



macOS SAFR® Documentation

Documentation Version = 2.000

Publish Date = April 1, 2020

Copyright © 2020 RealNetworks, Inc. All rights reserved.

SAFR® is a trademark of RealNetworks, Inc. Patents pending.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

August_2019_Release_Notes

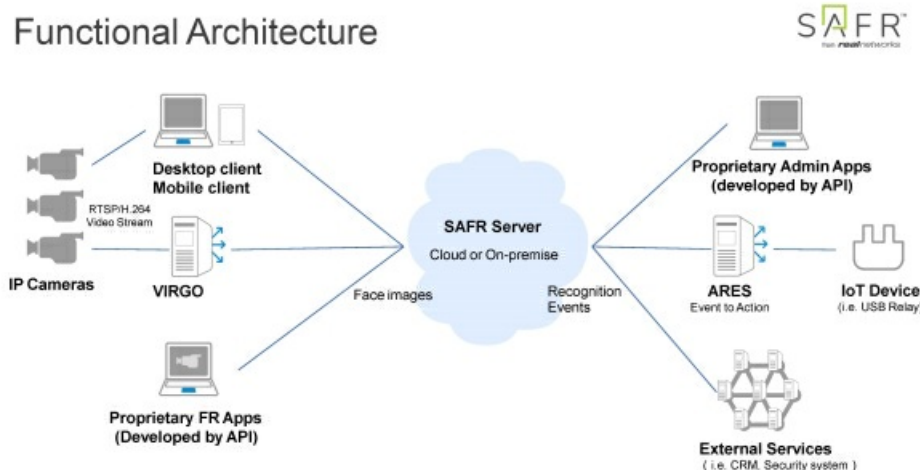
- [SAFR Overview](#)
- [SAFR System Requirements](#)
- [Licensing](#)
- [Getting Started with SAFR Platform on Windows or macOS](#)
- [Getting Started with SAFR Edge with a Cloud Account](#)
- [Recommendations for Camera Placement and Use](#)
- [Set up ONVIF IP Cameras](#)
- [Connect Cameras to SAFR](#)
- [Connect to a Video Feed](#)
- [Interpret Video Feed Overlays](#)
- [View Video Feeds Status](#)
- [Manage People in the Person Directory](#)
- [Importing and Registering People](#)
- [Image Quality Metrics Guidance](#)
- [Use Event Analytics to Gain Insight](#)
- [Actions Overview](#)
- [Actions Relay Event Service \(ARES\)](#)
- [SAFRActions.config](#)
- [SAFR Actions](#)
- [Large Scale Deployments](#)
- [Database Redundancy](#)
- [SSL Certificate Installation](#)
- [SAFR Support Tools and Scripts](#)
- [SAFR Server Backup and Restore](#)
- [Video Recognition Gateway \(VIRGO\)](#)
- [VIRGO Installation Guide](#)
- [VIRGO System Requirements](#)
- [VIRGO Command Line Interface](#)
- [Docker](#)
- [Factory Configuration](#)
- [Service Logging](#)
- [Service Monitoring](#)
- [Troubleshooting](#)
- [Command & Control Protocol \(COP\)](#)
- [COP Introduction](#)
- [COP Status Delivery](#)
- [COP Status Reply](#)
- [COP Image Capture](#)
- [Tracking Result Capture](#)
- [COP Logging](#)
- [COP Software Updates](#)
- [COP Errors](#)

- [COP State Update Algorithms](#)
- [COP Examples](#)
- [Desktop Client](#)
- [Camera Window](#)
- [View Menu Options](#)
- [People Window](#)
- [Events Window](#)
- [Video Feeds Status Window](#)
- [Account Preferences](#)
- [Camera Preferences](#)
- [Detection Preferences](#)
- [Tracking Preferences](#)
- [Recognition Preferences](#)
- [Events Preferences](#)
- [User Interface Preferences](#)
- [Connect a Face Recognition Panel](#)
- [Connect a Registration Kiosk](#)
- [Customize a Registration Kiosk](#)
- [Configure a Mobile Device into Locked Mode](#)
- [Install SAFR Beam](#)
- [Mobile Account Preferences](#)
- [Mobile Detection Preferences](#)
- [Mobile Recognition Preferences](#)
- [Mobile Events Preferences](#)
- [Mobile User Interface Preferences](#)
- [Web Console](#)
- [Status Page](#)
- [People Page](#)
- [Events Page](#)
- [Video Feeds Pages](#)
- [Reports Page](#)
- [Traffic Dashboard](#)
- [Queue Dashboard](#)
- [Attendance Dashboard](#)
- [Traversal Dashboard](#)
- [Traffic Report](#)
- [Pose Liveness Detection](#)
- [March 2020 Release Notes](#)
- [January 2020 Release Notes](#)
- [December 2019 Release Notes](#)
- [November 2019 Release Notes](#)
- [September 2019 Release Notes](#)
- [August 2019 Release Notes](#)

SAFR Overview

SAFR is a facial recognition system that integrates cameras, door locks, and alert systems with face recognition technology to enhance access control and security. It runs on a variety of operating systems, including Windows, macOS, Linux, iOS, and Android.

SAFR Components



SAFR primarily consists of the following components:

- **SAFR Server:** Available for Windows, macOS, and Linux. The SAFR Server installation contains the recognition engine, event server, several databases, and the Web Console. The databases contain stored enrolled face images, the identity information for the stored faces, and recognition events that have been generated by the SAFR system. The SAFR Server runs as several background services that automatically start on system reboot and are kept active by the operating system. They must be running at all times for the system to be operational. All other SAFR components must connect to a SAFR Server, although if you're doing a cloud deployment you'll be connecting to a SAFR Server in the cloud that RealNetworks maintains.
- **Desktop client:** Available for Windows and macOS. The Desktop client is one of the primary ways that administrators and operators can interact with the SAFR system. As such, the client can be used to enable camera connectivity, monitor video camera feeds, register users, view recognition events, and more.
- **Mobile client:** Available for Android and iOS. The Mobile client converts a mobile device into a registration kiosk or a recognition panel. Registration kiosks allow people to self-register their face into the Identity Database so they can be approved for access or granted other privileges. Recognition panels enable the mobile device to scan the faces of people that walk by and to compare those faces against faces in the Identity Database. Mobile devices set up as recognition panels can also provide visual or audio feedback to the person viewing the mobile device based on actions that a SAFR administrator has

configured.

- **VIRGO:** Available as a standalone download for macOS and Linux. It's also available as part of the SAFR Desktop, SAFR Edge, and SAFR Platform download packages for Windows, macOS, and Linux. The Video Recognition Gateway (VIRGO) is a daemon system which receives video feeds from one or more cameras and recognizes and tracks faces in those video streams in real time. It generates tracking events and sends those events to an event server. The VIRGO daemon can be controlled either by the command line tool or through the Video Recognition Gateway Administration (VIRGA) command & control server.
- **Web Console:** Available on all platforms. The Web Console provides administrators and operators web-based access to the SAFR system. As such, the Web Console can be used to generate analytical reports, monitor video camera feeds, register users, view recognition events, and more.
- **ARES:** Available as a standalone download for all platforms. Actions Relay Event Service (ARES) is a cross-platform Java application that acts as the event listener that dispatches configured actions in response to events. ARES can provide replies on any event handled by the client that originates an event and is normally installed as a service when either SAFR Platform or SAFR Edge are installed. It is constantly active and is automatically started by the operating system on power-up.
- **SAFR Actions:** SAFR Actions is a GUI that facilitates configuring SAFRActions.config. SAFRActions.config is the file that defines all the defined actions for your SAFR System, as well as a couple fields that are used to connect ARES (and SAFR Actions) to your primary SAFR Server, whether that server is local or in the cloud. See Actions for more information about actions in SAFR.

In addition to the SAFR components listed above, SAFR also relies on a couple additional non-SAFR components:

- **IP Cameras:** As you might expect, Internet Protocol (IP) cameras are absolutely integral to SAFR. Both the Desktop client and VIRGO automatically detect integrated, USB, and [Open Network Video Interface Forum](#) (ONVIF) IP cameras. If an IP camera does not support ONVIF or doesn't have ONVIF enabled, you can still manually add it to the SAFR system as described here.
- **Physical access control devices:** Door locks, electronic gates, etc. can all be used by SAFR to grant or deny access to people, depending on whether or not they're identified as having the proper authorization.
- **Notification systems:** Email can be used to discretely notify specified people of various events, while general alarms can be used to alert everybody in the vicinity when unauthorized people attempt to force entry.
- **Additional external peripherals:** Any device that can be controlled by a computer language or protocol can be incorporated into the SAFR system.

Available Download Packages

The following download packages are available on the [SAFR Download Portal](#):

- **SAFR Platform:** Available on Windows, macOS, and Linux. The SAFR Platform installs everything you need to set up a local deployment of SAFR. This download package enables a locally deployed system to be easily deployed on a single computer and afterwards expanded to additional computers as needed. See *Getting Started with SAFR Platform on Windows or macOS* for more information.
- **SAFR Desktop:** Available on Windows and macOS. Installs the Desktop client and one of the VMS extensions. Windows has an additional download variant called SAFR Desktop Lite which has fewer features and lower system requirements.
- **SAFR Edge:** Available on Windows and macOS. SAFR Edge installs the Desktop client as well as SAFR Actions, a programmable interface to create and manage responses to event triggers. For example, you can unlock a door, turn on a light, send an alert, and so on. See *Getting Started with SAFR Edge with a Cloud Account* for more information.
- **SAFR Mobile:** Available on Android and iOS. Installs the Mobile client. When you download SAFR Mobile for Android, you're also offered the SAFR Beam download. SAFR Beam allows you to enable the more secure Lock Task Mode on your Android device. If you don't install SAFR Beam, then Android devices can only enable the less secure Screen Pinning Mode. See *Configure Devices into Locked Mode* for more information.
- **Actions Relay Event Service (ARES):** Available on all platforms. Installs ARES.
- **Video Recognition Gateway (VIRGO):** Available on Linux and macOS. Installs VIRGO.

Deployment Types

There are two types of SAFR deployment: cloud and local. Each deployment type requires its own account type; a cloud deployment requires a SAFR Cloud Account, while a local deployment requires a SAFR Local Account. Contact your SAFR Account Manager to obtain either type of account.

Cloud Deployment

When SAFR is deployed as a cloud deployment, all your SAFR components are deployed locally except for the SAFR Server. Your components will connect to a SAFR Server located in the cloud which is operated by RealNetworks, Inc. Using the cloud SAFR Server greatly simplifies deployment and maintenance, but it requires a network connection to the cloud at all times in order to be operational.

A single installation of the Desktop client can handle about 16 connected cameras, assuming the hosting machine meets the recommended system requirements listed here. Expanding your SAFR system beyond this limitation is fairly easy; simply install additional Desktop clients onto additional machines.

Local Deployment

When SAFR is deployed as a local deployment, all of the SAFR components (including SAFR Server) are installed locally. During installation a connection is made to a SAFR License Server in the cloud to obtain a licence, but after a license has been obtained, local deployments do not require a connection to the cloud.

A single installation of the SAFR Server can handle about 25 viewed faces at one time, assuming the hosting machine meets the recommended system requirements listed here. Note that for the purposes of server capacity, “25 viewed faces” can mean “25 cameras with 1 face in each camera view” or “1 cameras with 25 faces in its camera view”, or anything in between. If you want to expand your SAFR system beyond this limitation please see Large Scale Deployments.

Environments

There are 2 environments available:

- **SAFR Cloud:** A general availability SAFR Server in the cloud maintained by RealNetworks. It is a stable, high availability environment intended for production use. Its SAFR Download Portal is located at <https://safr.real.com/products>.
- **SAFR Partner Cloud:** A SAFR Server in the cloud maintained by RealNetworks that is intended for use by partners that have special integration needs. Its SAFR Download Portal is located at <https://safr.int2.real.com/products>.

Each SAFR account that you’re issued applies to exactly one environment.

SAFR System Requirements

macOS Requirements

Product	Description	Minimum Requirements	Recommended Requirements
---------	-------------	----------------------	--------------------------

Product	Description	Minimum Requirements	Recommended Requirements
Desktop client	One of the administration consoles for SAFR. Use the Desktop client to connect to a camera, video feed, detect faces, and submit images to the SAFR Server or Cloud for recognition.	<ul style="list-style-type: none"> • macOS 10.12 • Dual Core i7 • 1GB RAM per connected camera • 0.5GB available storage • Supports one 4K camera¹ • Supports 2+ 1080p cameras¹ 	<ul style="list-style-type: none"> • iMac Pro: • macOS 10.12 • 8-core Intel Xeon • 1GB RAM per connected camera • 0.5GB available storage • Supports up to eight 4K cameras¹ • iMac, MacBook Pro, Mac mini: • macOS 10.12 • 6-core Intel i7 • 1GB RAM per connected camera • 0.5GB available storage • Supports up to three 4K cameras¹ • Supports 4+ 1080p cameras¹
SAFR Actions	Actions allow you to create and manage responses to event triggers; deploy them to unlock a door, turn on a light, send an alert, record data for reporting, or any security response to fit the use case.	<ul style="list-style-type: none"> • macOS 10.12 • Dual Core i5 • 1GB RAM • 0.5GB available storage 	<ul style="list-style-type: none"> • macOS 10.12 • Dual Core i5 • 1GB RAM • 0.5GB available storage

Product	Description	Minimum Requirements	Recommended Requirements
SAFR Server ²	The trusted engine of SAFR solutions, the SAFR Server includes: the facial recognition server, identity database, recognition event server, event archive, report server, and remote video feed administration servers.	<ul style="list-style-type: none"> • macOS 10.12 • Quad Core i7 2.6GHz • 16GB RAM • 1TB available storage • Supports one 4K camera¹ 	<ul style="list-style-type: none"> • iMac Pro: • macOS 10.12 • 10-core Intel Xeon • 32GB RAM • 1TB available storage • Supports up to six 4K cameras¹ • Supports 7+ 1080p cameras¹ • Mac mini: • macOS 10.12 • 6-core Intel i7 3GHz • 32GB RAM • 1TB available storage • Supports up to four 4K cameras¹ • Supports 5+ 1080p cameras¹
SAFR SDK		<ul style="list-style-type: none"> • macOS 10.12 • 1GB RAM per 4K video stream • 215MB available storage 	<ul style="list-style-type: none"> • macOS 10.14 • 1GB RAM per 4K video stream • 215MB available storage

1 = Number of cameras is based on an average of five visible faces in a 4K resolution camera view, running at 15 frames per second. Using fewer faces per camera and lower resolution will enable support for more cameras.

2 = Installed as part of the SAFR Platform installer.

Mobile Requirements

Product	Description	Minimum Requirements	Recommended Requirements
Mobile client for iOS	Set up a registration kiosk, perform facial recognition, and add users — all from a mobile device.	<ul style="list-style-type: none"> • iOS 11.0 • iPad Pro or iPhone 6/7/8/X 	<ul style="list-style-type: none"> • iOS 11.0 • iPad Pro or iPhone 6/7/8/X

Product	Description	Minimum Requirements	Recommended Requirements
Mobile client for Android	Set up a registration kiosk, perform facial recognition, and add users — all from a mobile device.	<ul style="list-style-type: none"> Android 5.0 with Google Play Services 13.2.74 or later Quad-core Snapdragon 802 2.5GHz 2GB RAM 13MB available storage 	<ul style="list-style-type: none"> Android 6.0 Quad-core Snapdragon 802 2.5GHz Samsung Galaxy Tab S4 Samsung Galaxy S8 Google Pixel 2 XL 2GB RAM 13MB available storage
SAFR Beam for Android	This SAFR utility allows you to configure Android mobile devices for secure SAFR operation.	<ul style="list-style-type: none"> Android 6.0 Near-Field Communication (NFC) support required 1MB RAM 8MB available storage 	<ul style="list-style-type: none"> Android 6.0 Near-Field Communication (NFC) support required 1MB RAM 8MB available storage
SAFR SDK for Android		<ul style="list-style-type: none"> Android 6.0 1GB RAM 0.5GB available storage 	<ul style="list-style-type: none"> Android 6.0 1GB RAM 0.5GB available storage
SAFR Embedded SDK for Android		<ul style="list-style-type: none"> Android 6.0 ARM Architecture 200MB RAM 150MB available storage 	<ul style="list-style-type: none"> Android 6.0 ARM Architecture 200MB RAM 150MB available storage
SAFR SDK for iOS		<ul style="list-style-type: none"> iOS 11 or higher iPhone 6 Swift 5 92MB available storage 	<ul style="list-style-type: none"> iOS 12 iPhone X or iPad Pro Swift 5 92MB available storage

Licensing

SAFR systems require a license to operate.

License Limit Metrics

SAFR licenses limit usage according to the following metrics:

- **Expiration date:** The date when the SAFR license expires. After this date, SAFR software discontinues operation.
- **Max Feeds per Hour:** Maximum number of video feeds that can be used at one time by the SAFR system. If you attempt to connect more video feeds than your license allows, the excess video feed connection attempts will all fail. Existing video feeds must be disconnected for a period of 1 hour before new video feeds are allowed to re-use the license.
Note: If a single camera is providing video feeds to 2 different Desktop client instances, that counts as 2 video feeds for licensing purposes.
- **Max Faces:** Maximum number of faces that can be stored in SAFR's database. Attempting to save more faces than this limit allows results in an error.
- **Max Days Between Reports:** The maximum elapsed time that can pass before the SAFR system can report its status to a SAFR License Server. SAFR Server discontinues operation if it is unable to reach the SAFR License Server after the specified time has elapsed. If you need to operate your SAFR system on a private network that isn't connected to the Internet, contact your SAFR account manager to acquire a special offline license.
Note: This metric is only applicable for local deployments.

License limit metrics for your SAFR license can be found on the Status page of the Web Console. Note that *Max Days Between Reports* won't appear on your Web Console if you have a cloud deployment.

Licensing for Local Deployments

In local deployments, SAFR licenses are attached to your SAFR system's primary server. The following describes how the SAFR license is managed:

- License Acquisition - Your SAFR Server attempts to acquire a license from the SAFR license server when it's first run. If your SAFR system doesn't have Internet connectivity, see the [Offline Licensing](#) section below to see how to obtain a SAFR license.
- Licenses are bound to the primary SAFR Server. If you install one or more secondary servers for the purpose of load balancing or redundancy, the secondary servers acquire their licenses through the primary server.
- If you want to move your primary server to a machine with a different IP address, you must wait 24 hours between uninstalling the server and reinstalling it on the new machine. If you try to reinstall the SAFR Server before 24 hours has elapsed, you will get an unauthorized access error when the SAFR Server unsuccessfully attempts to get a valid licence from the SAFR License Server. After 24 hours has elapsed, however, a reinstalled SAFR Server will automatically (and successfully) reacquire a SAFR license.
 - Note that the previous behavior only applies to SAFR servers that are **uninstalled**. If, on the other hand, the IP address of your SAFR Server changes or changes to a hostname while the server remains installed, there is no problem; your server simply

informs the SAFR License Server of its new IP address or hostname the next time it checks in with the SAFR License Server.

Offline Licensing

If your SAFR system doesn't have Internet connectivity, do the following to get a SAFR license:

1. Obtain a license request file for the machine on which SAFR Platform is installed.
 1. On the machine that has SAFR Platform installed, run `get-license-request.py`.
 - For macOS: `/Library/RealNetworks/SAFR/bin/get-license-request.py`
 2. When prompted, enter the SAFR account name and password.
 3. The script will attempt to read `safrports.conf` to communicate with CoVi. If `safrports.conf` can't be found, then the script will use the default port, 8080.
 4. The script can be copied and run from any system that has Python 3 installed. If you run the script on a machine other than the one hosting SAFR Platform, use the `-n` parameter to provide the hostname of the machine hosting SAFR Platform.
 5. Running the script generates a file called `safr_license_request.json` in the same working directory as the script. Make sure to run the script in a directory that you have write access to.
2. Retrieve the license by sending the license request to SAFR Cloud.
 1. Copy the newly generated `safr_license_request.json` file and the script `get-license.py` to a machine that has Internet access and has Python 3 installed. `get-license.py` can be found at the following locations:
 - For macOS: `/Library/RealNetworks/SAFR/bin/get-license.py`
Note: On a macOS machine you might receive an error message about being unable to load SSL root certificates. This is an issue with the way Python handles SSL certificates on Macs. More information about the issue can be found [here](#). A workaround can be found [here](#).
 2. When prompted, enter the SAFR account name and password.
 3. You can use the `-p` parameter to tell the script where `safr_license_request.json` is located.
 4. You can use the `-e` parameter to set the environment value. (i.e. `prod`, `int2`, or `dev`) The default is `prod`.
 5. Running the script generates a file called `safr_license.json` in the same working directory as the script. Make sure to run the script in a directory that you have write access to.
3. Install the retrieved license onto your installed SAFR Platform.
 1. Copy `safr_license.json` to the machine running your SAFR Platform.
 2. Run `insert-license.py` to install the license onto your SAFR installation.
 - For macOS: `/Library/RealNetworks/SAFR/bin/insert-license.py`
 3. When prompted, enter the SAFR account name and password.
 4. The script will attempt to read `safrports.conf` to communicate with CoVi. If

safrports.conf can't be found, then the script will use the default port, 8080.

Getting Started with SAFR Platform on Windows or macOS

The computer used for the first installation of SAFR Platform acts as the primary server for the entire SAFR system. The primary server acquires a SAFR license that is then restricted to that machine. (See Licensing for details.) Any additional instances of SAFR Server you install under the same SAFR Local Account must be configured as secondary servers for the purposes of load balancing or redundancy and are linked to the primary server as described in Large Scale Deployments.

SAFR Platform Contents

The macOS SAFR Platform installation includes the following:

- **SAFR Server:** Includes the recognition engine, event server, several databases, and the Web Console. The databases contain stored enrolled face images, the identity information for the stored faces, and recognition events that have been generated by the SAFR system.
- **Desktop client:** One of the primary ways that administrators and operators can interact with the SAFR system. As such, the client can be used to enable camera connectivity, monitor video camera feeds, register users, view recognition events, and more.
- **Web Console:** Provides web-based access to the SAFR system. As such, the Web Console can be used to generate analytical reports, monitor video camera feeds, register users, view recognition events, and more.
- **SAFR Actions:** SAFR Actions is a GUI that facilitates configuring *SAFRActions.config*. *SAFRActions.config* is the file that defines all the defined actions for your SAFR System, as well as a couple fields that are used to connect ARES (and SAFR Actions) to your primary SAFR Server, whether that server is local or in the cloud. See Actions for more information about actions in SAFR.
- **ARES:** Actions Relay Event Service (ARES) is a cross-platform Java application that acts as the event listener that dispatches configured actions in response to events. ARES can provide replies on any event handled by the client that originates an event and is normally installed as a service when either SAFR Platform or SAFR Edge are installed. It is constantly active and is automatically started by the operating system on power-up.

Prerequisites

Before you begin the installation, ensure that you have the following prerequisites:

- **SAFR Local Account:** If you're not sure which account type you have, go to the [SAFR Download Portal](#). If SAFR Platform is listed among the downloads, then you have a SAFR

Local Account.

- **System requirements:** Ensure that your system meets the minimum system requirements listed here.
- **An up-to-date SAFR License:** See Licensing for information about SAFR Licenses.
- **SSL certificate:** SSL certificates are required if you want your SAFR Server to support HTTPS connections. If you don't care if HTTPS connections are supported, this prerequisite may be skipped. See SSL Certificate Installation for information about how to get an SSL Certificate.

Note: There are 2 situations where SAFR requires that your server supports HTTPS connections:

1. **iOS Devices:** The iOS Mobile client can only connect to the SAFR Server over HTTPS, so you must obtain an SSL certificate if you want to run the Mobile client on any iOS devices.
2. **Additional SAFR Servers:** SAFR Servers can only connect to each other over HTTPS, so you must obtain an SSL certificate if you want to install additional SAFR servers. Additional SAFR Servers are used when you want to scale your SAFR system beyond the processing capacity of a single machine. See Large Scale Deployments for additional information.

Download and Install SAFR Platform

To download and install SAFR Platform using, do the following:

1. On the computer where you want to install SAFR Server, open a web browser and go to the [SAFR Download Portal](#).
2. Sign in with your SAFR Local Account's credentials.
3. Once signed in, select your operating system from the menu and download the appropriate SAFR Platform installer.
4. After the download is complete, start the installation.
5. The Platform installer displays a *Choose Components* window where you can choose the features you want to install, such as:
 - SAFR Face Attribute Recognition
 - Age
 - Gender
 - Occlusion
 - Sentiment
 - Optimize GPU models
 - SAFR Peripheral Sub-systems
 - SAFR Actions (If you choose to install SAFR Actions, ARES will automatically also be installed.)
 - SAFR Reports
 - SAFR Logs
 - SAFR Web Console
 - SAFR Video Recognition Gateway (VIRGO)

- SAFR Video Recognition Gateway Administrator (VIRGA)
- Camera Extensions
 - Ximea
- GPU Support
 - NVIDIA Accelerated Detection
 - NVIDIA Accelerated Recognition
- SAFR Application (This refers to the Desktop client)

You may uncheck the boxes for any features you do not want to install. We recommend installing all the components except the VMS extensions the first time you install SAFR Platform.

In addition to whatever components you selected, SAFR Platform also always installs SAFR Server.

6. Follow the installer prompts as they guide you through the rest of the installation process. The final phase of the installation may take a few minutes to complete as it installs dependencies and runs the configuration scripts. Allow it to continue without interruption.

After the installation finishes, two icons will appear on your desktop: one labeled “SAFRActions” and another labeled “SAFR”. *SAFRActions* launches SAFR Actions, while *SAFR* launches the Desktop client. The SAFR Server automatically runs as a collection of background services.

Immediately following installation, the Platform installer opens the Desktop client and prompts you to log in with your SAFR Local Account. Make sure to log in; it’s important in acquiring the SAFR license.

Check Server Status

To check the status of your SAFR Server, use the *check* script. See the table below for the location of the script.

Platform	Script Name	File Location
macOS	check.sh	/Library/RealNetworks/SAFR/bin

The *check* script displays the status of all SAFR services. The following screenshot shows a server installation with healthy statuses for all its services:

SAFR Service Health		
State	Service	Description
UP	MongoDB Server	example.real.com:27017
UP	CoVi API Service - HTTP	http://127.0.0.1:8080/covi-ws/version
UP	CoVi API Service - HTTPS	https://example.real.com:8081/covi-ws/version
UP	GPU Face Service	http://127.0.0.1:8888/status
UP	Object Storage Service - HTTP	http://example.real.com:8086/health
UP	Object Storage Service - HTTPS	https://example.real.com:8087/health
UP	Event Service - HTTP	http://127.0.0.1:8082/version
UP	Event Service - HTTPS	https://example.real.com:8083/version
UP	Virga - HTTP	http://127.0.0.1:8084/health
UP	Virga - HTTPS	https://example.real.com:8085/health
UP	Reports - HTTP	http://127.0.0.1:8088/version
UP	Reports - HTTPS	https://example.real.com:8089/version
UP	Web Console - HTTPS	https://example.real.com:8091/signin
UP	Ares - SAFR Actions	ares.jar
UP	Apache HTTPD	httpd
UP	Virgo Service	virgod

UP	= Service is online
CERT	= Service is online but SSL certificate is invalid
????	= Service status unknown
DOWN	= Service is offline

Connect Desktop Clients

A Desktop client that is installed on the same machine as the primary server is automatically connected with your primary server; no additional actions need to be taken.

Desktop clients that are installed on machines other than the primary server, however, need to be configured so they can connect with the primary server. Clients that aren't connected to a server are nearly useless and have very limited functionality.

To connect a remote Desktop client, do the following:

1. On the remote machine download and install either SAFR Desktop or SAFR Edge for your OS from the [SAFR Download Portal](#).
2. Start the Desktop client. If prompted, cancel the camera login screen. Also cancel the SAFR Account login if it is displayed.
3. Click **Tools > Preferences**. On the **Account** tab, enter your user identifier and password for your SAFR Local Account.
4. Select *SAFR Custom* from the drop down menu of the *Environment* setting. Do one of the following:

Note: If you customized ports when installing SAFR Server, use the customized port values instead of the values listed below.

- If you are running the server without an SSL certificate, enter the following in the associated fields, substituting the server IP Address for **localhost**:
 - CoVi Server: http://localhost:8080/covi-ws
 - Event Server: http://localhost:8082
 - Object Server: http://localhost:8086
 - VIRGA Server: http://localhost:8084
- If you are running the server with an SSL certificate, enter the following in the

associated fields, substituting your server's hostname for **localhost**:

- CoVi Server: <https://localhost:8081/covi-ws>
- Event Server: <https://localhost:8083>
- Object Server: <https://localhost:8087>
- VIRGA Server: <https://localhost:8085>

5. Click **OK** to save the preference changes.

Getting Started with SAFR Edge with a Cloud Account

SAFR Edge installs the Desktop client as well as SAFR Actions, a programmable interface to create and manage responses to event triggers.

SAFR Edge Contents

The SAFR Edge installation for macOS includes the following:

- **Desktop client:** One of the primary ways that administrators and operators can interact with the SAFR system. As such, the client can be used to enable camera connectivity, monitor video camera feeds, register users, view recognition events, and more.
- **SAFR Actions:** SAFR Actions is a GUI that facilitates configuring SAFRActions.config. SAFRActions.config is the file that defines all the defined actions for your SAFR System, as well as a couple fields that are used to connect ARES (and SAFR Actions) to your primary SAFR Server, whether that server is local or in the cloud. See Actions for more information about actions in SAFR.
- **ARES:** Actions Relay Event Service (ARES) is a cross-platform Java application that acts as the event listener that dispatches configured actions in response to events. ARES can provide replies on any event handled by the client that originates an event and is normally installed as a service when either SAFR Platform or SAFR Edge are installed. It is constantly active and is automatically started by the operating system on power-up.

Prerequisites

Before you begin the installation, ensure that you have the following prerequisites:

- **SAFR Cloud Account:** If you're not sure which account type you have, go to the [Download Portal](#). If SAFR Cloud is listed among the downloads, then you have a SAFR Cloud Account.
- **System requirements:** Ensure that your system meets the minimum system requirements listed [here](#).
- **An up-to-date SAFR License:** See Licensing for information about SAFR Licenses.
- **An Internet connection:** Cloud deployments must maintain a network connection with the SAFR Server maintained by RealNetworks in the cloud at all times. Any components

that lose their connection to the cloud will immediately lose almost all their functionality.

Download and Install SAFR Edge

To download and install SAFR Edge, do the following:

1. On the computer where you want to install SAFR Edge, open a web browser and go to the [Download Portal](#).
2. Sign in with your SAFR Cloud Account's credentials.
3. Once signed in, select your operating system from the menu and download the appropriate SAFR Edge installer.
4. After the download is complete, start the installation.
5. The Edge installer displays a *Choose Components* window where you can choose the features you want to install, such as:
 - SAFR Actions (If you choose to install SAFR Actions, ARES will automatically also be installed.)
 - SAFR Video Recognition Gateway Administration (VIRGO)
 - GPU Support
 - NVIDIA Accelerated Detection

You may uncheck the boxes for any features you do not want to install. We recommend installing all the components the first time you install SAFR Edge. In addition to whatever components you selected, SAFR Edge always installs the Desktop client.
6. Follow the installer prompts as they guide you through the rest of the installation process. The final phase of the installation may take a few minutes to complete as it installs dependencies and runs the configuration scripts. Allow it to continue without interruption.

After the installation finishes, two icons will appear on your desktop: one labeled "SAFRActions" and another labeled "SAFR". *SAFRActions* launches SAFR Actions, while *SAFR* launches the Desktop client.

Recommendations for Camera Placement and Use

The best placement for cameras is at choke points where subjects tend to be looking straight ahead and traffic is going in one direction. Doorways are a common example of such a choke point. Doorways are also ideal in that they channel all subjects to a narrow access point where the pixel density of even a low resolution camera can be concentrated for the highest possible pixel density, resulting in higher quality recognition.

The top of an escalator is another ideal spot for camera placement. Escalators have the added advantage that subjects tend to be looking up, which negates some of the problems introduced by headwear such as baseball caps. Just like doorways, escalators also allow the camera pixels to be concentrated on an very narrow region.

The one major challenge with placing cameras in doorways is the backlight. Strong backlight can result in either very dark or poor contrast on a person's face, thus diminishing recognition capability. See the Backlighting Considerations section below for a discussion about how to overcome backlight difficulties.

In the most common use cases, cameras are positioned either inside the building pointed at external doors or mounted just outside external entrances pointed at approaching subjects. The major challenge here is the lighting. You can face the following challenges:

- Very high contrast - The subjects might be incredibly dark with a very bright background.
- Highly variable lighting conditions - Opening doors can change lighting conditions right at the moment the subject is walking past; the camera is still trying to adjust to the lighting conditions. This is especially true with automatic doors that open wide and remain open for a period of time.
- Daily light variations - The amount and quality of the light varies greatly between direct sunlight, indirect sunlight, daylight when it's cloudy, and nighttime.

Camera recognition accuracy should be tested under the following conditions:

- Bright sunny daytime conditions.
- Cloudy daytime conditions.
- Nighttime conditions.

Camera Positioning

We recommend that you obey the following guidelines whenever possible:

- If inside a building, place camera 20-30 feet in front of the door and about 10-12 feet high.
- The subject should be facing the camera nearly straight on.
- Camera resolution should be such that subjects' faces are at least 80 pixels high (120 is optimal) when at the door, larger if possible.
- When attempting to learn subjects, pixel density should be 180 or higher. (220 is optimal.)

Backlighting Considerations

There are three basic strategies to overcome backlight:

1. Use cameras with good dynamic range control. These are often abbreviated WDR (wide dynamic range) or HDR (high dynamic range). These cameras allow for dark regions to appear brighter and very bright regions not to get over saturated with light.
2. Use cameras that have good shutter control. In some cases this might mean the ability to define the aperture (the degree to which the lense opens) and in other cases it means good exposure (the duration of time in which a shutter opens). A shutter that opens wide

can let in more light while a camera that holds the lense open longer lets in more light. Both techniques can be used to allow more light onto the region of the face but they both have the effect of oversaturating the bright regions.

3. A combination of strategies 1 and 2.

Given the choice, it's better to have good dynamic range than to use shutter controls to manage backlight. If you are using shutter control, however, some basic steps you can take are as follows:

- Place the bound on the slowest shutter speed allowed. As people are moving through the field of view (FOV), maintaining high shutter speed is important to obtain blur-free images.
 - Set the slowest shutter speed to at least 1/90 second.
 - In some cameras, if the shutter speed's lower bound is not settable for auto mode, you may need to use Shutter Priority mode.
- If faces are still dark, adjust the exposure compensation to brighten the faces. The background becomes overexposed, but that works for facial recognition.
- If there are specular reflections or faces that are still too dark, turn on Highlight Compensation.

The previous approach is appropriate for the situation where varying outdoor conditions also vary the amount of light reflected from the face. Light intensity is simply boosted above what the cameras would choose automatically and enhancing the image to reduce exposer variance.

In cases where outdoor conditions only generate backlight (light from behind the subject's face) and there is minimal variation in lighting from inside (e.g. there are few windows so indoor illumination on the face is mostly constant), it is more appropriate to place the camera in fully manual mode and set Shutter Speed, Iris, and Gain values manually to properly expose the face while allowing the background to be overexposed. In this mode, the camera makes no auto-adjustments and is not thrown off by momentary bursts of light due to a door opening or other momentary reflections. To do this:

- Set shutter speed to 1/90 or higher.
- Open the iris, increasing the f-stop for the iris (aperture) until the face is bright enough.
- Focus the camera on the sweet spot or optimal point of recognition where people are most likely to face the camera.

Increasing the iris reduces the depth of field (distance during which the face is in focus). Increasing the iris increases the quality of the image but reduces the amount of time the image is in focus and viability for recognition. In either case, focus the camera on the optimal point of recognition where people are most likely to face toward the camera.

General Lighting Considerations

Success with recognition indoors or outdoors depends on lighting conditions.

- Should have a light source that hits the front of the face.
- Outdoors:
 - Typically always has more light during daytime unless there is an awning blocking the light from above and front as the subject approaches.
 - At nighttime, a light source is required behind the camera to illuminate the front of the subject's face.
 - Avoid a direct line between the sun and the camera lens.
 - The camera lens should have sun/rain shade for effective operation.
- Indoors:
 - Need to contend with the backlight illumination.
 - There should be ample ambient lighting conditions and relatively uniform light on the subject's face for the best results.

Additional Information

- Anyone setting up cameras for facial recognition should be fully familiar with digital photography concepts described in the following 15 minute video:
 - <https://www.youtube.com/watch?v=F8T94sdiNjc&app=desktop>
- The following video covers some specifics on backlight compensation:
 - <https://www.dpmag.com/how-to/tip-of-the-week/combat-backlighting-with-exposure-compensation/>

Set up ONVIF IP Cameras

A camera must be correctly configured for authentication via the ONVIF protocol to work.

Enable ONVIF

Make sure that ONVIF is enabled in the camera settings. The precise procedure for how this is done depends on the make and model of the camera.

Configure the Date and Time

The camera's configured date & time must not differ by more than +/- 5 seconds from the machine you're connecting the camera to. Follow these steps to ensure that the camera date & time are configured correctly:

1. Set the camera Time Zone to the local time zone. (e.g. GMT-8 if you're in Seattle)
2. Disable daylight savings time (DST) adjustments. The Network Time Protocol (NTP) will take care of this automatically.
3. Set the NTP server to `time.google.com` and port 123.
4. Synchronize the camera time to the time on your computer. The web interface usually

has a button that allows you to do this.

5. Enable the NTP service.

The end result should be that the camera's date and time are up-to-date and that the NTP service is enabled to keep it up-to-date.

Some camera web UIs will show an incorrect/strange/nonsensical time after you've set the time zone to your local time zone. Do not change the time zone away from your local time zone! It must be set to the local time zone for ONVIF authentication to work.

Configure the Camera's ONVIF User

Many cameras maintain two sets of users: one set of web users and a second (and independent) set of ONVIF users. These cameras with 2 sets of users do not automatically create an ONVIF user even when a new web user is created.

Be sure that your camera has at least one ONVIF user with administration privileges. If there aren't any ONVIF with administration privileges, ONVIF authentication will not work.

Connect Cameras to SAFR

SAFR supports USB and integrated cameras, which are always auto-detected. SAFR also supports the standard Open Network Video Interface Forum (ONVIF) camera auto-discovery protocol used by most IP cameras. When ONVIF discovery is enabled, IP cameras are also auto-detected. To connect an auto-detected camera, simply click on the **Select Camera** menu on *Camera* window.

In some cases, however, ONVIF discovery is disabled by default. (ONVIF is sometimes disabled by default for security reasons.) This makes the camera effectively invisible to SAFR and SAFR is unable to discover or communicate with the camera automatically. See the [Manually Add and Configure IP Cameras](#) section to see how to manually add such cameras.

Manually Add and Configure Cameras

1. On the Desktop client, click **Tools > Preferences**, then select the **Camera** tab.
2. In the lower left corner, click **+**.
3. In the **Name** field, enter a descriptive name for the camera.
4. In the **URL** field, enter the RTSP (Real Time Streaming Protocol) URL to the live video feed of the camera or an RTSP server. For information on how to find the RTSP URL for your camera, see the **Determine the IP Address and Streaming URL for the Camera** section below.

Name:

URL:

Cancel

Add

After a camera is successfully connected, you can configure it. See camera preferences for more information on available options and how to use them.

Determine the IP Address and Streaming URL for the Camera

The IP addresses of many security cameras can't be auto-discovered because their ONVIF has been disabled for security reasons. This makes the camera effectively invisible to SAFR and thus SAFR is unable to find and communicate with the camera automatically. For SAFR to discover and connect to a camera automatically, you must enable ONVIF.

If you already know the IP address for the camera, do the following:

1. Connect to the camera from a web browser by typing the URL of the camera's IP address (e.g. `http://10.124.13.34`).
2. Find its streaming URL (starting with `rtsp://`). You need the streaming URL to enter when you add the camera to SAFR. The `rtsp://` URL you enter into SAFR most likely includes the camera user ID and password.

The streaming URLs, while different for different camera manufacturers, tend to follow the same format for different camera models of the same manufacturer. The following table includes a few examples of camera streaming URLs for different camera manufacturers. To use them for your camera, match your camera make to an example listed in the table and enter the SAFR provided streaming URL while replacing example values with actual values for your camera. Replace username, password, and IP address (for example, 10.124.13.32) with actual ones configured for your camera. Even if your camera model is not the same as the one listed, there is a good chance the streaming URL provided in the table works if your camera manufacturer is the same.

Camera Make	Camera Model	Example rtsp:// URL
Avigilon	5.0-H3-DO1-IR	<code>rtsp://username:password@10.124.13.32/defaultPrimary?streamType=u</code>
Axis	Q6128-E	<code>rtsp://username:password@10.124.13.32/axis-media/media.amp</code>
Dahua	HFW5421E-Z	<code>rtsp://username:password@10.124.13.32/cam/realmonitor?channel=1&subtype=0</code>
HikVision	DS-2CD4185F-IZ	<code>rtsp://username:password@10.124.13.32/h264</code>

Camera Make	Camera Model	Example rtsp:// URL
Mobotix	M26	rtsp://username:password@10.124.13.32/mobotix.h264
Panasonic	WV-SFV781L	rtsp://username:password@10.124.13.32/MediaInput/h264/stream_1
Samsung	SND-L6013R	rtsp://username:password@10.124.13.32/onvif/profile2/media.smp
Sony	SNC-VM772R	rtsp://username:password@10.124.13.32/video1

If you are unable to find your camera's RTSP address, do the following:

1. Determine the make, model, and manufacturer for the camera.
2. Contact your system administrator or the camera manufacturer. They should be able to provide you with instructions for configuring the camera, determining its IP address, enabling it to be discovered by IP address, and/or finding the camera rtsp:// URL.

Save your camera configuration

After you complete your manual camera connection, we recommend that you export and save the created camera connection configurations. Although your configurations are automatically stored in the Desktop client, they cannot be shared with other SAFR components that are installed on different computers unless you export the camera configurations. Exporting your camera connection configurations makes your setup work shareable to other SAFR components and preserves it in case the Desktop client is re-installed later on different hardware.

To export your camera configurations, do the following:

1. In the Desktop client menu, click **Tools > Preferences** and select the **Camera** tab.
2. In the lower left corner, click the gear icon.
3. Click **Export Configurations**.
4. Specify the file name and location where the configurations are to be saved.

Note: Your camera connection configurations are saved in a file with an .acc extension. This file may contain your camera access credentials, so save it in secure location.

To import camera configurations, do the following:

1. In the Desktop client menu, click **Tools > Preferences** and select the **Camera** tab.
2. In the lower left corner, click the gear icon.
3. Click **Import Configurations**.
4. Specify the file name and location where the configurations are located.

Interaction of Auto-detected and Manually Configured Cameras

Auto-detected camera configurations and manually entered camera configurations may exist for the same camera. This situation does not cause a conflict. Each configuration can be separately selected; you can choose a method of connection when selecting a camera from the feed window.

Your exported camera connection configurations include only manually configured cameras. Auto-detected camera configurations are dynamic and are discovered by the SAFR client at start and when *Camera Preferences* are opened. Because auto-discovered cameras are dynamically discovered, they may not appear for a few seconds after the application starts, so make sure to wait several seconds to give the discovery process time to complete.

To Delete a Camera Configuration

To delete a camera configuration, do the following:

1. In the Desktop client menu, click **Tools > Preferences** and select the **Camera** tab.
2. In the left hand field select the camera you want to delete.
3. In the lower left corner, click -.

Note: Auto-detected camera configurations cannot be deleted. They are dynamically discovered.

Connect to a Video Feed

The Desktop client requires a video feed window to be open for each camera feed (or video file) it is monitoring. The video feed window not only facilitates SAFR monitoring of the video but can also present additional information overlaid on top of the video to assist staff in interpreting the scene in the video. For more information on the additional overlaid information, see [Interpret Video Feed Overlays](#).

The Desktop client automatically detects integrated, USB, and [Open Network Video Interface Forum](#) (ONVIF) compatible IP cameras. While you're becoming familiar with SAFR, we recommended that you plug in a single USB camera (aka a web cam). Only after you've spent some time learning SAFR should you attempt to connect to additional IP cameras.

The instructions on this page assume you have at least one camera detectable from the computer where SAFR is installed. See [Connect Cameras to SAFR](#) for information about how to connect a camera.

