

Jul 27 00:27:58 ServerName charon: 11[NET] received packet: from <remote
peer address.[500] to <StrongSWAN public IP>[500] (196 bytes)
Jul 27 00:27:58 ServerName charon: 11[ENC] parsed ID_PROT response 0 [ KE
No ]
Jul 27 00:27:58 ServerName charon: 11[ENC] generating ID_PROT request 0 [
ID HASH N(INITIAL_CONTACT) ]
Jul 27 00:27:58 ServerName charon: 11[NET] sending packet: from
<StrongSWAN public IP>[500] to <remote peer address.[500] (108 bytes)
Jul 27 00:27:58 ServerName charon: 07[NET] received packet: from <remote
peer address.[500] to <StrongSWAN public IP>[500] (76 bytes)
Jul 27 00:27:58 ServerName charon: 07[ENC] parsed ID_PROT response 0 [ ID
HASH ]
#Phase 1 (IKE) Established
Jul 27 00:27:58 ServerName charon: 07[IKE] IKE_SA PeerProvider[1]
established b <StrongSWAN public IP>[<local server address>]...<remote
peer address.[<remote peer address.>]
Jul 27 00:27:58 ServerName charon: 07[IKE] scheduling reauthentication in
86056s
Jul 27 00:27:58 ServerName charon: 07[IKE] maximum IKE_SA lifetime 86236s
Jul 27 00:27:58 ServerName charon: 07[ENC] generating QUICK_MODE request
688951572 [ HASH SA No KE ID ID ]
Jul 27 00:27:58 ServerName charon: 07[NET] sending packet: from
<StrongSWAN public IP>[500] to <remote peer address.[500] (316 bytes)
Jul 27 00:27:58 ServerName charon: 13[NET] received packet: from <remote
peer address.[500] to <StrongSWAN public IP>[500] (316 bytes)
Jul 27 00:27:58 ServerName charon: 13[ENC] parsed QUICK_MODE response
688951572 [ HASH SA No KE ID ID ]
#Phase 2 (ISAKMP) Established
Jul 27 00:27:58 ServerName charon: 13[IKE] CHILD_SA PeerProvider{1}
established with SPIs X and TS <Dummy Address>/32 === <remote server
address>/32
Jul 27 00:27:58 ServerName charon: 13[ENC] generating QUICK_MODE request
688951572 [ HASH ]
Jul 27 00:27:58 ServerName charon: 13[NET] sending packet: from
<StrongSWAN public IP>[500] to <remote peer address.[500] (60 bytes)

```

You can verify the proper configuration of the iptables rule to mask the source IP with the dummy address. You should see the number of packets that have been applied to that rule:

```

$ iptables -t nat -L -v

Chain PREROUTING (policy ACCEPT 804 packets, 66289 bytes)
  pkts bytes target    prot opt in     out     source destination

Chain INPUT (policy ACCEPT 43 packets, 4165 bytes)
  pkts bytes target    prot opt in     out     source destination

Chain OUTPUT (policy ACCEPT 51100 packets, 3555K bytes)
  pkts bytes target    prot opt in     out     source destination

Chain POSTROUTING (policy ACCEPT 51076 packets, 3553K bytes)
  pkts bytes target    prot opt in     out     source destination
   785 63924 SNAT      all  --  any    eth0    <<local subnet>>/24 <remote
server address>      to:<Dummy Address>

```

You can use the 'ip route' command to ensure StrongSWAN has properly configured the route on the instance required to forward traffic into the VPN tunnel. The second and fourth lines of the following output are of particular importance:

```

$ ip route show table all
default via ...
<<Remote Address>>/24 dev dummy0 proto kernel scope link src <Dummy

```

```
Address>
... dev eth0
<local subnet> dev eth0 proto kernel scope link src <StrongSWAN public
IP>
broadcast .. dev dummy0 table local proto kernel scope link src <Dummy
Address>
<< deleted the rest of the output >>
```

3.10 Commercial Customer Relationship Management

To demonstrate integration with a commercial CRM service, ACE Direct connects to the Zendesk portal via the ESB. 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 Agent answers a call from a consumer, the application requests the ticket information from Zendesk and displays it on the ACE Direct Agent Desktop portal.

3.11 FenDesk Customer Relationship Management

FenDesk is a server that emulates the [Zendesk](#) ticketing system for ACE Direct or any client that needs a simple ticketing system. The software only implements the subset of Zendesk RESTful API calls that ACE Direct uses; however, it is expandable to include other API calls.

FenDesk uses a simple storage scheme. It creates, updates, and returns tickets as simple text files. The filename for a ticket follows the same naming convention as Zendesk: <ticketno>.json (e.g., 322.json). FenDesk offers RESTful API calls to test connectivity, add/update/delete/retrieve tickets, and search for all tickets with a specified VRS number.

3.12 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 Agent to document service cases.

3.12.1 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 APIs of the Java Business Integration (JBI) specification JSR 208. The software is distributed under the Apache License.

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 Enterprise Edition

(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

3.12.2 Installation Overview

First install and configure ServiceMix and its prerequisites on the host machine.

3.12.2.1 ServiceMix System Requirements

To run Apache ServiceMix itself, you will need 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) 1.8.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.

3.12.2.2 Installing the JDK

Issue the following commands to install the Java 8 JDK:

Redhat/Fedora/CentOS Systems (SystemV) :

```
sudo yum install java-1.8.0-openjdk
```

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= /opt/jdk1.8.0_111
export JAVA_HOME
JRE_HOME=/opt/jdk1.8.0_111/jre
export JRE_HOME
PATH=$PATH:/opt/jdk1.8.0_111/bin:/opt/jdk1.8.0_111/jre/bin
export PATH
```

3.12.2.3 Installing Apache Maven

To install Apache Maven, issue the following command:

Redhat/Fedora/Centos Systems (SystemV):

```
sudo yum install maven
```

3.12.2.4 Installing MySQL

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

Redhat/Fedora/Centos Systems (SystemV):

```
sudo yum install unixODBC unixODBC-devel libtool-ltdl
libtool-ltdl-devel mysql-connector-odbc
wget http://repo.mysql.com/mysql-community-release-el7-
5.noarch.rpm
sudo rpm -ivh mysql-community-release-el7-5.noarch.rpm
sudo yum install mysql-server
sudo systemctl start mysqld
sudo yum update
```

3.12.2.5 Configuring MySQL

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

To start the mysql in Redhat/Fedora/Centos Systems (SystemV):

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

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

```
mysql -u root -p=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 also requires setting the privileges for anonymous local users. Privileges must 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');
```

3.12.2.6 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 from esb.git

Navigate to the broker code folder:

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

Modify the applications blueprint file for your environment. The blueprint is configured to process messages intended for Zendesk. You will need to modify the blueprint to specify your Zendesk hostname and any proxy settings if you are accessing Zendesk from the ESB through a proxy.

Build the broker application with Maven:

```
mvn clean install
```

3.12.2.7 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
```

3.12.2.8 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
karaf@root>feature:install camel-http4
```

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 that 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
```

3.12.2.9 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 before installing and starting the service:

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

Redhat/Fedora/Centos Systems (SystemV):

To install the service:

```
$ ln -s ~/dev-tools/apache-servicemix-6.1.0/bin/KARAF-  
service /etc/init.d/  
$ chkconfig KARAF-service --add
```

To start the service when the machine is rebooted:

```
$ chkconfig KARAF-service on
```

To disable starting the service when the machine is rebooted:

```
$ chkconfig KARAF-service off
```

To start the service:

```
$ service KARAF-service start
```

To stop the service:

```
$ service KARAF-service stop
```

To uninstall the service :

```
$ chkconfig KARAF-service --del
```

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

For systemd compliant Linux:

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

```
$ systemctl enable~/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 ~/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.

3.12.3 Editing blueprint.xml Application File

The blueprint.xml file is provided with placeholders which will need to be edited for your specific environment.

3.12.3.1 Update Zendesk hostname

The blueprint.xml file has placeholders for the Zendesk hostname. You will need to provide your Zendesk hostname wherever you see the placeholder “<insert CRM hostname>”

3.12.3.2 Enable/Disable proxy

If there is a proxy between the ESB and Zendesk, you may need add the following parameters to set of parameters specified for each instance of the Zendesk endpoint. For example, you may need to replace

```
/api?bridgeEndpoint=true&throwExceptionOnFailure=false
```

with:

```
/api?bridgeEndpoint=true&proxyAuthHost=<replace with proxy
ip>&proxyAuthPort=<replace with proxy
port>&proxyAuthScheme=http4&
throwExceptionOnFailure=false
```

substituting your proxy host and port settings for the placeholders.

After editing the blueprint file, rebuild the application using maven:

```
cd ~/code/camel-rest-proxy-blue-print/
mvn clean install
cd ~/dev-tools/apache-servicemix-6.1.2/bin
./client
karaf@root>bundle:install -s mvn:org.apache.camel/camel-rest-
proxy-blueprint/2.16.3
```

Make sure the bundle is successfully installed with no errors and active.

3.12.4 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 in <https://github.com/FCC/ACEDirect> in the acrdemo-provider folder.

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
username@hostname/token:hLLUnPzJtpvMZ5WnntN3wCneKHkl20kP0Hh
n5NrD http://localhost:9090/api/v2/users/me.json --
insecure
```

where username is a Zendesk user account and hostname is your Zendesk host name.

You can check the status and statistics 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
ACR	Auto Call Routing
ADA	Americans with Disabilities Act
AMA	Automatic Message Accounting
AMI	Asterisk Management Interface
API	Application Programming Interface
ASL	American Sign Language
AWS	Amazon Web Services
CA	Communication Assistant, Certificate Authority
CAMH	CMS Alliance to Modernize Healthcare
CDR	Call Detail Record
COE	Center of Expertise
COTS	Commercial Off-the-Shelf
CRM	Customer Relationship Management
CSV	Comma Separated Value
DNS	Domain Name System
DVC	Direct Video Calling
EIP	Elastic Internet Protocol
ENUM	E.164 Number to URI Mapping
ESB	Enterprise Service Bus
FCC	Federal Communications Commission
FFRDC	Federally Funded Research and Development Center
GUI	Graphical User Interface
HTTPS	HyperText Transport Protocol Secure
iTRS	Internet Telecommunications Relay Service
I/O	Input/Output
IP	Internet Protocol
IPSec	Internet Protocol Secure

Java EE	Java Enterprise Edition
JB1	Java Business Integration
JDBC	Java Database Connectivity
JDK	Java Developer Kit
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
NAT	Network Address Translation
ODBC	Open Database Connectivity
OS	Operating System
OSGi	OSGi Alliance (formerly Open Systems Group Initiative)
PBX	Private Branch Exchange
POC	Proof of Concept
PSTN	Public Switch Telephone Network
QoS	Quality of Service
REST	Representational State Transfer
RFC	Request for Comment
RTCP	RTP (Real-time Transport Protocol) Control Protocol
RTT	Real-Time Text
SIP	Session Initiation Protocol
SOA	Service-Oriented Architecture
SQL	Structured Query Language
SRTP	Secure Real-Time Transport Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TURN	Traversal Using Relay NAT
UDP	User Datagram Protocol
URD	User Registration Database

URI	Uniform Resource Identifier
URL	Universal Resource Locator
VPN	Virtual Private Network
VRS	Video Relay Service