Connect to a Camera Video Feed with the Desktop Client

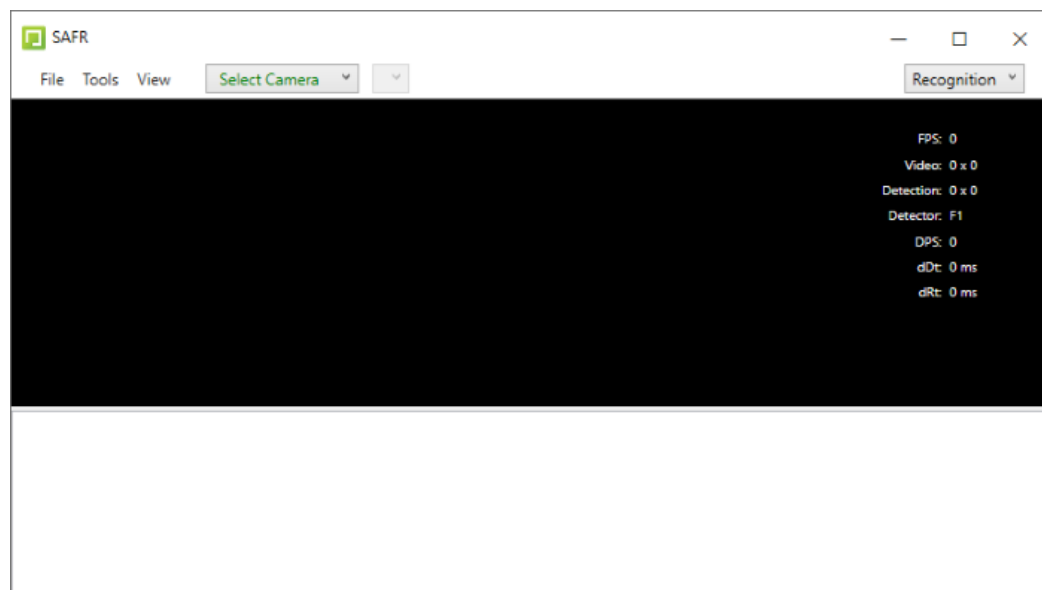
If you have a local deployment, you must first install and configure SAFR Platform before connecting to a live video feed. For information on how to do this, see [Getting Started with SAFR Platform on Windows or macOS](#) or [Getting Started with SAFR Platform on Linux](#).

If you have a cloud deployment, your Desktop client should automatically be configured to connect to your SAFR Server in the cloud when you install it.

To connect to a video feed, do the following:

1. From the Desktop client menu, click **File > New** to open a new *Camera* window. **Note:** It's possible that your client will default to the *Camera* window when you first start it. If this happens, there's no need to open a new *Camera* window.

Below is an example *Camera* window.



2. Along the top of the *Camera* window, select a camera from the **Camera Selector** drop-down menu. The menu displays all detected cameras.
3. Set the camera frame rate and resolution in the **Video Feed Profile Selector** to the right of the **Camera Selector** menu. Frame rate and resolution selection are only configurable for USB and IP cameras auto-discovered via ONVIF. For IP ONVIF cameras, only configured ONVIF media profiles are able to be selected. The higher the frame rate (i.e. frames per second) and resolution for a video feed, the more processing power is required to monitor and collect data from the video feed.

Once you complete this procedure, SAFR receives, monitors, and processes the video feed from the camera. The Desktop client *Camera* window must remain open for SAFR to continue to monitor the video feed. If the window is closed, SAFR no longer receives the video feed and no longer monitors it. The *Camera* window can be minimized without affecting the monitoring of the feed.

Select a Video Processing Mode

A variety of different video processing modes are supported to accommodate different

monitoring and security needs, as described in the table below. Each mode can be customized through the Detection, Tracking, Recognition, Events, and User Interface tabs of the **Tools -> Preferences** menu.

SAFR Video Processing Mode	Description
Recognition	<ul style="list-style-type: none"> • This is the default mode typically used for set-up, validation, and experimentation. • Only reports enrolled individuals. • No events are generated or recorded.
Import	<ul style="list-style-type: none"> • Any face that can be clearly seen but is unidentified will be automatically registered. • Faces that are already registered are only recognized and do not create additional entries in the Person Directory. • Additional different face images (e.g. from different expressions) may be added to the existing faces in the Person Directory if they improve the recognition of the person. • No events are recorded.
Similar	<ul style="list-style-type: none"> • Registered faces that are the most similar to the face observed in the video are shown in real time in the <i>Camera</i> window. • Each similar face displays a % match. Anything less than 100% match is not a certain match. • No event information or images are recorded.
Secure Access	<ul style="list-style-type: none"> • Secure access uses strict criteria for confirming the identity of the face in the view of the camera. • Records events and images for recognized faces. • Listens for event replies and displays them on the screen. • Typically used for door access control. • Events and images are not recorded for unrecognized faces.

SAFR Video Processing Mode	Description
Secure Access with Smile	<ul style="list-style-type: none"> • Uses strict criteria for confirming the identity of the face in the view of the camera. It also looks for transitions in the facial expression (e.g. non-smiling to smiling). • Records events and images for recognized individuals. • Listens for event replies and displays them on the screen. • Typically used for door access control where it is necessary to guard against identity impersonation via photo. • Includes all functionality of <i>Secure Access</i> mode and can be used to allow access on first sight recognition for a certain time (e.g. when security staff are on duty) and can change to a higher degree of security (e.g. recognition with smile expression change) at set times (e.g. when security staff is off duty). • Events and images are not recorded for unrecognized faces.
Secure Access with Liveness	<ul style="list-style-type: none"> • Secure access uses strict criteria for confirming the identity of the face in the view of the camera. • Records events and images for recognized individuals. • Listens for event replies and displays them on the screen. • Typically used for door access control where you want to use liveness detection to guard against face spoofing. • See Pose Liveness Detection for information about liveness detection.
Registration Kiosk	<ul style="list-style-type: none"> • Allows self-registration into the SAFR system. • Optimized to work from a tablet. We recommend the iPad Pro. • Can be customized in the User Interface preferences tab to display different prompts, gather additional information, or restrict registration with a minimum age. • Records registration and appearance events and associated images of registered persons. • Events and images are not recorded for unrecognized faces unless they initiate the registration process.

SAFR Video Processing Mode	Description
Enrolled Monitoring	<ul style="list-style-type: none"> • Facial recognition, events, and images are only recorded for registered/recognized faces. • Images and events are not recorded for unrecognized faces.
Anonymous Traffic Monitoring	<ul style="list-style-type: none"> • Only gender and age information is detected and recorded anonymously for faces viewed by cameras. • Images and biometric information are not recorded.
Enrolled and Anonymous Traffic Monitoring	<ul style="list-style-type: none"> • Recognition events are recorded for registered individuals. • Anonymous age and gender information is also recorded for unknown/unrecognized faces. • No images are recorded for recognized or unrecognized faces.
Enrolled and Unique Traffic Monitoring	<ul style="list-style-type: none"> • Events are recorded for registered and unknown individuals. • Any faces that can be clearly seen but are currently unknown are automatically registered. • Age and gender information is recorded for all clearly seen faces. • Images are recorded for all faces.
Enrolled and Stranger Monitoring	<ul style="list-style-type: none"> • Events are recorded for registered and unknown individuals. • Faces that are clearly seen but aren't currently registered are reported as strangers. • Faces that can't be seen clearly enough to attempt recognition are reported as unrecognizable. • Images are recorded for all faces.

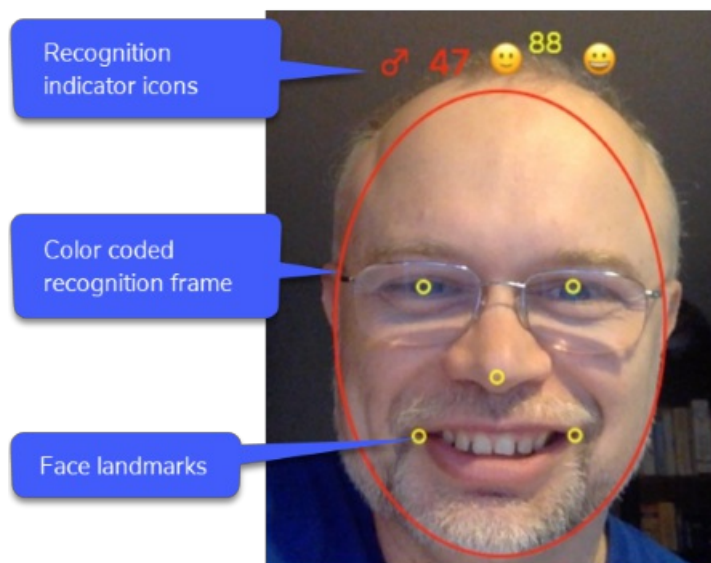
Recommendations for the Best Video Experience

- Use the highest resolution available (4K) if you need to monitor an area of 5 meters or wider.
- To monitor a narrow area of approximately 2-3 meters, 1080p video is sufficient.
- For up close door access applications, 720p video offers adequate quality.
- For resolutions of 1080p or higher, we recommend 15 frames per second.

- For cameras used in *Secure Access With Smile* mode, we recommend 30 frames per second at 720p resolution.
- Generally, one computer can support one 4K camera (or 2 HD cameras) for every 2 CPU cores depending on the camera make and model.

Interpret Video Feed Overlays

Video feed overlays are available in the video feed window. Depending on the View menu options you choose, various supplemental information is displayed in real time in the form of overlays and readouts to help monitor the video feed.



Color Codes for the SAFR Recognition Frame

The following colors are used to indicate the level of recognition for a face:

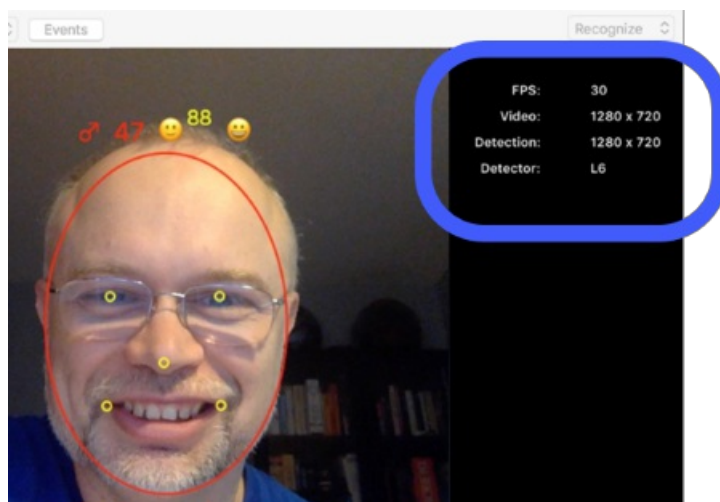
- **Gray**: Unrecognizable. Either the face does not meet minimum quality values to be recognized or the response from the attempted recognition has not yet arrived.
- **Purple**: Stranger. The face met sufficient quality for recognition but was not recognized and did not meet minimum quality to be registered.
- **Cyan**: Identified as a close match to recognized user but not 100% identification.
- **Blue**: Registered person without a name. The face was recognized as matching one already in the Person Directory.
- **Green**: Registered person with a name. The face was recognized as matching one already in the Person Directory.
- **Yellow**: Concern. The registered face has been tagged as a concern.
- **Red**: Threat. The registered face has been tagged as a threat.

Recognition Indicator Icons

The indicator icons indicate the following:

- Gender
- Age
- Sentiment
- Sentiment score
- Smile

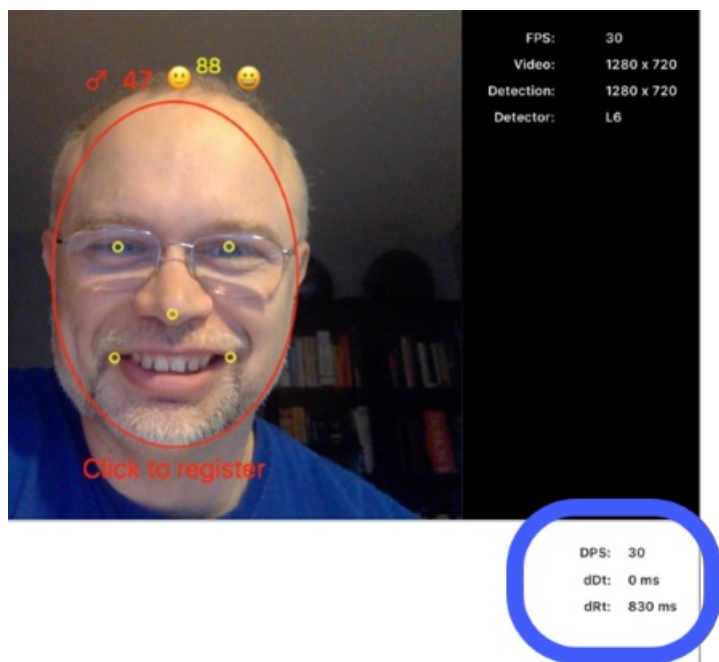
General Video Information



The following information is provided in the Desktop client video display:

- **FPS:** The number of frames per second being captured by the camera.
- **Video:** Video resolution. (e.g. 1280x720)
- **Detection:** Face detection resolution.
- **Detector:** The detector CPU capacity level. It's based on the number of CPU processing cores.

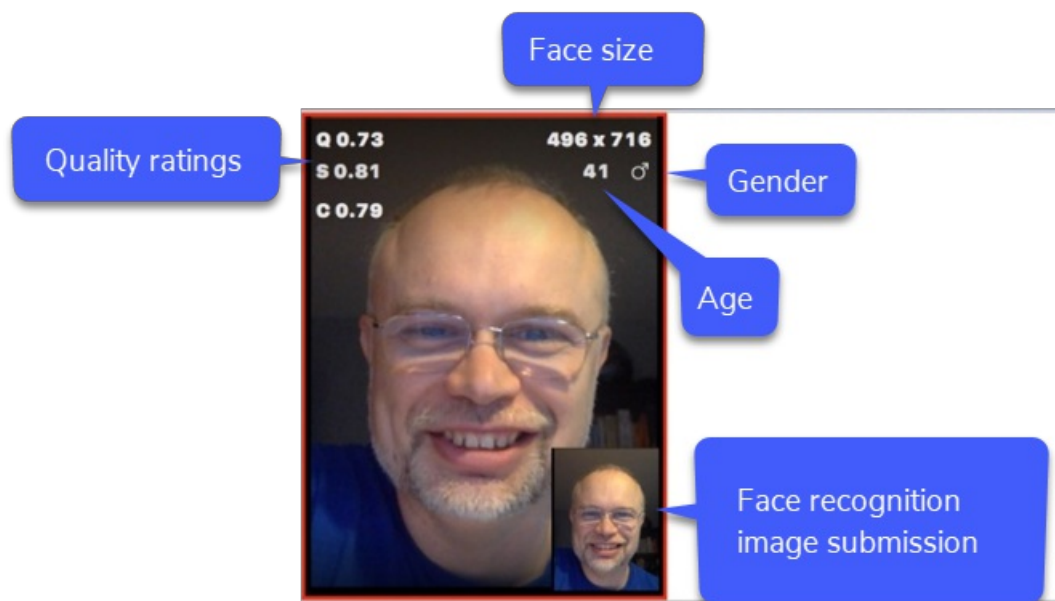
Video Processing Latency Information



- **DPS:** Frame detections per second.
- **dDt:** Detection time. For example, how long it is taking to detect a face.
 - This time should normally be in 20-50ms range.
- **dRt:** Recognition time. For example, how long it is taking to recognize a face.
 - This time should normally be in the 60 - 250 ms range for only identity recognition.
 - If age, gender, and sentiment are also being recognized, the time can be expected to be up to 450 ms.

If your processing times are longer than indicated here, it may be an indication your system is overloaded. You should look to offload some of your video feed processing to other computers.

Face Detection Information



There are 3 main image quality metrics:

- **Q:** Center pose. Represents how directly the face is looking at the camera. A face looking directly at the camera would receive a score of 1. The more the face looks up, down, left, or right of the camera, the more this metric is reduced.
- **S:** Face sharpness. Represents how clear the image is. A score of 1 represents a perfectly clear image, while 0 represents an extremely blurry image.
- **C:** Face contrast. Represents the color contrast within the image. A score of 1 represents an image with very high contrast, while 0 represents very low contrast.

For guidance on these 3 metrics, see [Image Quality Metrics Guidance](#). These metrics can be configured in the recognition preferences.

- **Face size:** Face size is the resolution of the image with a 25% margin. It can be used to ignore background (smaller) faces or to require an up-close high resolution image to be presented before the face is learned by the SAFR system. See detection preferences and recognition preferences for information on how to customize SAFR behavior based on the detected face size.
- **Gender:** Displayed as an icon if gender recognition is enabled.
- **Age:** Age of user if age recognition is enabled.
- **Face recognition image submission:** The thumbnail in the lower right corner is the image submitted for facial recognition. This is only displayed if you select *Recognition Candidates* (for macOS) or *Detection List and Recognition Details* (for Windows) from the View menu.

View Video Feeds Status

Video feeds status provides real-time monitoring of the Desktop client, the Mobile client, and

the VIRGO client within an account. While VIRGO clients support both remote monitoring and configuration, the Desktop and Mobile clients only support remote monitoring and must be configured locally in their GUI.

Note: This feature is available with the Web Console or on macOS Desktop client installations.

Enable the Desktop or Mobile Clients for Remote Monitoring

Both the Desktop and Mobile clients can be enabled for remote status monitoring and remote viewing.

To enable those clients for remote monitoring:

- For macOS, click the **SAFR > Preferences > Account** tab, and select the **Report Status** and **Allow Remote Viewing** check boxes.

The screenshot shows the 'Preferences' window with the 'Account' tab selected. The 'User Identifier' field is highlighted with a blue border and contains the text 'test304'. Other fields include 'User Password' (masked with dots), 'User Directory' (main), 'User Site' (Main Building), 'User Source' (South Entrance), and 'Similar Directory' (main). The 'Environment' is set to 'LOCAL' with a dropdown arrow and a note '(requires app restart)'. At the bottom, there are two checked checkboxes: 'Report Status' and 'Allow Remote Viewing'.

- In the Web Console, click **Video Feeds > Processor Status**.

The screenshot shows the 'Processor Status' page in the SAFR Web Console. It features a table with columns for device name, CPU usage, date added, last config, and last status. There are also filter and sort dropdowns at the top of the table.

Filter By:	Sort By:
All	Date Added

<input type="checkbox"/> argus/argus-MacBook.../1.3.069	CPU: 33%	Date Added: 07/22/19, 10:00	Last Config: 07/22/19, 11:17	Last Status: 07/23/19, 11:27	Config
* Feed: watchlist macbook Status: OK? FPS: 30 DPS: 30 CPU: 33% View					
<input type="checkbox"/> argus/argus-MacBook.../1.3.068	CPU: 22%	Date Added: 07/16/19, 23:54	Last Config: 07/16/19, 23:54	Last Status: 07/16/19, 23:55	Config
* Feed: m1-BFD0F92E-EB74-51D... Status: OK? FPS: 30 DPS: 29 CPU: 22% View					

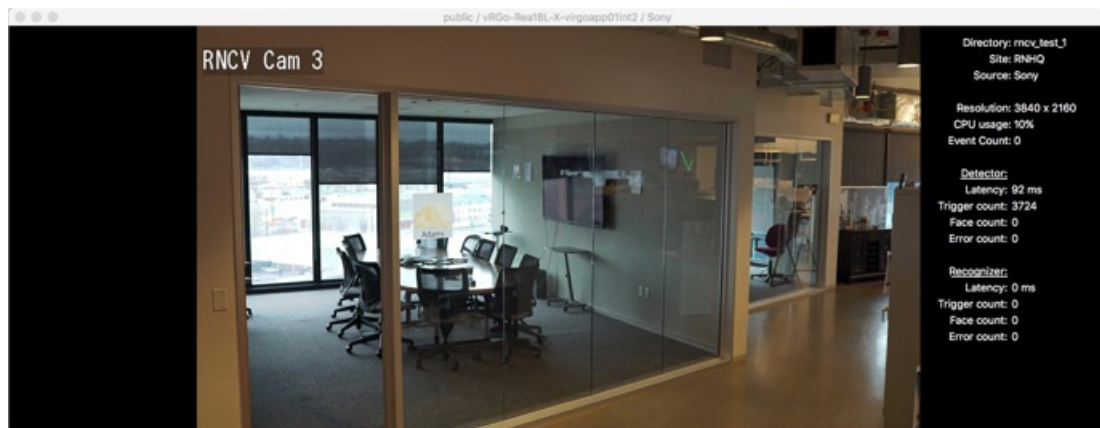
The Video Feeds Status Window (aka Processor Status Window)

The Video Feeds Status (or Processor Status) window can be accessed from the **SAFR** menu of the Desktop client for macOS and from the Web Console Video Feeds page. It displays real-time status of all VIRGO, Desktop, and Mobile clients enabled for status reporting.

The screenshot shows the 'Video Feeds Status' window. Annotations include:

- Shows all VIRGO and SAFR clients and their associated feeds
- Account to which client belongs
- Client unique id
- Red background indicates client is offline
- Client version
- CPU used by client
- Date client came online
- Date when last configuration was applied to the client
- Date when client last reported status
- iPad client processing 1 video feed
- VIRGO client on Linux processing 2 out of configured 5 video feeds
- Names of feeds configured in the client
- Feed Status:
 - INACTIVE: not enabled
 - PREROLLING: starting to process
 - OK: normally processing
 - ERROR: attempting to recover from error
 - FAILED: failed to recover from error
 - COMPLETED: processing ended
- Video feed frames per second
- Video feed frames processed for detection per second
- CPU utilization for feed processing
- Enables client feed configuration
- Shows feed live video view

The Live Status Video View



The **Live Status Video** view can be accessed by clicking the **View** button next to an active feed in the **Video Feeds Status** window. You can display the **Live Status Video** view for all active VIRGO clients and for all Desktop and Mobile clients for which you have enabled **Allow Remote Viewing**.

You can simultaneously view live status video of as many feeds as you like from a single remote viewer. Only one viewer of the status video is allowed at any given time, however.

The **Live Status Video** view is displayed at low frame rates (~1 frame per second) and is intended for cursory inspection or monitoring. Full fidelity live video view, including recognition overlays, is available only in the video feeds window of the Desktop and Mobile clients to which the camera is connected.

The top of the status video view displays the *Account*, *Client ID*, and the *Name* of the video feed.

On the right-hand side of the view, overlaid text displays additional information about the feed.

- **Directory:** Face Directory used for face recognition in the video feed, as specified in Account Preferences.
- **Source:** Source label for the video feed, as specified in Camera Preferences.
- **Site:** Site label for the video feed.
- **Source:**Source label for the video feed.
- **Resolution:** Video feed resolution.
- **CPU Usage:** CPU percent used for video feed processing.
- **Event Count:** Number of events reported since start of video feed processing.

Detector:

- **Latency:** Face detector latency in the last second of operation.
- **Trigger count:** Number of times face detector was triggered.
- **Face count:** Number of faces detected since the start of the video feed.
- **Error count:** Number of face detector errors since video processing started.

Recognizer:

- **Latency:** Face recognition latency in the last second of operation.
- **Trigger count:** Number of times face recognition was triggered since the start of the video feed.
- **Face count:** Number of faces recognized since video processing started.
- **Error count:** Number of face recognition errors since video processing started.

Manage People in the Person Directory

The Person Directory contains a list of all people stored in the user directory location specified under Account Preferences. To open the directory from the Desktop client:

- For macOS, the **People** menu item is available under the **SAFR** menu.

By default, the list is displayed in chronological order with the most recently added displayed first. You can also search and filter identities by *Name*, *Person Type*, *ID Class*, and *Home Location*. All 4 of those properties can be changed by clicking the available fields to the right of the identity's picture.

- Metadata applied to identity groups is applied to all identities within the group. Changing

these properties for any identity within a group will cause the change to be applied to all identities within that group.

- Groups are alternative identities belonging to a single person. While rare, a person may require such grouping to fully cover all different face modalities by which he or she can be recognized.

Double click the identity entry to view or edit even more information associated with the identity.

- The *Id Class* field is important and can be used to define a person as a *Concern* or *Threat*.
- *Moniker* is an advanced feature used to realize two factor authentication with visual badges.

You can also perform the following actions on identities in the People Directory:

- **Regroup**: Removes selected face from their existing groups (if any) and forms a new group of faces to represent a new identity. Root identity is always the earliest one added to the directory.
- **Delete**: Deletes selected identities and all information associated with the identity from the directory. All information associated with the identity is removed.
- **Export**: Exports a face image into an image (.jpg) file on the local drive.
- **Refresh**: Reloads the people directory page making sure up-to-date information is displayed.

Add a Person Type or Home Location

In the Person Directory, click **Add Person Type**, and then type the *Person Type* you want to assign (for example, Staff, Guest, or Maintenance). Likewise, you can click **Add Home Location** and type text representing a person's home location.

Best Practice: You can create and customize as many *Person Types* and *Home Locations* as you like, but we recommend keeping the list short (less than a dozen or so) because short lists are easier to maintain. As *Person Types* are entered for a few registered individuals, *Person Types* that are already entered become available for selection once **Add Person Type** is clicked, which makes designation easier for new registration. The same is the case for *Home Location*. The system knows of all previously entered *Home Locations* and offers them in the menu when **Add Home Location** is clicked.

Importing and Registering People

There are three main ways to register people to SAFR's Person Directory: cameras, photos, and recorded video. Imported people are registered to the Person Directory and stored in the directory specified in the User Directory setting of your Account preferences.

Register People Using a Camera Connected to the Desktop Client

1. Select the connected camera you want to use by clicking on **File > New** to open the Desktop client's *Camera* window, then select a camera from the **Select Camera** menu.
2. Set your **video feed processing mode** (located in the upper right hand corner) to one of the following modes: *Recognition*, *Import*, or *Learn and Monitor*. *Recognition* is considered the default mode for set-up validation and experimentation.
3. All the faces in view of the camera will initially have a grey overlay, which indicates one of two things:
 1. Your client or console hasn't received a response from its attempted recognition from the SAFR Server yet, or
 2. The face does not meet minimum image quality metric values and recognition cannot be successful. If the grey overlay persists, then the problem is the image quality. Try cleaning your camera lens or adjusting your camera placement to fix the problem.
4. When the overlay turns purple, the face has sufficient image quality for recognition by SAFR but it isn't recognized by SAFR because it hasn't been registered yet. To register the face, double click the face and the *Register* dialog will open.
5. You can choose to enter a name for the face if you want, but it's not required. Click **Register** to complete the registration.
6. The color of the face's overlay will change to either green (if you named the face) or blue (if you didn't name the face). Both of those colors mean that SAFR recognizes the face. For more information on overlays, see *Interpret Video Feed Overlays*.

Register People Using the Mobile Client

Another way to register faces is by using a Mobile client installed on an iOS or Android device. For more information, see *Connect a Registration Kiosk*.

Register People by Importing Faces from Picture Files

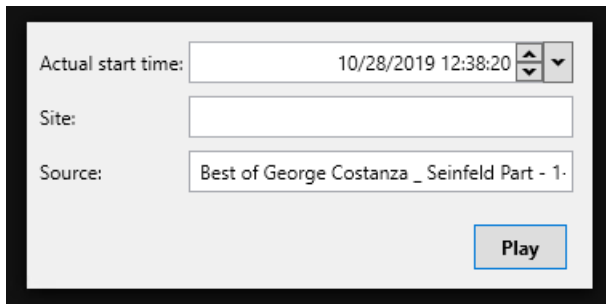
To import faces from picture files, do the following:

1. Open either the Desktop client (by clicking on the **SAFR** icon on your desktop) or the Web Console. (See *Access the Web Console* for information on how to do this.)
2. On the Desktop client, click **File > Open** and select an image file. On the Web Console, click on the **People** tab, click on the up arrow symbol in the upper right hand corner, select **Pick File** in the dialog window that pops up, and select an image file.
3. Image files are usually .jpg, .jpeg, or .png files. If the file you selected has multiple faces on it, then SAFR will import all the faces on the image.
4. When you import facial images, you may be prompted to resolve any duplicate and/or low-quality image conflicts that may have arisen.

Register People from a Video File

You can open a saved video file to recognize and extract facial recognition data. To do so, do the following:

1. Open the Desktop client.
2. Click **File > Open**, and then browse to any saved .mp4 file to open it.
3. If you're on a Windows machine and you have event reporting enabled for the currently selected video processing mode, (located on the Events Preferences tab) the dialog below will open. (If you don't meet both of these conditions, then the video will simply open.)








- **Actual start time:** The timestamp that the video will acquire when you press **Play**. (e.g. In the example above, the played video's timestamp would start at 12:38, 10/28/2019) The input box starts 'live' and keeps up-to-date with the local time. When you interact with the time or set the focus, the input box stops being live. **Note:** Deleting the timestamp and leaving the field blank is valid, despite the red outline that the field acquires. Of course, if you do leave the field blank, the video won't have a timestamp, as expected.
 - **Site:** The *Site* label that will be applied to all events generated by the video. This field is auto-populated with your *User Site* preference located in the Account Preferences.
 - **Source:** The *Source* label that will be applied to all events generated by the video. This field is auto-populated with the name of the video.
4. Set the video file's video feed processing mode to *Recognition*.
 5. SAFR will proceed to register any unregistered faces that appear in the video.

Image Quality Metrics Guidance

Choosing to import images that have been flagged as "low-quality" will cause more false positives to occur as SAFR incorrectly identifies newly scanned faces as identical to the low-quality facial image. Greater discrepancies between the recommended metric value and the

actual metric value will result in more false positives. Similarly, having more than one metric value be poor or very poor will also result in more false positives.






Center Pose

				
Center Pose = .89	Center Pose = .76	Center Pose = .54	Center Pose = .34	Center Pose = .21

Center pose represents how directly the face is looking at the camera. The more the face looks up, down, left, or right of the camera, the more this metric value is reduced from 1. Similarly, if the face is tilted in any way (e.g. the person's chin is pointing at a corner of the image) this metric value is reduced. The default recommended minimum value for this metric is .59. You can adjust the recommended minimum value by going to **Tools** → **Preferences**, clicking on the **Recognition** tab, then adjusting the **For merging** slider in the **Minimum required center pose quality** section.

Quality Label	Metric Range	Description
Excellent	0.7 - 1.0	Full recognition accuracy can be expected under all conditions.
Good	0.6 - 0.7	Very good recognition accuracy can be expected in general but may confuse closely related family members.
Marginal	0.45 - 0.6	Good recognition but may result in occasional failures.
Poor	0.3 - 0.45	Recognitions can be performed to significant extent but may produce false recognitions.
Very Poor	0.0 - 0.3	Recognitions can still be performed but with significant possibility of confusing similar faces.




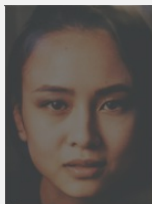

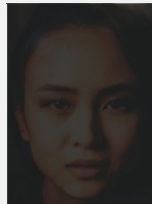
Sharpness

				
Sharpness = .79	Sharpness = .62	Sharpness = .58	Sharpness = .35	Sharpness = .22

Sharpness represents how clear the facial image is. The more blurry the face is, the more this metric value is reduced from 1. The default recommended minimum value for this metric is .45. You can adjust the recommended minimum value by going to **Tools** → **Preferences**, clicking on the **Recognition** tab, then adjusting the **For merging** slider in the **Minimum required face sharpness** quality section.

Quality Label	Metric Range	Description
Excellent	0.7 - 1.0	Full recognition accuracy can be expected under all conditions.
Good	0.6 - 0.7	Very good recognition accuracy can be expected in general but may confuse closely related family members.
Marginal	0.45 - 0.6	Good recognition but may result in occasional failures.
Poor	0.3 - 0.45	Recognitions can be performed to significant extent but may produce false recognitions.
Very Poor	0.0 - 0.3	Recognitions can still be performed but with significant possibility of confusing similar faces.

Contrast

					
Contrast = 1	Contrast = .87	Contrast = .63	Contrast = .47	Contrast = .40	Contrast = .20

Contrast represents the color contrast within the facial image. The less color contrast a face has, the more this metric value approaches 0. The default recommended minimum value for

this metric is .45. You can adjust the recommended minimum value by going to **Tools** → **Preferences**, clicking on the **Recognition** tab, then adjusting the **For merging** slider in the **Minimum required face contrast quality** section.

Quality Label	Metric Range	Description
Excellent	0.7 - 1.0	Full recognition accuracy can be expected under all conditions.
Good	0.6 - 0.7	Very good recognition accuracy can be expected in general but may confuse closely related family members.
Marginal	0.45 - 0.6	Good recognition but may result in occasional failures.
Poor	0.3 - 0.45	Recognitions can be performed to significant extent but may produce false recognitions.
Very Poor	0.0 - 0.3	Recognitions can still be performed but with significant possibility of confusing similar faces.

Face Size

Face size defines the minimum required face size in pixels. The metric also includes a margin around the face. The margin is required when learning a face. The face itself (without the margin) includes the area ranging from the top of the forehead to the bottom of the chin and across the full width of the face excluding ears.

The recommended minimum value for this metric is 220 pixels. You can adjust the recommended minimum value by going to **Tools** → **Preferences**, clicking on the **Recognition** tab, then adjusting the **For learning / strangers** slider in the **Minimum Required Face Size** section.

Note that only the shortest side of the image is used for the purpose of determining the metric value. For example, a facial image that is 200 x 300 (including the margin) would be classified as *Marginal*, since the shortest side (200) falls in the *Marginal* range.

Quality Label	Metric Value	Description
Excellent	260 px and greater	Full recognition accuracy can be expected under all conditions.
Good	210 px - 260 px	Very good recognition accuracy can be expected in general but may confuse closely related family members.

Quality Label	Metric Value	Description
Marginal	160 px - 210 px	Good recognition but may result in occasional failures.
Poor	110 px - 160 px	Recognitions can be performed to significant extent but may produce false recognitions of blurry or otherwise not clearly visible faces.
Very Poor	60 px - 110 px	Recognitions can still be performed but with significant possibility of confusing similar faces.

Occlusion

Occlusion represents how much of the face is occluded. Faces can be occluded by masks, baseball caps, or even the person's hands held between the face and the camera. The default recommended maximum value for this metric is .5. You can adjust the recommended maximum value by going to **Tools** → **Preferences**, clicking on the **Recognition** tab, then adjusting the **For learning / strangers** slider in the **Maximum allowed occlusion** section.

Quality Label	Metric Range	Description
Occluded	0.5 - 1.0	At least one of the facial features is not clearly visible thus potentially preventing full recognition accuracy. Recognition based on occluded features will not be possible and incorrect recognition of similar faces occluded in similar manner is possible. Recognition is generally possible as long as two out of three key features (eyes, nose, mouth) are visible.
Not Occluded	0.0 - 0.5	All facial features are clearly visible and full recognition accuracy can be achieved.

Sentiment

Sentiment represents how happy (a positive sentiment score) or angry (a negative sentiment score) a face is. 0 sentiment (a neutral or serious expression) yields the most accurate facial recognition.

Use Event Analytics to Gain Insight

The **Event Analytics** page displays an overview of the number and type of events occurring for each connected camera.

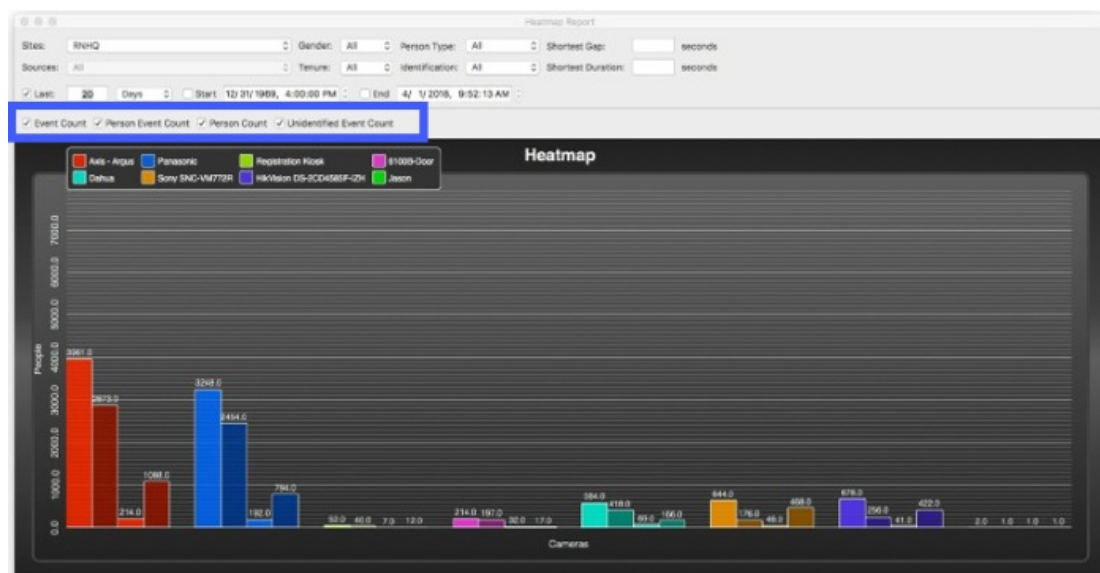
Use the Heat Map Report

The Event Heat Map shows the number of event occurrences per camera source. It lets you filter, sort, and view data to gain insight into event activity within your SAFR system.

The following types of counts are reported in the Heat Map:

- **Event Count:** Total number of events.
- **Person Event Count:** Number of known person appearance events.
- **Person Count:** Number of unique known persons.
- **Unidentified Event Count:** Number of events associated with unknown persons.

You can show or hide event counts by checking or unchecking the corresponding check boxes.



Use Heat Map Filters

You can filter displayed information using the following settings:

- **Sites:** Filter by sites.
- **Sources:** Filter by camera sources.
- **Tenure:** Filters based on a person being new in the chosen time interval or known from before the chosen time interval.
- **Person Type:** Filter by person type.
- **Id Class:** Filters by identity classification. This filter must be one of the following:
 - Unrecognizable

- Stranger
- No-Concern
- Concern
- Threat
- **Shortest Gap:** Multiple events that occur within the specified number of seconds or less are presented as one combined event.
- **Shortest Duration:** After combining events based on the **Shortest Gap** setting, the **Shortest Duration** setting filters out any events shorter than the specified number of seconds.

Actions Overview

In SAFR an action is essentially a script/macro that communicates a desired action in a language/protocol the receiving device or system understands. It can be written in any language supported by the computer where ARES is installed. It only needs to be invocable as an executable directly or through the use of another executable (usually a script interpreter such as Python).

Actions Components

There are 3 principle components involved with actions:

- **Actions Relay Event Service (ARES):** ARES is a cross-platform Java application that acts as an event listener that dispatches configured actions in response to events, as defined in the SAFRActions.config file. ARES can provide replies on any event to be handled by the client originating the event and is normally installed as a service by either the SAFR Platform or SAFR Edge installers. It is constantly active and is automatically started by the operating system on power-up.
- **SAFRActions.config:** The SAFRActions.config file defines which events will trigger specified actions. It also can specify additional condition constraints before the action(s) will trigger.
- **SAFR Actions:** Only available on macOS and Windows. SAFR Actions is a GUI tool that makes editing the SAFRActions.config file much easier. It presents the JSON information of the config file in a visual and easy to understand manner, offers drop-down menus when appropriate so you can quickly and easily what values are available and valid, makes the JSON element heirarchies easy to understand, and ensures that your changes will validate against the SAFRActions.config JSON schema.

Action Config Overview

```

<name: value connection attributes>
rules: [
  {
    event: { },
    triggers: [
      <time of day and week properties>
      actions: [ ],
      reply: { },
      conditionalReply: { },
    ],
    excludeDates: [ ]
  }
]
noTriggerReply: { }
nFactorDef: [ { }, { }, ... ]
emailDef: [ { }, { }, ... ]
smsDef: [ { }, { }, ... ]

```

- **rules:**
 - 1 or more rules can be defined.
 - When an event occurs each rule is checked to see if any of its events match.
 - A rule's event matches an occurring event when:
 - All attributes rules[i].events match the event.
 - Each rule has 1 or more triggers.
 - Each Trigger inside a matching rule is fired as long as time of day conditions match. **Exception:** If 2 *triggerIds* are identical only the first trigger is fired.
 - Each trigger has one or more actions.
 - Actions are either:
 - A shell command or a batch/shell script to be executed.
 - A send email command that has the syntax of: @emailSend <value of emailDef.label>
 - All actions are run asynchronously unless a *conditionalReply* is specified in which case the first rule is run synchronously (and the return code of that rule is used for the conditionalReply) and all other rules are run asynchronously.
- *noTriggerReply* is used to perform a reply if none of the triggers are fired.
- *nFactorDef* can define 2 or more conditions that must occur within the specified time window.
- *emailDef* defines one or more email message attributes (subject, from, message, etc).

Examples:

- Send email when visitor arrives during work hours
 - rules
 - Rule 1
 - event (hasPersonId=false)
 - trigger (day/hours: 8-5, M-F)
 - action: @emailSend visitorEmail

- emailDef
 - label=visitorEmail
 - subject="Visitor Arrived"
 - message="A visitor has arrived at #I - #S.
 - ...
- Log all events to a CSV and send one type of email for a known person event and another for a threat event.
 - rules
 - Rule1 (known person email)
 - event (hasPersonId=true, idClass=No-Concern)
 - trigger
 - action: @emailSend knownEmail
 - Rule 2 (threat email)
 - event (hasPersonId=true, idClass=[Threat, Concern])
 - trigger
 - action: @emailSend threatEmail
 - Rule 3 (log)
 - trigger
 - action: ".\scripts\log_event.bat "#D" "#N" "#F" ..."
 - If editing config file, escape backslash or quotes with another backslash. (In SAFR Actions no escaping is needed.)
 - The file 'log_event.bat' should be placed in C:\Program Files\RealNetworks\SAFR\ares\scripts (for Windows) or /Library/RealNetworks/SAFR/ares/scripts (for macOS).
 - emailDef
 - 1 (label=knownEmail, subject, message, etc)
 - 2 (label=threatEmail, subject, message, etc)

Actions Relay Event Service (ARES)

ARES is a cross-platform Java application that acts as SAFR Platform event listener that dispatches configured actions (macros) in response to events. The recommended Java version is 9.0.4 or later. ARES can provide replies on any event to be handled by the client originating the event and is normally installed as a service by either SAFR Platform or SAFR Edge installers. It is constantly active and is automatically started by the operating system on power-up.

ARES Installation Locations

- For Windows: C:\Files
- For macOS: /Library/RealNetworks/SAFR/ares

- For Linux: /opt/RealNetworks/SAFR/ares

Command Line Start

```
java -jar Ares.jar
```

Command line supports the following options:

```
-u <UserId>      - provides RealCV account User Id
-p <Password>    - provides RealCV account password
-q              - turns on quiet mode which suppress most console output
```

Command line UserId/Password override those configured in SAFRActions.config.

Re-configuration:

- ARES dynamically applies any changes to config file without restarting:
 - ARES monitors config file for any changes.
 - ARES examines config file for modifications every 2 seconds
- When a change is noticed, ARES reads and reconfigure atomically (event polling is to suspend briefly and then promptly resumed after reconfiguration).
- Reconfiguration action is indicated in the log:

```
--- RECONFIGURED at <date>
```

Console Output:

- At start, ARES displays any errors or warning based on contents of the config file.
- ARES displays all received events, triggered actions, and replies issued unless it was given -q (quiet) option at start.

Tip: In the Mac terminal or in the Windows Cygwin shell, the `tail -f ares.log` command is a convenient way to monitor the SAFR Action service in real time.

SAFRActions.config

The SAFRActions.config file defines which events will trigger specified actions. You can also specify additional condition constraints before the action(s) will trigger. It also contains basic configuration information so that ARES can communicate with other SAFR components, such

as the Event Archive.

SAFRActions.config JSON Schema

```
{
  environment : "string",
    <optional,
      - values: "LOCAL", "DEV", "INT2", "PROD", "Custom"
      - if not specified assumed PROD >
  eventServer : "string",
    <optional,
      - required in case of Custom environment
      - only affects Custom environment>
  replyServer : "string",
    <optional,
      - only affects Custom environment>
  coviServer : "string",
    <optional,
      - only affects Custom environment>
  reportServer : "string",
    <optional,
      - only affects Custom environment>
  configServer : "string",
    <optional, "https://cvos.int2.real.com" for
      partner cloud environment
      "https://cvos.real.com" for
      cloud environment
      - if specified config is retrieved from the cloud using the
        following address: <configServer>/obj/ares/<aresId> >
  userId : "string", <optional>
  userPwd : "string", <optional>

  directory : "string", <required>
  site : "string", <optional>
  source : "string", <optional>

  aresId : "string", <optional>

  maxEventLatency: <long>, <optional, in milliseconds, default = 8000>

  rules: [
    {
      event : {
        type: [ "string", ... , "string" ],
          <optional, values=(person, badge, action or object),
default = all>
        personType: [ "string", ... , "string" ],
          <optional, default = all, "" = no personType>
        personTags: [
          [ "string", ... , "string" ],
          ...
          [ "string", ... , "string" ]
        ]
          <optional, default = all>
        tagType: [ "string", ... , "string" ]
          <optional, values=(april), default = all, "" = no tagType>
        tagId: [ "string", ... , "string" ],
          <optional, values=(Ids of tagType) default = all, "" = no
tagId>
        actionType: [ "string", ... , "string" ],
          <optional, values=(smileToActivate) default = all, "" = no
```

```

        <optional, values={smile|noSmile}, default = all, "" = no smile>
    actionType>
        actionId: [ "string", ... , "string" ],
            <optional, default = all, "" = no actionId>
        name: [ "string", ... , "string" ],
            <optional, default = all, "" = no name>
        company: [ "string", ... , "string" ],
            <optional, default = all, "" = no company>
        moniker: [ "string", ... , "string" ],
            <optional, default = all, "" = no moniker>
        personId: [ "string", ... , "string" ],
            <optional, default = all, "" = no personId>
        hasPersonId: <boolean>,
            <optional, default = all>
        hasName: <boolean>,
            <optional, default = all>
        hasMoniker: <boolean>,
            <optional, default = all>
        hasRootEventId: <boolean>,
            <optional, default = all>
        gender: [ "string", ... , "string" ],
            <optional, default = all>
        age: [
            <optional, default = all>
            {
                min: <float>,
                max: <float>
            },
            ...
        ],
        smile: <boolean>,
            <optional, default = all>
        avgSentiment: [
            <optional, default = all>
            {
                min: <float>,
                max: <float>
            },
            ...
        ],
        similarityScore: {
            <optional, default = all>
            min: <float>,
            max: <float>
        },
        occlusion: {
            <optional, default = all>
            min: <float>,
            max: <float>
        },
        site: "string",
            <optional if specified at the root>
        source: "string",
            <optional if specified at the root>
        idClass: [ "string", ... , "string" ],
            <optional, default = all, "" = no idClass>
        directGazeDuration: {
            <optional, default = all>
            min: <long>,
            max: <long>
        }
        objectType: [ "string", ... , "string" ]
            <optional, default = all, "" = no objectType>
        objectId: [ "string", ... , "string" ],
            <optional, default = all, "" = no objectId>
    }
}

```

```

triggers : [
  {
    triggerId : "string",
      <optional>
    daysOfWeek: ["Mon","Tue","Wed","Thu","Fri","Sat","Sun"],
      <optional, default = all>
    timesOfDay: [
      <optional, default = all>
      {
        start: "11:00",          <required>
        end: "17:00"            <required>
      },
      ...
    ],
    actions: [
      <required - can be empty (no actions)>
      "string",
      ...
    ],
    reply: {
      <optional, default = no reply>
      "replyDelay": long,
        <optional, in milliseconds, default = 0>
      "message": "string",
        <optional, default = no message>
      "disposition": double,
        <optional, range [-1 .. 1], default = 1>
      "tags": [ "tag1", ... "tagN" ]
        <optional, default = no tags>
    },
    conditionalReply: [
      <optional, default = no conditional reply>
      {
        "actionResponse": [ integer, ..., integer ],
          <required>
        "replyDelay": long,
          <optional, in milliseconds, default = 0>
        "message": "string",
          <optional, default = no message>
        "disposition": double,
          <optional, range [-1 .. 1], default = 1>
        "tags": [ "tag1", ... "tagN" ]
          <optional, default = no tags>
      }
      ...
    ],
  },
  ...
],
excludeDates : [
  <optional, default = none>
  "7/4",
  "12/25",
  "4/10/2017",
  ...
]
...
],
noTriggerReply: {
  <optional, default = no reply>
  "replyDelay": long,
    <optional, in milliseconds, default = 0>
  "message": "string",
    <optional, default = no message>

```

```

        "disposition": double,
            <optional, range [-1 .. 1], default = -1>
        "tags": [ "tag1", ... "tagN" ]
            <optional, default = no tags>
    },
    nFactorDef: [
        {
            "name": string,
                <required>
            "failOnMismatch": string,
                <optional: "delayed"/"immediate"/"none", default = "delayed">
            "maxDelay": <milliseconds>,
                <optional, default = 60000 (1min)>
            "factors": [
                "<factor_name>|<factor_value>",
                ...
            ],
            "actions": [
                "<action_command>",
                ...
            ]
        },
        ...
    ],
    emailDef: [
        {
            "label": string,
                <required>
            "recipients": [ "recipient1", ... "recipientN" ],
                <required, escape sequences can be used>
            "subject": string,
                <required, escape sequences can be used>
            "cc": [ "cc1", ... "ccN" ],
                <optional, escape sequences can be used>
            "bcc": [ "bcc1", ... "bccN" ],
                <optional, escape sequences can be used>
            "message": string,
                <optional, escape sequences can be used>
            "attachments": [ "attachment1", ... "attachmentN" ],
                <optional, escape sequences can be used
                http://, https://, cvs:// url schemes are supported>
        },
        ...
    ]
    smsDef: [
        {
            "label": string,
                <required>
            "recipients": [ "recipient1", ... "recipientN" ],
                <required, escape sequences can be used, phone numbers using the
the E.164 format required>
            "maxPrice": string,
                <optional>
            "message": string,
                <optional, escape sequences can be used>
        },
        ...
    ],
}

```

- Events that are older than maxEventLatency will be ignored. Event time is defined as the difference between the time the event was generated - as measured by the SAFR Cloud

(or machine Platform is running) and the time the event is processed – as measured on the machine the SAFR Actions app is running.

rules

event

- For rules.events that allow arrays, the new event must contain all the specified array elements to match. For example, if a config file specified rules.events. "personType": ["staff", "admin", "guest"], Then the new event's personTags array would have to have all 3 specified personTypes for it to match the rule.
- personTags: all elements in one of sub-arrays need to exist in event's personTags array to match the rule.

trigger

- Event (id) can trigger actions only once (albeit multiple triggers can be activated simultaneously).
- Event (id) can trigger replies only once per reply context (triggered, notTriggered). Multiple replies can be triggered simultaneously (one reply per triggered action).
- triggerId - ID Unique within the triggers array used in rare case where you want only 1 trigger to fire. If triggerId is same on 2 or more, only 1st of all matching get triggered.
- Useful if date filters are overlapping and during overlap times only wish to actions from single trigger.

conditionalReply and reply

- disposition refers to how the reply should be perceived by the recipient:
 - Replies with disposition in range [-1 .. 0 > are interpreted as negative replies and can thus be expected to be presented (color, sound, voice) in manner consistent with rejection.
 - Value of 0 is a neutral reply and can thus be expected to be presented in a neutral manner (color, sound, voice).
 - Replies with disposition in range <0 .. 1] are interpreted as positive replies and can thus be expected to be presented (color, sound, voice) in manner consistent with acceptance.
- When conditional reply is specified, non-conditional reply is used only as catch-all if none of the action response codes match.
- When conditional reply is specified, execution of the FIRST action in trigger will occur in blocking manner to enable retrieval of the response code from that FIRST action.

- If any other actions are specified, they will be performed in non-blocking manner and their response codes will not be retrieved or used.
- When conditional reply is not specified, execution of all actions will occur in non-blocking manner.
- A reply is generated as follows:
 - One or more matching conditionalReply entries are sent
 - In addition, either the reply or noTriggerReply is sent
- URL used to post the reply: <replyServer>/stream/reply.<Base64 (event Id)>
 - By default the reply is posted to the CVOS server (replyServer)
 - POST is a file of the following format.
 - The reply object (JSON file) can be obtained by querying the CVOS server after some delay after the event was fired

Structure of the reply:

```
{
  triggered : boolean,
               <required - either true or false>
  replyTime : <epoch time>,
               <required - milliseconds since start of epoch>
  eventStartTime: <epoch time>,
               <required, indicates time of the start of the event being
replied to>

  replies: [
    <optional, default = no replies>
    {
      replyDelay : long,
                   <optional, in milliseconds, default = 0>
      message: "string",
                   <optional, default = no message>
      disposition: double,
                   <optional - [-1 .. 1]>
      tags: [ "tag1", ..."tagN" ],
                   <optional, default = no tags>
      actions: [ "action1", ..."actionN" ]
                   <optional, default = no actions>
    }
  ]
}
```

- actions returns the values passed in the actions list

actions

- actions
 - Each action is a command string that will be executed.
 - Any tokens matching Action and Reply tokens below will be replaced with the respective value before command is executed
 - Commands are executed asynchronously unless conditionalReply is set
 - If conditionalReply set, the 1st command is executed synchronously (See below)
 - Some windows programs (particular Windows programs that do not have a message pump) may not run in background and block until command returns
 - If multiple actions, each action is executed in sequence

Action and Reply Message Escape Sequences

```
#N - name
#F - first name (name prefix up to first white-space)
#U - surname (name postfix: staring after first white-space sequence to the
end of name string)
#T - person type
#S - source
#I - site
#D - person id
#R - root person id
#E - person external id
#G - gender
#A - age      (###)
#M - sentiment (#.##)
#L - smile    (true/false)
#V - event type
#v - event id
#B - tag type
#C - action type
#b - tag id
#c - action id
#s - event start time (milliseconds since epoch)
#r - event start date/time (local time)
#p - validation phone
#e - validation email
#H - home location
#t - personTags (comma separate list of personTags)
#O - company
#m - moniker
#<d>m - moniker substring (delimited by white-space)
      indexed by single decimal digit 0-9 . E.g.: #0m or #3m
#l - similarityScore (#####)
#a - idClass
#Z - directGazeDuration
#o - objectType
#d - objectId
#u - occlusion (###)
```

N-factor Actions

- nFactor actions are started via internal @nFactorStart action within standard trigger actions array:

```
{
  triggerId : "string",
  ...
  actions: [
    "@nFactorStart <name>",
    ...
  ],
  reply: {
    ...
  },
  conditionalReply: [
    ...
  ]
}
```

At the time of starting, the following occurs:

- @nFactorStart action just as any other action is first resolved for escape sequences
- factors (names and values) defined in corresponding nFactorDef are also resolved for escape sequences
- actions defined in corresponding nFactorDef are also resolved for escape sequences
- eventStartTime is retrieved from the triggering event

Response codes for nFactorStart action:

- 0 = nFactor monitoring for action started successfully

nFactorStart-ed action are resolved via nFactorResolve commands. When all factors needed for the actions are resolved, actions are executed:

```
{
  triggerId : "string",
  ...
  actions: [
    "@nFactorResolve <name> <factor_name>|<factor_value>",
    ...
  ],
  reply: {
    ...
  },
  conditionalReply: [
    ...
  ]
}
```

- At the time of resolving the following occurs:

- @nFactorResolve action just as any other action is first resolved for escape sequences.
- Each factor can resolved at most one not yet resolved factor requirement.
- Response codes for nFactorResolve action:
 - 0 = resolved last unresolved factor
 - Executed action response supersedes
 - >=1 resolved other than last unresolved factor
 - -1 = no matching <Site>/<Source>/<name>
 - -2 = <mismatched factor - ignored since failOnMismatch = none>
 - -3 = <matches but already resolved>
 - -4 = <matches but too late to resolve>
 - -5 = <mismatched factor - error since failOnMismatched = delayed/immediate>
 - -6 = unknown (not defined in nFactorDef) factor_name.
- @nFactorStartOrResolve combines starting and resolving into one action. Usually used for generating pseudo events from monikers.

```

{
  triggerId : "string",
  ...
  actions: [
    "@nFactorStartOrResolve <name> <factor_name>|<factor_value>",
    ...
  ],
  reply: {
    ...
  },
  conditionalReply: [
    ...
  ]
}

```

@personEventFromMoniker action generates a pseudo person event from moniker created by combining all the resolved factor values (separated by space) in order listed in factors array. The generated event is of type person which is populated with meta-data of person with moniker matching the assembled moniker value.

```

{
  nFactorDef : [ {
    factors : [
      "moniker|**",
      "moniker|1**",
      "moniker|2**",
      "moniker|3**"
    ],

    actions : [
      "@personEventFromMoniker"
    ]
  }
]
}

```

Email Actions:

Action syntax:

```

{
  triggerId : "string",
  ...
  actions: [
    "@emailSend <label>",
    ...
  ]
}

```

SMS actions

Action syntax:

```

{
  triggerId : "string",
  ...
  actions: [
    "@smsSend <label>",
    ...
  ]
}

```

SAFR Actions

SAFR Actions is a GUI tool to aid users to edit the SAFRActions.config configuration file.

It comes already installed with SAFR Platform. It requires separate downloading and installation only under the following circumstances:

- The computer on which SAFR Platform is installed is not sufficiently close to mechanisms SAFR Actions service needs to control.
- The SAFR Platform computer does not have enough capacity to perform SAFR actions.
- You're using a cloud deployment, so SAFR Platform is not installed locally.

Note: SAFR Actions is generally very light in terms of system resource usage but can be burdensome if the rate of events requiring handling is high (for example, hundreds per second) and actions scripts are computationally or I/O intensive. However, this is not a common occurrence.

Large Scale Deployments

At some point, your SAFR system's capacity and/or performance may become limited by your SAFR Server; your server's load is primarily limited by the number of *face recognitions* occurring per second and the number of people in your Person Directory. You can install additional SAFR Servers on other machines in order to achieve higher capacity, improve performance, and improve resiliency. The first SAFR Server you install is your primary server, while all additional servers are secondary servers.

In order to install additional servers, you must first install an SSL certificate on your primary server. See SSL Certificate Installation for information about how to do this.

Note: You can change which machine is the primary server by uninstalling the primary server, waiting 24 hours, and then re-installing the SAFR Server on a different machine. If you want to preserve existing data, you should create a backup prior to the change.

There are three different load balancing configurations you can choose from.

- **Prescribed Configuration**: Cameras are connected to Desktop clients or VIRGO daemons running on the same machines that are hosting your SAFR Servers. This gives you tight control over how your face recognition load is distributed, since the video feeds' face recognition requests are processed on the same machine where the video feeds are connected.
- **Software-Based Load Balancing Configuration**: In this configuration the machines hosting SAFR Servers do not also have cameras connected to them. All face recognition requests are initially sent to the primary server. The primary server acts as the load balancer for the server cluster.
- **External Load Balancing Configuration**: All recognition requests are directed at one or more external load balancer(s), which handle load balancing duties for the SAFR system.

Understand When to Scale

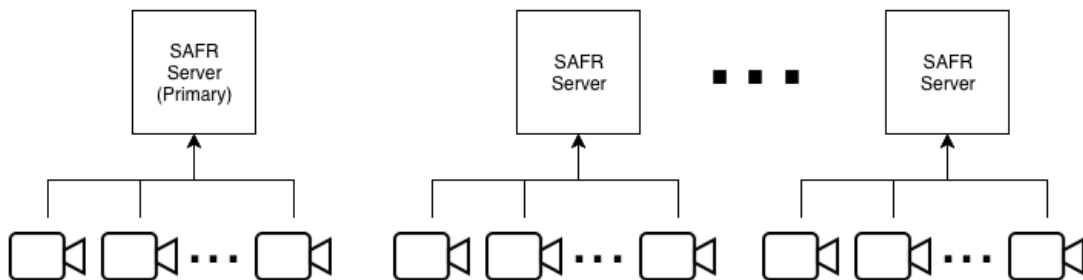
A single SAFR Server that's also running a Desktop client can handle up to 16 cameras, (assuming each camera view contains just a single face), as long as the host machine meets the recommended hardware requirements. If the machine running the server doesn't have any cameras directly connected to it, then the server's capacity increases to 25 cameras, each camera view containing a single face. A higher number of faces per camera or a higher number of cameras requires either vertical scaling of a single server (i.e. more or faster CPUs) or horizontal scaling by installing more SAFR Servers.

For prescribed deployments, the system requirements of the Desktop client need to be combined with those of SAFR Server. A single Desktop client typically handles up to 16 cameras as long as it is equipped with a GPU card (see SAFR System Requirements). In this way, running SAFR Server and the Desktop client on the same machine using the recommended configuration can host up to 16 cameras, each camera with a single face.

Prescribed Configuration

In the prescribed configuration, you run multiple SAFR Servers by connecting cameras to Desktop clients or VIRGO daemons running on the same machines that are hosting SAFR Servers. In this way, you have tight control over which servers take the video feed load. This is also a useful configuration for very small stream count loads where running a Desktop client on a separate machine from the SAFR Server would take more resources than are required for the given use case.

The following illustration demonstrates this setup:



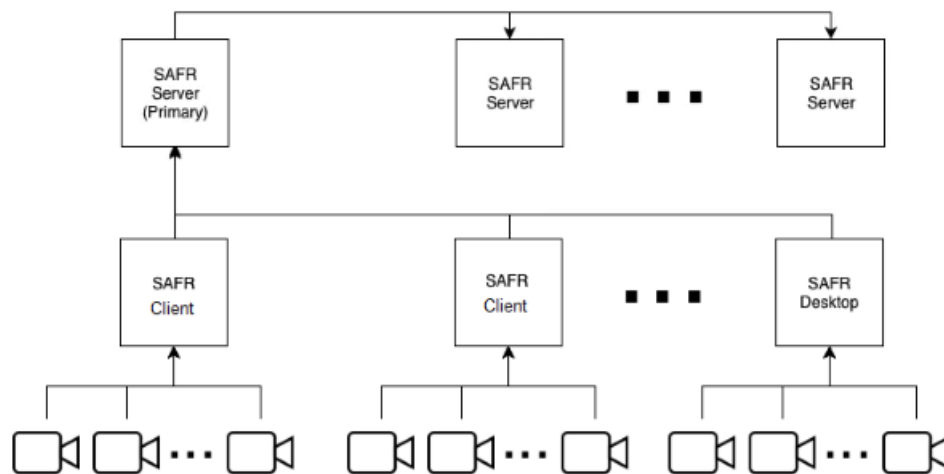
Most services (e.g. face service, events, and reports) are performed on the server where recognition requests are sent.

See [Add a Secondary Server](#) for information about how to add secondary servers.

Software-Based Load Balancing Configuration

In the software-based load balancing configuration, cameras aren't connected to machines

running SAFR Servers. When newly installed secondary servers are configured, they check in with the primary server and announce that they're ready to receive load-balanced traffic. All recognition requests go through the primary server, which balances the load among itself and all other servers in the SAFR system. The following illustration demonstrates this setup:



See [Add a Secondary Server](#) for information about how to add secondary servers.

Secondary SAFR Server Health Checks

- At startup each server, both primary and secondary, registers itself by posting its status to the database on the primary server.
- The primary server directs requests to all secondary servers in a *least connection method* that keeps the load evenly balanced among all secondary servers.
- As long as a secondary server remains healthy, the primary server keeps the secondary server in its load balance rotation.
- Status information about all secondary servers is stored in the primary server database. In this way, it is not lost on restart of the primary server.
- Every minute the secondary servers and the primary server send a status update to the database on the primary server.
- Every five seconds, the SAFR load balancer process on the primary server calls a health check API on each secondary and the primary server.
- If the health check fails for 15 seconds, the server is pulled out of rotation and is no longer sent requests. If the health check succeeds for that server for ten seconds, the server is returned to accepting requests.
- If a server's status has not been reported for over five minutes, it is removed from the load balancer configuration. In this case, it is no longer sent requests or health check requests.
- If a secondary server has been pulled out of rotation for not responding to health checks, or is removed from the load balancer config for not reporting status for more than five minutes, it can still be put back in rotation through any of the following:
 - If a network interruption prevents the secondary server from sending a request, the

secondary server continues to send a status update at its regularly scheduled interval after it goes back online and its status is updated in the primary server.

- If the secondary server is restarted, it sends a status update after all services are started and ready.
- If the secondary server IP address is changed, the server must be manually restarted to force it to send a status update to the primary server with the new IP address.

Manually Configure Load Balancing

SAFR Servers can be manually enabled or disabled to accept load balancing traffic.

Note: If the server you want to disable is the only one configured to take traffic, you receive a warning and prompt to continue. In this case, should you proceed, your system will most likely go offline.

Disable Load Balancer Traffic

To stop receiving traffic on a server, log in to a shell on the server and run the appropriate command for your server's OS:

OS	Command
macOS	<code>/Library/RealNetworks/SAFR/bin/server-status.py --disable</code>

It may take up to one minute for the desired traffic state to change.

Enable Load Balancer Traffic

To stop receiving traffic on a server, log in to a shell on the server and run the appropriate command for your server's OS:

OS	Command
macOS	<code>/Library/RealNetworks/SAFR/bin/server-status.py --enable</code>

It may take up to one minute for the desired traffic state to change.

External Load Balancing Configuration

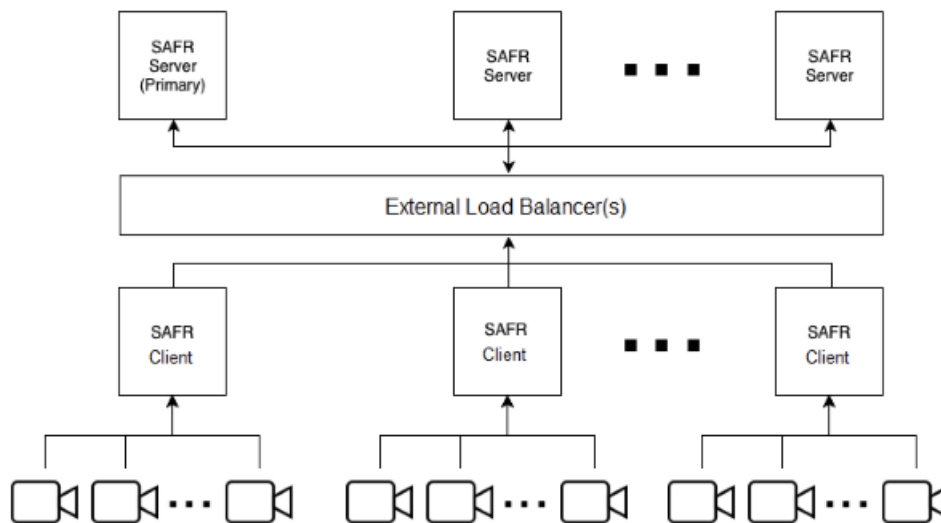
The software-based load balancing for SAFR is limited by the primary server being a single point of failure. All traffic must be routed through the primary server to reach the rest of the servers. If the primary server is down, all traffic will stop.

External load balancing may be used to provide a more robust setup that can better deal with server failure than software load balancing.

When using external load balancing solutions, you can route traffic to one or more load

balancer(s), have HTTPS/SSL terminate there, and then proxy requests to the backend servers over either HTTP or HTTPS. HTTP would be OK in situations where network traffic is isolated to a trusted network, or when network sniffing by non-target hosts is impossible.

If HTTPS is used to proxy traffic to SAFR servers, you should manually disable load balancing on all secondary servers [as described above](#) so that the primary server isn't double load balancing traffic to them. A valid (i.e. non self-signed) SSL certificate would still need to be installed and configured on the primary server. Secondary servers should be fine with the default (i.e. self-signed) certificate, if your load balancer allows it.



See Add a Secondary Server for information about how to add secondary servers.

Troubleshooting Tips

- The network throughput of the primary server is a possible performance bottleneck. Monitor the primary server network throughput during maximum concurrency times to make sure the network is not over-saturated.

Database Redundancy

The first SAFR Server you install will automatically become the primary server. All subsequent servers you install will be secondary servers. There are two types of secondary servers:

- **Simple:** Does not replicate the database data.
- **Redundant:** Replicates database data from the primary server. If there are at least two redundant secondary servers (three servers total), fail-over functionality is enabled, which means that if the primary server is offline, the secondary servers will continue to function.
Note: Only Windows and Linux SAFR Servers can become redundant secondary servers.

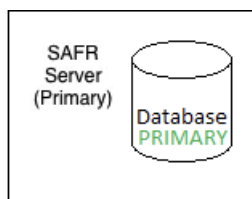
With both types of secondary servers the traffic for the CoVi and Event API services are load-balanced across all servers. Other services, such as VIRGA (feed management), Reports, and the Web Console are not load-balanced and are always served from the primary server.

Multiple Server Installations

1 Server

Install a single SAFR Server. The database runs on the primary (and only) server.

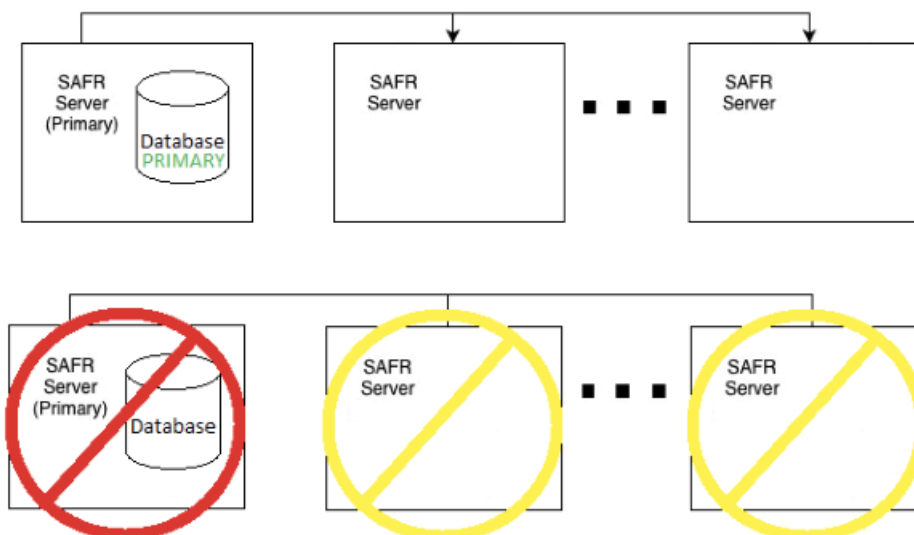
- This configuration provides no redundancy if the primary server is offline.



2+ Servers (Simple)

Install a primary server and one or more simple secondary servers. The database runs on the primary server only.

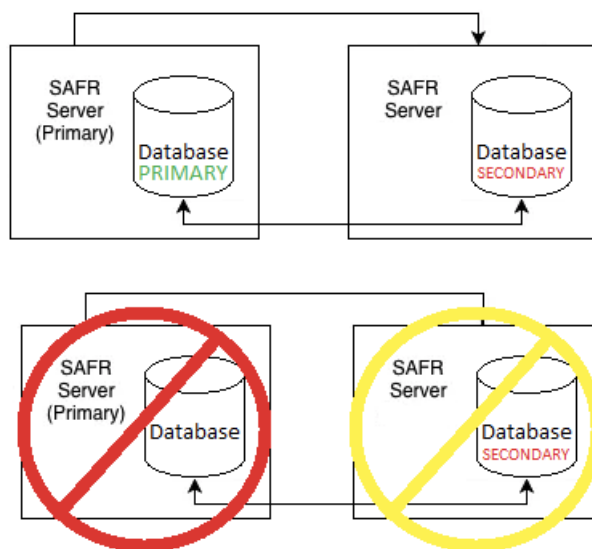
- This configuration provides no redundancy if the primary server is offline.
- The secondary servers can be offline without impacting functionality, although performance may suffer.



2 Servers (Redundant)

Install a primary server and a redundant secondary server. The database runs on both the primary and secondary servers.

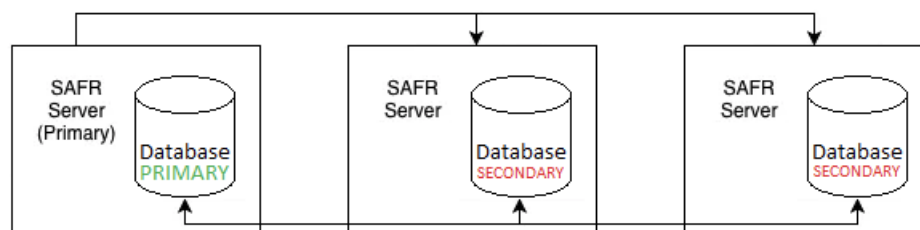
- This configuration provides no redundancy if the primary server is offline.
- The secondary server can be offline without impacting functionality, although performance may suffer.
- Database content is replicated to the secondary server, which provides another copy of the data. This can act as a backup in case of emergencies, but the backup & restore scripts should be used for proper and complete backups.

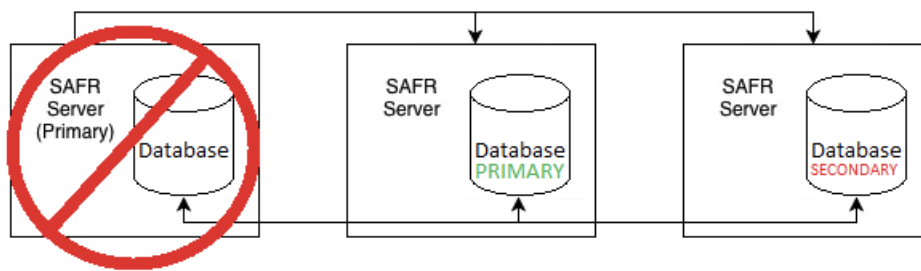


3 Servers (Redundant)

Install a primary server and 2 redundant secondary servers. The database runs on both the primary and both secondary servers.

- This configuration provides redundancy if the primary server is offline.
- A single server can be offline without impacting functionality. If the primary and one secondary, or both secondary servers go offline, the whole cluster will go offline. A majority of the servers are required to be online for your SAFR system to function.

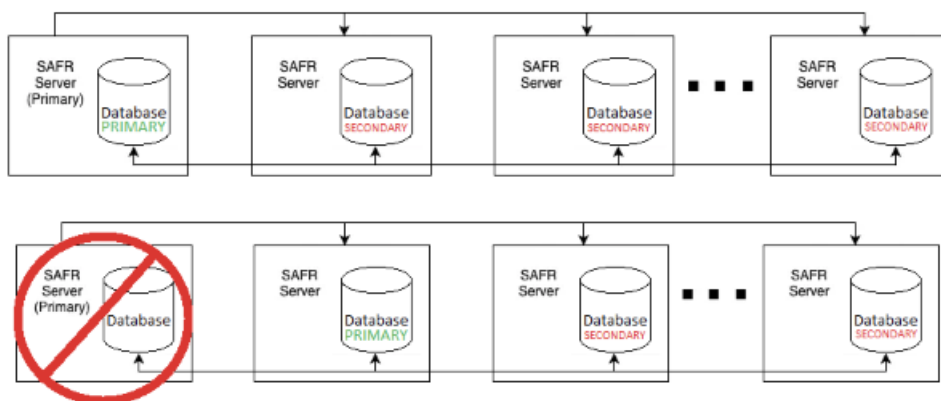




4+ Servers (Redundant)

Install a primary server and 3 or more redundant secondary servers. The database runs on both the primary server as well as all the secondary servers. Only the first two secondary servers that were added can act as the primary database host, though. Additional secondary servers will continue to replicate database data but cannot become the primary database host and do not count towards the “majority” count required for a primary database host to be elected.

- This configuration provides redundancy if the primary server is offline.
- A single server can be offline without impacting functionality. If the primary and one of the first two secondary, or both of the first two secondary servers go offline, the database cannot have a primary member, and the whole cluster will go offline. A majority of the first three installed servers is required to be online for SAFR to function. Additional secondary servers past the first two can be offline without any impact to functionality.



Add a Secondary Server While Connected to the Internet

If your system is connected to the Internet, do the following to add a secondary server to your existing primary server:

1. Download and install SAFR Platform on the additional machine.
2. To start the SAFR auto-discovery process:
 - Connect the Desktop client to the primary server as described here .
 SAFR now executes auto-discovery.

3. During auto-discovery, the following automatically happens:
 1. The secondary server contacts a SAFR Licensing Server in the cloud to acquire a license.
 2. The SAFR Licensing Server authenticates the SAFR account credentials.
 3. The SAFR Licensing Server identifies the license and deployment type.
 4. A suitable license is returned to the secondary server and information about the primary server is returned to the secondary server, including the hostname.
4. If your new secondary server is on a Windows or Linux machine, you will be prompted to choose which kind of secondary server you want: simple or redundant. If your new secondary server is on a macOS machine no prompt will occur; macOS secondary servers are always simple.
5. Auto-discovery will now continue, with the following automatically occurring:
 1. The secondary server re-configures itself to reference the primary server.
 2. The secondary server registers itself with the primary server.
 3. The primary server updates its local database and adds the new secondary server to its load balancer configuration.
 4. From this point on, the primary server uses the secondary server as an additional node in its cluster.

Add a Secondary SAFR Server While Offline

If you are not connected to the Internet, you can still connect to the primary SAFR Server, but the auto-discovery process is not available. You must instead manually configure the newly installed secondary server to locate the primary server.

1. Download and install SAFR Platform on the second machine.
2. Run the `safr-worker` script on your secondary server by doing the following:
 1. Open Terminal.
 2. Run the following command, substituting the primary SAFR hostname for `HOSTNAME`:
 - `sudo /Library/RealNetworks/SAFR/bin/safr-worker HOSTNAME`

Error Messages

When attempting to join a new secondary server, you might encounter the following error messages:

Error Message	Description
System is offline	Network or system connectivity issue. Attempt to access the system at a later time.
SAFR master host is not reachable	Ensure all servers are connected to the same network and try again.

Error Message	Description
Improperly configured SSL certificate	SSL certificates are required to set up multiple servers. See the SSL Certificate Installation page for information about how to install an SSL certificate.
Secure connection error. Check server for valid SSL certificate	SSL certificates are required to set up multiple servers. See the SSL Certificate Installation page for information about how to install an SSL certificate.
Incomplete server connection	Attempt to join again; a persistent issue may require either uninstalling and reinstalling SAFR Platform on your servers or contacting your SAFR support representative.

SSL Certificate Installation

A properly installed secure sockets layer (SSL) certificate is critical to the secure operation of your SAFR Server. SAFR uses SSL certificates to establish secure network connections and data transfers. (i.e. https connections) SAFR requires https connections between SAFR Servers and between SAFR Servers and iOS Mobile clients. None of the other SAFR components require https connections.

Before you can install an SSL certificate on your SAFR Server, you must first configure a Domain Name System (DNS) hostname for your server within your network domain, as described below.

DNS Hostnames

If you do not currently have a domain, you need to first obtain a domain name registered and configured with an accredited domain registrar.

How to Obtain a Domain Name

In order to set up a DNS, you need a domain within which you can register hostnames. [ICANN](#) maintains a list of accredited registrars from which to choose.

The following is a list of common registrars:

- [GoDaddy](#)
- [Google Domains](#)
- [AWS](#)
- [HostGator](#)

Follow the processes on these websites to find, purchase, and configure your domain name. Most registrars offer the ability to host your DNS for you and most also give you a web

interface for managing it.

The following links lead to instructions on how to modify DNS entries:

- [GoDaddy](#)
- [Google Domains](#)
- [AWS](#)
- [HostGator](#)

After you have your domain, you can create a DNS hostname entry for your SAFR Server.

What a DNS Hostname Entry Does

DNS is a system that translates a hostname to a network IP address. For example, when a user types `www.example.com` into their browser, DNS servers resolve it to the IP address where the website is hosted.

To provide this translation, DNS requires an entry for each hostname. This entry typically takes the form of an *A record* (the A stands for “Address”) which defines the hostname to IP address translation in DNS. An *A record* is the most basic type of syntax used in DNS records.

The following is an example of an *A record*:

```
safr.example.com    A    12.34.56.78
```

Set Up a DNS Hostname Entry for your Primary Server

DNS can be managed in numerous ways. This might be a text file or a web interface for configuring the DNS entries. If you are not sure, contact the person managing the domain name for your network.

What Type of IP Address Should I Use?

You should use a static IP address. If you instead choose to use DHCP to get a dynamically assigned IP address, and your IP address happens to change, your DNS hostname entry will stop working until you update the entry.

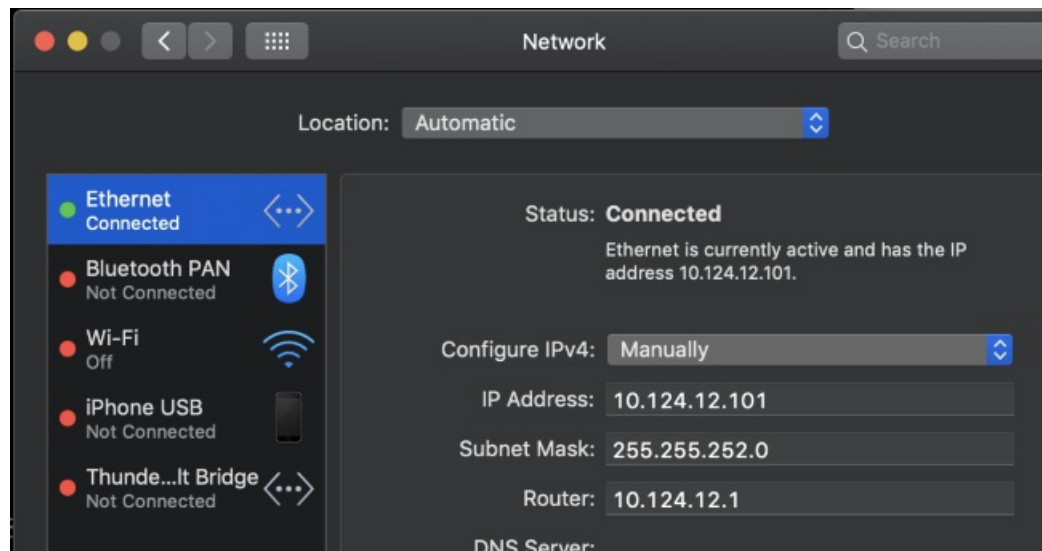
Configuring a Static IP

1. Obtain a static IP from your network administrator. The information should include the following:
 - Static IP address
 - Subnet mask
 - Default gateway

2. Configure your system as described in these guides:
 - For macOS, see <https://www.howtogeek.com/howto/22161/how-to-set-up-a-static-ip-in-mac-os-x/>.

The IP address should be the internal IP address of the computer running the SAFR Server. This should not be your public IP address because the public IP address usually points at your router, modem, or similar device. The internal IP address is the IP used locally by the computer. It can be determined by doing the following:

1. Open **System Preferences**.
2. Open **Network**.
3. Click the active network connection (usually WiFi or Ethernet).
4. The IP address is displayed in the dialog.



SSL Certificates

After you have configured a DNS hostname for your primary server, you can now install an SSL certificate.

What an SSL Certificate Does

SSL certificates are small data files that digitally bind a cryptographic key to an organization's information. When installed on a server, an SSL certificate allows secure connections from the server to a browser or other program and protects sensitive information.

A common use for SSL certificates is to enable a web server to provide a secure connection with a web browser (i.e. an `https://` connection instead of an `http://` connection).

Obtain an SSL Certificate

SSL certificates need to be issued from either a trusted certificate authority or from an accredited domain registrar.

Browsers, operating systems, and mobile devices maintain lists of trusted certificate authority root certificates, which must be present on a computer for it to trust the certificate.

The following is a list of popular certificate authorities from which you can obtain an SSL certificate:

- [Comodo](#)
- [IdenTrust](#)
- [GoDaddy](#)
- [GlobalSign](#)
- [Digicert](#)
- [Certum](#)
- [Entrust](#)

Go to [ICANN](#) for a complete list of accredited domain registrars.

Because SAFR uses Apache as its web server, request SSL certificate files for Apache web server. You will receive the following three files SAFR uses to configure the Apache web server:

- **Key:** This is your key file and should not be shared publicly.
- **Certificate:** The SSL certificate for your domain.
- **Ca_bundle:** Signer root/intermediate certificate. **Note:** This file is optional; it's not always provided by the SSL certificate provider.

Note: Self-signed certificates do not work.

Provision SSL Certificates for your Primary Server

Do the following to configure Apache to serve the request over HTTPS:

1. Log in to your primary server.
2. It is recommended that you make a backup of the default SSL files and save them in case you need to perform a rollback to the earlier version.
 - On macOS, back up the following files:
 - `/etc/apache2/ssl/SAFR.key`
 - `/etc/apache2/ssl/SAFR.crt`
3. Upload the certificate-related files to the SSL certificate folder:

- SSLCertificateFile – Certificate CRT
- SSLCertificateKeyFile – Private.a key file

4. Change the names of the following files:

- Rename *_certificate.crt to SAFR.crt
- Rename *_private.key to SAFR.key

5. If your certificate authority provided an intermediate certificate chain, do the following:

1. Save your SSL intermediate certificate chain file to the following location:

- **On macOS:**

- /etc/apache2/ssl/SAFR-ca.crt

2. Check the SAFR-ssl-cert.inc file to connect your SSL certificate to the certificate chain.

- **On macOS:**

- /etc/apache2/other/SAFR-ssl-cert.inc
- #Define ssl_certificate_chain_file
"/private/etc/apache2/ssl/SAFR-ca.crt"

- **Certificate file mappings**

Certificate file	Certificate file in SAFR
*.domainname.key	SAFR.key
*.domainname_chain.crt	SAFR-ca.crt
*.domainname_public.crt	SAFR.crt

6. Run the SAFR **reconfigure** script, as described below.

- **On macOS:**

- Open **Applications > Utilities > Terminal** to open a Terminal window.
- Run the following command after replacing hostname.domain.com with your hostname and domain:
 - /Library/RealNetworks/SAFR/bin/reconfigure
hostname.domain.com

```

C:\Windows\system32\cmd.exe
Enter new hostname: hostname.domain.com
Does your SSL Certificate use a Certificate Chain? [Y,N]? Y
Administrative permissions required. Detecting permissions...
Current permissions inadequate - escalating.
  
```

See SAFR Support Tools and Scripts for more information about this script.

7. Verify that your services are running and your SSL certificate is properly installed by opening a browser and opening `https://hostname.domain.com:8085/health`. (Replace `hostname.domain.com` with your hostname and domain.)

You should receive the following message:

```
{ "status" : "up" }
```

Troubleshoot

Database Service Down

Problem: You receive an error report saying Database (MongoDB) Service Down when you run the **check** command after you install SSL.

Solution: The cause may be that the DNS hostname IP is different from the IP when you installed SAFR without SSL installed.

Use the following workaround:

Add the following line to your primary server `/etc/hosts` file: `127.0.0.1`
`hostname.domain.com`

SAFR Support Tools and Scripts

The SAFR Platform installation includes several scripts to manage and monitor your server. They are located in the `bin` folder under the SAFR Platform installation location.

- On macOS: `/Library/RealNetworks/SAFR/bin`

Note: Some of the scripts below may not work if you're accessing the SAFR Platform through the NVIDIA Metropolis Application Framework (MAF).

Tools

check

Use the **check** command to check the status of SAFR Server services.

- On macOS, run `/Library/RealNetworks/SAFR/bin/check`

configure-ports

Use the **configure-ports** command to customize the ports SAFR services listen on. This is

typically done only if there is a conflict with existing software on the same server.

If port conflicts are detected during SAFR Platform installation, the following occurs:

1. The ports in conflict are reported.
2. Notepad is launched to edit safrports.conf
3. The SAFR Platform installer is automatically relaunched after new non-conflicting ports are chosen.

This command is executed as part of the installation when appropriate, so it does not need to be executed manually unless you are changing the port settings after installation.

This command takes no arguments but relies on the safrports.conf file to determine what ports are to be used.

safrports.conf is located at the following locations:

- On macOS: `/Library/RealNetworks/SAFR/safrports.conf`

reconfigure

Use the **reconfigure** command to configure the hostname used by the SAFR Server. Run this command when configuring the server to use a DNS hostname with an SSL certificate.

This command can be run with arguments specifying the hostname and whether an SSL certificate chain is used by your SSL certificate. If no arguments are passed, you will be prompted for those values.

This command requires administrator privileges. It automatically asks for admin privileges on Windows and requires `sudo` on macOS and Linux.

- On macOS, run `/Library/RealNetworks/SAFR/bin/reconfigure`

Examples:

macOS:

- `/Library/RealNetworks/SAFR/bin/reconfigure`
- `/Library/RealNetworks/SAFR/bin/reconfigure safr.example.com y`

start

Use the **start** command to start up the SAFR Server. It starts all server services on the current machine.

- On MacOS, run `/Library/RealNetworks/SAFR/bin/start`

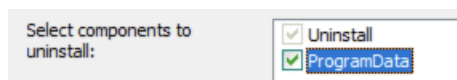
stop

Use the **stop** command to shut down the SAFR Server. It stops all server services on the current machine.

- On macOS, run `/Library/RealNetworks/SAFR/bin/stop`

uninstall

Use the **uninstall** command to remove the SAFR Platform entirely. This closes all SAFR applications, stops all SAFR services, and then removes all SAFR services and data.



- On macOS, run `/Library/RealNetworks/SAFR/bin/uninstall`

SAFR Server Backup and Restore

The backup process backs up and restores the entire SAFR Server, including the various databases, configuration files, images, and objects.

On macOS

Backup

command path: `/Library/RealNetworks/SAFR/bin`

run command: `sudo python backup.py`

The backup command generates a backup file at the path

`/opt/RealNetworks/SAFR/backups/SAFR-backup-YYYYMMDD-HHMMSS.tgz`

You'll receive the following message when the backup is complete:

- Backup File: `/Library/RealNetworks/SAFR/backups/SAFR-backup-20190814-003342.tgz` SAFR Backup Complete.

Restore

command path: `/Library/RealNetworks/SAFR/bin`

run command: `sudo python restore.py BACKUPFILENAME`

- **Example:** `sudo python restore.py /opt/RealNetworks/SAFR/backups/SAFR-backup-20190814-083700.tgz`

Press Y when asked, “Are you sure? (Yy/Nn)”

You’ll receive the following message when the restore is complete:

- SAFR Restore Complete.

Video Recognition Gateway (VIRGO)

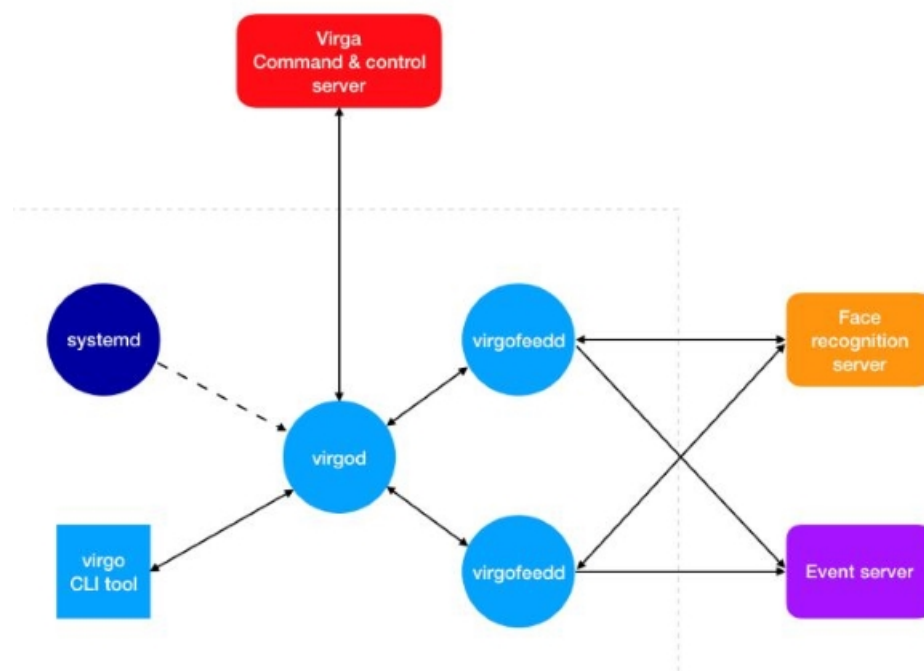
VIRGO (Video Recognition Gateway) is a daemon system which runs on a POSIX compatible system. It receives video feeds from one or more cameras and recognizes and tracks faces in those video streams in realtime. It generates tracking events and sends those events to an event server. The VIRGO daemon can be controlled either by the command line tool or through the VIRGA command & control server.

Architecture

A single VIRGO installation consists of the following components:

- **virgod:** The VIRGO control daemon. One such daemon is spawned and maintained per VIRGO hardware.
- **virgofeedd:** A *virgod* child process which handles a single video feed.
- **virgo:** The locally available VIRGO command line tool which acts as a Command Line Interface (CLI)-based user interface to the VIRGO daemon.

This diagram shows how those components fit together:



virgod:

- Spawned by the operating system systemd/launchd service. The daemon is automatically restarted by the OS if the hardware power cycles or virgod terminates for some unexpected reason.
- Runs as its own user. The VIRGO user is limited to read/write access to the “virgo” home directory.
- The VIRGO user home directory contains just the ~/Library directory which is the place where libFoundation (used in the implementation of VIRGO) stores the daemon settings.
- Is responsible for spawning the per-video-feed child processes: virgofeedd.
- *virgod* monitors each *virgofeedd* child process that it has spawned and it automatically restarts a *virgofeedd* if it unexpectedly terminates for some reason. (e.g. it ran out of memory)
- Is responsible for carrying out all the necessary steps for an update to the VIRGO daemon system.
- Is the only process on the machine which talks to the VIRGA command & control server.
- carries out any command sent by VIRGO to *virgod*.
- regularly informs VIRGA about the current status of *virgod*.

virgofeedd:

- Spawned by *virgod*.
- Runs as the same user as *virgod*.
- Receives a video stream. Detects and recognizes faces in that video stream, generates events and reports them to the event server.
- Receives commands from *virgod*.

virgoupdaterd:

- Spawned by *virgod* after it has received an update request.
- Runs as the same user as *virgod*.
- Downloads the update archive, extracts it, installs the update bundle, and saves the current persistent *virgod* state.
- Restarts *virgod*. (*virgod* takes care of data migration.)
- Monitors *virgod* after restart and rolls back to the previous *virgod* version if the new *virgod* fails to startup or fails to check back in with a commit message in less than a couple seconds.
- Once the update has finished, the updater exits.

virgo:

- Implements the local (CLI-based) user interface to *virgod*.
- Offers commands to show the current status, select the cloud environment, get a screen capture from a feed, etc.

VIRGO Bundle (File System Layout)

VIRGO ships as a bundle which supports multiple versions of the VIRGO daemon. The VIRGO bundle directory contains a “versions” directory which in turn contains one sub-directory per installed VIRGO version. The name of a version sub-directory is the semantic version number of the VIRGO installation. The “versions” directory also contains a symlink named “current”. This symlink points to the version sub-directory which is currently active.

The version sub-directory stores all necessary executable, library, and data files for the VIRGO version.

VIRGO bundle layout:

```
virgo/
  versions/
    1.0.0/
      virgo
      virgod
      virgofeedd
      virgoupdaterd
      lib/
        <shared libraries>
      model/
        <tensor flow model files>
      virgo-factory.config
      current -> ./1.0.0
    virgo -> ./versions/current/virgo
```

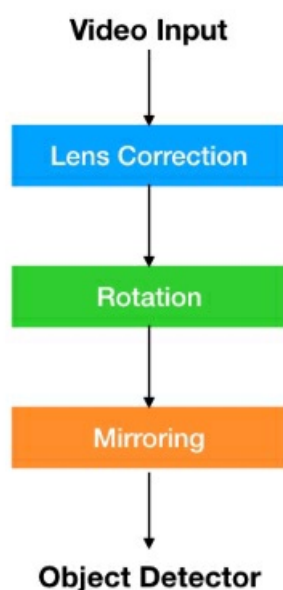
VIRGO Feeds

A single *virgod* instance manages a set of feeds. Each feed represents a video stream from a camera, a file, or some other video source. Each feed is associated with a set of configuration information which is stored persistently by VIRGO. The configuration information for the feeds is either provided by the VIRGO server through the COP-HTTP protocol or through the VIRGO command line tool and the COP-DTP protocol.

Each feed has a name which is unique among the set of feeds of a single *virgod* instance. These names are used as a simple and convenient way to refer to a feed and its configuration. Each feed is managed by a separate *virgofeedd* instance which is started and monitored by *virgod*. *Virgod* will automatically restart a *virgofeedd* instance if it dies for some unexpected reason.

A feed may be enabled or disabled. Only enabled feeds are associated with a *virgofeedd* instance. The enabled state of a feed may be changed through the VIRGO command line tool by issuing a **feed start** or a **feed stop** command. A feed may also be enabled or disabled through the COP-HTTP protocol by changing the **enabled** setting in the feed configuration dictionary. This allows the system to reclaim resources like memory and network bandwidth if a feed is temporarily not needed. Feeds which are no longer needed at all should be removed altogether.

A feed has an input which connects the feed to a video stream. The only type of input currently supported is “stream”. A stream input is specified by a URL which may point to a publicly accessible RTSP, HTTP, or FILE video stream. Each video frame from the input is first sent through a video post-processing pipeline before it is fed into the object detector and recognizer sub-systems:



First a lens correction algorithm is applied to an incoming video frame. This step removes distortions that may be introduced by the optical system of a camera. After that the image will be rotated to compensate for any undesired rotation that may have been introduced by the physical orientation of the camera. Finally the image may be mirrored to ensure that a camera that is facing a user will produce an image that aligns with what a user expects to see.

VIRGO Installation Guide

System Requirements

See the VIRGO System Requirements page before you start the VIRGO installation process. Note that VIRGO depends on certain 4r party libraries which must be installed before installing VIRGO.

Download the VIRGO Installer

The macOS and Linux VIRGO installers can be downloaded from the SAFR Download Portal here: <https://safr.real.com/developers>

VIRGO Installer Package

This package installs VIRGO as a system or user daemon. The system daemon installation ensures that VIRGO will be able to run independently of any logged in user and it will start running as soon as the computer is booted up. Administrator privileges are required to complete the installation. VIRGO will look for factory default settings in the `/etc/virgo-factory.conf` file. The user installation on the other hand links Virgo to the user who installed it. The VIRGO daemon will only be accessible to this user and it will only run while this user is logged in. However no administrator privileges are required to install and operate VIRGO in this mode. VIRGO will look for factory default settings in the `~/virgo-factory.conf` file.

The following sections describe how to use the platform-specific version of the VIRGO installer package.

Installer name: `Virgo.pkg`

Follow these steps to install VIRGO:

1. Download the macOS VIRGO installer from the SAFR Download Portal here: <https://safr.real.com/developers>
2. Double click it.
3. The installer will guide you through the necessary steps to complete the installation. Note that you can choose between a system and user installation by selecting the appropriate install location option.

VIRGO will be installed into the following location:

- For a user installation: `~/Library/RealNetworks`
- For a system installation: `/Library/RealNetworks`

The installer script will ask you for all necessary information and guide you through the installation process.

The final VIRGO configuration information is written to a factory configuration file which is stored in the required file system location from where VIRGO is able to read it. Note that for security reasons the factory configuration file is only readable and writeable by the user who owns the VIRGO daemon. The following code block shows an example of how to install VIRGO as a system daemon:

- `> sudo virgo_installer/install.sh`

FAQ for macOS Installations

1. I've installed VIRGO as a system daemon. How do I change the factory configuration?

Place your custom factory configuration file in the `/etc` directory and then reset the VIRGO service like this:

```
Assuming that the factory configuration file is at:  
  
/etc/virgo-factory.conf  
  
> virgo service reset
```

2. I've installed the VIRGO Package. How do I uninstall VIRGO?

For system daemon installations, execute the following command from the Terminal:

```
> sudo /Library/RealNetworks/virgo/uninstall.sh
```

For user daemon installations, execute the following command from the Terminal:

```
> ~/Library/RealNetworks/virgo/uninstall.sh
```

3. I've installed the VIRGO as a user daemon. How do I stop *virgod*?

Execute the following command from the Terminal:

```
> launchctl bootout gui/${id -u}  
~/Library/LaunchAgents/com.real.virgod.plist
```

This command terminates the *virgod* daemon. Keep in mind that the VIRGO command line tool will automatically restart *virgod* when you use it again.

VIRGO System Requirements

- macOS 10.11 or newer is required.
- Swift 5 Runtime Support for Command Line Tools must be installed. See [this Apple support article](#) for more information.

VIRGO requires the following x86_64 CPU features:

- Ivy Bridge or better CPU architecture
- SSE 4.2
- AVX

VIRGO on macOS is able to take advantage of GPUs to accelerate video decoding, image processing, face detection, and object detection. The GPU requirements are:

- Metal version 1

VIRGO Command Line Interface

The command line interface is designed based on an object - verb structure.

- VIRGO is conceptually organized into sub-systems which are represented by “objects”.
- “Verbs” are commands that can be issued on an object.
- Some verbs may require additional parameters.

VIRGO currently defines the following types of objects (subsystems):

- **Service:** The VIRGO daemon itself.
- **Feed:** A video stream. (e.g. from a camera)
- **Environment:** The environment to which *virgod* connects.

The sections below describe the VIRGO command line syntax. Note that VIRGO command line options follow the standard POSIX convention. This means that many of those options come in a short (single dash prefix) and a long (double dash prefix) form.

Command Line Options

Help

```
> virgo --help
> virgo -h
<help text>
```

Shows all available VIRGO command line options.

Administrator

Get the current administrator configuration

```
> virgo administrator get
```

This command causes VIRGO to print the current administrator configuration. VIRGO may either be administrated by a cloud server (aka VIRGA) or it may be self-administrated. 'Virgo' is printed in the former case 'Virga' in the later.

Setting the administrator configuration

```
> virgo administrator set <name> // <name> is either 'virga' or 'virgo'
Administrator: <name>
```

This command causes VIRGO to switch to the specified administrator. Pass 'virga' if VIRGO should be administrated via the VIRGA server. Note that the environment definition must contain an admin-server-url entry in this case. Pass 'virgo' if VIRGO should be used standalone without a cloud command & control server. Standalone mode allows you to freely add, remove, and change feeds whereas the VIRGA administration mode requires that feeds are added, removed, and changed via VIRGA.

Service

Get information about the VIRGO service

```
> virgo service info
Version:      1.0.0
Target:       x86_64-macos
Domain:       System
Administrator: Virga
Environment:  PROD
Client ID:    <client-id>
Client Type:  <client type>
```

This command prints the following information about the installed VIRGO daemon build and its fundamental configuration.

- **Version:** The build version of the VIRGO daemon.
- **Target:** Specifies for which operating system and CPU architecture the VIRGO daemon was built.
- **Domain:** Specifies whether the VIRGO daemon is running as a system-wide daemon (system) or a daemon which is only available to the currently logged in user (user). Note that user-wide VIRGO daemons will terminate when the user logs out.
- **Environment:** The environment to which the VIRGO daemon connects in order to receive commands from the command & control server.
- **Client ID:** The client ID that the VIRGO daemon sends to the command & control server to identify itself.
- **Client Type:** TBD

Get the current service status

```
> virgo service status
camera_1: ok
camera_2: ok
camera_3: inactive
```

This command tells VIRGO to print the current status of all registered feeds.

Monitor the current status of all feeds

```
> virgo service monitor
```

This command enables the service monitor. See Service Monitoring for more information.

Logging

```
> virgo service log <log specification>
```

This command displays the current service log. See Service Logging for more information.

Resetting the VIRGO daemon state

```
> virgo service reset
```

This command tells VIRGO that it should delete its current state and reinitialize it from the contents of the factory configuration file. This effectively resets the daemon back to the factory state.

Updating VIRGO

```
> virgo service update <version> <url> [--verbose]    // download an install
a new version.
> virgo service update <version> [--verbose]          // switch virgo to a
previously installed version. E.g. downgrade to an old version.
```

This command causes VIRGO to upgrade or downgrade to the specified version. <version> is the version to upgrade or downgrade to and <url> is a file or HTTP/HTTPS URL that points to VIRGO update archive. Specifying the update archive URL is only necessary if the version you are trying to switch to isn't already installed on the machine. By default VIRGO shows the current update status and progress. Specify the “--verbose” switch to cause VIRGO to show the full update log instead.

VIRGO update bundles are available from the Jenkins build machine.

Get information about the installed VIRGO versions

```
> virgo service versions
Installed:
  1.0.0
  1.1.0
-> 1.2.0

Current:
  1.2.0
```

This command causes VIRGO to print the version numbers of all installed VIRGO packages plus the version number of the currently active and running VIRGO daemon.

Environment

A VIRGO daemon has a built-in list of supported environments. Only one of those environments can be active at a given time. The active environment determines to which VIRGA, face recognition, and event servers *virgod* and its *virgafeedd* child processes will talk.

List supported environments

```
> virgo environment list
DEV
INT2
LOCAL
PROD
```

Lists all environments supported by VIRGO.

Get the active environment

```
> virgo environment get [--verbose]
INT2
```

Returns the currently active environment. This is the environment to which *virgod* and all of its *virgofeedd* daemons connect. Additionally VIRGO will show the URLs of the individual servers in the environment if you pass the `--verbose` flag.

Set the active environment

```
> virgo environment set <environment name> [--verbose]
OK
```

Sets the environment which VIRGO and its feeds will use. Note that `<environment name>` must be one of the supported environments or one of the custom environments defined in the factory configuration file. Note that changing the environment also resets the VIRGO daemon back to the factory configuration.

By default the command prints “OK” if the switch to the new environment succeeds, while it prints an error if one or more services can not be contacted. You can pass the `--verbose` flag to get a detailed status for each service.

Cloud User

Get cloud account details

```
> virgo user get
User ID: <user id>
Password: ***
```

Prints the *User ID* and an indication whether a password was provided. Three asterisk characters indicate that VIRGO has a password on file, while an empty password line indicates that VIRGO doesn't have a password for the user on file.

Set the cloud account

```
> virgo user set
User ID: <user id>
Password: ***
```

Replaces the current cloud account's credentials with the provided *User ID* and *Password*. All currently enabled feeds are automatically restarted with the new account information.

Feeds

A single *virgod* daemon instance is capable of managing a set of video feeds. *Virgod* spawns one *virgofeedd* instance per feed and this *virgofeedd* instance is exclusively responsible for tracking its assigned feed. *Virgod* automatically respawns a *virgofeedd* instance if it dies unexpectedly.

A feed has:

- A name which is used to identify a particular feed.
- An RTSP URL which provides access to the video stream.
- Default face detection, recognition, and tracking parameters.
- Additional information to control features like lens correction.

Virgod stores the configuration information for a feed persistently. A feed can be added, removed, started, and stopped at any time. A VIRGO instance may come prepackaged with the configuration information for one or more feeds. New feeds may be added dynamically any time as long as *virgod* is running.

List feeds

```
> virgo feed list
camera_1
camera_2
```

Lists all enabled and disabled feeds that have been registered with VIRGO.

Get the configuration information for a feed


```
> virgo feed get <feed name>
{
  "active":true
  "url":"rtp://camera.is.here/with/stream:8789"
  ...
}
```

Prints the feed configuration JSON dictionary. See VIRGO-COP for a description of the feed configuration format.

Update/set the configuration information for a feed

```
> virgo feed set <feed name> <feed config file path>
```

Updates the current configuration of the feed with name <feed name>. The feed configuration file is read and the properties in the configuration file are applied to the current feed configuration stored in VIRGO. The feed configuration file is a JSON file with a single dictionary which contains the feed properties that you want to change. Note that the feed configuration file only needs to contain those properties that you want to change. See VIRGO-COP for a list of supported feed properties.

Get the PID of a feed

```
> virgo feed get-pid <feed name>
53280
```

Prints the *PID* of the feed. -1 is printed if the feed is currently not active and thus no feed daemon is running to process the feed video stream.

Get the status of a feed

```
> virgo feed status <feed name>
ok
```

Prints the current status of a feed.

Add a new feed

```
> virgo feed add <feed name> <feed config file path>
```

Adds a new feed configuration to the persistent list of feeds. The feed name must be unique with respect to the VIRGO instance. The feed configuration is read from the supplied feed

configuration file. See VIRGO-COP for a list of supported feed configuration keys. The feed will immediately start processing if it is marked as “enabled” in the configuration file; otherwise the feed will be added to the persistent list of feeds but a separate “virgo feed start <feed name>” command will have to be issued to cause the feed to start running.

Remove an existing feed

```
> virgo feed remove <feed name>
```

VIRGO will stop the feed and then remove the feed configuration information from its persistent feed table.

Starting a feed

```
> virgo feed start <feed name>
```

VIRGO will mark the feed as active and start processing it. A video file feed starts processing from the beginning of the video while a camera feed starts processing from the current time code of the video stream. If the feed is already active and running this command instead does nothing.

Stopping a feed

```
> virgo feed stop <feed name>
```

Marks the feed as inactive and stops processing the video stream. If the feed is already marked as inactive, then this command instead does nothing.

Capturing an image from a feed

```
> virgo feed capture-image <feed name> <url or path> [--size <image_size>]
[--max-frames <max_number_of_frames>] [--frame-delay
<delay_in_milliseconds>]
```

Enables capturing of a single image or a series of consecutive images from the specified feed. <url or path> is a file or HTTP URL or a file system path. The URL/path is expected to point to a directory. VIRGO will create the directory if necessary and it will write all captured images to this directory. The size of the larger side of the capture image can be specified with the `--size` option. The default capture image size is 720 pixels. The maximum number of consecutive frames that should be captured can be specified with the `--max-frames` option. The default is to capture a single image. The `--frame-delay` option allows you to specify the delay between consecutive frames in milliseconds.

Docker

The VIRGO application runs as a Docker Container alongside all the other native services as part of the SAFR Linux Platform.

If all the services are running as Docker Containers then the system is in **Swarm Mode**.

Initial Configuration

The VIRGO container starts for the first time with no factory configuration file. It will remain in this state until a new configuration has been generated and activated.

Configuration

The factory configuration file is generated when the following configuration script is called by CoVi during licensing (kickoff):

```
/opt/RealNetworks/SAFR/virgo/app/virgo/app/virgo_configure.sh
```

The script requires both a *username* and a hashed *password* to be passed in.

NOTE: If either of these are missing the script will not generate the configuration.

The script requires a template file

```
/opt/RealNetworks/SAFR/virgo/app/virgo/config/virgo-factory.template
```

in order to generate a new configuration.

Once executed the script will generate a working configuration and will store it in the following file. **NOTE:** The existing configuration will be overwritten!

```
/opt/RealNetworks/SAFR/virgo/app/virgo/config/virgo-factory.conf
```

After the configuration is generated the VIRGO container will be restarted to activate the newly generated configuration.

Service Status

There are two ways to confirm if VIRGO is running or to confirm how long it has been operational.

1. Use the **check** utility located in `/opt/RealNetworks/SAFR/bin`
2. Use Docker command to show active running containers:
 - **# sudo docker ps**

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS NAMES		
cf2a2dd33875	safr_virgo:1.1.38	"/bin/sh -c \$VIRGO_A..."	18 hours ago
Up 18 hours	safr_virgo		

If there is no output check the following:

- Run the same command again with -a to determine if the container is stopped or restarting (failing);
- Verify there is a valid virgo-factory.conf located in
/opt/RealNetworks/SAFR/virgo/configs/
 - Correct user name and IP Address (Same as host IP)
 - Password is not readable so hard to validate

Swarm Mode

In addition to checking the container status we also need to confirm the services are all up and stable.

The following command will output a table showing each service and how many instantiations there are.

```
docker service ls
```

The following command will tell you how many nodes are running in the Swarm Cluster

```
docker node ls
```

NOTE: If either of these commands fail to execute it's likely that you are not in **Swarm Mode** or the swarm node is down.

Execution

VIRGO container will remain operational both after a failure has occurred or if the OS is restarted.

The container is started by the SAFR Platform Installer and stopped by the SAFR Platform Uninstaller.

Logging

Execute the following command to provide logging output.

```
sudo docker exec -it safr_virgo /opt/RealNetworks/virgo/virgo service  
log <log options>
```

NOTE: Refer to VIRGO Logging for more information on logging options.

Swarm Mode

Slightly more complex as you need to obtain the container id first.

```
sudo docker ps | grep virgo | awk '{print $1}'
```

Once you have the container id run this command by substituting in the container id.

```
sudo docker exec -it {container_id} /opt/RealNetworks/virgo/virgo service  
log <log options>
```

Service Monitor

Live view

```
sudo docker exec -it safr_virgo /opt/RealNetworks/virgo/virgo service  
monitor
```

Active Feeds to CSV

```
sudo docker exec -i safr_virgo /opt/RealNetworks/virgo/virgo service monitor  
> {CSV File} --active-only
```

NOTE: The stats are added to the CSV file every second so the usable data can be large depending on the number of active feeds.

Upgrade

To upgrade VIRGO you need to perform the following steps depending the platform architecture.

Standalone Container

- Upload new VIRGO Docker Image to the deployment server (location does not matter).
- Load image into local registry.
 - `docker load < {image_file}`
- Update `/opt/RealNetworks/SAFR/virgo/app/docker-compose.yml`.
- Restart VIRGO container.
 - `docker restart safr_virgo`

Swarm Mode

- Upload new VIRGO Docker Image to the deployment server.
- Load image into local registry.
 - `docker load < {image_file}`
- Update VIRGO service with new image. (Service will restart automatically.)
 - `docker service update --image safr_virgo:{version string} safr_virgo`

Add Volume Mount to Existing Container

1. Update the compose file to add the additional volume instructions (Below is just an example local folder name).

- The format is `<local folder>:<docker folder>`
- The `<docker folder>` will be created if not already existing.

```
version: "3.6"
services:
  virgo:
    image: safr_virgo:1.2.12
    container_name: safr_virgo
    restart: on-failure
    pid: "host"
    volumes:
      - /opt/RealNetworks/SAFR/virgo/config:/etc/virgo
      - /opt/RealNetworks/SAFR/virgo/files:/opt/RealNetworks/virgo/files
```

2. Create (or check it already exists) the local folder to mount into the container.

```
# mkdir -p /opt/RealNetworks/SAFR/virgo/files
```

3. Stop and delete the container.

```
# docker-compose -f /opt/RealNetworks/SAFR/virgo/app/docker-  
compose.yml down
```

4. Create container instance with new volume mount.

```
# docker-compose -f /opt/RealNetworks/SAFR/virgo/app/docker-  
compose.yml up -d
```

5. Create test file in local mount point.

```
# touch /opt/RealNetworks/SAFR/virgo/files/testfile
```

6. Check file exists inside the container's mount location.

```
# docker exec -it safr_virgo ls -l /opt/RealNetworks/virgo/files
```

Swarm Mode

When all applications are running in Docker containers we are running as a Docker Swarm.

Factory Configuration

Every VIRGO daemon ships with factory settings which define the default configuration that the daemon should use the first time it starts up. *Virgod* also reverts the current configuration back to the factory settings if it is unable to load the current configuration because of a version mismatch and it is unable to automatically convert the old configuration to the new format.

The factory settings are stored in a JSON file with the name `virgo-factory.conf`. *Virgod* looks in the following locations to find a factory configuration file:

- The home directory of the user who started *virgod*.
- The `/etc` directory.
- The VIRGO bundle directory.

Virgod loads the first factory configuration file that it finds. If it can't find any factory configuration file, it falls back to hardcoded defaults.

Factory Configuration File Format

The factory configuration file is a JSON file which is organized into (optional) sections:

```

{
  "global": {           // [optional]
    // global state
  },
  "environments": {     // [optional]
    "Foo": {
      // environment specific URLs
    }
  },
  "feeds": {            // [optional]
    "camera_1": {
      // feed state
    }
  }
}

```

Note: Nearly all keys in a factory configuration file are optional. Only those keys that you explicitly want to override with a custom value need to be specified. *Virgod* uses hardcoded default values for keys that are missing from a factory configuration file.

The Global Section

The following properties are supported in the global section:

Property	Type	Default	Description
status-interval	Int?	5000	Status reporting time interval in ms.
environment	String?	PROD	The name of the environment which should be used by <i>virgod</i> . See the “Environments Section” below for a list of pre-defined environment names.
machine-id-prefix	String?	empty string	The machine ID prefix. The default machine ID prefix is the empty string.
machine-id	String?	OS defined machine ID	The machine ID. The default machine ID is derived from the OS provided machine ID. The concatenation of the machine-id-prefix and the machine-id is sent to the cloud in the X-CLIENT-ID header.

Property	Type	Default	Description
client-type	String?	OS defined client type	The client type. This value is sent to the cloud in the X-CLIENT-TYPE header.
user-id	String		The user ID for the cloud account.
user-password	String		The password for the cloud account. Note that the password is stored in clear text. Use <i>user-encrypted-password</i> whenever possible instead.
user-encrypted-password	String		The encrypted password for the cloud account.
administrator	String?	cloud	Specifies whether VIRGO should be administrated by VIRGA or whether it should be self-administrated. A self-administrated VIRGO allows you to manage feeds via the VIRGO command line tool.
visible-accelerator-ids	[Int]?		Allows you to specify which GPUs/accelerators VIRGO is allowed to use for video decoding and detection tasks. Only the accelerators listed in this array will be used by VIRGO; all others will be ignored. The value is an array of accelerator IDs. VIRGO will use all available accelerators if this property is not set.

Note that feeds which are assigned to a specific accelerator ID will fail with an error at startup if that accelerator is not in the set of visible accelerator IDs.

The Environments Section

The environments section defines the available cloud environments. Each environment has a name and a set of URLs that point to the hosts in the cloud that provide the required services. An environment may override one of the pre-defined environments. The environment name is

used to identify the environment and to switch among environments with the `virgo environment set` command.

The following properties are supported in the environments section:

Property	Type	Default	Description
covi-server-url	URL	none	The face recognition service.
rncv-server-url	URL?	none	The face detection service.
event-server-url	URL	none	The detection and recognition event recording service.
object-server-url	URL	none	The service which stores objects such as images and logs.
admin-server-url	URL	none	The VIRGO administration service.

The following table lists the pre-defined environments:

Name	Alternative name
SAFR Local	LOCAL
SAFR Developer Cloud	DEV
SAFR Partner Cloud	INT2
SAFR Cloud	PROD

You can use the alternative environment name in place of the full environment name.

The Client ID

VIRGO computes the client ID by concatenating the `machine-id-prefix` and the `machine-id` properties.

Example Configuration Files

The following subsections show some typical factory configuration files.

Using VIRGO with a VIRGA server

This is an example of a configuration file which configures VIRGO to run as a slave to a VIRGA server. VIRGO will continuously report its status to the VIRGA server and the VIRGA server is

responsible for pushing state changes to VIRGO.

```
{
  "global" : {
    "environment": "DEV",
    "machine-id-prefix": "foo",
    "user-id": <user ID>,
    "user-password": <password>
  }
}
```

Using VIRGO standalone

This is an example of a configuration file which configures VIRGO to run as a standalone daemon which does not connect to a VIRGA server. VIRGO starts processing the declared feeds as soon as it starts up. Note that you still have to provide a user ID and a password to allow VIRGO to use the (cloud-based) face recognition and event recording service.

```
{
  "global":{
    "environment": "DEV",
    "machine-id": "argusrn",
    "user-id": <user ID>,
    "user-password": <password>,
    "administrator":"self"
  },
  "feeds":{
    "camera_1":{
      "directory":"test",
      "input.type":"stream",
      "input.stream.url":"file://<absolute path to a movie file>",
      "recognizer.learning-enabled":true,
      "enabled":true
    }
  }
}
```

Defining Custom Environments

This is an example of a configuration file which defines two custom cloud environments. Note that the first custom environment has a new unique name that is separate from any of the pre-defined environments. The second custom environment, on the other hand, overrides the pre-defined environment name `PROD`. Consequently VIRGO will use the URLs of the custom environment if the `PROD` environment is selected. This allows you to replace the built-in definition of the pre-defined environment.

```

{
  "global": {
    "environment": "Test"
  },
  "environments": {
    "Test": {
      "covi-server-url": "https://covi.test.real.com",
      "event-server-url": "https://event.test.real.com",
      "object-server-url": "https://object.test.real.com",
      "admin-server-url": "https://admin.test.real.com"
    },
    "PROD": {
      "covi-server-url": "https://covi.sim.real.com",
      "event-server-url": "https://event.sim.real.com",
      "object-server-url": "https://object.sim.real.com",
      "admin-server-url": "https://admin.sim.real.com"
    }
  }
}

```

Service Logging

The VIRGO command line tool has a simple logger built in. You enable logging by executing the following command in a shell:

```
> virgo service log <log specification>
```

where the log specification is a space separated list of log predicates. A log predicate looks like this:

```

level/tag
level/tag[feedName]

```

The first variant sets the log level for the package *tag* to *level* on a global basis. Consequently this log predicate applies to the VIRGO daemon and all feeds it spawns. The second variant allows you to apply the log predicate to a single feed with the name *feedName*. If you specify both a global- and a feed-specific log level for a tag then the level with higher priority is applied.

Note: The VIRGO daemon does not keep a log history. Log information is only generated and retained while you are actively running a `virgo service log` command.

Examples:

```
> virgo service log D/tracking
```

Enables DEBUG level logging for the 'tracking' package in all feeds.

```
> virgo service log D/capture D/cop-http
```

Enables DEBUG level logging for the 'capture' and the 'cop-http' packages in all feeds.

```
> virgo service log D/tracking[foo]
```

Enables DEBUG level logging for the 'tracking' package in the feed 'foo'. This does not change the current log configuration for any other feed.

The following log levels are supported:

Level	Description
V	Verbose
D	Debug
I	Info
W	Warn
E	Error
O	Off

The order in terms of verbosity, from least to most verbose is OFF, ERROR, WARN, INFO, DEBUG, and VERBOSE.

The following log packages are supported:

Package	Supports feed name?	Description
detection	yes	Object detector related messages
recognition	yes	Face recognizer related messages
tracking	yes	Object tracker messages
capture	yes	Image capture related messages
events	yes	Event reporting related messages
pose-liveness	yes	Pose Liveness Action Recognizer related messages
feed	yes	Feed life cycle related messages
cop-http	no	COP over HTTP related messages

Package	Supports feed name?	Description
config	no	<i>Virgod</i> configuration management related messages
updates	no	<i>Virgod</i> update initiation mechanism related messages

Service Monitoring

The VIRGO command line tool has a service monitoring user interface built in. Execute the following command in a shell window to activate continuous monitoring:

```
> virgo service monitor
```

After executing this command, VIRGO clears the terminal window and presents the following live screen:

Status	Feed	PID	Epoch	P-Time	Resolution	FPS	DPS		
dDt	dRt	#D	#D-Badge	#D-Face	#D-Skip	#R	#R-Face	#R-Err	#R-
Skip	#Evt	%CPU	GPU#	GPU	GPU-Name				
ok	camera_1	14536	12/06/17	00:24:13.450	1280x720	120	8ms		
250ms	120	18	10	0	0	8	0	0	0
1240	1%	0	VF	GTX 1060					
ok	camera_2	67289	13:07:12	80:10:00.000	1920x1080	29.97	8ms		
250ms	1920	1400	0	0	0	1000	50	1	0
10	4%	1	VF	GTX 1050					
inactive	camera_3								

Note that the screen is live, which means that VIRGO continuously updates it every second. You can quit monitoring by pressing the 'q' key or by pressing Ctrl-C. Also please keep in mind that VIRGO only shows as many columns as fit on the screen. If you do not see all columns then this means that your terminal window is not wide enough. Make the window wider to see all of the columns.

The service monitor UI allows you to scroll up and down when there are more feeds than fit vertically in the terminal window. Use the cursor up key to scroll up and the cursor down key to scroll down.

The following table explains what the various columns in the monitoring output mean:

Column Name	Description
Status	The feed status. This is one of ok, inactive, eos, error or failure.
Feed	The feed name.
PID	The PID of the feed daemon if the daemon is running

Column Name	Description
Epoch	The time when the feed processed the first frame in the video stream.
P-Time	The amount of time that the feed has spent on processing the video stream. This is in terms of milliseconds.
Resolution	The width and height of a video frame in pixels
FPS	The frames per second of the input video.
DPS	The number of detections per second.
dDt	The latency of a single detection operation in milliseconds.
dRt	The latency of a single recognition operation in milliseconds.
#D	The number of detection operations that have been triggered.
#D-Badge	The number of badges that have been detected.
#D-Face	The number of faces that have been detected.
#D-Skip	The number of detection operations that have been skipped due to detector overcommitment. This means that no detector was available for a video frame because all detectors were busy at that time.
#R	The number of face recognition or reconfirmation operations that have been triggered.
#R-Face	The number of successful face recognition or reconfirmation operations that have been run.
#R-Err	The number of face recognition or reconfirmation operations that have failed for some reason.
#R-Skip	The number of recognition operations that have been skipped due to recognizer overcommitment. This means that no recognizer was available for a face image because all recognizers were busy at that time.
#Evt	The number of events that have been reported.
%CPU	How CPU is used by the feed. Note that this number is in the range 0% to CPU_COUNT * 100%.
GPU#	The GPU ID. Every GPU in the system is assigned a unique ID. This entry is blank if the feed does not use a GPU.

Column Name	Description
GPU	A string which indicates which modules in the feed are using the GPU: V → video decoder F → face detector B → badge detector O → object detector An empty/non-existing string indicates that the feed is not using the GPU at all.
GPU-Name	The name of the GPU. Note that the name is not unique because a system may be equipped with more than one GPU of the same model and make. This entry is blank if the feed does not use a GPU.

Creating CSV Files

You can create a CSV file with all the information from the live service monitor screen by invoking the service monitor like this:

```
> virgo service monitor > my.csv
```

This command tells VIRGO that it should write the service monitor information into a CSV file instead of showing it on the screen. VIRGO will continue to write feed statistics once per second to the CSV file until you stop it by pressing Control-C in your terminal window.

VIRGO writes one line per feed to the CSV file and it repeats this process every second. It even includes inactive feeds by default. If you only want to include active feeds in the CSV file then pass the “--active-only” command line switch to VIRGO.

Troubleshooting

1. VIRGO crashes when I try to use it

You are most likely trying to run VIRGO on a system which does not have the Swift 5 runtime libraries installed. VIRGO depends on those libraries and Apple started shipping them with macOS beginning with version 14.4.4. If you are running an older OS and are not able to upgrade to a recent version of macOS then you should download the Swift 5 runtime libraries from Apple. See [this support article](#) for instructions on how to do this.

Docker

1. Feed reports “No Recogniser Available” after feed is added.

This type of error is normally produced when the Face Service is too busy to accept additional requests for recognition.

It can also be generated when the VIRGO configuration is incorrect and as such the requests are not getting sent to CoVi and time out.

Command & Control Protocol (COP)

The VIRGO-COP (Command & Control Protocol) enables you to control the VIRGO daemon system and to get information from it.

The full COP is only supported over the Daemon Transport Protocol (DTP). This COP variant is known as COP-DTP. DTP is a very efficient transport protocol which is used to send messages across Unix Domain Sockets. COP-DTP supports a number of request types which enable fine control over VIRGO and efficient realtime information gathering (e.g. getting feed statistics and log statements) without the need for repeated polling.

A subset of the COP is supported over the HTTP protocol. This COP variant of COP is known as COP-HTTP. COP-HTTP has fewer request types and slightly less efficient real-time information gathering than COP-DTP.

COP-DTP is used to enable the interaction between the VIRGO daemon and the VIRGO command line tool while COP-HTTP is used for the interaction between VIRGO and VIRGA.

COP Introduction

The VIRGO-COP network protocol enables a VIRGA server to update the global and per-feed state of a VIRGO instance. A VIRGO instance maintains a global state and a per-feed state which are stored persistently on the computer on which VIRGO is running. The currently active VIRGO state is associated with a modification date which indicates the generation of the state. New state may be applied to VIRGO at any time by including the new state with a new modification date in the reply to a VIRGO status message. However, VIRGO will only update its state if the new modification date is strictly greater than the current modification date. New state with a modification date earlier than or equal to the current modification date is ignored by VIRGO.

A good way to look at the state of a VIRGO daemon is that it moves along a timeline. The timeline always starts at 0 and with the factory settings. Each update to the VIRGO state advances the timeline to a new modification date. A state update may only move the timeline forward and not backward. Note however that the VIRGO-COP does offer a mechanism for a VIRGA server to reset the timeline back to 0 before a new state update is applied. This allows a VIRGA server to reset a VIRGO daemon, which may be in an unknown state, back to a well defined state.

Client Identity and Type

Each VIRGO instance is identified with an immutable identifier. This identifier is securely configured on the machine VIRGO runs on during deployment. It can only be changed in the factory by another deployment. It is passed along in every call the VIRGO instance makes to a VIRGO server in an X-CLIENT-ID header.

VIRGO also sends an X-CLIENT-TYPE header which is a combination of the VIRGO application name and the platform on which VIRGO is running. Currently the following client types are supported:

X-CLIENT-TYPE	Platform
Virgo-macOS	macOS
Virgo-Linux	Linux

VIRGO Configuration

The configuration of a VIRGO daemon can be broken down into two subsets:

- **global:** The state that applies to the daemon itself. (e.g. the current semantic version number and the update URL)
- **per-feed:** The state that applies to an individual feed. (e.g. the activity state, video source URL, etc.)

The current VIRGO configuration is associated with a modification date which is initially 0 and which is advanced to a new date every time the state is updated. All state information including the modification date is stored persistently by the VIRGO daemon and is guaranteed to survive daemon shutdowns and restarts.

Note: A description of semantic versioning is available here: <http://semver.org>

Data Types

The following table specifies the data types that may appear in a COP request or response:

COP Type	JSON Type	Description
Bool	Bool	true or false
Int	Int64	64-bit wide signed integer
Float	Float64	Double precision IEEE 754 floating point
String	UnicodeString	Unicode compliant string encoded in UTF-8

COP Type	JSON Type	Description
URL	UnicodeString	URL compliant with standards RFC 1808, RFC 1738, and RFC 2732
EpochTime	Int64	Milliseconds since January 1, 1970 at 00:00:00 GMT (GMT == UTC)
Milliseconds	Int64	Time interval expressed in milliseconds
Version	UnicodeString	Semantic version (e.g. "10.2.4")

A COP type may be optional which is indicated by a '?' (question mark) after the type name. An optional property may be left out in a request and may or may not appear in a response. Non-optional properties are always required in a request and are guaranteed to appear in a response.

COP Status Delivery

VIRGO sends a status message to VIRGA every couple seconds. The body of the status message lists all active feeds by feed name and it lists the current state of every feed. The purpose of this message is to keep the VIRGA server up-to-date about the current state of all active feeds. (e.g. whether the feed went offline because of an error)

Keep feed names short. A feed name is used as a simple and short handle to identify a feed in the context of the COP protocol. Feed name do not need to be globally unique; they only need to be unique with respect to a single VIRGO instance.

A feed name is NOT:

- A URL.
- A HTTP session ID.
- A cookie.
- Any kind of state.

Feed names should be at most be 16 characters in length.

VIRGO posts a status message via an HTTP POST request. The body of the message looks like this:

```

{
  "mod-date": "656756"           // [required][epoch time] the
modification date of the currently active virgo state
  "version": "1.0.0"             // [required][semantic version] the
semantic version number of the currently active virgo instance

  "capabilities": {              // [optional] Specifies the capabilities
of the client. If this is not specified then the default values are used.
    "config": "true"             // [optional][bool][default=true]
Indicates whether configuration of this client is allowed. If not present
the default value is used.
    "capture": "true",           // [optional][bool][default=true]
Indicates whether viewing of streams from this client is allowed. If not
present the default value is used.
  }

  // The most recent COP-HTTP error
  "error": {                     // [optional][dictionary] only present
if the previous COP configuration request had a syntax or semantic error
    "code": <int: error code>
    "message": <string: error message>
  }

  // The per-feed state.
  "feeds": {
    "camera_1": {
      "status": "ok",            // [required][string] this feed is up
and running
      "pid": 56757,              // [optional] the PID of the feed daemon
if the daemon is running; missing otherwise
      "start-date": "...",       // [optional][epoch time] the date &
time when the feed was most recently enabled (not updated if the feed is
restarted because of an error)
      "p-time": 68767,           // [optional][milliseconds] how much
time the feed has spent on processing the video stream in terms of
milliseconds
      "capturing" : true,        // [optional][boolean][default=false]
indicates whether the stream is currently depositing frames to its capture
deposit url.
      "statistics": {            // [optional][dictionary] only present
if feed statistics is enabled (see chapter "Feed Statistics" below
        ...
      }
    },

    "camera_2": {
      "status": "error",         // [required][string] This feed has
encountered an error and virgod is retrying
      "error": {                 // [optional][dictionary] only present
if the feed status is "error" or "failed"
        "code": <int: error code>,
        "message": <string: error message>,
        "retry-count": <int: number of retries>
      }
    }
  }
}

```

A VIRGO feed is always in one of the following states:

Feed status	Has statistics	Has start date	Has p time	Has error	Description
prerolling	no	yes	yes	yes	The feed is in the process of starting up, connecting to the video source, and priming the video decoder. Note that the feed dictionary may contain an error dictionary if an error was encountered while prerolling. The feed automatically retries in this case.
ok	yes	yes	yes	no	The feed is up and running without problems. VIRGO is able to receive a video stream from the feed URL, it is able to decode it, and it is able to run face detection and face recognition on it.

Feed status	Has statistics	Has start date	Has p time	Has error	Description
error	no	yes	yes	yes	<p>The feed has encountered an error while processing the video stream. VIRGO expects that it will be able to recover from this error without user intervention. (e.g. a temporary resource shortage or an unexpected crash of the feed daemon)</p> <p>The feed dictionary contains an additional error dictionary with the error code, error message, and retry count.</p>
failed	no	yes	yes	yes	<p>The feed has encountered a fatal error and VIRGO is unable to recover from it without the help of the user. User intervention is required to fix the problem. E.g. the feed is a video file and the file was not found. The feed dictionary contains an additional error dictionary with the error code and error message.</p>

Feed status	Has statistics	Has start date	Has p time	Has error	Description
eos	yes	yes	yes	no	The feed has encountered an end-of-stream condition. For example, the feed URL might point to a video file and the whole video file has been processed. The feed will remain in the end-of-stream state until it is deactivated or until it is updated with a new feed URL and the state "active": true. Note that the statistical information in this state represents the last know statistics. (e.g. the final statistics at the end of a video file)
inactive	no	no	no	no	The feed is currently disabled. A disabled feed exists but does not process the incoming video stream.

The following HTTP custom headers are included with every status message request:

Header	Description
X-CLIENT-ID	The client id of the <i>virgod</i> instance. This is immutable, factory configured, unique and descriptive <i>vrgo</i> instance identifier. For example: VRGO-LNX-TRPR-16-123

Header	Description
X-CLIENT-TYPE	The type of the client. This is a combination of the client name and the platform name. For example: <code>Virgo-Linux</code>

Feed Error

A feed dictionary contains an error dictionary if the feed has encountered an error. Most errors are recoverable and *virgod* automatically retries the feed. A few errors are fatal and require action by the user to make the feed work again. The following table lists the available information in an error dictionary:

Property	Type	Description
code	Int	The error code.
message	String	The error message.
retry-count	Int?	Exists only if <i>virgod</i> has retried the operation. The number of retries.

Feed Statistics

The status request may include per-feed statistics in the form of a statistics dictionary. Delivery of feed statistics is enabled by setting the “statistics.enabled” key in the feed state dictionary to “true”. The following table lists the available statistics and what information it represents:

Property	Type	Description
w	Int?	The width of a video frame in pixels. This value is sent after the resolution of the input stream has become available.
h	Int?	The height of a video frame in pixels. This value is sent after the resolution of the input stream has become available.
fps	Float?	The nominal frame rate of the source video. This is the frame rate at which the video should be played back, not the rate at which frames are being processed. This value is sent after the frame rate of the input stream has become available.

Property	Type	Description
cpu.usage	Float?	The amount of CPU time (user + system) that the feed is using. This is a percentage value in the range 0...1. This value is only sent while the feed is actively processing the input stream.
gpu.usage	Float?	The amount of GPU processing power used by the feed. This is a percentage value in the range 0...1. The entry only exists if the feed is using a GPU.
gpu.uses	String?	A string which indicates which modules in the feed are using the GPU: V → video decoder F → face detector B → badge detector O → object detector empty/non-existing string indicates that the feed is not using the GPU at all.
gpu.id	Int?	The unique ID of the GPU used by the feed. This entry only exists if the feed is using a GPU.
gpu.name	String?	The name of the GPU which the feed is using. This entry only exists if the feed is using a GPU.
detector.dps	Float?	The rate at which detection operations are executed on incoming frames. This effectively represents the rate at which frames are processed. This value is only sent after object detection has started for the input stream.
detector.latency	Milliseconds?	The time in milliseconds it takes to run a single detection operation. This value is only sent while object detection is active for an input stream.
recognizer.latency	Milliseconds?	The time in milliseconds it takes to run a single recognition operation. This value is only sent while object recognition is active for an input stream.
detector.trigger-count	Int	The number of detection operations that have been triggered so far. This value is initially 0.

Property	Type	Description
detector.badge-count	Int	The number of badges that have been detected. This value is initially 0.
detector.face-count	Int	The number of faces that have been detected. This value is initially 0.
detector.error-count	Int	The number of detection operations that have failed for some reason. This value is initially 0.
detector.skipped-count	Int	The number of detection operations that were skipped because at the time of detection there were no available detectors. This can indicate too much load on the machine. This value is initially 0.
recognizer.trigger-count	Int	The number of face recognition and reconfirmation operations that have been triggered so far. This value is initially 0.
recognizer.face-count	Int	The number of faces that have been successfully recognized or reconfirmed. This value is initially 0.
recognizer.error-count	Int	The number of faces recognition operations that have failed for some reason. This value is initially 0.
recognizer.skipped-count	Int	The number of recognition operations that were skipped because at the time of recognition there were no available recognizers. This can indicate too much load on the machine. This value is initially 0.
reporter.event-count	Int?	The number of events that have been reported. Only appears if reporting is turned on. This value is initially 0.

COP Status Reply

The reply to a VIRGO status request may contain a new JSON configuration that should be applied to VIRGO or it may be empty. The configuration may be changed in one of two different ways:

- full update: In this case the new configuration completely replaces the existing

configuration and all properties have to be provided.

- delta update: In this case the new configuration contains the difference to the existing configuration. Only properties which should be changed should be provided.

200 - State Change

VIRGA responds with an HTTP status code 200 if it has determined that the configuration stored in *virgod* is not up-to-date with respect to the configuration stored in VIRGA. The body of the reply should contain the new configuration and the new modification date.

204 - No Change

VIRGA responds with an HTTP status code 204 if it has determined that the configuration stored by *virgod* is up-to-date and requires no change. VIRGA does this determination by comparing the “mod-date” sent by *virgod* with the “mod-date” stored in its own persistent store.

Delta Updates

The following code block shows the outline of a delta update:

```

{
  // [required][epoch time] The new modification date associated with the
  new state
  "mod-date": "767878"

  // [optional] Tells Virgo how to apply the new state to its current
  state:
  // "current": means that the new state should be applied on top of the
  current virgo state. This is the default behavior.
  // "initial": means that Virgo should FIRST reset its state back to the
  factory settings before it applies the new state. This allows you to reset
  virgo
  "relative-to": "current"

  // [required] Tells virgo that this is a delta update that contains
  changes which should be applied relative to the current configuration.
  "apply-as": "delta"

  // [optional] The new global state.
  // The current global state is retained if no new global state is
  provided.
  "global": {
    "status-interval": 200    // [milliseconds] status reporting interval
    in ms (default: 500)
    ...
  }

  // [optional] Specifies which feeds should be removed. Note that removals
  are always
  // carried out before additions.
  "feed.removals" = [
    "video_1", "video_2", ...
  ]

  // [optional] Specifies which feeds should be added.
  "feed.additions" = {
    "camera_1": { ... }
    "camera_2": { ... }
    ...
  }

  // [optional] Specifies which feeds should be updated.
  "feed.updates" = {
    "camera_1": { ... }
    "camera_2": { ... }
    ...
  }
}

```

Full Updates

Note that you should always prefer delta updates over full updates because full updates are inherently inefficient and suffer from race conditions.

The following code block shows the outline of a full configuration update:

```
{
  // [required][epoch time] The new modification date associated with the
  new state
  "mod-date": "767878"

  // [optional] Tells Virgo how to apply the new state to its current
  state:
  // "current": means that the new state should be applied on top of the
  current virgo state. This is the default behavior.
  // "initial": means that Virgo should FIRST reset its state back to the
  factory settings before it applies the new state. This allows you to reset
  virgo
  "relative-to": "current"

  // [optional] Tells virgo that the update is a full update that should
  replace the current configuration.
  "apply-as": "full"

  // [optional] The new global state.
  // The current global state is retained if no new global state is
  provided.
  "global": {
    "status-interval": 200    // [milliseconds] status reporting interval
    in ms (default: 500)
    ...
  }

  // [optional] The new per-feed state. This is a dictionary. The
  dictionary key is the name of a feed
  // and the value is another dictionary which contains the feed's new
  state.
  // The current feed state is retained if no new per-feed state is
  provided.
  "feeds": {
    "camera_1": { ... }
    "camera_2": { ... }
    ...
  }
}
```

Configuration Sections

The configuration is organized into sections. Sections are optional. A section which is not mentioned in the reply is not applied and *virgo*d retains the currently active state for this section. This is true for both “full” and “delta” “apply-as” modes.

A status reply message may contain the following sections:

Section	apply-as	Description
global	delta, full	Contains state that applies to the <i>virgod</i> daemon itself.
feeds	full	Contains per-feed state information.
feed.additions	delta	Contains dictionaries of feeds that should be added. See the feeds section below for a description of a feed dictionary.
feed.removals	delta	Contains the names of feeds that should be removed. Note that this is an array of feed names.
feed.updates	delta	Contains dictionaries of feeds that should be updated with new state. See the feeds section below for a description of a feed dictionary.
log	delta, full	Contains information to configure the logging behavior.
update	delta, full	Contains information about the version to which VIRGO should be upgraded or downgraded.

The “feed.xxx” sections are applied in the order “feed.removals” followed by “feed.additions” and finally “feed.updates”.

Global Section

The following properties are supported in the global section which contains configuration information that applies to VIRGO itself:

Property	Type	Default	Description
status-interval	Milliseconds	500	Status reporting time interval in ms.

Feeds Section

The following properties are supported in the feeds section which contains feed-specific configuration information:

Property	Type	Default	Description
----------	------	---------	-------------

Property	Type	Default	Description
directory	String?	client ID	Directory name
source	String?	client ID	Source name
site	String?	client ID	Site name
enabled	Bool	false	Marks the feed as enabled or disabled.
input.type	String		The type of feed input. Must be "stream".
input.loop	Bool	false	Enables looping of the feed input. Only video file-based feeds support looping. Ignored for cameras
input.video-clock.enabled	Bool	false	Enables enforcement of the video clock. Video files will be processed as fast as possible if the video clock is turned off.
input.lens-correction.enabled	Bool	false	Enables or disables lens correction for the camera.
input.lens-correction.k1	Float	0	The "k1" lens correction factor.
input.lens-correction.k2	Float	0	The "k2" lens correction factor.
input.mirroring.enabled	Bool	false	Whether the video image should be mirrored before detection and recognition operations are executed.
input.rotation.angle	Int	0	Whether the video should be rotated before detection and recognition operations are executed. Valid values are 0, 90, 180, and 270.

Property	Type	Default	Description
input.crop-rectangle.enabled	Bool	false	When this is true the defined crop rectangle is used for the camera feed. The crop rectangle is specified in a normalized coordinate system, which means the rectangle is (0, 0) x (1, 1).
input.crop-rectangle.left	Double	0	The normalized left coordinate relative to the video of where the crop rectangle origin should be.
input.crop-rectangle.top	Double	0	The normalized top coordinate relative to the video of where the crop rectangle origin should be.
input.crop-rectangle.width	Double	1	The normalized width value relative to the video of how big the crop rectangle size should be.
input.crop-rectangle.height	Double	1	The normalized height value relative to the video of how big the crop rectangle size should be.
input.contrast-enhancement.enabled	Bool	false	Enables contrast enhancement of the input video frame.
input.contrast-enhancement.low-light-threshold	Double	0.02	Low-light-threshold for contrast enhancement.
input.contrast-enhancement.exposure-boost	Double	0	Exposure boost for contrast enhancement.

Property	Type	Default	Description
input.contrast-enhancement.detection-only	Bool	false	If true then contrast enhancement is applied to the image which is handed off to the face detector only. If false then contrast enhancement is applied to the video frame as delivered by the camera. Consequently the contrast enhancement effect is visible in the video preview if this option is off but not if it is on.
accelerator	String	"auto"	The type of acceleration that a feed should use. See the table "Feed accelerator types" below for a list of the supported acceleration types.
accelerator.gpu-id	Int		The GPU identifier to use when GPU acceleration is in use. This is only used if the "accelerator" property is set to gpu or auto (and gpu is used). If this is specified this will force the specific GPU to be used and if failure occurs it will fallback to CPU. This is an advanced setting that should only be used in very specific cases.
statistics.enabled	Bool	false	Whether VIRGO should record and report statistics for this feed.
detector.detect-badges	Bool	false	Whether detection of badges should be enabled for this feed.
detector.maximum-input-resolution-badges	Int	4320	Maximum resolution of the Input image. Bigger images are scaled down (aspect-ratio preserving) to this resolution before detection.

Property	Type	Default	Description
detector.minimum-searched-badge-size	Int	20	The badge detector is advised to search for badges of at least this size. This value is applied while searching the image.
detector.minimum-required-badge-size	Int	0	The minimum size of badges to accept from the detector. Only badges with at least this size are eligible for recognition.
detector.detect-faces	Bool	true	Whether detection of faces should be enabled for this feed.
detector.minimum-searched-face-size	Int	80	The face detector is advised to search for faces of at least this size. This value is applied while searching the image.
detector.minimum-required-face-size	Int	0	The minimum size of faces to accept from the detector. Only faces with at least this size are eligible for recognition.
detector.maximum-input-resolution	Int	720	Maximum resolution of the Input image. Bigger images are scaled down (aspect-ratio preserving) to this resolution before detection.
detector.maximum-concurrent-detections	Int	0	The maximum number of concurrent detections to allow. 0 means to automatically set this.
detector.detect-people	Bool	false	Whether detection of people should be enabled for this feed. This detects any part of a person's body and not just the face.

Property	Type	Default	Description
detector.minimum-required-person-to-screen-height-proportion	Double	0	Specifies the ratio of the person to the screen height. This can be between 0 - 1 and allows for decimal precision. For example, if you don't want the person to show up unless they are greater than 25% of the image height then specify a value of 0.25.
detector.minimum-consecutive-detections-required-person	Int	0	This is the number of consecutive detections that are required before reporting that the person (based on object id) was actually detected and can be used to filter out false positives.
detector.detect-people-every-n-frames	Int	1	This can be used to avoid running person detection on every frame. Since person detection requires a lot of GPU processing if the hardware is not powerful enough this value can be changed so that we only attempt to detect people every Nth frame to save processing power to keep up with realtime detection.
detector.person-detection-threshold	Double	0.4	This is the detection threshold to use when matching objects. The higher the threshold the more strict the matching will be and the higher the confidence will be that the actual object matches.

Property	Type	Default	Description
detector.person-separation-threshold	Double	0.45	This threshold controls the object separation when the objects are overlapping. This determine how much overlap is needed before no longer detecting the object with the weaker footprint.
detector.detect-people-model	String	"balanced"	Valid values: "max-accuracy" - Use a larger model for better accuracy, but the speed will be slower. "max-speed" - Use a smaller model for faster speed, but the accuracy will be lower. "balanced" - Use a larger model for better accuracy, but the precision will be slightly lower resulting in faster speeds than the "max-accuracy" model without sacrificing too much accuracy.
detector.initial-face-selection-threshold	Double	0.8	The initial face candidate threshold that is used during face detection.
detector.middle-face-selection-threshold	Double	0.85	The middle face candidate threshold that is used during face detection.
detector.final-face-selection-threshold	Double	0.9	The final face candidate threshold that is used during face detection.
recognizer.minimum-face-size	Int	120	The minimum size of faces to detect. This value is applied after searching the image.
recognizer.minimum-face-size-merging	Int	220	The minimum resolution a recognition candidate must have in order to allow merging.

Property	Type	Default	Description
recognizer.minimum-face-size-identification	Int	220	The minimum resolution that a recognition candidate image must have in order to allow the insertion of the candidate image into the Cloud database.
recognizer.minimum-center-pose-quality	Float	0.05	The minimum center pose quality that a face image must have before we try to recognize the face.
recognizer.pose-configuration-identification-enabled	Bool	false	If this is true then pose configuration is enabled for identification. The pose configuration allows for replacing center pose quality with advanced parameters such as yaw, pitch and roll. If this is true then recognizer.minimum-center-pose-quality is ignored and the pose configuration parameters are used instead. Currently these are recognizer.maximum-yaw-identification, recognizer.maximum-pitch-identification, and recognizer.maximum-roll-identification.
recognizer.maximum-yaw-identification	Double	0.4	This is the maximum yaw value used to determine if the face is looking straight. The yaw value is the side to side movement of the face.
recognizer.maximum-pitch-identification	Double	0.4	This is the maximum pitch value used to determine if the face is looking straight. The pitch value is the forward/backward movement of the face.

Property	Type	Default	Description
recognizer.maximum-roll-identification	Double	0.15	This is the maximum roll value used to determine if the face is looking straight. The roll value is the side to side tilt movement of the face.
recognizer.minimum-center-pose-quality-merging	Float	0.59	The minimum CPQ that a recognition candidate must have in order to allow merging.
recognizer.minimum-center-pose-quality-identification	Float	0.59	The minimum CPQ that a recognition candidate must have in order to allow the insertion of the candidate image into the Cloud database.
recognizer.minimum-face-contrast-quality	Float	0.2	The minimum face contrast quality that a face image must have before we try to recognize the face.
recognizer.minimum-face-contrast-quality-merging	Float	0.59	The minimum FCQ that a recognition candidate must have in order to allow merging.
recognizer.minimum-face-contrast-quality-identification	Float	0.59	The minimum FCQ that a recognition candidate must have in order to allow the insertion of the candidate image into the Cloud database.
recognizer.identity-recognition-threshold	Float	0.54	The identity recognition threshold.
recognizer.minimum-face-sharpness-quality	Float	0.3	The minimum face sharpness quality that a face image must have before we try to recognize the face.
recognizer.minimum-face-sharpness-quality-merging	Float	0.59	The minimum FSQ that a recognition candidate must have in order to allow merging.

Property	Type	Default	Description
recognizer.minimum-face-sharpness-quality-identification	Float	0.59	The minimum FSQ that a recognition candidate must have in order to allow the insertion of the candidate image into the Cloud database.
recognizer.maximum-clip-ratio	Float	0.2	The maximum clip ratio on either side the recognition candidate might have.
recognizer.maximum-clip-ratio-identification	Float	0	The maximum clip ratio on either side the insertion candidate might have.
recognizer.detect-gender	Bool	false	Whether to enable the detection of gender information.
recognizer.detect-age	Bool	false	Whether to enable the detection of age information.
recognizer.detect-sentiment	Bool	false	Whether to enable the detection of sentiment information.
recognizer.learning-enabled	Bool	false	Whether the recognizer is allowed to learn new identities.
recognizer.maximum-concurrent-recognitions	Int	0	The maximum number of concurrent recognitions to allow. 0 means to automatically set this.
recognizer.detect-occlusion	Bool	false	Whether to enable occlusion detection during recognition.

Property	Type	Default	Description
recognizer.maximum-occlusion	Double	0.5	Valid values are in the range of 0.0 - 1.0. This is the maximum occlusion value that is allowed when inserting new recognition candidate images into the Cloud database. If the face is occluded with a value greater than this then the face will not be added, but if it is less than or equal to this value then it will be added.
recognizer.learn-occluded-faces	Bool	false	Whether to enable learning of occluded faces regardless of the maximum occlusion setting. If this is true then the server configuration will be used, which by default doesn't do any occlusion detection.
recognizer.identity-proximity-threshold-allowance	Double	0.13	The identity recognition threshold proximity allowance. The lower the value to more strict recognition is.
tracker.maximum-linger-frames	Int	30	Determines for how many frames more we continue to keep a tracked face around after we have failed to detect it in the most recent frame. This makes the tracker resilient against intermittent loss of face.
tracker.minimum-number-identical-recognitions-lock	Int	1	The minimum number of consecutive recognition attempts that we must run and produce the same person identity before we lock onto this identity.

Property	Type	Default	Description
tracker.minimum-required-consecutive-badge-detections	Int	0	This is the number of consecutive detections that are required before reporting that the object (based on object id) was actually detected and can be used to filter out false positives.
tracker.reconfirmation-interval	Int	1000	Identity reconfirmation time interval in ms.
tracker.initial-recognition-attempts	Int	3	The number of initial recognition attempts to make on an unrecognized person as fast as possible.
tracker.failed-recognition-back-off-interval	Milliseconds	340	After making the initial recognition attempts as fast as possible back up this amount for each subsequent recognition to slow down. This goes on until the retry interval is reached.
tracker.failed-recognition-retry-interval	Milliseconds	1	The interval in which to run recognition requests if the face has not been recognized.
tracker.identity-relearn-interval-days	Float	0	Updates the identity only in the case where the identity currently saved is older than the updated identity.
tracker.identity-update-better-image	Bool	false	Updates the identity in the case where the identity currently saved is of lower quality (in all aspects) than the updated identity.
tracker.max-position-change-relative-to-face	Int	115	The maximum position change, specified in percentage relative to the object size, to continue tracking.
tracker.max-size-change-relative-to-face	Int	50	The maximum size change, specified in percentage relative to the object size, to continue tracking.

Property	Type	Default	Description
tracker.minimum-number-identical-recognitions-learn	Int	2	This is the number of consecutive recognitions that need to occur before adding a new identity to the system.
tracker.enable-face-size-correlation	Bool	true	Enable face correlation of tracked faces, which compares detected faces looking for a change in area.
tracker.enable-face-bounds-prediction	Bool	true	Enable face bounds prediction, which predicts which direction the face is moving to maintain tracking.
tracker.stop-tracking-on-failed-re-recognition	Bool	false	If recognition fails when re-recognizing a person then delete the identity that was created.
tracker.reconfirm-identity-in-video-on-every-key-frame	Bool	false	When a key frame is encountered in a video file all the faces that are being tracked are marked as unconfirmed so that their identities are reconfirmed to make sure they are the same person. This setting only applies to video files and not live video. If a video file does not represent recorded live video then this can typically be set to true for better tracking during scene changes.

Property	Type	Default	Description
tracker.min-failed-recognitions-to-stop-tracking-identity	Int	3	When a face is being tracked recognitions are continually confirming the identity. The identity is also being verified if it is transferred from a person object. In these cases, if the recognition or verification fails this number of consecutive times then the identity will be reset and no longer associated with the face because we are no longer sure it is the same identity.
tracker.detect-unauthorized-movement.person.left	Bool	false	Enabled unauthorized movement detection in the left direction.
tracker.detect-unauthorized-movement.person.left-distance	Double	0.1	The distance the tracked object is allowed to move to the left. The distance is provided in relative terms as a fraction of screen width in range 0 - 1.
tracker.detect-unauthorized-movement.person.right	Bool	false	Enabled unauthorized movement detection in the right direction.
tracker.detect-unauthorized-movement.person.right-distance	Double	0.1	The distance the tracked object is allowed to move to the right. The distance is provided in relative terms as a fraction of screen width in range 0 - 1.
tracker.detect-unauthorized-movement.person.up	Bool	false	Enabled unauthorized movement detection in the upward direction.
tracker.detect-unauthorized-movement.person.up-distance	Double	0.1	The distance the tracked object is allowed to move to the up. The distance is provided in relative terms as a fraction of screen height in range 0 - 1.

Property	Type	Default	Description
tracker.detect-unauthorized-movement.person.down	Bool	false	Enabled unauthorized movement detection in the downward direction.
tracker.detect-unauthorized-movement.person.down-distance	Double	0.1	The distance the tracked object is allowed to move to the down. The distance is provided in relative terms as a fraction of screen height in range 0 - 1.
reporter.enabled	Bool	true	Enables or disables event reporting.
reporter.report-event-face	Bool	true	Enables the inclusion of face thumbnails in event reports.
reporter.report-event-scene	Bool	false	Enables the inclusion of scene images in event reports.
reporter.minimum-event-duration-identified	Milliseconds	0	The minimum allowed recognized person event duration in seconds. Events below this value will not be reported.
reporter.minimum-event-duration-unidentified	Milliseconds	0	The minimum allowed unrecognized person event duration in seconds. Events below this value will not be reported.
reporter.delay	Milliseconds	0	Delay the event reporting to the server by this amount in seconds.
reporter.events-initial-date-offset	EpochTime	nil	When processing a video file for events this value can be used to set the initial date offset to use for the events being processed. By default video events use the timestamps.
reporter.report-unrecognizable-events	Bool	true	Reports events for people that are not recognizable.

Property	Type	Default	Description
reporter.report-stranger-events	Bool	true	Reports events for people that are strangers. These are people not registered by the system after running facial recognition on the face.
reporter.report-speculated-events	Bool	true	Reports events for speculated people. This means faces that aren't a 100% match, but are close.
reporter.update-images	Bool	true	Update the thumbnail images with higher quality images during the course of the event if possible.
reporter.update-in-progress-event-properties	Bool	false	If this is enabled then any event properties that change will be updated at the specified interval. Many properties do change periodically, such as images or other averages that are continually computed.
reporter.update-in-progress-event-interval	Milliseconds	1000	This specifies the interval time in which to update event properties that change.
reporter.stranger-events.only-if-occluded	Bool	false	This specifies whether only occluded stranger events should be reported. By default stranger events are only generated if the face is not occluded, if occlusion detection is enabled, otherwise they are generated when the face meets the identification image quality metrics. If this option is set to true then stranger events will be reported only if the face is occluded.

Property	Type	Default	Description
reporter.report-secondary-events	Bool	false	Reports secondary events. Secondary events are events that are associated with a primary event via the rootEventId property in the event. It is usually preferred to only report the primary events and the secondary events need to only be reported if there is more detail needed. If this is disabled then all events with a rootEventId property set to a primary event will not be reported. Only events with rootEventId not set to anything will be reported, which are the primary events.
capture.lease-date	EpochTime	0	The date of the capture lease
capture.size	Int	480	Specifies size of the smaller dimension of the image that will be sent
capture.maximum-frames	Int	1200	If > 0, enables the capture of "max-frames" frames; if 0, disables capture
capture.frame-delay	Milliseconds	200	Wall-clock time between consecutive frame captures. If this value is set to 0 then VIRGO will capture frames as fast as the native frame rate is playing the video.
capture.deposit-base-url	URL?	none	The base URL to which captured frames should be posted.
recognizer.detect-smile-action	Bool	false	Enables the smile action recognizer.
recognizer.smile-pre-delay	Milliseconds	100	The amount of time that there should be no smile.
recognizer.smile-duration	Milliseconds	0	The amount of time that the smile should last.

Property	Type	Default	Description
recognizer.smile-identity-threshold-boost	Double	0.13	The smile threshold to boost temporarily during the smile action.
recognizer.smile-thresholds-enabled	Bool	false	Enables the smile threshold values.
recognizer.smile-threshold-neutral	Double	-0.1	The threshold in which there is no smile.
recognizer.smile-threshold-smiling	Double	0.7	The threshold in which there is a smile.
recognizer.detect-pose-action	Bool	false	Enables the pose liveness action recognizer.
recognizer.pose-action-min-center-pose-quality	Double	0.5	The minimum center pose quality to use when detecting the initial center pose.
recognizer.pose-action-max-profile-pose-quality	Double	0.26	The maximum center pose quality to use when detecting the final profile pose.
recognizer.pose-action-min-profile-confidence-start	Double	0.35	The minimum profile pose confidence to allow during the initial center pose detection phase.
recognizer.pose-action-max-profile-confidence-end	Double	0.60	The maximum profile pose confidence to allow during the final profile pose detection phase.
recognizer.pose-action-min-transition-poses	Int	2	The minimum number of required center pose samples during the transition from center to profile pose.
recognizer.pose-action-required-confirmations	Int	3	The number of consecutive confirmations required to enter the initial center pose detection phase.
recognizer.pose-action-min-profile-similarity	Double	0.86	The minimum similarity score required when verifying the final profile pose.

Property	Type	Default	Description
recognizer.pose-action-min-detections-per-second	Int	15	The minimum number of frames per second that is required during the process.
recognizer.pose-action-max-cpq-jump-after-discontinuity	Double	0.15	The maximum change between samples while the pose is changing from center to profile if lingering.
recognizer.pose-action-max-cpq-jump-in-continuity	Double	0.18	The maximum change between samples while the pose is changing from center to profile.
recognizer.pose-action-max-profile-pose-roll	Double	0.3	The maximum roll threshold in either direction in which the face can rotate when determining whether the face is in profile pose.
recognizer.pose-action-min-profile-pose-yaw	Double	0.81	The minimum profile pose yaw value that is required during the final profile pose detection phase.
recognizer.pose-action-profile-pose-required-confirmations	Int	1	The number of consecutive confirmations required to enter the final profile pose detection phase.

Feed Properties for “Stream” Inputs

Property	Type	Default	Description
input.stream.url	URL		The video stream URL. The URL must point to a RTSP, HTTP, or FILE stream.
input.stream.name	String		A friendly name used for display purposes.
input.stream.id	String		Identifier used to connect to a stream if the URL is blank.

Property	Type	Default	Description
input.stream.rtsp.transport	String	udp	The transport protocol that should be used while accessing the RTSP video stream. Must be one of “udp”, “tcp”, or “udp-multicast”.

Feed Acceleration Types

Property	Description
auto	VIRGO will automatically pick the best available acceleration type. For example VIRGO will assign the feed to one of the available GPUs if there is still processing capacity available. Otherwise VIRGO will assign the feed to the CPU.
cpu	The feed should exclusively run on the CPU and not use any GPU even if a GPU would be available.
gpu	The feed should exclusively run on a GPU and not use the CPU for video decoding, graphics processing, or detection. The feed will fail if no GPU is available.

Feed.Additions, Feed.Removals, and Feed.Updates Sections

The feed.removals section lists the names of the feeds that should be removed. This section is always applied first. The feed.additions sections lists the descriptions of the feeds that should be added. This section is always applied after removals. Finally the feed.updates section lists the description of feeds that should be updated with new state and this section is always applied last.

Log Section

The following properties are supported in the log section which contains logging related configuration:

Property	Type	Default	Description
lease-date	EpochTime	0	The date of the log lease.

Property	Type	Default	Description
deposit-url	URL?	none	The URL to which the most recently recorded log statements should be posted.
deposit-interval	Milliseconds	5000	The minimum time interval between consecutive log deposit operations.
levels	Dictionary<String, String>	empty	A dictionary which maps a log package name to a log level.

See Logging for a description of the logging mechanism.

Update Section

The following properties are supported in the update section which contains information related to upgraded or downgrading the currently installed VIRGO version:

Property	Type	Default	Description
version	Version	none	The version to which the current VIRGO installation should be upgraded or downgraded to.
download-url	URL	none	The URL from which VIRGO should fetch the update archive.
progress-url	URL?	none	The URL to which update events should be sent. Update events are sent periodically at a time interval equal to "progress-interval".
progress-interval	Milliseconds?	1000	The time interval at which update events should be sent to the "progress-url" URL
log.enabled	Bool?	false	Set to true to enable the inclusion of logging information in the update events.

See Software Updates for a description of the updates mechanism.

“relative-to” and resetting the current state

The current state stored in VIRGO may be reset back to the factory defaults by including the “relative-to” JSON key with a value of “initial”. This causes VIRGO to delete its persistently stored state and to reload its state from the factory defaults. This action also resets the modification date back to 0. Virgo then applies the new state as listed in the status message reply. This new state together with the new modification date is then persistently stored.

“apply-as” and delta vs full updates

VIRGO is able to interpret the state included in a status message reply as either the description of a complete (full) configuration or as a delta relative to the current VIRGO configuration. The configuration included in a status message reply is by default interpreted as a full configuration update which completely replaces the current state. You may change this behavior by adding a “apply-as” to the reply:

- “full”: this is the default behavior. VIRGO expects that the new configuration is complete and it will replace the current configuration.
- “delta”: VIRGO interprets the new configuration as a change relative to the current configuration.

An “apply-as”: “delta” mode means that you may leave out key-value pairs which are not supposed to change. VIRGO automatically reuses the current value for any key that is missing in the new global or per-feed state. Here is an example:

This is the current state of the "camera_1" feed as stored by Virgo. Note that we show only some of the state here, however you should assume that all state is fully defined:

```
{
  "source" : "foo"
  "site": "bar"
  ...
  "lens-correction.enabled": false
  ...
  "detector.minimum-searched-face-size": 80
  ...
  "recognizer.detect-age": false
  ...
  "tracker.maximum-linger-frames": 20
  ...
  "reporter.enabled": true
  ...
  "capture.max-frames": 0
  ...
}
```

Now all we want to do is to enable lens correction. We don't want to change any other feed state. To achieve this, we simply send a new feed state with just those keys and values that we want to change. Virgo will retain all other values as they are:

Status message reply:

```
{
  "mod-date": 878789
  "apply-as": "delta"
  "feeds": {
    "camera_1": {
      "lens-correction.enabled": true
      "lens-correction.k1": 0.6
      "lens-correction.k2": 0.7
    }
  }
}
```

After the update:

All state remains as it was before the update except that the "lens-correction.enabled", "lens-correction.k1" and "lens-correction.k2" key-value pairs have been applied to the previous state and the modification date has been advanced to the new date.

Note that "delta" updates are generically preferred over "full" updates because:

- delta update messages can be significantly more compact and smaller than full update messages. This means less bandwidth consumption and faster updates.
- delta updates can be processed much more efficiently than full updates.
- full updates inherently suffer from race conditions because VIRGO continues to operate asynchronously after it has sent a status message to VIRGA. But the status of a feed may change in-between the time the status message was sent off by VIRGO and the time VIRGO receives a new configuration from VIRGA. E.g. a feed may transition from OK to

EOS state in the meantime. Since VIRGA is not aware of that, pushing a full state update out would involuntarily restart the feed from scratch.

COP Image Capture

Image capture is a mechanism which allows you to capture individual frames from a VIRGO feed. Image capture is by default turned off. It may be turned on for a feed at any time but only one capture session can be active at any given time. A “capture lease” is required to turn image capture on or to restart an already active capture session. A “capture lease” is represented by its “capture lease date” which you must send to VIRGO. VIRGO does not interpret the lease date value - it only cares about a change in the lease date value.

You turn image capturing on by sending a “capture.max-frames” with a value > 0, a valid “capture.deposit-base-url” and a unique “capture.lease-date”. The “maximum-frames” value defines how many frames should be captured before image capturing is automatically turned off again. This mechanism ensures that image capturing can not be accidentally left turned on by, for example, forgetting to send a new image capture state. You turn image capture off by sending “maximum-frames” with a value of 0 or by sending “capture.deposit-base-url” with an empty string value.

VIRGO sends a HTTP POST request with the JPEG compressed image in its body to the final per-image deposits URL. The final deposit URL is formed by adding the image file name to the “capture.deposit-base-url”. The image file name is computed as follows:

```
<feed name>_<yyyy>-<mm>-<dd>_<pts>.jpg
```

Where “yyyy-mm-dd” is referring to the current year, month and day and “pts” refers to the presentation time stamp of the captured frame. The presentation time stamp is in terms of microseconds from the start of the stream.

The following HTTP custom headers are included with every capture POST request:

Header	Description
X-CLIENT-ID	The client id of the <i>virgod</i> instance. This is immutable, factory configured, unique and descriptive VIRGO instance identifier. For example: VRGO-LNX-TRPR-16-123
X-CLIENT-TYPE	The type of the client. This is a combination of the client name and the platform name. For example: Virgo-Linux
X-FEED-ID	The ID of the feed from which the image was captured. For example: VRGO-LNX-TRPR-16-123-camera_1

Tracking Result Capture

VIRGO - Posting the Tracking Result Metadata

Tracking result capture is automatically enabled when Image Capture is enabled. As each image is being captured the tracking result metadata is also being captured. This tracking result metadata is sent to a different deposit url, but it is derived from the image capture deposit URL. More details about this process are documented in the Image Capture document so this document just focuses on the changes needed to post and retrieve the tracking result metadata.

VIRGO sends a HTTP POST request with the JSON metadata in its body to the final per-tracking result deposit URL. The final deposit URL is formed by appending "_tracking_result" to the "capture.deposit-base-url" and then adding the JSON filename. Here is an example showing a sample capture deposit base url and a final URL modified by VIRGO. The filename is optional on the server side, but it is useful for logging, which is why VIRGO appends it.

capture.deposit-base-url = https://cvos.dev.real.com/sharedStream/video_d98fe652-9a76-4578-863a-104c1b86dec3

final-deposit-tracking-result-url = https://cvos.dev.real.com/sharedStream/video_d98fe652-9a76-4578-863a-104c1b86dec3_tracking_result/Samsung_2020-02-05_56677000.json

The JSON file name is computed as follows:

```
<feed name>_<yyyy>-<mm>-<dd>_<pts>.json
```

Where "yyyy-mm-dd" is referring to the current year, month, and day, and "pts" refers to the presentation time stamp of the captured frame. The presentation time stamp is in terms of microseconds from the start of the stream.

The following HTTP custom headers are included with every capture POST request:

Header	Description
X-RPC-DIRECTORY	The directory that is in use for the current account.
X-RPC-AUTHORIZATION	The authorization header for the request that contains the current user information.
X-FEED-ID	The ID of the feed from which the image was captured. E.g.: VRGO-LNX-TRPR-16-123-camera_1

Header	Description
X-CLIENT-TYPE	The type of the client. This is a combination of the client name and the platform name. E.g.: Virgo-Linux
X-CLIENT-ID	The client id of the virgod instance. This is immutable, factory configured, unique and descriptive vrgo instance identifier. E.g.: VRGO-LNX-TRPR-16-123

VIRGO will always posts the video frame images and tracking results in the proper order. This means that the timestamps are always moving forward. The client doesn't need to synchronize the image timestamp with the tracking result timestamp because of the way VIRGO posts these they should already be synchronized. What this means is that video frame images and tracking results are posted as quickly as possible and generally have minimal delay. VIRGO posts the video frame images and the tracking results in parallel.

SAFR Client - Reading the Tracking Result Metadata

The SAFR client will retrieve capture images and capture tracking results metadata as fast as possible. In parallel to retrieving the video frames the client can retrieve the tracking results from the corresponding stream. Following the same approach as outlined above the client will append "_tracking_result" to the URL in which to receive capture information. It will retrieve the frames and tracking results as fast as possible, by making a new GET request right after receiving the previous one. The client needs to manage renewing the capture stream before it ends so that there is no stalling of the video. Generally a safe rule of thumb is to renew the stream after half of the frames has been posted. The formula below illustrates how to calculate the estimated posted frame count.

```
let configFramesPerSecond = 1 / (TimeInterval(configFrameDelay) / 1000)
let estimatedPostedFrameCount =
Int((currentDate.timeIntervalSince(initialImageReceivedDate!) *
min(configFramesPerSecond, videoFramesPerSecond) + 0.5))

// So if there is no image and no error we should renew the stream. In this
// case it is possible that Virgo quit posting the images for some
// or it just reached the limit. This should really never happen unless
// there is a problem, but this will recover from that case.
// If the estimated posted frame count is greater than half of the total
// frames then we renew the stream. This is a light weight operation that
// just makes sure that Virgo continues to post frames.
if (image == nil && error == nil) || estimatedPostedFrameCount >= maxFrames
/ 2 {
    renewImageStream()
}
```

- **configFrameDelay**: The user configured frame delay in VIRGA.
- **initialImageReceivedDate**: This is the date/time when the first image was received.
- **videoFramesPerSecond**: This is the frame rate that VIRGO reports it is running at.
- **maxFrames**: This is the maximum number of frames that is configured in VIRGA.
- **image**: The current image received.
- **error**: The current error received if there is one.

Tracking Result Metadata JSON Format

```

Timestamp = {
  "microseconds": "Int64",
  "date": "Int64"
}

Badge = {
  "badgeId": "Int64",
  "detectionService": "String" // "apriltags", "rhinotagsLite",
  "rhinotagsTeam", "rhinotagsFlex", "rhinotagsFull"
}

RecognizedObject = {
  "objectId": "String",      // "person"
  "objectType": "String",    // "person"
  "idClass": "String",       // "unknown", "unidentified", "stranger",
  "noconcern", "concern", "threat"
  "enabled": "Bool"
}

DetectedObject = {
  "objectType": "String",    // "face", "badge", "recognizedObject"
  "localId": "Int64",
  "normalizedBounds": {
    "x": "Double",
    "y": "Double",
    "width": "Double",
    "height": "Double"
  }
  "thumbnailBoundsExpansionFactor": "Double",
  "confidence": "Double",
  "centerPoseQuality": "Double",
  "imageSharpnessQuality": "Double",
  "imageContrastQuality": "Double",
  "yaw": "Double",
  "pitch": "Double",
  "roll": "Double",
  "clipRatio": "Double",
  "pixelBounds": {
    "x": "Double",
    "y": "Double",
    "width": "Double",
    "height": "Double"
  },
  // Face only (objectType = "face")
  "validatorConfidence": "Double",
  // Badge only (objectType = "badge")
  "badge": "Badge",

```



```
// RecognizedObject only (objectType = "recognizedObject")
"recognizedObject": "RecognizedObject",
}

PersonUpdatableProperties = {
  "name": "String",
  "tags": [
    "String"
  ],
  "ignore": "Bool",
  "mergedWithPersonId": "String",
  "gender": "String",
  "age": {
    "lowerBound": "Int64",
    "upperBound": "Int64",
  }
  "externalId": "String",
  "personType": "String",
  "validationPhone": "String",
  "validationEmail": "String",
  "homeLocation": "String",
  "company": "String",
  "moniker": "String",
  "idClass": "String", // "unknown", "unidentified",
  "stranger", "noconcern", "concern", "threat"
  "rootPersonExpirationDate": "Int64"
}

Person = {
  "personId": "String",
  "imageUrl": "String",
  "unmergedImageUrl": "String",
  "rootPersonAddDate": "Int64",
  "sentiment": "Double",
  "smile": "String",
  "occlusion": "Double",
  "updatableProperties": "PersonUpdatableProperties",
  "similarPeople": [
    "Person"
  ],
  "similarityScore": "Double",
  "similarDirectory": "String",
  "confidence": "Double",
  "hasMergedPeople": "Bool",
  "profilePose": "Bool",
  "profilePoseConfidence": "Double",
  "isOccluded": "Bool",
  "faceConfirmed": "Bool"
}

TrackedObject = {
  "objectType": "String", // "face", "badge",
  "recognizedObject"
  "localId": "Int64",
  "person": "Person",
  "isNew": "Bool",
  "detectedObject": "DetectedObject",
  "occluded": "Bool",
  "isolated": "Bool",
  "state": "String", // "detected", "recognizing",
  "recognized", "unconfirmed", "reconfirming"
  "allowsMerging": "Bool",
  "allowRecognizerToLearn": "Bool",
  "timeOfInitialDetection": "Timestamp",
}
```

```

    "timeOfMostRecentConfirmationAttempt": "Timestamp",
    "lingeringCount": "Int64",
    "isZombie": "Bool",
    "identityRecognitionThresholdBoost": "Double",
    "completedSuccessfulRecognitionAttempt": "Bool",
    "completedSuccessfulIdentificationAttempt": "Bool",
    "receivedNotOccludedRecognitionResult": "Bool",
    "disableOcclusionForStrangerClassification": "Bool",
    "identityVerificationComplete": "Bool",
    "consecutiveFailedIdentityVerifications": "Int64",
    "receivedPositiveFaceConfirmation": "Bool"
}

TrackingResult = {
  "disappeared": [
    "TrackedObject"
  ],
  "updated": [
    "TrackedObject"
  ],
  "lingering": [
    "TrackedObject"
  ],
  "appeared": [
    "TrackedObject"
  ],
  "zombies": [
    "TrackedObject"
  ],
  "isSceneChange": "Bool",
  "timestamp": "Timestamp"
}

```

COP Logging

VIRGO supports logging and storing the log information in a file and posting it to a DTP or HTTP URL. Logging is by default disabled. Logging is turned on by sending a log section as part of a configuration message which contains at least a lease date, log deposit URL and a dictionary which stores the desired log levels. The following code block shows an example of a log section which enables logging for the package 'tracking' across all feeds:

```

{
  ...
  "log": {
    "lease-date": 12716,
    "deposit-url": "http://object-server.real.com/virgo-logs"
    "deposit-interval": 12000
    "levels": {
      "tracking": "d"
    }
  },
  ...
}

```

A log section must contain a lease date which is greater than the lease date currently stored by virgo to enable logging. VIRGO enables logging for up to 1 minute. VIRGO automatically disables logging for all log packages after one minute. A new lease date must be sent periodically to allow logging to continue uninterrupted.

VIRGO automatically disables logging after one minute in order to guarantee that logging will not accidentally stay turned on even if the connection to the VIRGA server or the local VIRGO command line tool is lost.

A log level inside the “levels” dictionary is expressed as a mapping from a log package name to a log level. Log package names may optionally be scoped to a feed:

```
# Enable debug level logging for the 'tracking' package on a global level
which means that logging will happen for all existing and future feeds.
"tracking": "d"

# Enable debug level logging for the 'tracking' package for the feed with
the name "foo" only. Other feeds will not generate log statements for this
log package.
"tracking.foo": "d"
```

See Service Logging for a list of all supported log packages and log levels.

VIRGO delivers the log statements to the deposit URL as a JSON array of log statements. The JSON array contains all log statements that have been generated since the last time a log deposit operation was executed. A log statement is a single line of text which is organized into individual fields separated by a tab character. The line is terminated by a newline character. The structure looks like this:

```
<time>\t<level>\t<tag>\t<message>\n
```

where:

Field	Type	Description
time	EpochTime	The time when the log line was generated.
level	String	The log level. See Service Logging for a list of supported log levels.

Field	Type	Description
tag	String	The log package name. See Service Logging for a list of supported log package names. If the log statement was generated by a specific feed then the feed name is appended to the tag and the package and feed components are separated by a single dot. E.g. if a feed "foo" generates a log statement for the package "tracking" then the tag would be "tracking.foo".
message	String	The log message.

The following code block shows an example of a log deposit:

```
[
  "567567\td\ttracking.camera_1\t...\n",
  "567590\td\ttracking.movie\t...\n",
  ...
]
```

The following HTTP headers are included with every log deposit POST request:

Header	Description
X-CLIENT-ID	The client id of the virgod instance. This is immutable, factory configured, unique and descriptive virgo instance identifier. For example: VRGO-LNX-TRPR-16-123
X-CLIENT-TYPE	The type of the client. This is a combination of the client name and the platform name. For example: Virgo-Linux

COP Software Updates

VIRGO supports upgrades to new versions and also downgrades to older versions. An upgrade or downgrade is triggered by including an update section in the status reply (configuration message). VIRGO triggers an upgrade if the version number listed in the update section is greater than the version of the currently running VIRGO and it triggers a downgrade if the version number listed in the update section is smaller than the version of the currently running VIRGO. The update section is ignored if the version number listed in that section is equal to VIRGO's current version number. VIRGO will preserve the existing configuration in the case of an upgrade and it will automatically migrate the existing configuration if the storage format has

changed. Note however that configuration information is not preserved in the case of a downgrade because VIRGO does not support backward migration. Instead VIRGO will initially run with the factor configuration after the downgrade and it expects to receive the version-appropriate configuration information in response to the first status message that VIRGO sends to VIRGA.

VIRGO downloads the update archive from the provided download URL. The download URL may be a HTTP, HTTPS, or file URL. The archive must be a tar.gz file.

VIRGO is able to continuously send update progress events to the progress URL mentioned in the update section. One update progress event is sent every “progress-interval” milliseconds. These events represent the current update progress and they provide additional information about the currently active update stage. If an update fails then an event is sent which includes information about the cause of the update failure.

An update is a multi stage process. VIRGO executes the following stages one after the other to install an update bundle.

Stage	Update Event Status	Description
Downloading	downloading	The update archive is being downloaded to the machine on which VIRGO is running.
Dearchiving	dearchiving	The downloaded update archive is expanded into the update bundle. The bundle is then validated to ensure that it is a well-formed update bundle.
Migration	migrating	The existing VIRGO configuration data is converted to the format expected by the new virgo version. This stage is skipped for downgrades.
Completed	completed	The update has completed successfully.
Failure	failed	VIRGO was unable to complete the update successfully. Note that in this case VIRGO automatically rolls the update back to the previous version.

Once the update process has completed the VIRGO daemon is automatically restarted and it reloads the configuration and it automatically restarts all feeds that are marked as enabled. If on the other hand the update process could not be completed successfully because of some problem then VIRGO is automatically rolled back to the previous version and the VIRGO daemon is restarted.

The following code block shows an example of a update section:

```

{
  ...
  "update": {
    "version": "1.0.240",
    "download-url": "http://virga.real.com/virgo-updates/1.0.240.tar.gz",
    "progress-url": "http://virga.real.com/virgo-updates/progress-1-0-
240",
    "progress-interval": 500
  },
  ...
}

```

This update section will cause VIRGO to be updated to version 1.0.240. Progress events with information about the current state of the update will be sent every half second to the provided progress URL.

Update Events

VIRGO continuously sends update progress events to the provided progress URL to provide information on the current progress of an ongoing upgrade or downgrade operation. The following code block shows the structure of an update progress event:

```

{
  "from-version": "1.0.0",           // [required][semantic version]
  "to-version": "1.0.140",          // [required][semantic version]
  "status": "downloading",          // [required][string]
  "progress": 15,                   // [required][int]
  "log": [                           // [optional][array of strings]
    "576567\td\tupdates\t...\n",
    ...
  ],
  "error": {                         // [optional][dictionary] only provided if
    "status" == "failed"
    "code": 4,
    "message"...
  }
}

```

The following table lists the properties that may appear inside an update progress event:

Property	Type	Description
from-version	Version	The version from which the update was started.
to-version	Version	The version to which VIRGO is being upgraded or downgraded.

Property	Type	Description
status	String	The current update status. See the table listing the “stages” above.
progress	Int	The current progress as a percentage value in the range 0% to 100%
error	Dictionary	The error code and message if the update has failed. Note that this property only exists if the “status” == “failed”.
log	Array<String>?	The log statements that have been recorded since the previous update event. See Logging for a description of how log statements are encoded.

Update progress events are sent whenever VIRGO transitions from one update stage to the next stage and after every “progress-interval” milliseconds.

The following HTTP custom headers are included with every update event:

Header	Description
X-CLIENT-ID	The client id of the virgod instance. This is immutable, factory configured, unique, and descriptive vrgo instance identifier. For example: VIRGO-LNX-TRPR-16-123
X-CLIENT-TYPE	The type of the client. This is a combination of the client name and the platform name. For example: Virgo-Linux

COP Errors

VIRGO includes information about errors in the status message that it regularly sends out. The status message may include a top-level error dictionary and a per-feed error dictionary:

- The top-level error dictionary may appear in any status message and indicates errors that happened on a global level and in the communication between VIRGA and its administration server.
- The per-feed error dictionary always appears in feeds with state “error” or “failed” and may appear in feeds with state “prerolling”.

The error dictionary includes the error code, an error message, and if applicable the number of times that VIRGO has retried an operation.

Do not match errors by their error message. Only match errors by error code. The error message may change in the future while the error code is guaranteed to not change.

The following table lists the errors codes that the error dictionary in a VIRGO status message may contain:

Error Code	Meaning	Description
0	Feed launch failure	A feed could not be launched because there was not enough memory or the feed daemon was not found. If you ever get this kind of error then this means that your VIRGO installation is broken beyond repair or the machine on which VIRGO is running is completely out of resources.
1	Unexpected feed termination	The feed terminated unexpectedly. Usually this means that the feed daemon has crashed or the system is low on memory and the OS decided to kill the feed daemon to recover memory.
2	Decoder not found	The feed daemon is unable to decode the video stream because it lacks the necessary video decoder.
3	Demuxer not found	The feed daemon is unable to process the video stream because it lacks the necessary demuxer component.
4	Protocol not found	The feed daemon is unable to process the video stream because it lacks the necessary network protocol handler.
5	Invalid data	The video stream can not be parsed because it contains unknown/invalid data.
6	Stream not found	There is no stream/file at the location indicated by the stream URL.
7	Bad HTTP request	The server vending the video stream has returned an "HTTP bad request" error.

Error Code	Meaning	Description
8	HTTP unauthorized	The server vending the video stream has returned an "HTTP unauthorized" error. This usually means that the password embedded in the streaming URL is incorrect.
9	HTTP forbidden	The server vending the video stream has returned an "HTTP forbidden" error. This usually means that the video stream you are trying to connect to is not a publicly accessible video stream.
10	HTTP not found	There is no video stream available at the feed's stream URL.
11	Other 4xx HTTP error	The server vending the video stream has returned some other kind of 4xx HTTP error.
12	HTTP server error	The server vending the video stream has returned a HTTP 500 class error.
13	No network	The feed lost network connection. This may mean that the feed is no longer able to receive a video stream or it may mean that it is unable to continue to do (cloud-based) recognition. The feed will automatically try to regain network connection. This error may also appear in the top-level status message. In this case it means that VIRGO itself was unable at some point to communicate with VIRGA.
14	File not found	The feed points to a file (file:// scheme URL) and it is unable to find the video file at that location.
15	Access denied	The feed is unable to open the video stream or video file because it lacks the necessary permissions to do so. For example, the feed points to a video file in the local file system and the video file belongs to a different user which does not allow reading of the file.

Error Code	Meaning	Description
16	Timeout	Some kind of network timeout has occurred. This error is generated for any kind of network operation that may trigger a timeout.
17	I/O error	A generic I/O error has occurred. Eg the feed tried to read from the network and the operation has failed for some reason (but did not time out). Note that I/O errors may be indicative of resource shortage.
18	Unknown environment	You tried to switch VIRGO to another environment and VIRGO has no definition for this environment. An environment name must be one of the predefined environments or one of the custom defined environments listed in the VIRGO factory config file.
19	Unknown feed	You specified a feed name which does not refer to an existing feed.
20	Feed already exists	You tried to add a feed with a name to VIRGO which is already claimed by another feed.
21	Mandatory key missing	The COP message that you sent to VIRGO is missing a required property.
22	Mandatory feed key missing	This is the version of (21) which is returned if a feed-specific key is missing.
23	administrator	You tried to execute a VIRGO function which requires you to be the administrator of VIRGO. E.g. the VIRGO administrator is currently set to "VIRGA" and you tried to add a new feed to VIRGO via the VIRGO command line tool rather than VIRGA. Adding a feed to VIRGO in this case requires that you first switch the administrator from "VIRGA" to "VIRGO".

Error Code	Meaning	Description
24	Unknown administrator	You tried to switch VIRGO to an unknown administrator. The administrator name must be one of "VIRGA" (alternative name "cloud") and "VIRGO" (alternative name "self").
25	administrator URL missing	You tried to switch VIRGO from self-administration mode to VIRGA administration but the environment does not have a URL defined at which VIRGO could contact the administration (VIRGA) server.
>= 1000	Update failed	Applying a VIRGO update has failed for some reason. VIRGO automatically rolls back the previous version.
26	Other	Some other kind of error has occurred.
27	Codec parameters not found	The video decoder was unable to find the required decoder parameters in the video stream.
28	Detector service unavailable	The face detector service is currently unavailable. E.g. because of resource shortage or license restriction.
29	Detector service not authorized	The face detection service can not be used because access to it is not authorized.
30	Recognizer service unavailable	The face recognizer service is currently unavailable. E.g. because of resource shortage or a missing network connection.
31	Recognizer service not authorized	The face recognizer service can not be used because access to it is not authorized. One reason may be that you have forgotten to specify a directory name in the feed configuration.
32	Recognizer SSL error	The face recognizer service can not be accessed because of an SSL error.

Error Code	Meaning	Description
33	Feed unresponsive	The feed appeared to be unresponsive and because of that was restarted. Too many unresponsive feeds or feeds which are repeatedly unresponsive are indicative of resource shortage. E.g. the machine on which VIRGO is running does not have enough processing power or memory to run them all at the same time.
34	No accelerator	The feed is configured to use a GPU exclusively but not enough GPU capacity is available to run the feed. You should assign the feed to the CPU or leave the decision making whether to use GPU or CPU to VIRGO by setting the feed accelerator to "auto".
35	Detector out of memory	The detector is out of CPU memory.
36	Detector out of GPU memory	The detector is out of GPU memory.
37	Detector unable to load model	The detector is unable to find or load its neural network model file.
38	Detector unsupported GPU	The detector does not support the type of GPU on which you are trying to run it.
39	Detector failure	Some other kind of fatal detector failure has occurred.
40	Unknown accelerator ID	The feed is bound to a specific accelerator ID but VIRGO was not able to find this accelerator when it attempted to start up the feed.

The following table lists the errors that are generated if an update from the current VIRGO version to a different VIRGO version failed:

Error Code	Meaning	Description
1000	Update already in progress	You issued another update while an update is already in progress. Note that only one update at a time can happen.

Error Code	Meaning	Description
1001	Invalid version number	The version number in the update metadata is not a valid semantic version number.
1002	Invalid attempt to downgrade	You attempted to downgrade the current VIRGO version to an older version for which no VIRGO exists on the machine.
1003	Invalid update token	An internal server error. If you ever see this then you have to reinstall VIRGO from scratch in order to update.
1004	No active update	Same as 1003.
1005	Missing download URL	The update requires a download but no download URL was specified in the update metadata.
1006	Invalid download URL	The provided download URL is not a valid URL.
1007	No network	The update could not be completed because the network was not available when VIRGO tried to download the update package.
1008	HTTP error	Some HTTP error occurred when VIRGO tried to download the update package.
1009	Corrupted archive	The VIRGO update package is corrupted and VIRGO is unable to decompress it.
1010	Archive validation failure	The downloaded VIRGO update package is missing components or has an incorrect structure.
1011	Unable to relaunch daemon	The VIRGO updater is unable to relaunch the VIRGO daemon after the update. This error should never occur in actual practice, If it does then you'll have to reinstall VIRGO from scratch.
1012	Unable to migrate data	VIRGO was unable to migrate its data from the old version to the new version.
1013	Version already active	You attempted to "upgrade" VIRGO to the version that is already installed and running.
1014	Other	other kind of error has occurred.

Note that the VIRGO updater will always automatically roll back VIRGO to the previous version if an error is encountered while trying to install a new version. The old version is then restarted and will continue to operate the feeds.

COP State Update Algorithms

This page describes the COP update algorithm which allows a control server to update the state in a VIRGO instance.

A control server and the VIRGO instances tethered to it share state. This states describes among other things which feeds exist and what the feed settings are. Delta updates are the preferred mechanism to update a VIRGO instance to new state. They are very efficient and free from data races. Nevertheless the COP protocol supports full state updates because they are key to enabling reliable resynchronization in the event that VIRGO and its control server got out of sync.

Delta updates are reliable because the design of the delta update mechanism is based on the following key principles:

- The control server defines the truth with respect to the shared state.
- The control server decides when to do a delta update and when to do a full update.
- VIRGO provides the control server with the mod-date of its state which allows the control server to efficiently verify that VIRGO's state is in sync with the control server state.

The key principle which informs every other aspect of the design is that the control server and only the control server defines at all times what the truth of the shared data is. The state stored in VIRGO is in principle untrusted. Only after the control server has received a status message from VIRGO and the control server has validated that the mod-date that VIRGO sent is the expected mod-date, is the VIRGO state considered trustworthy and correct until the next status message is received.

The Nature of a Mod-Date

Mod-dates are simple 64 bit integers which represent the current state of the shared VIRGO-VIRGO state. Every time the state changes for some reason the mod-date has to change too. Mod-dates have to be unique in the sense that if you have two states A and B which differ in some form then state A and state B have to identified by different mod-dates.

The meaning of mod-dates is defined by the control server and the control server decides how mod-dates are generated and changed over time. VIRGO does not interpret the bits of a mod-date. It only cares about the fact that two mod-dates which are associated with different sets of data have to be different bit patterns.

Note that although mod-dates are called "mod-dates", they technically do not have to be dates. A mod-date may be any random but unique number. The only thing that is important is that

they are unique.

That said a simple way to generate unique mod-dates is by using the current time when the data is changed. Another simple way is to atomically increment an integer every time a change is applied to the data stored in the control server database.

The Update Timeline

There is a timeline associated with the shared state. This timeline starts at an epoch point and then continues to move along the time axis as the state continues to evolve. Every time the state changes from a previous version to a new version the associated mod-date is changed too.

There are really three mod-dates associated with the shared state:

- **virga-mod-date:** This is the mod-date that the control server generates and stores. It represents the current state of the data stored in the control server database. This mod-date is incremented every time the data in the control server database changes because the user changes one or more settings.
- **expected-mod-date:** This is the mod-date which accompanied the data that the control server has pushed most recently to VIRGO. The reason why this mod-date is called an expected-mod-date is because the control server VIRGO to receive this mod-date back in subsequent status messages from VIRGO. The control server uses this mod-date to detect out-of-sync situations.
- **virgo-mod-date:** This mod-date is stored inside of VIRGO and reflects the current state of the data stored in VIRGO's local database.

Both the virga-mod-date and the expected-mod-date are persistently stored in the control server's database while the virgo-mod-date is stored in VIRGO's local database.

The following sections explain how the update algorithm works.

Timeline Epoch

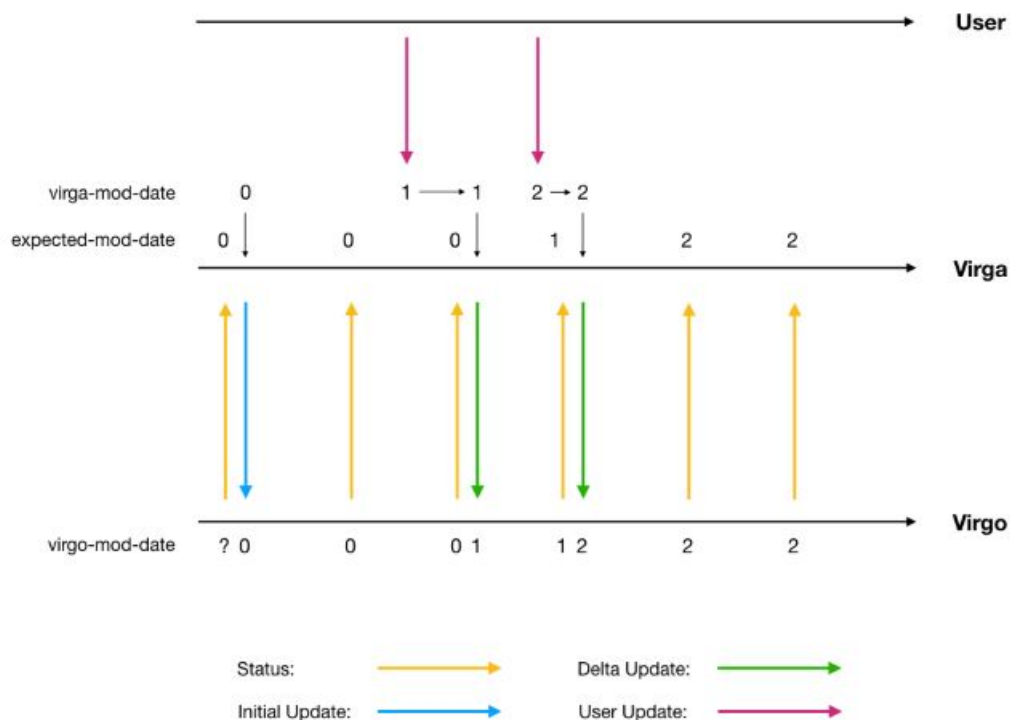
The very first time a control server communicates with a VIRGO instance, the control server does not know which state the VIRGO instance stores and neither does it know what the virgo-mod-date is. Consequently the control server has to do a full state update to adopt the VIRGO instance and to sync it up to its own state.

The control server does this by sending a delta update with the "relative-to" property set to "initial". This tells VIRGO that it should remove all stored data and revert back all its stored settings to its factory defaults. It also tells VIRGO that it should set its virgo-mod-date to the mod-date of the initial message.

From that moment on the VIRGO instance can be considered linked/tethered to the control server and it accurately reflects the current state of the control server.

Updates

The following graphic shows how the timeline of the shared data evolves as updates are applied to the data.



“Virga” here refers to the control server and “User” refers to some kind of user interface which allows the user to view and update the data stored in the control server. Typically VIRGA and VIRGO will run on different machines and are linked through a reliable or unreliable network connection.

The VIRGO state and `virgo-mod-date` are unknown to VIRGA in the very beginning. This is why in response to the very first status message that VIRGA receives from VIRGO it sends an “initial update” to VIRGO. This initial update is an update with “relative-to”: “initial” and whatever else properties VIRGA wants to push out to VIRGO to sync it up with its own state. From that moment on VIRGO status messages will contain a mod-date which is equal to the `expected-mod-date` that VIRGA has stored.

VIRGA maintains two mod-dates in its local database: an `expected-mod-date` and a `virga-mod-date`. The `virga-mod-date` is updated by VIRGA every time the user changes the data. The `expected-mod-date` on the other side is only then updated by VIRGA when it pushes its state to VIRGO. At this time the `expected-mod-date` is set to the current state of the `virga-mod-date`. The two mod-dates are used by VIRGA to detect out-of-sync conditions (see next section). Note that the inequality $\text{virga-mod-date} \geq \text{expected-mod-date} \geq \text{virgo-mod-date}$ is universally true in this scheme.

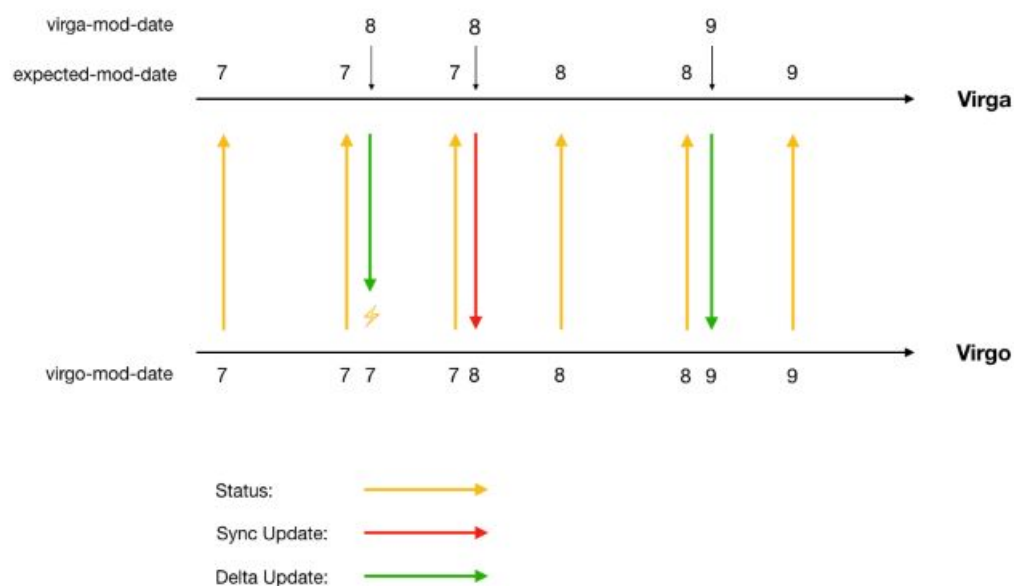
Every time the user changes the data in the VIRGA database, VIRGA increments its virga-mod-date. This new mod-date together with the new data is pushed to VIRGO in response to the next status message that it receives from VIRGO. VIRGO then applies the new data to its current state and it sets its virgo-mod-date equal to the mod-date that was passed along with the update message.

Detecting Out-Of-Sync Situations

It is the responsibility of the control server to detect out-of-sync situations. It can do this easily by comparing its expected-mod-date with the mod-date provided by VIRGO in a status message. Assuming that the mod-date that VIRGO sends in a status message is called “status-mod-date” then VIRGA and VIRGO are out-of-sync iff $\text{expected-mod-date} \neq \text{status-mod-date}$.

Resyncing the Shared State

The control server should initiate a resync of the shared state as soon as it has detected an out-of-sync situation. The following graphic shows an out-of-sync situation and how the control server is expected to detect and correct it:



Note that VIRGA sends an update message to VIRGO with a mod-date of 8 but VIRGO for some reason failed to apply this update. Consequently the virgo-mod-date (the mod-date stored inside of VIRGO) remains at 7 but the control server has advanced its expected-mod-date to 8 because the virga-mod-date was 8 at the time when the control server pushed the update to

VIRGO (remember that the expected-mod-date is set to be equal to the virga-mod-date at the time when an update is pushed to VIRGO).

The next time VIRGO sends a status message to VIRGA, VIRGA's expected-mod-date == status-mod-date (the mod-date from the VIRGO status message) fails. Because of this VIRGA realizes that VIRGO is no longer in sync and that the current state of VIRGO can no longer be trusted. VIRGA now generates a full update with the mod-date 8 and pushes this to VIRGO. This forces VIRGO to replace its current state with the VIRGA provided state.

This full update can be achieved in one of two different ways:

- either send an "apply-as": "full" update with all required properties
- or send an "apply-as": "delta" update with "relative-to" set to "initial" plus all the properties that should be changed to accurately reflect the current VIRGA state

Why Resyncing is Important

The ability to reliably detect out-of-sync conditions and to efficiently and reliably correct them is a major capability of the COP protocol. But you may be wondering how it is possible for the state of the control server and VIRGO to get out-of-sync. Here are some possible reasons why:

- Bugs or (temporary) resource shortages may cause VIRGO to fail to apply an update.
- The control server may not actively check for errors that VIRGO sends back to the control server in response to a failed update. Instead the control server relies on the workings of the COP update algorithm to ensure that state remains synced even in the presence of communication and resource shortage errors.
- A VIRGO instance may be temporarily tethered to a different control server.
- A VIRGO instance may be switched into self-administration mode and then back to cloud administration mode.

No matter what the reason for an out-of-sync situation is the control server is always able to resync the VIRGO instance if it implements the COP update algorithm correctly.

COP Examples

Here are some examples of how to generate VIRGO status message replies for various use cases.

Replacing All Feeds

Assuming that the objective is to unconditionally replace all feeds currently managed by VIRGO:

Feeds known to VIRGO before the update:

```
"camera_1"  
"camera_2"  
"camera_foo"  
"camera_bar"
```

Update:

```
{  
  "mod-date": 767868,  
  "feeds": {  
    "camera_1": { ... },  
    "camera_2": { ... },  
    "camera_3": { ... }  
  }  
}
```

Feeds known to VIRGO after the update:

```
"camera_1"  
"camera_2"  
"camera_3"
```

->

```
"camera_foo" and "camera_bar" have been deleted  
"camera_3" has been added  
"camera_1" and "camera_2" states have been updated to the new state
```

Note: Replacing all feeds is an exceedingly disruptive operation and you should only execute this operation if the goal is truly to replace all existing feeds. If the goal is to add, remove, or update individual feeds then you should use one of the techniques outlined below.

Add a New Feed

Assuming that the objective is to add a new feed without changing any other feeds:

Feeds known before the update:

```
"camera_1"  
"camera_2"
```

Update:

```
{  
  "mod-date": 767898,  
  "apply-as": "delta",  
  
  "feed.additions": {  
    "camera_3": { ... }  
  }  
}
```

Feeds known after the update:

```
"camera_1"  
"camera_2"  
"camera_3"
```

Remove an Existing Feed

Assuming that the objective is to remove an existing feed without changing any other feeds:

Feeds known before the update:

```
"camera_1"  
"camera_2"  
"camera_3"
```

Update:

```
{  
  "mod-date": 7867867,  
  "apply-as": "delta",  
  
  "feed.removals": [ "camera_1" ]  
}
```

Feeds known after the update:

```
"camera_2"  
"camera_3"
```

Update an Existing Feed

Assuming that the objective is to update the state of an existing feed without changing any other feeds:

Feeds known before the update:

```
"camera_1"  
"camera_2"
```

Update:

```
{  
  "mod-date": 7867867,  
  "apply-as": "delta",  
  
  "feed.updates": {  
    "camera_2": { ... }  
  }  
}
```

Install a New VIRGO Version

Assuming that the objective is to update VIRGO to the new version 2.0.3 without changing any of the other states:

```
{  
  "mod-date": 7867887,  
  "apply-as": "delta",  
  
  "update": {  
    "version": "2.0.3",  
    "download-url": "https://virga.int2.real.com/virgo-  
updates/2.0.3.tar.gz",  
    "progress-url": "https://virga.int2.real.com/virgo-  
updates/progress/2.0.3",  
    "progress-interval": 500  
  }  
}
```

Reset VIRGO

Assuming that the objective is to do a full reset of VIRGO back to the factory settings without applying any new state at the same time:

```
{
  "relative-to": "initial"
}
```

Note that it is not necessary to send a mod-date in this case because the reply contains no new state and VIRGO will reset back to modification date 0 and the factory settings.

Reset VIRGO and Apply a New Configuration

Assuming that the goal is to completely replace the existing (and unknown) configuration of a VIRGO instance:

Before:

Some unknown configuration.

Update:

```
{
  "relative-to": "initial",
  "mod-date": 65768678,

  "feeds": {
    "camera_1": { ... },
    "camera_2": { ... }
  }
}
```

After:

- "camera_1 and "camera_2" feeds. All other feeds have been removed because of the reset.

Enable Image Capture

Assuming that the goal is to turn image capture on for a feed without changing any of the other feed state:

Before:

Image capture is turned off for the feed "camera_1".

Update:

```
{
  "mod-date": 76789,
  "apply-as": "delta",

  "feed.updates": {
    "camera_1": {
      "capture.lease-date": 6587687,
      "capture.maximum-frames": 1,
      "capture.deposit-base-url": "https://virga.int2.real.com/virgo-
captures/",
    }
  }
}
```

After:

VIRGO delivers 1 image to this URL:

https://virga.int2.real.com/virgo-captures/camera_1_2017-10-17_266.jpg

Note that image capture will automatically turn off after the image has been delivered because "capture.max-frames" is 1.

Disable Image Capture

Assuming that the goal is to disable image capture for a feed for which it is currently turned on:

Before:

Image capture is turned on for the feed "camera_1".

Update:

```
{
  "mod-date": 76978789,
  "apply-as": "delta",

  "feed.updates": {
    "camera_1": {
      "capture.lease-date": 687678,
      "capture.maximum-frames": 0
    }
  }
}
```

After:

VIRGO no longer captures images for "camera_1".

Renew the Capture Lease

Assuming the goal is to renew the lease of an existing capture stream:

Before:

Image capture for the feed "camera_1" is turned on and active but about to hit its current maximum-frames limit.

Update:

```
{
  "mod-date": 7697678,
  "apply-as": "delta",

  "feed.updates": {
    "camera_1": {
      "capture.lease-date": 78678,
      "capture.maximum-frames": 60
    }
  }
}
```

After:

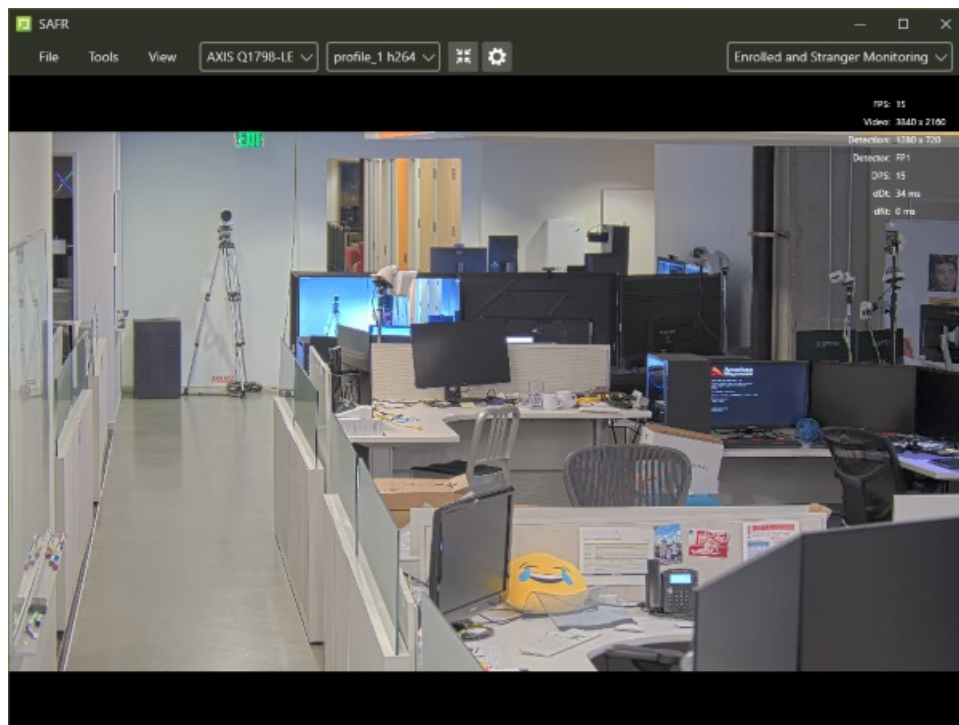
VIRGO has renewed the capture lease and another 60 frames will be captured and delivered.

Desktop Client

The Desktop client is used to add and configure cameras, monitor feeds, get alerts, and view activity. It is also used to update and manage the Identity Database. The Desktop client can be installed on additional laptops or desktops to allow administration and monitoring.

Camera Window

This window enables you to easily view and configure the live feeds from the cameras connected to this client. You can select any of the cameras connected to this Desktop client from the drop-down menu at the top of the window.



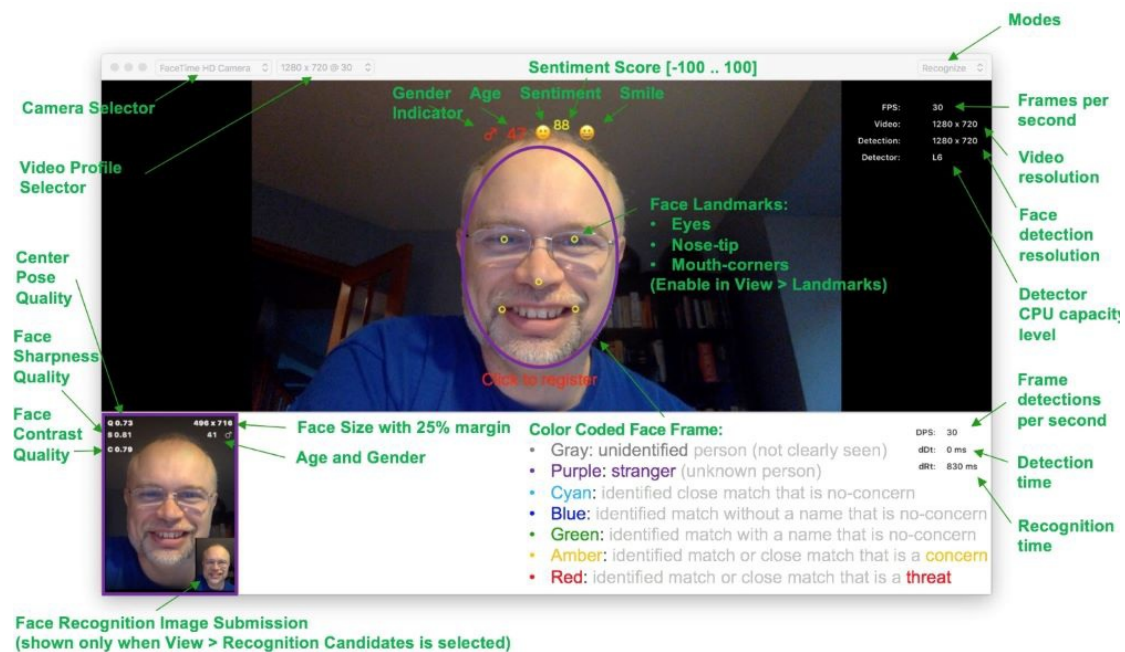
View Menu Options

You can customize the information displayed in the Desktop client's *Camera* window by clicking on the **View** menu when the client has the *Camera* window open. The following options are available:

- **Tracking Frames:** Enables colored indicator frames overlaid around detected faces and objects. See the color codes section here for a description of what each color indicates.
- **Attributes:** Enables the the selected attributes located in the *Detect* section of the Recognition preferences tab to be displayed above the faces seen in the *Camera* window's video feed.
- **Names:** Displays the name (if known) of recognized people below their faces.
- **Full-Screen Names:** Flashes recognized people's names or badge IDs (if known) over the entire video.
- **Motion Vectors:** Shows a motion vector for each tracked face. <doctag=mac>*
Landmarks: Displays the five face landmarks (eyes, nose tip, and the corners of the mouth) on faces viewed in the *Camera* window's video feed. See Interpret Video Feed Overlays for more information about face landmarks.
- **Recognition Details:** Displays image quality metric values on the facial image(s) along the bottom of the *Camera* window. See Image Quality Metrics Guidance for more information about image quality metrics.
- **Flash on Key Frames:** Causes a white backdrop to flash when a key frame is encountered while processing video files. A key frame is a location on a timeline that marks the beginning or end of a transition.
- **Video:** Disables the video, causing only the overlay elements to show.

- **Performance Metrics:** Displays feed window metrics, including frames per second, video and face detection resolution, and CPU capacity level.
- **Pose Liveness State Data:** Displays pose liveness data. For more information, see Pose Liveness Detection

The following depicts various information that is displayed in the feed view, whether it is live from the camera or recorded video:

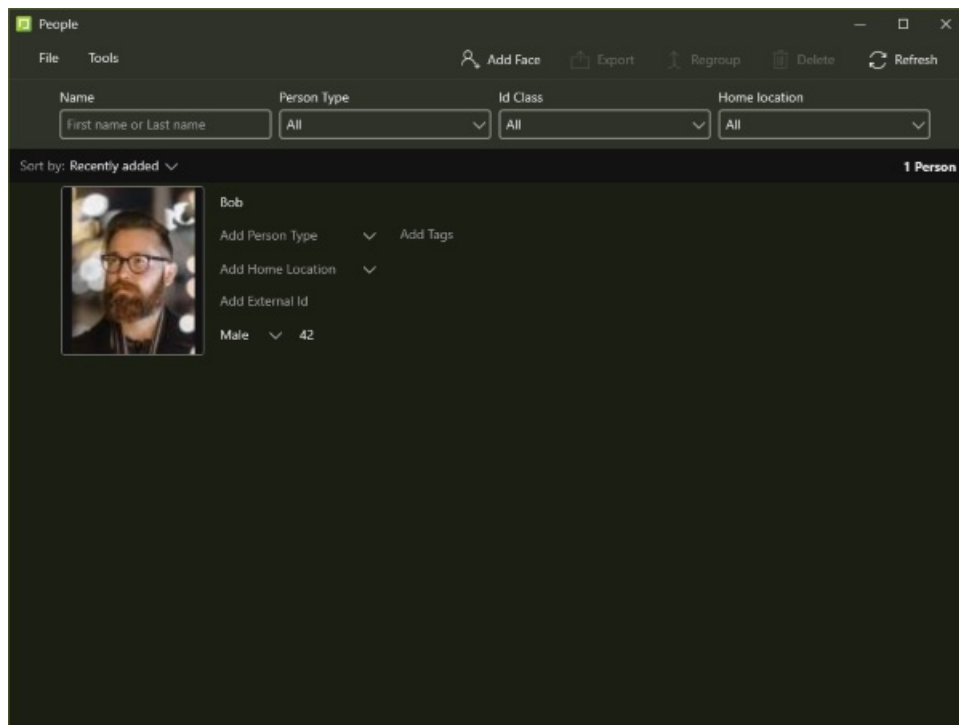


People Window

This window allows you to view and manage registered people. Clicking on the **Add Face** button at the top of the window allows you to use saved image files (e.g. photos) to register people, while clicking on the **Export** button allows you to export the selected faces to image files.

The **Regroup** button allows you to tell SAFR that two different listings in the Identity Database are actually the same person, and they should be merged together. Merging two separate listings together greatly improve, as that particular person will have two reference images that the SAFR system will be able to use to better identify him or her.

Conversely, the **Regroup** button can be used to separate an Identity Database listing that had been previously erroneously merged from two separate listings.



People Filters

You can choose which people are listed in the bottom half of the window by specifying any of the following filters:

- **Name:** Filter based on the people's names.
- **Person Type:** Filter based on the people's `Person Types`.
- **Id Class:** Filter based on the people's `Id Classes` (i.e. their threat level).
- **Home location:** Filter based on the people's `Home locations`.

Identity Database

All the people in the Identity Database that match your specified filters will be listed in the bottom half of the window. Any people that have an `Id Class` of "Threat" will be outlined in red, while any people with an `Id Class` of "Concern" will be outlined in yellow.

All listed people will display the reference image for that person in the Identity Directory, (if available) information about the event, and an image from the event along the right (if available).

You can sort the people based on the following criteria:

- **Recently added:** Sort the people based on when they were registered to the Identity Database, with the latest registrants at the top.
- **First added:** Sort the people based on when they were registered to the Identity

Database, with the earliest registrants at the top.

- **Last name A-Z:** Sort the people alphabetically based on their last names, with last names beginning with “A” at the top.
- **Last name Z-A:** Sort the people alphabetically based on their last names, with last names beginning with “Z” at the top.

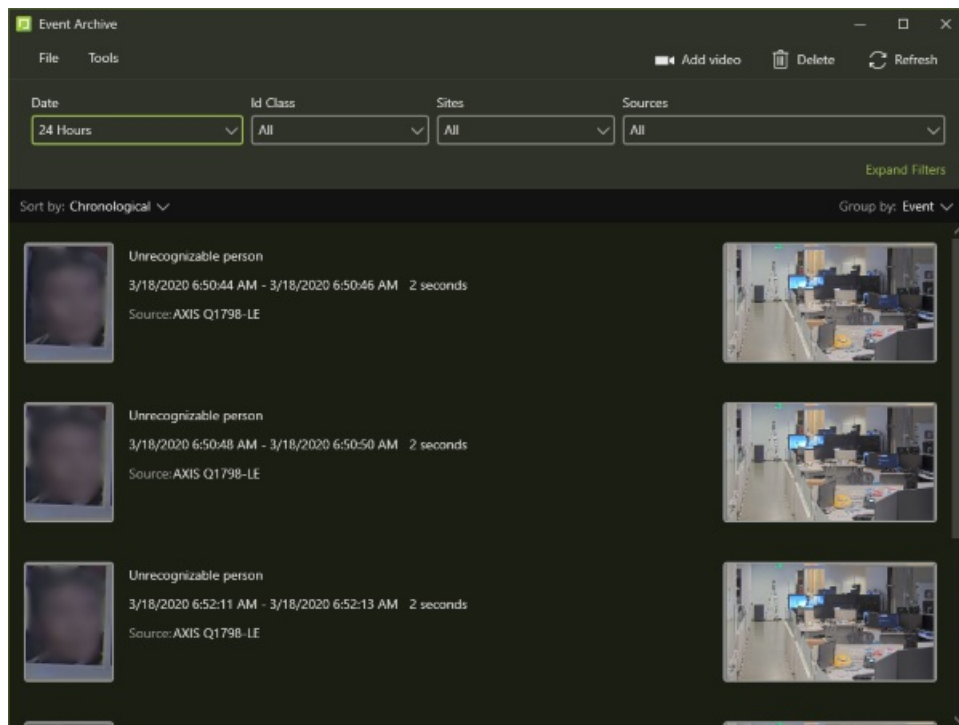
Identity Attributes

Some of the identity attributes are exposed in the default view of the Person Window, but if you double click on people’s faces, you can view and configure all the identity attributes. Note that most of their identity attributes will be empty until manually enter the information.

- **Name:** The person’s name.
- **Identifier:** The person’s unique identifier within SAFR. This value is automatically assigned to them when they’re registered, and cannot be changed.
- **Enrolled Since:** The date when the person was registered.
- **Gender:** The person’s gender.
- **Age:** The person’s age.
- **Company:** The company the person works for.
- **Moniker:** Used to realize two-factor authentication with visual badges.
- **Validation Phone:** The person’s validation phone number.
- **Validation Email:** The person’s validation email address.
- **Enrollment Expiration:** The expiration date of the person’s enrollment.
- **Id Class:** The person’s threat level. (i.e. Threat, Concern, or No-Concern)
- **Person Type:** The person’s `Person Type`. `Person Types` are groupings that you define to differentiate the people registered in your Identity Database. (e.g. “student”, “teacher”, and “staff”)
- **External Id:** If the person has been imported from another database, this value can be used to track the identity in both databases.
- **Home Location:** The person’s Home Location. Home Locations, much like `Person Types`, are labels that you define to help differentiate the people registered in your Identity Database.
- **Tags:** Any custom tags that you have defined. People can have multiple tags assigned to them.

Events Window

This window allows you to view and manage recorded events. By clicking on the **Add video** button at the top of the window you can also use saved video files to generate events and register people.



Event Filters

You can filter the events based on the following criteria. The first 4 filters are always visible, while the others become visible when you click on the **Expand Filters** button.

- **Date:** The date when the event was recorded.
- **Id Class:** The threat level of the person that triggered the event.
- **Sites:** The camera or set of cameras that recorded the event. **Note:** Usually Sites are set to multiple cameras.
- **Sources:** The camera or set of cameras that recorded the event. **Note:** Usually Sources are set to single cameras.
- **Name:** The name of the person that triggered the event.
- **Person Type:** The Person Type of the person that triggered the event.
- **Gender:** The gender of the person that triggered the recording of the event.
- **Tenure:** The date when the identity that triggered the event was registered to the Identity Database.
- **Shortest Gap:** If a person is viewed by one or more cameras multiple times within this time period, all those appearances are considered the same event.
- **Shortest Duration:** The minimum event duration, in milliseconds, to include in your search.
- **Disparate Sources:** If a person is viewed by multiple cameras at the same time, all those appearances are considered the same event when this filter is enabled.

Event Archive

All the events that match your specified filters will be listed in the bottom half of the window. All listed events will display an image of the person who triggered the event on the far left of the listing, the reference image for that person in the Identity Directory, (if available) information about the event, and an image from the event along the right (if available).

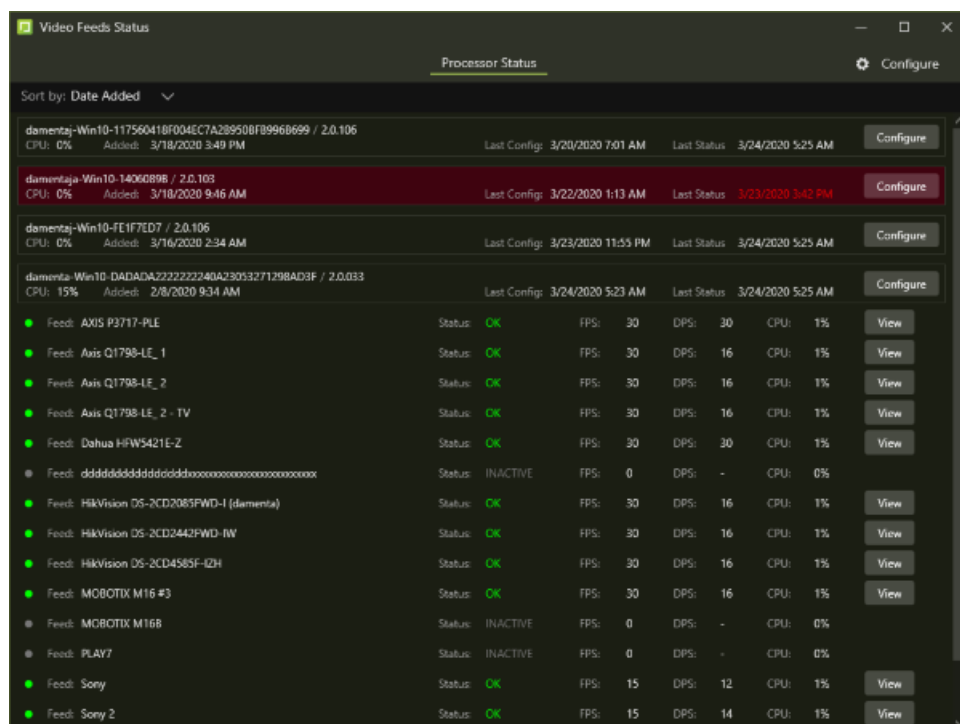
You can sort events by the following criteria:

- **Chronological:** Sort the events based on when they were recorded.
- **Duration:** Sort the events based on how long they last.
- **Name:** Sort the events based on the name of the person that triggered the event, if known.

You also have the option to group events by the person that triggered the event by selecting **Group by: Person** on the right side of the window. If you instead select **Group by: Event**, then the events aren't grouped by person and will instead only be sorted based on the sorting criteria that you have selected.

Video Feeds Status Window

This window allows you to view and configure your video feeds. Any feed that's highlighted in red is currently inactive.



The video feeds can be sorted by one of the following criteria:

- **Date Added:** Sort the feeds based on the date when they were added.
- **Status Date:** Sort the feeds based on the date when the feeds' status last changed.

- **Client Id:** Sort the feeds based on the Id's of the clients to which the feeds are connected.
- **Tenant:** Sort the feeds based on the feeds' tenants.
- **Version:** Sort the feeds based on the feeds' version.

You can click on the **Configure** button to the right of a feed to configure it. Similarly, you can click on the **View** button the right of a camera to see a live video view of the camera.

Add a VIRGO Daemon

To add a VIRGO daemon, there must be at least one active video feed connected to this Desktop client. See Connect Cameras to SAFR for information about how to connect cameras to the client.

Note: You can't add a VIRGO daemon to a video feed that is currently being displayed by the Camera Window.

To add a VIRGO daemon, do the following:

1. Click the **Configure** button on the active video feed that you want to associate the VIRGO daemon with.
2. Hover your mouse over the **feeds** entry. You'll see a **+** button and a **-** button. Click the **+** button. You'll be prompted for the following information:
 - **Feed Name:** Enter any name for the video feed you wish.
 - **Camera:** Select the camera that is providing the video feed.
 - **Mode:** Select the video processing mode that you want the VIRGO daemon to operate in. For a description of the video processing modes, see [here](#).
 - **Apply Mode Customizations from Preferences:** Enable if you want the Desktop client preferences applied to the new VIRGO daemon.
3. Press the **Add** button. The VIRGO daemon has been created. **Note:** At this point, the VIRGO daemon exists independently of both the Desktop client and the video feed that you cloned to create the daemon.

Account Preferences

The Account Preferences tab allows you to configure the user account currently logged into the Desktop client. You can change which user is logged in entirely by clicking on the **Logout** button next to the *User Identifier*.

- **User Identifier:** The username of the logged-in user.
- **User Password:** The password for the user entered in the *User Identifier* field.
- **User Directory:** The user directory to use. (The default *user directory* is *main*.)
- **User Site:** The *User Site* label for this desktop client. All SAFR event data is tagged by site and source labels. These labels are used to help filter and analyze collected recognition events, such as where a face was recognized.

- **User Source:** The *User Source* label for this desktop client. All SAFR event data is tagged by site and source labels. These labels are used to help filter and analyze collected recognition events, such as where a face was recognized.
- **Similar Directory:** The directory to use when using the *Similar* video processing mode. If this setting is left blank, then the *User Directory* is used when using *Similar* mode. See *Connect to a Video Feed* for more information on video processing modes.
- **Environment:** Determines which operating environment your client contacts. The possible values for this field are as follows:
 - *SAFR Developer Cloud:* Internal use only.
 - *SAFR Partner Cloud:* Internal use only.
 - *SAFR Cloud:* Used for cloud deployments. This is a general availability SAFR Server in the cloud maintained by RealNetworks.
 - *SAFR Custom:* Used for local deployments. If you select *SAFR Custom*, you will be asked to provide the URLs for the primary SAFR Server services.
- **Report Status:** When enabled, video feeds will report their status to the SAFR Server even when a VIRGO daemon isn't associated with it.
 - **Allow Remote Viewing:** Enables video feeds to be viewed by any SAFR client, even when a VIRGO daemon isn't associated with them.

Camera Preferences

The Camera Preferences tab allows you to add, remove, or configure cameras connected to this Desktop client. Cameras that have already been automatically discovered are displayed here.

SAFR normally auto-detects all integrated (built-in) cameras, USB-connected cameras, and IP (internet protocol) cameras that support the ONVIF (Open Network Video Interface Forum) protocol as long as they are present on the same network to which the Desktop client is connected. For more about ONVIF and the ONVIF specification, see the [ONVIF home page](#).

ONVIF Cameras

- Each camera discovered via ONVIF must have a username and password.
- Be sure that the camera has at least one ONVIF user with administrative privileges, or ONVIF authentication will not work.
- ONVIF video profiles configured on the camera will be available to select in the live video recognition view.
- The date and time configured on the camera must be within five seconds of the system time SAFR is running on.

Camera Preferences

To configure a camera, select it in the left panel. If you need to add a camera, see *Connect*

Cameras to SAFR. When a camera is selected, a set of preferences will appear on the right of the window. Which preferences are exposed depends on which kind of camera is selected.

USB Camera Preferences

USB cameras have the following preference settings.

- **Source:** Use this field to override the default source label for the camera. For example, you might want to change a default source label to West Hall or a more meaningful label for your specific environment.
Multiple cameras can have the same source label. For example, you could have several cameras labeled as South Exit, or you could label each camera differently (“South Exit 1”, “South Exit 2”, or other labels that make sense for you and your SAFR environment).
- **Contrast Enhancement:** When selected, enhances low-light images and videos by enhancing the contrast.
 - **Low Light Threshold:** Determines the level of light when the exposure boost will be applied. If an image or video has more light than the threshold allows for, then the exposure boost will not be applied.
 - **Exposure Boost:** Determines how much to boost the contrast.
- **Front facing:** Indicates if the camera is front facing.
- **Rotate Image:** Use the dropdown menu to set a value that rotates to correct for camera orientation. Typically, this correction can be done directly by the camera, but if not, you can adjust it here. Recognition does not work properly unless an image is properly oriented.
- **Enforce Low Latency:** This option is only available for integrated and USB cameras. *Enforce Low Latency* optimizes display and processing by allowing the video frame rate to drop if CPU resources are low.
This may be needed for some 4K webcam models or any cameras that support particularly high frame rates.
- **Unauthorized movement detection for person:** When any of the 4 settings below are enabled, the Desktop client will generate an unauthorized movement event when a person’s face travels further than the specified percentage within the camera view field. For example, if `Left travel distance` were set to 50 then an unauthorized movement event would be generated if somebody’s face entered the camera view field from the right edge of the frame, and moved more than halfway across the camera field view. Note that you can set multiple values below.
 - **Left travel distance:** Generate an unauthorized movement event when a face enters from the right edge of the camera view field and travels more than the specified percentage across the field towards the left edge.
 - **Right travel distance:** Generate an unauthorized movement event when a face enters from the left edge of the camera view field and travels more than the specified percentage across the field towards the right edge.
 - **Up/Away travel distance:** Generate an unauthorized movement event when a face enters from the bottom edge of the camera view field and travels more than the

specified percentage across the field towards the top edge.

- **Down/Towards travel distance:** Generate an unauthorized movement event when a face enters from the top edge of the camera view field and travels more than the specified percentage across the field towards the bottom edge.

IP Camera Preferences

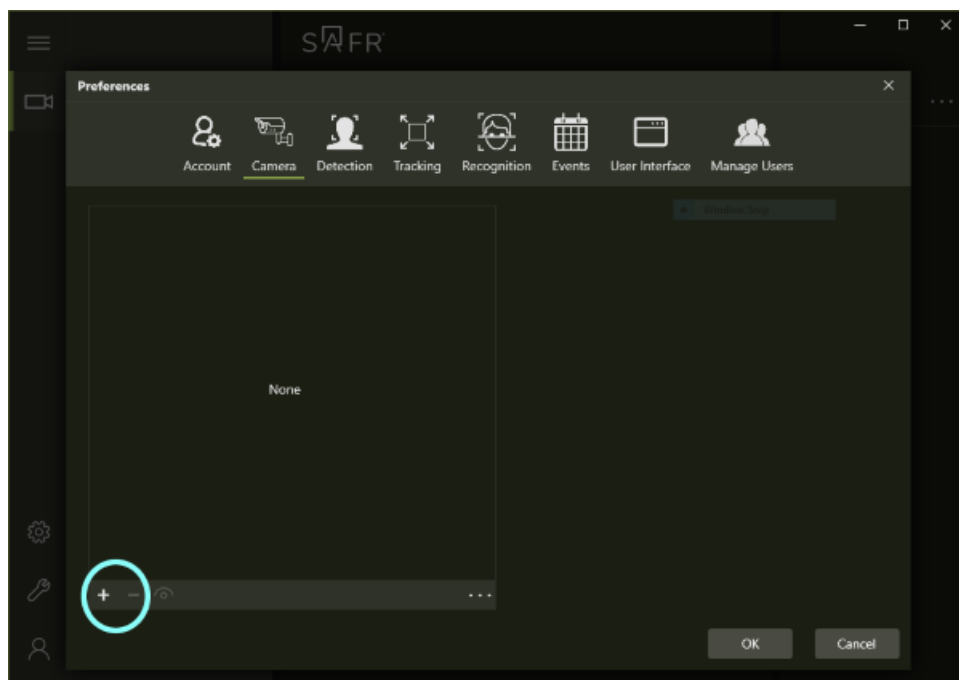
IP cameras have the following preference settings.

- **Source:** Use this field to override the default source label for the camera. For example, you might want to change a default source label to West Hall or a more meaningful label for your specific environment.
>Multiple cameras can have the same source label. For example, you could have several cameras labeled as South Exit, or you could label each camera differently (“South Exit 1”, “South Exit 2”, or other labels that make sense for you and your SAFR environment).
- **URL:** URL of the camera.
- **RTSP Transport Protocol:** For the lowest latency, select UDP (User Datagram Protocol). If network packet loss is an issue, select TCP (Transmission Control Protocol). Generally, UDP is a faster best effort communication system, whereas TCP is more reliable but slower. For example, if network connectivity is a concern for a particular camera, you might want to change this to TCP. Otherwise, UDP should be adequate in most situations.
- **Contrast Enhancement:** When selected, enhances low-light images and videos by enhancing the contrast.
 - **Low Light Threshold:** Determines the level of light when the exposure boost will be applied. If an image or video has more light than the threshold allows for, then the exposure boost will not be applied.
 - **Exposure Boost:** Determines how much to boost the contrast.
- **Lens Correction:** Enable lens correction to help correct the fisheye effect for very wide-angle lenses. This also improves recognition accuracy although is not needed in most cases.
 - **Coefficient K1:** The “K1” lens correction factor.
 - **Coefficient K2:** The “K2” lens correction factor.
- **Enforce timing for video frames:** When enabled, playback frame syncing to video clock is enforced.
- **Enable mirroring:** When enabled, the camera’s video feed is mirrored.
- **Front facing:** Indicates if the camera is front facing.
- **Rotate Image:** Use the dropdown menu to set a value that rotates to correct for camera orientation. Typically, this correction can be done directly by the camera, but if not, you can adjust it here. Recognition does not work properly unless an image is properly oriented.
- **Back Channel:** If you’re connecting to a Mobotix camera, you can set this field to *Mobotix MX* to have SAFR report *STRANGER* and *RECOGNIZED* event types to the Mobotix camera. This feature is necessary if you want to make use of the Mobotix app.

If this connection is to a non-Mobotix camera, setting this field to *Mobotix MX* has no effect.

- **Cash Point:** This value must match the configured *cash point* within the Mobotix app. If this value doesn't match the cash point within the Mobotix app, the *back channel* won't work.
- **Frame buffer size:** Sets the size of the camera's frame buffer for buffering network streams.
- **Unauthorized movement detection for person:** When any of the 4 settings below are enabled, the Desktop client will generate an unauthorized movement event when a person's face travels further than the specified percentage within the camera view field. For example, if `Left travel distance` were set to 50 then an unauthorized movement event would be generated if somebody's face entered the camera view field from the right edge of the frame, and moved more than halfway across the camera field view. Note that you can set multiple values below.
 - **Left travel distance:** Generate an unauthorized movement event when a face enters from the right edge of the camera view field and travels more than the specified percentage across the field towards the left edge.
 - **Right travel distance:** Generate an unauthorized movement event when a face enters from the left edge of the camera view field and travels more than the specified percentage across the field towards the right edge.
 - **Up/Away travel distance:** Generate an unauthorized movement event when a face enters from the bottom edge of the camera view field and travels more than the specified percentage across the field towards the top edge.
 - **Down/Towards travel distance:** Generate an unauthorized movement event when a face enters from the top edge of the camera view field and travels more than the specified percentage across the field towards the bottom edge.

Use Additional Options



Click + to manually add an IP camera.

Click - to delete the configuration for manually added cameras. This option is not available for auto-discovered cameras.

Click the *Settings* icon to export or import cameras and their associated configurations that have been manually added to the system.

Exporting enables you to create a copy of a camera configuration. Exporting saves a camera configuration to an .acc file in JSON format.

Importing enables you to import a copy of a camera configuration. Use this feature to copy a camera configuration from one SAFR system to another.

Detection Preferences

The Detection Preferences tab allows you to configure facial, badge, and person detection characteristics.

- **For Mode:** Selects which mode is affected by the current settings on this page.

Enable Face Detector

The **Enable face detector** check box must be selected to enable face recognition.

- **Reduce vertical input image size to:** This setting represents the vertical size in pixels to which the image scanned by the camera is scaled in order perform face detection.

Scaling down reduces CPU usage. 720-pixel resolution is usually sufficient for high-quality face detection without slowing down the CPU.

Vertical size can be reduced to 640-pixels and even 480 to greatly reduce CPU usage for face detection, but this reduces the ability to detect and recognize smaller faces.

(e.g. faces farther away from the camera)

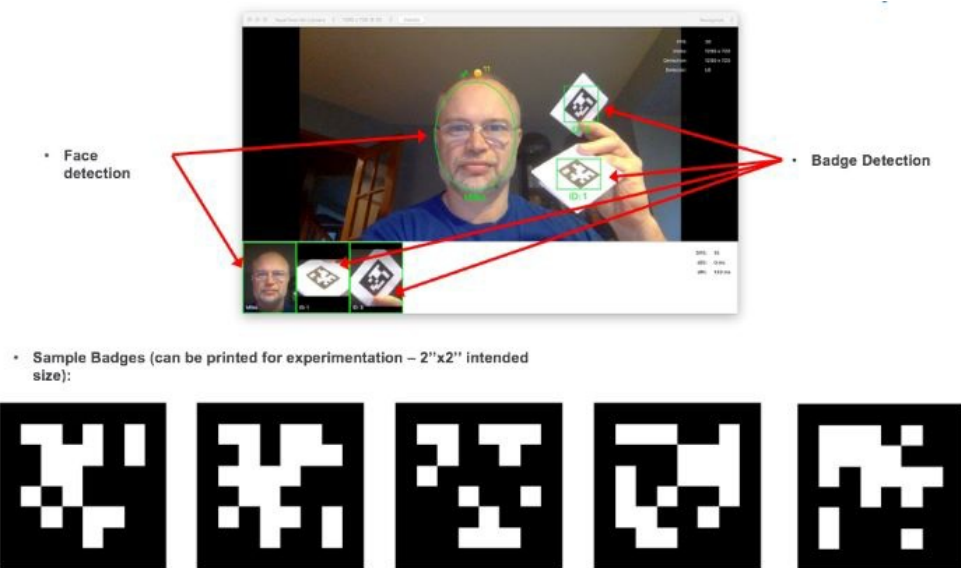
- **Minimum searched face size:** This setting defines the minimum face size that can be detected. A searched size of 80, for example, can still manage to detect faces as small as 60x60, but with lower certainty. Lowering this number enables SAFR to detect much smaller faces but also greatly increases CPU usage.
- **Minimum required face size:** This setting defines the minimum required size for a face to be detected. Any face smaller than the height or width is ignored.
This is typically set when face detection needs to be limited only to large faces, which indicates a face being closer to the camera. It may not be desirable, for example, to cause a lot of detection events for faces that are too far from a camera to be considered necessary for attention by SAFR.
- **Consecutive confirmations required:** This setting adds consecutive confirmations to the SAFR facial recognition to create more reliable detections. Increase this setting for more reliable but slower facial detection. Decrease it for faster but slightly less reliable detection.
- **Generate recognizer hint for detected faces:** This setting optimizes facial recognition and should be turned on for most cases. If it is turned off, recognition accuracy may be reduced if detection is performed at very low resolutions.
- **Use custom detection thresholds:** Allows you to customize the detection threshold. When this setting is checked, you can click on the *Configure* button to do the customization.
 - **Initial candidate selection threshold:** Initial face candidate threshold that is used during face detection.
 - **Middle candidate selection threshold:** Middle face candidate threshold that is used during face detection.
 - **Final candidate selection threshold:** Final face candidate threshold that is used during face detection.
- **Detection services:** Specifies which face detection service your Desktop client will use:
 - *SAFR*: The face detection service that ships with SAFR. We strongly recommend that you use SAFR.
 - *Coreimage*: A face detection service native to Apple. Its performance is comparable to SAFR's, but slightly less accurate.
 - *Vision*: Another face detection service native to Apple that's faster but much less accurate than either SAFR or Core Image.
- **Frame buffer size:** Sets the size of the frame buffer.

Enable Badge Detector

The **Enable badge detector** check box must be selected to enable badge detection. Badges are visual representation of users that are quicker and easier to detect than faces. Compared

to faces, they are easier to detect and recognize when rotated and when used in low light conditions.

- **Reduce vertical input image size to:** This setting is the vertical size a scanned image (badge) is scaled to in order to detect the badge. Scaling down the image reduces CPU usage.
If your badges are very small, we do not recommend using this setting.
- **Minimum searched badge size:** This setting defines the minimum badge size that can be detected. Lowering the number enables SAFR to detect very small badges (down to 15x15 pixels), but it also increases CPU usage. Increasing this setting reduces CPU usage but requires larger badges for successful detection.
- **Minimum required badge size:** Use this setting to require a minimum badge size in pixels. Any badge smaller than the number entered here in either height or width is ignored. This is mainly used when badge detection is limited to large badges, which indicates badges are close to the camera. If set too small, SAFR could create many detection events for badges that are far from the camera and would not be considered of interest.
- **Consecutive confirmations required:** This setting adds consecutive confirmations to the SAFR detection to create more reliable detections. Increase this setting for more reliable but slower badge detection. Decrease it for faster but slightly less reliable detection.
- **Detection service:** The different possible values for this setting represent different detection methods. Depending on the capabilities of your cameras, lighting conditions, and other variables, certain options may work better with your environment than others. You can choose from the following options:
 - **apriltags:** Basic badge detection.
 - **rhinotagsLite:** The fastest badge detector, but it has a lower tolerance for motion blur. It requires cameras with a fast shutter speed.
 - **rhinotagsTeam:** Faster badge detector, but it has little resilience to motion blur.
 - **rhinotagsFlex:** Fast badge detector with moderate resilience to motion blur.
 - **rhinotagsFull:** Badge detector with robust handling under various conditions and a strong resilience to motion blur.
rhinotagsFull is the recommended option.



A full set of badge images supported by SAFR is available at

<https://github.com/anqixu/apriltag/tree/master/tag36h11>.

- It is recommended these images be re-sized to at least 2" x 2" size using the nearest neighbor algorithm (to maintain sharp edges) before use with SAFR.
- While a single badge of displayed format can express only 587 IDs, multiple badges can be combined to increase the number of expressible IDs into the billions.

Note: 6 badges combined can express over 827 billion identifiers.

- **Frame buffer size:** Sets the size of the frame buffer.

Enable Object Detector

Internal use only.

Workload Limits

- **Max Simultaneous Detections:** Specifies the maximum number of detectors across all video feeds that can be used on this Desktop client. This setting defaults to the number of cores that your machine has. We strongly recommend that you do not change this setting; doing so could greatly decrease your system's performance.

Tracking Preferences

The Tracking Preferences tab allows you to configure settings for tracking detected people.

- **For Mode:** Selects which mode is affected by the current settings on this page.
- **Maximum change relative to face size**
 - **Position:** The maximum position change, specified in percentage relative to the object size, to continue tracking.

- **Size:** The maximum size change, specified in percentage relative to the object size, to continue tracking.
- **Stop tracking a face after it has lingered for:** Determines for how many frames more the system will continue to keep a tracked face around after it has failed to detect it in the most recent frame.
- **Minimum recognitions to lock on to identity:** Minimum number of consecutive recognition attempts that we must run and produce the same person identity before we lock onto this identity.
- **Minimum recognitions to learn identity:** Minimum number of consecutive face recognitions required to register it into the Identity Database. This setting also affects the quality of the reference face signature recorded for an identity.
Increasing the number increases the minimum quality of the face signature registered for a newly registered identity, but it decreases how quickly the SAFR system registers new identities.
- **Initial recognition attempts:** Number of initial recognition attempts to make on an unrecognized face as quickly as possible.
- **Failed recognition back-off interval:** After making the initial recognition attempts as quickly as possible, back up the amount specified by this setting for each subsequent recognition to slow down. This goes on until the retry interval is reached.
- **Retry failed recognition after every:** The interval in which to run recognition requests if the face has not been recognized.
- **Reconfirm identity after every:** This number defines how often a face's identity is reconfirmed, in milliseconds. If you set this value to zero, SAFR continues to re-confirm a face's identity at its normal rate. If you increase this number, it increases the confirmation rate and improves SAFR tracking subjects in crowded settings. However, increasing the number also increases CPU and network usage.
- **Update identity every:** Updates the identity when the currently saved identity is older than the updated identity.
- **Minimum failed recognitions to stop tracking identity:** When a face is being tracked recognitions are continually confirming the identity. The identity is also being verified if it is transferred from a person object. In these cases, if the recognition or verification consecutively fails this number of times then the identity will be reset and no longer associated with the face because we are no longer sure it is the same identity.
- **Update identity with better image:** Updates the identity only when the identity currently saved is older than the updated identity.
- **Enable correlation of faces by size:** Enables face correlation of tracked faces, which compares detected faces looking for a change in area.

This should usually be enabled.

Disabling it may help when there is only a single face to track and head movements are very fast.

- **Enable motion prediction:** Enables face motion prediction, which predicts which direction the face is moving in order to maintain tracking.

This should usually be enabled.

You may want to disable this option only when tracked faces might be moving in highly irregular motion patterns.

- **Stop tracking on failed recognition:** Enabling this option causes identity tracking to stop when SAFR doubts a confirmation of face tracking failures. SAFR is then forced to obtain a new identity lock. Enabling it may produce more discontinuity in recognition events and provide additional protection against mistaken tracking.

You rarely need to enable this option.

- **Reconfirm identity in video after each Key Frame:** When a key frame is encountered in a video file all the faces that are being tracked are marked as unconfirmed so that their identities are reconfirmed to make sure they are the same person. This setting only applies to video files; it can't be used with live video. If a video file does not represent recorded live video then this can typically be set to true for better tracking during scene changes.

Recognition Preferences

The Recognition Preferences tab allows you to configure a variety of settings that determine if SAFR detects, tracks, and recognizes faces and identities.

Understand Facial Recognition

There are three key elements of SAFR facial recognition, each of which consists of a variety of settings that can be changed to help you fine tune your SAFR system:

- **Detection:** When a face is detected by a SAFR camera, as well as various settings that can affect how a face is recognized; minimum size, and resolution for a face to be detected, and more.
- **Recognition:** When a face is recognized it is compared against the SAFR database of recognized faces so it can be identified. Over time, SAFR collects and compares more images of an individual's face to help it build a catalog of variations to enable it to better recognize someone's face under different lighting conditions, angles, and with differing characteristics, such as a beard or different colored hair.
- **Tracking:** When a face is tracked, how long it is tracked, and other characteristics such as how tolerant the tracking is of motion, and when to stop tracking.

Understand Occlusion Detection

SAFR can detect occluded faces. Occlusion constitutes any obstruction of the key facial features by, for example, a scarf, a hand, glasses, a mask, or hair draping over the face.

Occlusion detection can be used to:

- Filter out highly occluded faces while learning them in the wild and preventing the storing of ambiguous face references in the Identity Database. For example, occlusion detection might be used when attempting to register players sitting at a table to prevent registering them with an occlusion feature, such as wineglass in front of their faces that may later create recognition inaccuracies.
- Update the occurrence event record with better face images without the occlusion to increase the value of the image stored with the event for presentation and investigation purposes.

Occlusion Detection Related to Events

When the server returns an occlusion threshold in a recognition response, the most recent value is passed to the posted event under the following circumstances:

- When the event image is posted or updated.
- When the event `idClass` is set or updated.

The effect is the occlusion value in the event reflects occlusion of either the most recent event image update or an `idClass` change, whichever occurs last.

Recognition Preference Options

- **For Mode:** Selects which mode is affected by the current settings on this page.
- **Minimum Required Face Size**



- **For Recognition:** This setting defines the minimum required face size in pixels to attempt recognition. It includes a 25% margin around the face, for example, the part of the image captured by the camera that is not part of the face itself.
The minimum face size (with +25% percent margin) for maximum recognition accuracy is 240 pixels.
Faces smaller than 240 pixels may have slightly reduced recognition accuracy.
This setting can be as low as 60 pixels if other image characteristics are favorable.
- **For Merging:** This setting defines the minimum required face size (+25% margin) to attempt merging a captured face with the existing reference for an already

recognized identity.

When in Learning mode, the SAFR recognition system may merge references for an identity to improve its understanding of different face characteristics for that identity. Consider this as a catalog of variations for a face.

- **For Learning/Strangers:** This setting defines the minimum required face size (+25% margin) to enable SAFR to store a reference image for a new identity. When in Learning mode, the SAFR recognition system only learns faces that meet this quality metric.
- **For Full Accuracy:** Set to establish the point below which face size is considered when determining image quality. Once the face size goes above this setting, face size is no longer used to determine the quality of the image. This setting should be changed only in very special circumstances.
- **Minimum Required Center Pose Quality**
 - **For Recognition:** This setting defines the minimum required quality for a face posed directly in front of the camera (center posed) to attempt recognition. Center pose quality (CPQ) ranges between one and zero. A score of 1 is given to a face looking straight into the camera. Any deviation from this position diminishes center pose quality. Center pose quality of a face in full profile position is given a score of zero. Recognition from any pose is possible, but accuracy is reduced for faces that are in extreme profile positions.
 - **For Merging:** Indicates minimum required face center pose quality to attempt merging with existing reference for a recognized identity.
 - **For Learning/Strangers:** Indicates minimum required face center pose quality to attempt merging with existing reference for a recognized identity.
 - **For Direct Gaze Detection:** Events include a `directGazeDuration` property describing how long the person was looking at the camera. The `directGazeDuration` value is controlled by this For Direct Gaze Detection threshold in the settings. If the CPQ value is above this threshold, the face is determined to be looking at the camera, but if the value is below this threshold, the face is determined to be turned away. The longer the value is above this threshold, the longer the direct gaze duration is.
 - **Use advanced settings for learning/strangers:** Enable if you want to modify the advanced learning setting below.
 - **Max Yaw:** Indicates minimum required yaw value to attempt registering somebody into the Person Directory. Yaw measures how much a face is turned to the left or right; a value of 0 indicates that the face is looking straight ahead.
 - **Max Pitch:** Indicates minimum required pitch value to attempt registering somebody into the Person Directory. Pitch measures how much a face is tilted up or down; a value of 0 indicates that the face is looking straight ahead and isn't tilted at all.
 - **Max Roll:** Indicates minimum required roll value to attempt registering somebody into the Person Directory. Roll measures how much a face is tilted to one side or the other; (i.e. the person's ear is moved closer to their

shoulder) a value of 0 indicates that the face isn't tilted to either side.

- **Minimum Required Face Sharpness Quality**

- **For Recognition:** Indicates minimum required face sharpness quality to attempt recognition.
- **For Merging:** Indicates minimum required face sharpness quality to attempt recognition.
- **For Learning/Strangers:** Indicates minimum required face sharpness quality to store as a reference for a new identity.

- **Minimum Required Face Contrast Quality**

Contrast quality defines the difference between the color of a subject's face and the background.

- **For Recognition:** This setting indicates the minimum amount of contrast quality (lower or higher contrast) for SAFR to attempt a recognition.
- **For Merging:** Defines the minimum required face contrast quality to attempt merging a captured face with its existing references in the SAFR system.
- **For Learning/Strangers:** Indicates the minimum required face contrast quality to store as a reference for a new identity.

- **Maximum Allowed Occlusion**

- **For Learning/Strangers:** Indicates the maximum occlusion value allowed for a face to be registered to the Person Directory. When this setting is set to 1, no occlusion filtering is applied, and the default configuration is ignored.
- **Learn Occluded Faces:** Select to have the system learn the faces. The check box is cleared by default.

- **Clipping Tolerances**

Clipping occurs when a face is only partially captured by a camera.

- **For Recognition:** This value defines the maximum amount of clipping tolerance (as a percentage of width or height) to attempt recognition. Faces not fully in the field of view are not recognized unless within this clipping tolerance threshold.
- **For Learning:** Indicates maximum allowed face clipping tolerance (as percent of width or percent height) to store as a reference for a new identity.

- **Identity Recognition Threshold:** The recognition threshold governs the strictness of the face recognition in declaring an identity match between a face and a user.

Increasing the threshold makes the system less strict and can help it overcome hindrances such as face masks covering the nose and mouth while still providing accurate recognition. However, increasing this threshold increases the chance of incorrect (false positive) identifications.

When faces are imported from the Desktop client's **File > Import Faces** menu option, Camera Recognition Threshold is used to help recognize faces within the imported image.

The following threshold values can be used as a basic reference for different scenarios:

- 0.54 – Secure Access; very high confidence identification with very low false positives.
- 0.67 – Traffic monitoring; good for general monitoring in busy areas.
- 0.84 – Celebrity matching; low/unreliable recognition and face matching.

- 0.94 – Least secure; matching very small, grainy, or blurry images or video. For convenience, the Identity Recognition Thresholds can be set separately for video from the camera and video file sources.
- **Camera:** TBD
- **Video:** TBD
- **Similar:** TBD
- **Proximity threshold allowance:** TBD

Detect

Select the check box to enable the detection of the following characteristics:

- **Identity:** The identity of the user in the SAFR system, such as their name.
- **Occlusion:** Obstructing the full view of the face by using, for example, a mask, glasses, or using a hand to block a part of the face.
- **Gender:** Enables the detection of gender information.
- **Age:** Enables the detection of age information.
- **Sentiment:** Enables the detection of sentiment information.
- **Smile action:** Enables the smile action recognizer.
 - **Pre-smile delay:** The amount of time, in milliseconds, that there should be no smile.
 - **Smile duration:** The amount of time, in milliseconds, that the smile should last.
 - **Identity recognition threshold boost:** The smile threshold to boost temporarily during the smile action.
- **Pose liveness action:** Enables the pose liveness action recognizer. See Pose Liveness Detection for more information.
 - **Center pose quality:** Minimum center pose quality to use when detecting the initial center pose.
 - **Profile pose quality:** The maximum center pose quality to use when detecting the final profile pose.
 - **Max profile confidence at start:** Maximum profile pose confidence to allow during the initial center pose detection phase.
 - **Min profile confidence at end:** Minimum profile pose confidence to allow during the final profile pose detection phase.
 - **Min profile pose yaw:** The minimum profile pose yaw value that is required during the final profile pose detection phase.
 - **Center pose consecutive confirmations required:** Number of consecutive center pose confirmations required to enter the initial center pose detection phase
 - **Profile pose consecutive confirmations required:** Number of consecutive profile pose confirmations required to enter the initial center pose detection phase.
 - **Min profile similarity:** Minimum similarity score required when verifying the final profile pose.
 - **Min Detections Per Second:** Minimum number of frames per second required during the process.
 - **Min transition poses:** Minimum number of required center pose samples during

the transition from center to profile pose.

- **Max CPQ jump in continuous tracking:** Maximum change between samples while the pose is changing from center to profile.
- **Max CPQ jump after tracking loss:** Maximum change between samples while the pose is changing from center to profile if lingering.
- **Max profile pose roll:** The maximum roll threshold in either direction in which the face can rotate when determining whether the face is in profile pose.

Workload Limits

- **Max Simultaneous Recognitions:** Specifies the maximum number of recognizers across all video feeds that can be used on this Desktop client. We strongly recommend that you do not change this setting; doing so could greatly decrease your system's performance.

Events Preferences

The Events Preferences tab allows you to configure event reporting as well as event replies. Every reported event contains start and end times as well as information about where it occurred and who or what triggered it. The location of the event is indicated by the site and source labels, which can be defined for each camera. Events are stored in the Event Archive and can be easily retrieved in real time or on demand for search and analytics. Events are reported promptly as they occur and when they end. Events are updated as a better understanding of what they represent becomes available.

- **For Mode:** Selects which mode is affected by the current settings on this page.

Report Events

Enables event reporting. Event reporting enables SAFR to log and track events over time and gain additional insight into your SAFR system and usage patterns.

- **Include Unrecognizable Events from Camera:** Enable this option to report the appearance of unrecognized people captured by camera feeds. Unrecognized people are people that the SAFR system can't see well enough to compare it to its People Directory.
- **Include Unrecognizable Events from Video:** Enable this option to report the appearance of unidentified individuals captured in video files imported into SAFR. Unrecognized people are people that the SAFR system can't see well enough to compare it to its People Directory.
- **Include Stranger Events:** Enable this option to report when cameras see strangers. Strangers are people who the SAFR system can see well enough to compare him/her to individuals stored in its People Directory, and there isn't a match.

- **Min Age:** The minimum age of strangers that will trigger stranger events. If a stranger younger than the specified minimum age is detected, no stranger event is generated.
- **Max Age:** The maximum age of strangers that will trigger stranger events. If a stranger older than the specified maximum age is detected, no stranger event is generated.
- **Only if occluded:** When enabled stranger events are only reported when the stranger is occluded. This can be useful if you want to catch people who are attempting to bypass your security system by intentionally occluding their faces.
- **Include Speculated Identity Events:** Enables reporting events for speculated people. A “Speculated Identity” is a face that isn’t a 100% match with a face in the Person Directory, but is close.
- **Include Secondary Events:** When enabled, face events that are tied to a person event will be included. Enabling secondary events often produces undesired “noise”, so it is turned off by default.
- **Preserve Event Face Image:** Select this check box if you want the images that trigger an event to be saved with the event report.
 - **Max Image Size:** The maximum size of the event face images, in pixels.
 - **Image Format:** Choose whether you want the event face image saved as a JPG or a PNG.
- **Preserve Event Scene Thumbnail Image:** Select this option if you want a thumbnail of the scene image in which the event occurred to be saved with the event report.
 - **Max Image Size:** The maximum size of the event scene thumbnail images, in pixels.
 - **Image Format:** Choose whether you want the event scene thumbnail image saved as a JPG or a PNG.
- **Include Unrecognizable Event Images:** Specifies if event images should be stored for events triggered by unrecognizable people.
- **Include Stranger Event Images:** Specifies if event images should be stored for events triggered by people who aren’t registered in your Identity Database.
- **Reporting Delay:** The number of seconds an event report is delayed in order to properly assess the nature of the event. For example, a person who may at first seem unknown may become known after a second observation.
- **Min Identified Event Duration:** The minimum duration required for an event representing a known person to be recorded as an event.
 This setting helps filter out noise or brief appearances that may not be worth reporting as a system event.
 If this setting and *Reporting Delay* have different settings, the greater number is used.
- **Min Unrecognizable Event Duration:** The minimum duration of an event representing an unrecognizable person to be recorded as an event.
 If this setting and *Reporting Delay* have different settings, the greater number is used.
- **Min Stranger Event Duration:** The minimum duration of an event representing a stranger to be recorded as an event.
 If this setting and *Reporting Delay* have different settings, the greater number is used.
- **Update in-progress event attributes:** If this is enabled then any event properties that

change will be updated at the specified interval. Many properties do change periodically, such as images or other averages that are continually computed.

- **Update interval:** Specifies the interval time in which to update event properties that change.
- **Update with higher quality image:** Update the thumbnail images with higher quality images during the course of the event if possible.

Listen for Event Replies

- **Display Reply Message:** Enables SAFR to display reply messages on the screen.
- **Display until end of event:** When enabled, the event reply is continuously displayed until the event ends. When this isn't enabled, the event reply is only shown for a couple seconds.
- **Generate auto-replies:** Internal use only. Do not enable this setting.
- **Positive Reply:** Allows you to configure positive replies.
 - **Sound:** Specifies which sound, if any, is played when a positive event occurs.
 - **Voice:** Specifies which voice, if any, will be used.
 - **Background:** The background image to be displayed while a positive reply is being displayed. If no image has been uploaded, then the live view of the video feed is used as the background.
- **Neutral Reply:** Allows you to configure neutral replies.
 - **Sound:** Specifies which sound, if any, is played when a neutral event occurs.
 - **Voice:** Specifies which voice, if any, will be used.
 - **Background:** The background image to be displayed while a neutral reply is being displayed. If no image has been uploaded, then the live view of the video feed is used as the background.
- **Negative Reply:** Allows you to configure negative replies.
 - **Sound:** Specifies which sound, if any, is played when a negative event occurs.
 - **Voice:** Specifies which voice, if any, will be used.
 - **Background:** The background image to be displayed while a negative reply is being displayed. If no image has been uploaded, then the live view of the video feed is used as the background.
- **Reaction Delay:** Delays the event reporting to the server by this amount in seconds.

Remove events

Enables removing of events according to the preferences selected within this section.

- **Remove events:** It's *Enabled* if either anonymous or non-anonymous events are configured to be removed. Click the **Change...** button to configure the event removal.
 - **Remove Anonymous Events after:** Determines how many days to wait before removing events triggered by people without a *name* attribute in the Person Directory. If this value is set to zero, then anonymous events won't be automatically removed.

- **Remove Known Identity Events after:** Determines how many days to wait before removing non-anonymous events. If this value is set to zero, then non-anonymous events won't be automatically removed.
- **Remove identities:** It's *Enabled* if either anonymous or non-anonymous events are configured to be removed. Click the **Change...** button to configure the event removal.
 - **Target Directory:** Determines the directory whose identities are to be automatically removed.
 - **Remove Anonymous Identities after:** Determines how many days to wait before identities that don't have a *name* attribute. If this value is set to zero, then anonymous identities won't be automatically removed.
 - **Remove Identities of person type:** Select the *Person Type* of the identities you'd like removed. If you don't modify this field, then identities of all *Person Types* will be removed.
 - **after:** Determines how many days to wait before removing identities of the specified *Person Type*. If this value is set to zero, then identities with *Person Types* won't be automatically removed.

User Interface Preferences

The User Interface tab allows you to customize your Desktop client's user interface.

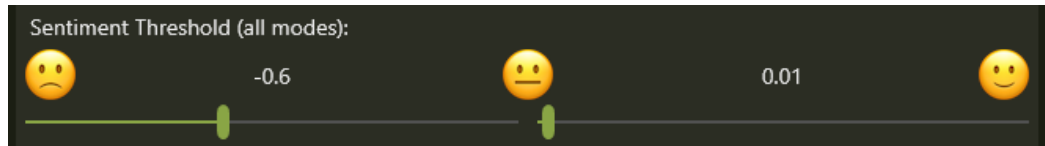
- **For Mode:** Selects which mode is affected by the current settings on this page.

Video

- **Loop playback:** Enable this to loop playback of a video file. This is primarily useful for looping video demos.
- **Display time:** Use the slider to set the number of seconds a newly recognized person's name is flashed on the screen.
- **Minimum name refresh time:** This setting defines how long (in seconds) a recognized person must be out of camera view before their name is flashed again should they reappear in the camera view.
- **Highlight border thickness:** Use the slider to set the thickness (in pixels) of the frame displayed around faces and badges.
- **Overlay text size:** Specifies the size of the text in the video feed overlay.
- **Name display message (#N = Full name, #F = First name, #U = Last name):** Use this option to display a custom message to registered and recognized entrants. Use #N as a placeholder for the name of any recognized person. For example, Welcome, #N would display "Welcome, <recognized person's name>." The message is only displayed to registered persons.
- **Speak name display message:** When enabled, the *Name display message* will be spoken aloud.
- **Average Age and Gender:** During its normal operation, SAFR estimates the age and

gender of people in every frame of a video stream independently. Thus, a person's displayed age can fluctuate by 10 years frame to frame. When this setting is enabled, ages and gender for a detected person are averaged over time, which creates a much smoother and more accurate experience. This setting has no effect on recognized people whose age or gender are specified within the Identity Database; SAFR will always display the stored values.

- **Sentiment Thresholds:** Specifies what range of sentiment values are classified as unhappy, neutral, and happy. The two Sentiment Threshold sliders are arranged as shown below.



- **Background:** Default background image that shows all the time. It can change based on events. If this is left blank, the video feed is shown.

Similar

- **Overlay zoom:** Sets the size of the comparison faces.
- **Maximum faces to select:** Sets the maximum number of faces to compare to the target face.

Registration

- **Enable Registration:** Enable this to allow unknown users to register their faces.
 - **Min Age:** Sets the minimum age that a person must be in order to register themselves. This setting is optional.
 - **Person Type:** Automatically sets the *Person Type* of all people who register at this Desktop client to the specified value.
 - **Home Location:** Automatically sets the *Home Location* of all people who register at this Desktop client to the specified value.

Connect a Face Recognition Panel

A face recognition panel is a mobile device running the Mobile client placed in **Secure Access** or **Secure Access With Smile** video processing mode. It is used at the door (usually placed behind safety glass on the inner side of a door) as part of the SAFR Secure Access setup. A face recognition panel provides an event to SAFR Server that is then picked up by SAFR Actions, which in turn triggers the door unlock action.

Download and Install the Mobile Client

To install the Mobile client, simply go to the SAFR Download portal, download the Mobile client specific to your mobile device's OS, and then run the installer.

iOS devices have additional potential download locations:

- Go to the *Apple App Store* and search for *SAFR Recognition*.
- Using your browser, navigate to *itunes.apple.com/app/id1376830890*.

Note: In local deployments, iOS devices require that the primary SAFR Server have an SSL certificate. See SSL Certificate Installation for instructions on how to do this.

Connect the Mobile Client to a SAFR Server

To connect your Mobile client to a SAFR Server, do the following:

1. Make sure your mobile device is connected to the internet and that it can make a network connection either to the SAFR Cloud (for cloud deployments) or to your SAFR Server (for local deployments).
2. Start the Mobile client.
3. Sign in using your credentials.

- If you have been issued an account for the Cloud environment, enter your user ID and password in the sign-in dialog that appears on the screen.

Note: Make sure the front facing camera of your mobile device has a view of your face when signing in. Your face is not recorded, but it must be detected for sign-in to be offered.

- If you instead have an account for the Partner Cloud environment:
 1. Cancel the sign-in dialog.
 2. Open the Mobile client settings by tapping the gear icon in bottom left.
 3. In the Account tab of the Mobile client settings, change the environment to SAFR Partner Cloud.
 4. Close the settings.
 5. Make sure the front facing camera of your mobile device has a view of your face. Your face is not recorded, but it must be detected for sign-in to be offered.
- Tap the **Sign In** button that appears at the top of the screen.
- Enter your credentials, select the agreement to terms of service check box, and tap **Sign In**.

If successful, the **Sign In** button disappears and a purple frame is displayed around your face with **Tap to Register** displayed underneath.

Configure the Mobile Client as a Face Recognition Panel

To configure the Mobile client as a face recognition panel, do the following:

1. Start the Mobile client.
2. Open the settings menu by tapping the gear icon in bottom left corner of the screen.
3. Tap the video processing mode selector at the top center of the screen, and select either **Secure Access** or **Secure Access With Smile**.
 - **Secure Access** mode generates an event when a person is recognized.
 - **Secure Access with Smile** mode generates an event when a recognized person is observed changing expression from non-smiling to smiling.

Note: When in **Secure Access** or **Secure Access With Smile** mode, video is turned off by default. If you want to show the video, you can override this behavior from the settings menu (gear icon) in the **User Interface** tab.

4. Complete the **User Site** and **User Source** fields.
 - The **User Site** labels the site (e.g. My-Office) at which you are deploying SAFR Secure Access.
 - The **User Source** labels the entrance location (e.g. Front-Door) at which the mobile device is placed.

Note: *Site* and *Source* labels are associated with every registration as well as with every other event and are crucial in making the source of registrations as well as other events traceable.

5. (Optional) Configure the mobile device into Locked Mode to lock in the Mobile client as the exclusive application for the device.

Note: Locking your mobile device locks the phone to your Mobile client and prevents any disruption in the registration kiosk operation due to operating system updates or unauthorized user interference. It isn't necessary to lock your mobile device if you merely want to try out the Mobile client as a registration kiosk. However, you should lock the device before deploying the registration kiosk in a production environment.

Connect a Registration Kiosk

A SAFR registration kiosk is a mobile device running a Mobile client that has been placed in Registration Kiosk mode. It is used to take pictures of users and enable them to register their faces and identity information with the SAFR system.

Download and Install the Mobile Client

To install the Mobile client, simply go to the SAFR Download portal, download the Mobile client specific to your mobile device's OS, and then run the installer.

iOS devices have additional potential download locations:

- Go to the *Apple App Store* and search for *SAFR Recognition*.
- Using your browser, navigate to *itunes.apple.com/app/id1376830890*.

Note: In local deployments, iOS devices require that the primary SAFR Server have an SSL certificate. See *SSL Certificate Installation* for instructions on how to do this.

Connect the Mobile Client to a SAFR Server

To connect your Mobile client to a SAFR Server, do the following:

1. Make sure your mobile device is connected to the internet and that it can make a network connection either to the SAFR Cloud (for cloud deployments) or to your SAFR Server (for local deployments).
2. Start the Mobile client.
3. Sign in using your credentials.

- If you have been issued an account for the Cloud environment, enter your user ID and password in the sign-in dialog that appears on the screen.

Note: Make sure the front facing camera of your mobile device has a view of your face when signing in. Your face is not recorded, but it must be detected for sign-in to be offered.

- If you instead have an account for the Partner Cloud environment:
 1. Cancel the sign-in dialog.
 2. Open the Mobile client settings by tapping the gear icon in bottom left.
 3. In the Account tab of the Mobile client settings, change the environment to SAFR Partner Cloud.
 4. Close the settings.
 5. Make sure the front facing camera of your mobile device has a view of your face. Your face is not recorded, but it must be detected for sign-in to be offered.
- Tap the **Sign In** button that appears at the top of the screen.
- Enter your credentials, select the agreement to terms of service check box, and tap **Sign In**.

If successful, the **Sign In** button disappears and a purple frame is displayed around your face with **Tap to Register** displayed underneath.

Configure the Mobile Client as a Registration Kiosk

Do the following:

1. Start the Mobile client.
2. Open the settings menu by tapping the gear icon in bottom left corner of the screen.
3. Tap the mode selector at the top center of the screen, and select **Registration Kiosk**.
4. Complete the **User Site** and **User Source** fields.
 - The **User Site** identifies the site (e.g. My-Office) at which you are deploying the SAFR System.
 - The **User Source** identifies the registration kiosk (e.g. Registration-Kiosk) as the source of registrations. **Note:** *Site* and *Source* labels are associated with every registration as well as with every other event and are crucial in making the source of registrations as well as other events traceable.
5. (Optional) Configure the mobile device into Locked Mode to lock in the Mobile client as the exclusive application for the device.

Note: Locking your mobile device locks the phone to your Mobile client and prevents any disruption in the registration kiosk operation due to operating system updates or unauthorized user interference. It isn't necessary to lock your mobile device if you merely want to try out the Mobile client as a registration kiosk. However, you should lock the device before deploying the registration kiosk in a production environment.

Register and Organize SAFR Users in your System

Although users can self-register their face at a registration kiosk, they are not automatically registered and approved in the system or granted access privileges. SAFR administrators can classify and control access to resources by using the Person Directory to assign various categories and tags to registrants. For more information on searching, viewing, and organizing registrants, see *Manage People* in the Person Directory.

For example, you can require every registrant to be assigned a **Person Type** property and base access to certain resources on that property. Think of **Person Type** as a category for your users, such as Staff, Maintenance, Administrator, or anything else you might like to define. The **Home Location** and **Person Type** properties associated with registrants can be adapted to different needs for different organizational purposes. You can also use the **Home Location** and **Person Type** properties to filter information. For example, in a school setting you might use **Home Location** to denote the grade of a student, and **Person Type** might be

defined as *student*.

Click **Add Home Location** or **Add Person Type** to add new options or choose from the existing ones. Existing options appear as options in the menu.

Best Practices for Organizing your SAFR Registrants

- You can create and customize as many **Person Types** and **Home Location** as you like, but we recommend keeping the number of defined values to less than a dozen or so for each property, for ease of maintenance.

Customize a Registration Kiosk

Each registration kiosk can be customized to prompt for additional required or optional information from the registrant. You can also customize:

- The registration prompt.
- Registration completion message.
- The default **Person Type** or **Home Location** for the registrant.
- A minimum age requirement for registrants (estimated based on the registrant's face).

Customize the Registration Prompt

To customize the registration prompt, do the following:

1. In the Mobile client, tap the gear icon (settings) > **User Interface**.
2. Enter a new text next to **Prompt**.

Assign Default **Person Type** or **Home Location** Values

It may be desirable to assign a default **Person Type** or **Home Location** value to all registrants who complete registration at a particular registration kiosk. For example, if a registration kiosk is located in the admissions office, anyone registered there could be assigned the **Person Type** of *Student* or perhaps *Employee*. Anyone registered at the registration kiosk placed at a specific location could be given a default **Home location** corresponding to the town in which the registration kiosk is located. This can save administrative time. Both **Person Type** and **Home Location** can be changed by the administrator after the registration when needed.

To configure the default **Person Type** or **Home Location**:

1. In the Mobile client, tap the gear icon (settings) > **User Interface**.
2. Enter a value for **Person Type** if desired. By default, **Person Type** is not assigned.
3. Enter a different value for **Home Location**. By default, **Home Location** is set the same as the *Site* label specified in the **Account** settings.

The **Home Location** field associated with every person registered can be used for various purposes. For example, in a school settings, it could be used by the administrator to enter the building name in which a student's home classroom is located. **Home Location** and **Person Type** fields offer filtering based on labels used for these fields and can become important organizational tools. They are named generically to allow labels to be created on the fly by simply entering them. You should decide how to use these labels and then use them consistently to get the most value from them.

Note: As a best practice, neither of these fields should have more than two dozen labels for ease of use.

Restricting Registration to a Minimum Age

It may be desirable to prevent registration of people below a certain age. The Mobile client can be configured to assess a person's age and not offer registration to people below a specified minimum age.

To configure the minimum registration age:

1. In the Mobile client, tap the gear icon (settings) > **User Interface**.
2. Enter the desired value for **Min Age**.
3. (Optional) Change **Show Attributes** to *Off*.

Tip: Switching **Show Attributes** to *Off* prevents displaying the assessed age to the registrant. Because some people may be sensitive to this feedback, it is recommended that age not be shown.

4. On the **Recognition** tab, change **Detect Age** to *On*. With age detection set to *On*, the restriction is now active.

Customize the Registration Form

To customize the message your kiosk displays to registrants:

1. In the Mobile client, tap the gear icon (settings) > **User Interface** > **Form: Customize**.
2. For any fields you want to add to your form, change the *Hidden* indicator to either *Required* or *Optional*. Any field that is marked as *Required* needs to be filled out by the registrant before registration is allowed to be complete.
 - The *Name*, *Company*, *Mobile*, and *Email* fields have fixed meanings. While you can customize prompt names for these fields, information entered for these fields is registered under the prescribed meaning. If you do not want to have this information gathered during registration, keep these fields hidden. Do not re-label them to a different meaning.

Note: *Name* cannot be hidden and must be entered by the registrant.

- If you need to gather information in addition to these prescribed fields, use the generic fields labeled by default as *Field*. These form entries have no prescribed meaning. Any information provided through these fields appears as tags in the registered person's record. If you want to give the entered information a tag name, complete the *Tag* field for each entry. If *Tag* is completed, information the registrant fills out for this field is prefixed with "Tag=" when appearing in person's record (e.g. Car Make=Ford). If the tag is not filled out, the information provided by the registrant appears on its own in the list of tags associated with the registered person.
3. Enter the names for the fields and add any information placeholder text. (e.g. "Type Your Name Here")
 4. Change the labels for the actions buttons if desired.
 5. Enter the completion message displayed once the registration process is successfully completed.

Configure a Mobile Device into Locked Mode

Single App Mode for iOS, or Lock Task Kiosk Mode for Android, allow you to lock an iOS or Android device into a single application. When enabled, the mobile device is restricted to running only one application even if it is rebooted. This mode allows the device to be fully locked from any unauthorized access, and it will remain locked until Single App or Lock Task Mode is explicitly disabled.

You can control how users interact with devices using Single App and Lock Task Modes by enabling or disabling any of the following features:

- Screen auto-lock
- Touch input
- Screen rotation
- Volume control
- Sleep/wake button
- Side switch

Warning: Putting an iOS device in Supervised Mode wipes all the information on the device and resets it. Likewise, when using an Android device, you must return the target device to factory settings which causes all information to be wiped from the device resets it.

Requirements

- For macOS, you'll need a Macintosh computer running 10.14 Mojave or later.
- For Android, you'll need 2 Android devices to set up the most secure mode, Lock Task Mode. If you only have a single Android device, then you can only set up the less secure

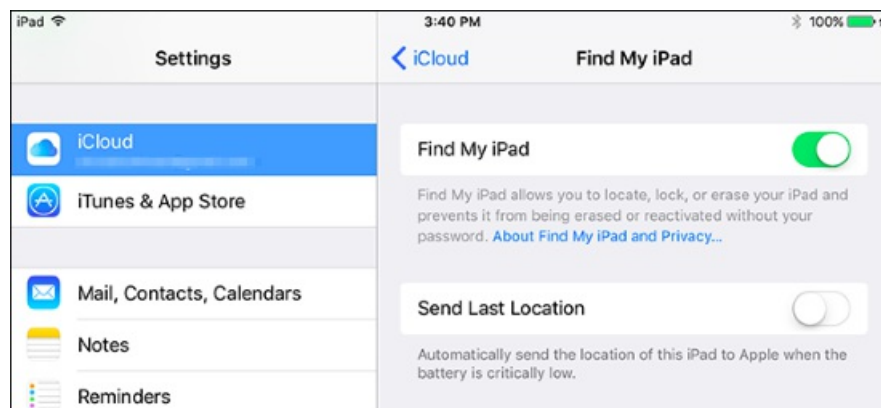
Screen Pinning Mode.

Put an iOS Device into Supervised Mode

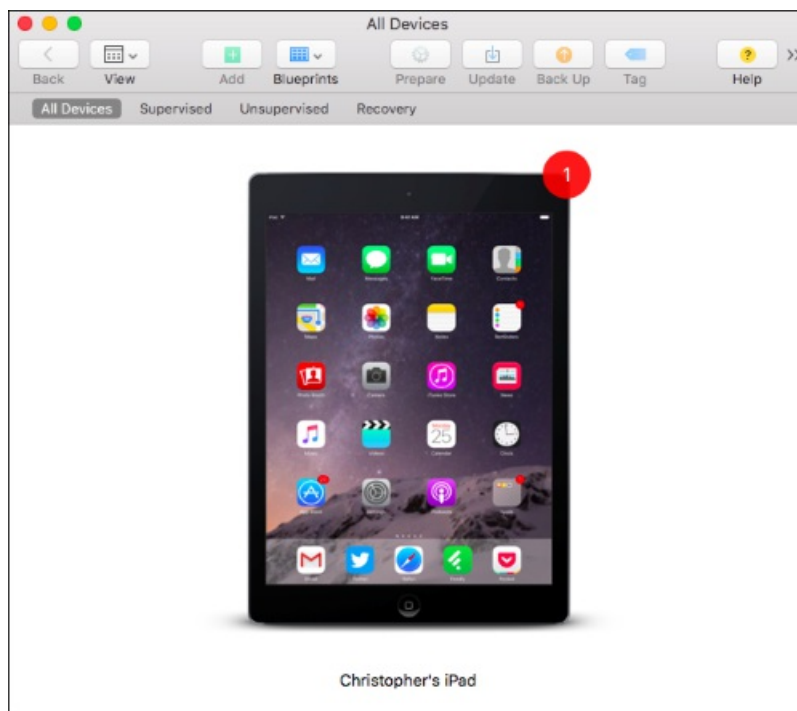
While the procedure described here manually puts a mobile device into Supervised Mode, there are other ways to do this via mobile device management (MDM).

To put an iOS device into Single App Mode, the device must first be put into Supervised Mode. To do this, do the following:

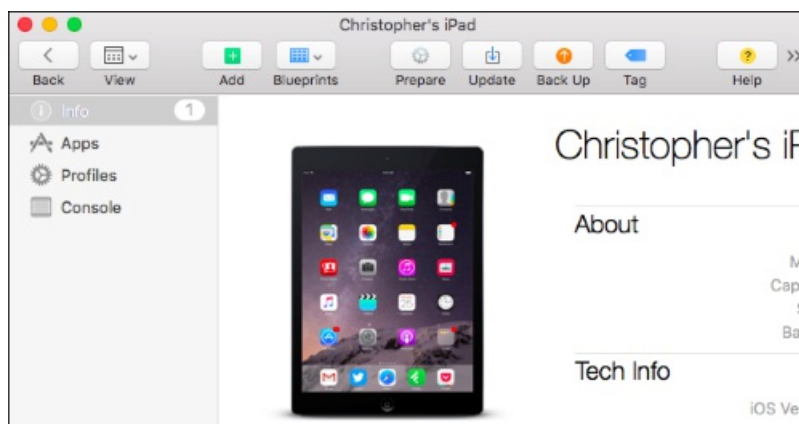
1. Go to **Settings > (User) > iCloud > Find My iPad/iPhone**. Disable the **Find My iPad/iPhone** switch by entering the password.



2. On your Mac, launch the **App Store** application and search for **Apple Configurator 2**. Download and install this application on the computer.
3. Plug in your iOS device to your Mac.
4. Launch **Apple Configurator 2**. You should see something that looks like the image below.



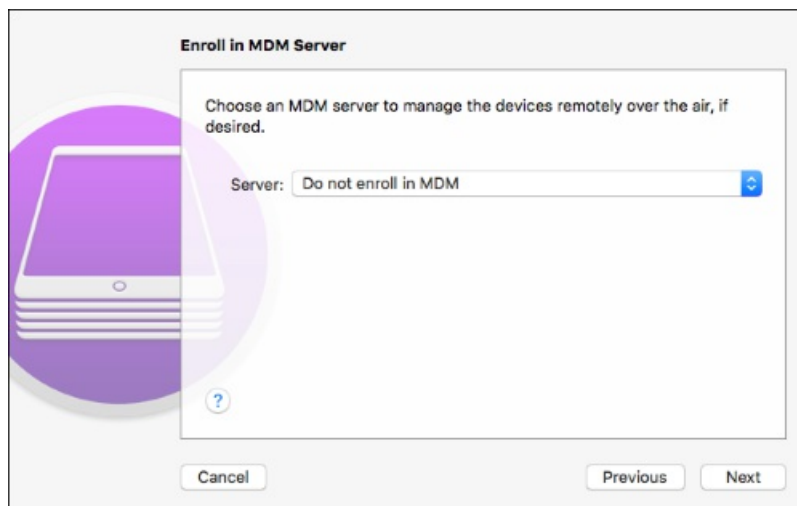
5. Double-click the device.
6. On the **Details** screen about the device, click the **Prepare** button.



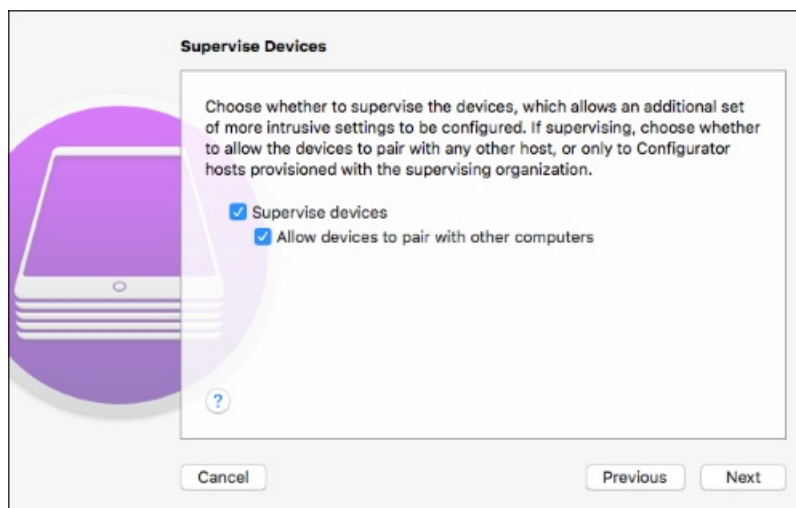
7. From the **Configuration** menu, select **Manual**.



8. From the **Server** menu, select **Do Not Enroll in MDM** unless you have an MDM server you want to use and enroll your device to.



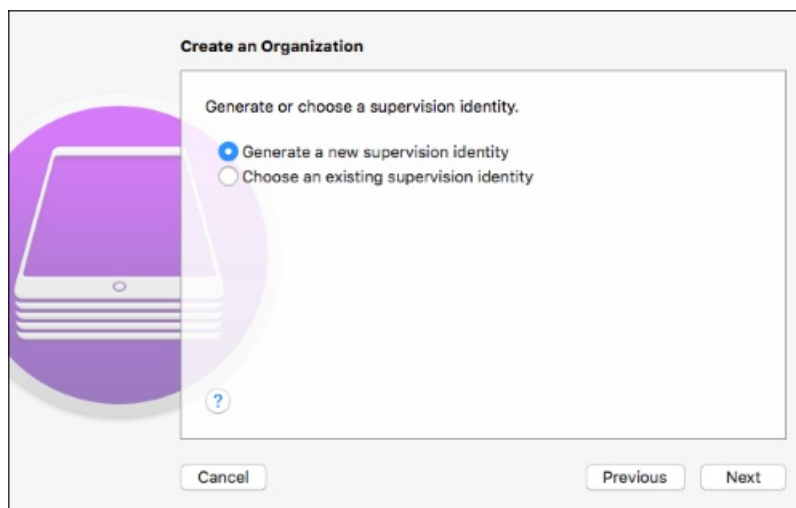
9. If you selected **Do Not Enroll**, you must now plug the mobile device into your Mac to configure it.
10. Click the **Supervise Devices** check box. If you want the device to be configured on multiple computers leave the default **Allow Device to Pair with Other Computers** selected.



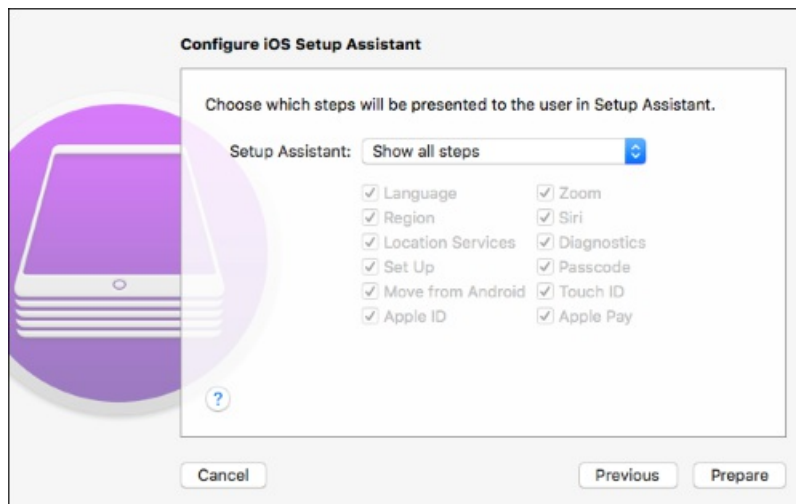
11. Enter your organization information.

A screenshot of a software window titled "Create an Organization". On the left is a purple circular icon containing a white graphic of a tablet and a stack of papers. The main text area contains the instruction: "Enter information about the organization." Below this are four input fields: "Name:" (containing "How-to Geek"), "Phone:", "Email:", and "Address:". A small blue question mark icon is at the bottom left of the text area. At the bottom of the window are three buttons: "Cancel", "Previous", and "Next".

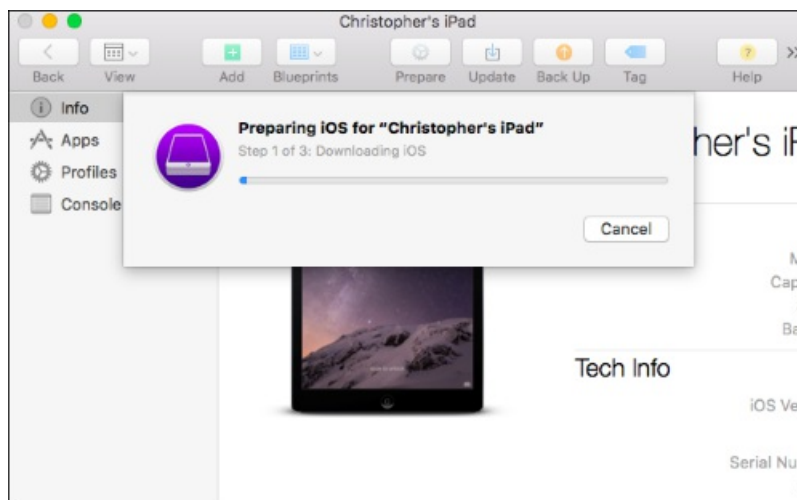
12. If you've previously generated a supervision identity at some point, select **Choose an Existing Supervision Identity**. Otherwise, you'll need to generate one by selecting **Generate a New Supervision Identity**.



13. Select the options you want the device to run after it is reset. The default options are generally sufficient.



14. Click **Prepare**. A status bar will be displayed as the iOS device is configured in supervised mode. **WARNING:** Clicking the **Prepare** button wipes all information on the device and resets it.



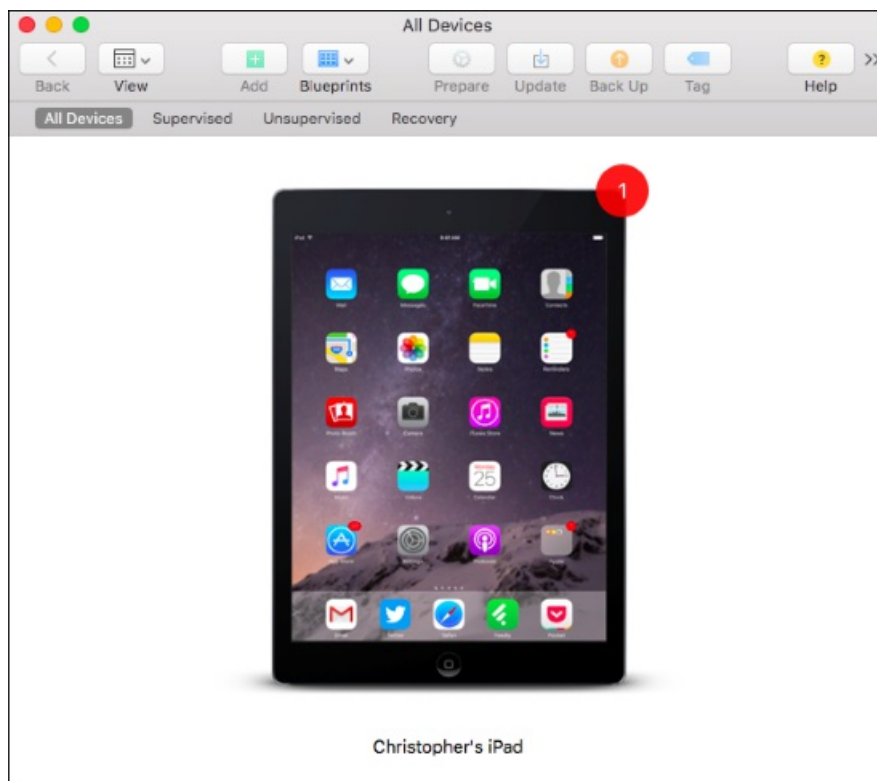
After the device is wiped and rebooted it will be running in supervised mode.

Enable Single App Mode

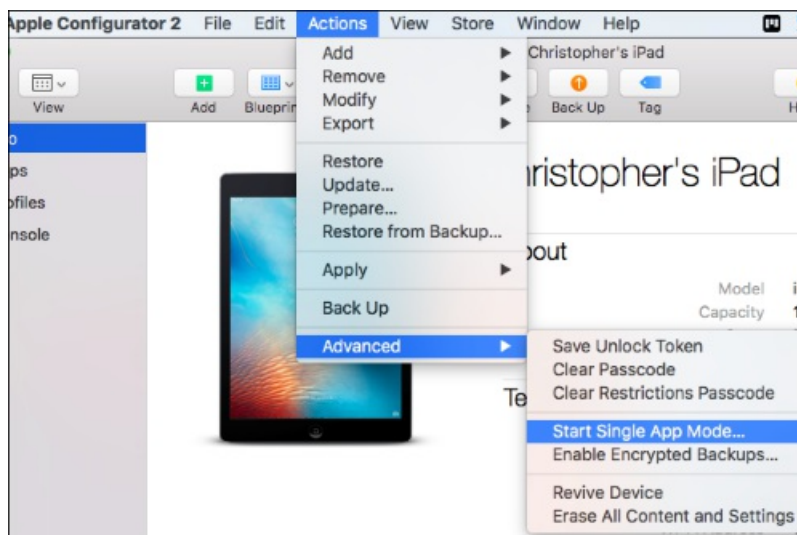
Note: To continue from this point, the iOS device should be in supervised mode. If the iOS device is not in supervised mode, repeat the instructions from the prior section first to put it in supervised mode.

To enable or disable Single App Mode, do the following:

1. On your Mac running 10.14 or greater Mojave, launch the *App Store* application and search for *Apple Configurator 2*.
2. Download and install *Apple Configurator 2* to your Mac.
3. Plug in your iOS device to your Mac computer.
4. Launch *Apple Configurator 2*. You should see something that looks like the image below. Double-click the device.



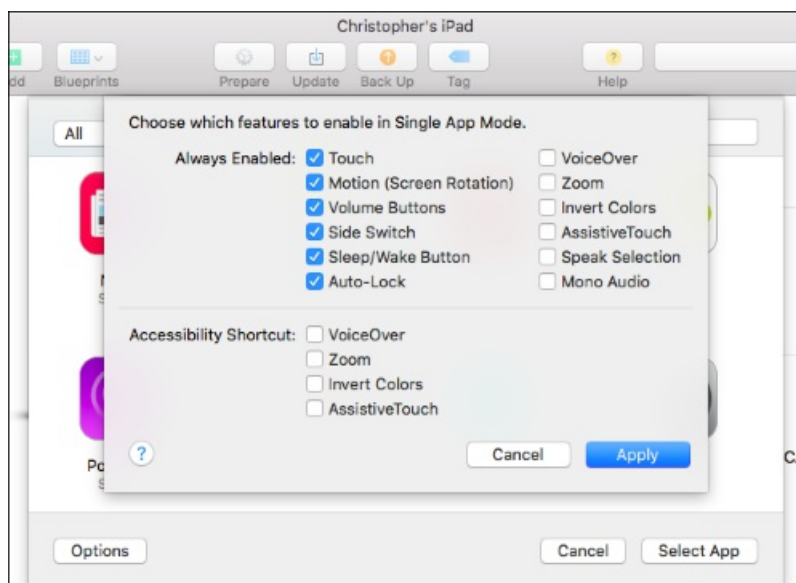
5. On the **Device Details** screen, from the **Actions** menu, click **Advanced > Start Single App Mode**.



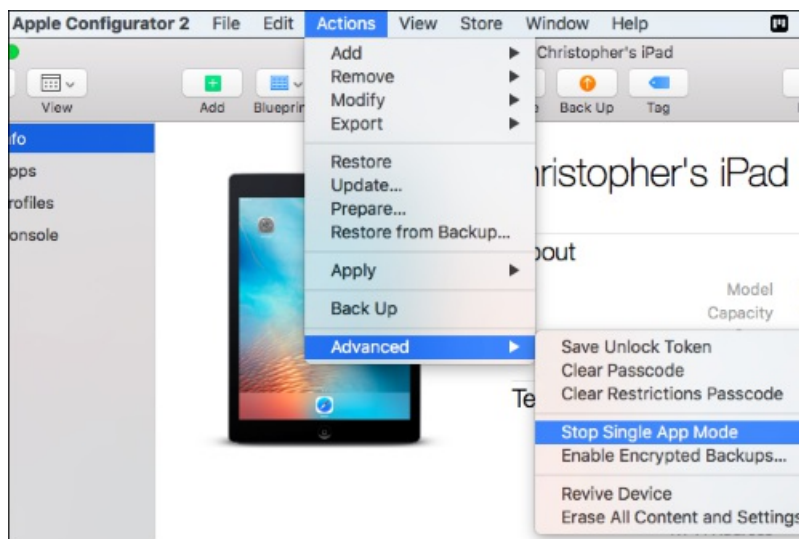
6. Select SAFR from the list of applications.
7. Click the **Select App** button when you're ready to launch SAFR. The iOS device is now locked in Single App Mode.



8. OPTIONAL: If you want to configure advanced options, click **Options**. From the dialog, select the options you want enabled, and click **Apply**. However, usually the defaults are sufficient.



9. When you return to the applications screen, click the SAFR application and click **Select App**.
10. To disable Single App Mode, plug the iOS device into the computer. In the **Actions** menu, click **Advanced > Stop Single App Mode**.



Enable Kiosk Mode for Android

There are two kiosk modes available in the Android Mobile client:

- Lock Task mode (LTM): A robust kiosk mode where only administrators are able to alter the configuration or access the data on the device. The device is locked into one application until the mode is explicitly disabled. You must install the Mobile client using SAFR Beam to use this mode.
- Screen Pinning mode (SPM): A less secure kiosk mode without device administrator registration. When using the device you can exit the mode at any time. Available for any Android device with the Mobile client installed.

Note: While this procedure explains how to manually set up a device using SAFR Beam, you can also use the Android Debug Bridge (ADB) command line tool.

To set up and enable Lock Task mode:

1. Go to the SAFR download portal and from the menu, select Android.
2. Install SAFR Beam on your primary device.
3. Set your target device in factory reset prior to use.
4. Follow the instructions on the primary device for installing the Mobile client on a target device.
5. Once the Mobile client is installed on the target machine, click the lock icon next to the settings gear icon. Follow the instructions for setting the device up for Lock Task mode.
Note: In this mode, the client has full control over the device and only the client can request exiting the mode.
6. Exiting can be done by tapping the screen three times (3-taps gesture) which displays the system's security dialog. (assuming that one has been configured) In the dialog, you are prompted to confirm your identity by entering the device's credentials (PIN, gesture, or fingerprint). If the device does not have security settings in place or your identity is

confirmed, the Mobile client restarts in an unlocked state.

Important: You should configure device security either with a PIN, a gesture, or a fingerprint. That way, if a device is turned off while the Mobile client is locked (either by the power button or as the result of drained battery), only a credible user is able to start the device and re-run the Mobile client. When re-run, the Mobile client enters the mode it was in prior to turning off the device.

Note: If you install the Mobile client apart from SAFR Beam, you can still set up security by clicking the lock icon. However, because the Mobile client has not been registered as a device administrator, its security is not as strong as the Lock Task mode.

The following scenarios occur when using the kiosk modes when the Mobile client is or is not registered as a device manager:

Scenario	Action
No device security configured (not registered); you confirm to enter SPM on the security dialog	Exits via 3-taps gesture, or by holding the Recents and Back keys at the same time; the Mobile client is restarted in unlocked state (Screen Pinning mode)
No device security configured (not registered); you deny entering SPM on the security dialog	The Mobile client is in locked state but is restarted in unlocked state after approximately ten (10) seconds; a timer is triggered that queries for locked state and corrects it if needed
PIN device security configured (registered); you confirm to enter SPM on the security dialog	Exit by 3-taps gesture or by holding the Recents and Back keys at the same time; SAFR prompts you to confirm your identity by entering PIN and if successful, it is restarted in unlocked state

Note: On some devices, SPM can be explicitly enabled in system's setting with an option to ask for a PIN upon unlocking/PIN device security configured. If you confirm to enter SPM on the system dialog by exiting by holding the Recents and Back keys at the same time, you are prompted to confirm your identity by entering PIN. If successful, the device home screen is displayed. The next time, SAFR restarts in an unlocked state.

Install SAFR Beam

Install the SAFR Beam for Android utility onto one device and use this primary device to install the Mobile client in a Lock Task kiosk mode on a second target device. Using SAFR Beam provides added security to the target device, locking it down in cases where added security is required. For example, using the device camera to identify employees and open secured door to them. For more information, see Configure Devices into Locked Mode section.

To Install and Use SAFR Beam

1. Secure two Android devices capable of running SAFR. One device serves as the primary

and the other as the target. For more information, see SAFR System Requirements.

2. Log into the SAFR download portal and install SAFR Beam on the primary device.
3. On the primary device, turn on Near Field Communication (NFC). Make sure the target device has NFC capabilities.
4. Reset the target to its factory settings.
5. Place the target device back to back with the primary device.
6. Once the target device is detected, tap the screen on the primary device to start the beam.
7. Follow the instructions on the target device to complete the installation.
8. Although not required, we highly recommend that you set up security access on the target device. (e.g. a PIN or gesture)
9. Run the Mobile client on the target device. If prompted, set SAFR as the default launcher app.

Mobile Account Preferences

The Account preferences tab is where you configure your organization's SAFR accounts and related information, such as the directory for your facial recognition database.

- **Environment:** Determines which operating environment your client contacts. The possible values for this field are as follows:
 - *SAFR Developer Cloud:* Internal use only.
 - *SAFR Partner Cloud:* Internal use only.
 - *SAFR Cloud:* Used for cloud deployments. This is a general availability SAFR Server in the cloud maintained by RealNetworks. It is a stable, high availability environment intended for production use.
 - *SAFR Custom:* Used for local deployments. If you select *SAFR Custom*, you will be asked to provide the URLs for the primary SAFR Server services.
- **User Identifier:** The account can have multiple user identifiers with different access privileges.
- **User Password:** The password for the user entered in the *User Identifier* field.
- **User Directory:** The directory in the account where the data used for facial recognition is stored.
- **User Source:** The *User Source* label for this mobile device. All SAFR event data is tagged by site and source labels. These labels are used to help filter and analyze collected recognition events, such as where a face was recognized.
- **User Site:** The *User Site* label for this mobile device. All SAFR event data is tagged by site and source labels. These labels are used to help filter and analyze collected recognition events, such as where a face was recognized.
- **Enable Active Camera Connect:** Only available on Android devices. When enabled, the mobile device's connected rtsp:// camera will continue to be processed even when the Mobile client is in the background or when the mobile device is asleep.
- **Report Status:** Enables a preview of the video stream in the video feed status window. The feed view is a simple low frame-rate stream (1 frame per second). It is only intended

for inspecting camera orientation and lighting conditions. It is not intended for actively monitoring feeds for security purposes.

- **Allow Remote Viewing:** Enables remote monitoring for your mobile device's video feed.

Mobile Detection Preferences

Use detection preferences to enable and configure facial detection characteristics.

- **Enable Face Detection:** The check box must be selected to enable face detection.
- **Min Searched Face Size:** Defines the minimum face size that can be detected. A searched size of 80, for example, can still manage to detect faces as small as 60x60, but with lower certainty. Lowering this number enables SAFR to detect much smaller faces but also greatly increases CPU usage.
Note: This setting does not impact face recognition accuracy.
- **Min Required Face Size:** Defines the minimum required size for a face to be detected. Any face smaller than the height or width is ignored.
- **Generate Recognizer Hint:** Optimizes facial recognition. It should be turned on for most cases. If it is turned off, recognition accuracy may be reduced if detection is performed at very low resolutions.

Mobile Recognition Preferences

Use Recognition preferences to adjust the range for a variety of settings that determine whether or not SAFR detects, tracks, and recognizes faces and identities.

- **Detect Identity:** Select to enable identity detection.
- **Detect Gender:** Select to enable gender detection.
- **Detect Age:** Select to enable age detection.
- **Detect Sentiment:** Select to enable sentiment detection.
- **Detect Smile Action:** Select to enable smile detection.
- **Pre-smile Delay (seconds):** The amount of time that there should be no smile.
- **Smile Duration (seconds):** The amount of time that the smile should last.
- **Identity Threshold Boost:** The smile threshold to boost temporarily during the smile action.
- **Minimum Recognition Face Size (pixels):** Defines the minimum required face size in pixels to attempt recognition. It includes a 25% margin around the face.
- **Minimum Learning Face Size (pixels):** Defines the minimum required face size in pixels to enable SAFR to store a reference image for a new identity. It includes a 25% margin around the face.

Mobile Events Preferences

Use the Events preferences tab to configure event reporting as well as how your client listens for event replies.

- **Report Events:** Enables event reporting. Event reporting enables SAFR to log and track events over time and gain additional insight into your SAFR system and usage patterns.
- **Include Unrecognizable Events:** Enable to report the appearance of unrecognizable people captured by camera feeds. Unrecognized people are people that the SAFR system can't see well enough to compare it to its Person Directory.
- **Include Stranger Events:** Enable this option to report when the appearance of strangers. Strangers are people that the SAFR system can see well enough to compare to individuals stored in the People Directory, but for whom there isn't a match.
 - **Min Age:** The minimum age of strangers that will trigger stranger events. If a stranger younger than the specified minimum age is detected, no stranger event is generated.
 - **Max Age:** The maximum age of strangers that will trigger stranger events. If a stranger older than the specified maximum age is detected, no stranger event is generated.
- **Include Speculated Identity Events:** Enables reporting events for speculated people. A "Speculated Identity" is a face that isn't a 100% match with a face in the Person Directory, but is close.
- **Preserve Event Face Image:** Enable if you want the images that trigger an event to be saved with the event report.
- **Preserve Event Scene Thumbnail Image:** Enable if you want a thumbnail of the scene image in which the event occurred to be saved with the event report.
- **Reporting Delay:** The number of seconds an event report is delayed in order to properly assess the nature of the event. For example, a person who may at first seem unknown may become known after a second observation.
- **Min Identified Event Duration:** The minimum duration required for an event representing a known person to be recorded as an event.

This setting helps filter out noise or brief appearances that may not be worth reporting as a system event.

If this setting and *Reporting Delay* have different settings, the greater number is used.
- **Min Unrecognizable/Stranger Event Duration:** The minimum duration of an event representing an unrecognizable person to be recorded as an event.

If this setting and *Reporting Delay* have different settings, the greater number is used.
- **Min Stranger Event Duration:** The minimum duration of an event representing a stranger to be recorded as an event.

If this setting and *Reporting Delay* have different settings, the greater number is used.
- **Listen For Event Replies:** Select to enable listening for event replies. Listening for event replies enables the client to display reply messages on the screen.
- **Display Reply Message:** Select to enable the display of reply messages on the screen.
- **Reaction Delay:** Delays the event reporting to the server by the specified number of seconds.

Mobile User Interface Preferences

The User Interface preferences tab is where you can customize your user interface.

- **Enable Registration:** Select to enable unknown users to register their faces.
- **Min Age:** The minimum age for unknown users to register their own faces.
- **Highlight Border Thickness:** Use the slider to set the thickness (in pixels) of the frame displayed around faces.
- **Overlay Text Size:** Specifies the size of the text in the video feed overlay.

Web Console

The Web Console provides administrators and operators web-based access to the SAFR system. It allows you to make changes to your account, manage the Person Directory, view events in the Events Archive, manage video feeds, and generate reports.

Access the Web Console with a Cloud Deployment

To access the Web Console with a cloud deployment, do the following:

1. Go to the **Products** tab of the [SAFR Download Portal](#).
2. Click on the **System Console** link located under the first listed product, **SAFR Cloud**.
3. Log in using your SAFR Cloud Account credentials.

It's also possible to go straight to the Web Console login page.

- For the SAFR Partner Cloud environment, the login page is located at <https://safr.int2.real.com/console>.
- For the SAFR Cloud environment, the login page is located at <https://safr.real.com/console>.

Access the Web Console with a Local Deployment

To access the Web Console with a local deployment, do the following:

If you're on the same machine as your primary SAFR Server:

1. Open a web browser.
2. Go to either <http://localhost:8090/> or <http://localhost:8091/>.
3. Sign in using your SAFR Local Account credentials.

If you're on any machine other than your primary SAFR Server:

1. Open a web browser.
2. Go to either `http://<ServerIP>:8090` or `http://<ServerIP>:8091`, where

<ServerIP> = the IP address of your primary SAFR Server.

3. Sign in using your SAFR Local Account credentials.

Status Page

The Status page includes general system, directory, and licensing information. It also allows you to set a deadline for event removal and to set the system's display language.

The screenshot shows the SAFR Status Page. At the top, there is a navigation bar with the SAFR logo (with 'by realnetworks' below it) and links for Status, People, Events, Video Feeds, and Reports. A user profile icon is on the right. The main content area is divided into sections: General, Usage Summary, Configuration, and License Information. The General section shows Environment: SAFR Cloud, Tenant ID: markproddcloud, User Directory: main, and Display Language: English. The Usage Summary section shows various metrics like Number of People, Faces, Sites, Sources, and Feeds, all at 0, along with Load and Latency. The Configuration section has checkboxes for Event removal, Identity removal, Identity synchronization, SMTP Email Service, and SMS, with sub-options and buttons for reconfiguration. The License Information section shows Expiration Date: Never, Max Feeds per Hour: Unlimited, and Max Faces: Unlimited.

General	
Environment:	SAFR Cloud
Tenant ID:	markproddcloud
User Directory:	main
Display Language:	English

Usage Summary	
Number of People:	0
Number of Faces:	0
Number of Sites:	0
Number of Sources:	0
Number of Feeds:	0
Load:	0.0 recognitions/second
Latency:	0 ms

Configuration	
<input checked="" type="checkbox"/> Set up Event removal	
Remove Anonymous Events after 30.00 Days	
Remove Known Identity Events after 30.00 Days	
Reconfigure Event Removal	
<input checked="" type="checkbox"/> Set up Identity removal	
Target Directory: main	
Remove Anonymous Identity after 30.00 Days	
Remove Identities of All person types after 30.00 Days	
Reconfigure Identity Removal	
<input type="checkbox"/> Set up Identity synchronization	
<input type="checkbox"/> Set up SMTP Email Service	
<input type="checkbox"/> Set up SMS	

License Information:	
Expiration Date:	Never
Max Feeds per Hour:	Unlimited
Max Faces:	Unlimited

General

- **Environment:** Environment associated with the user's account. There are two possible values for this field:
 - *SAFR Cloud:* A SAFR Server in the cloud maintained by RealNetworks. Cloud deployments use this environment.
 - *SAFR Local:* A locally installed SAFR Server that the user maintains. Local deployments use this environment.
- **Tenant ID:** The name of the person currently logged in.
- **User Directory:** User directory where the user's data is stored. The default value for this is main.

- **Display Language:** Language used by SAFR.

Usage Summary

- **Number of People:** Number of people currently registered.
- **Number of Faces:** Number of faces currently stored in SAFR's database.
- **Number of Sites:** Number of defined sites. A site can consist of one or more cameras, although usually it consists of multiple cameras.
- **Number of Sources:** Number of defined sources. A source can consist of one or more cameras, although usually it consists of a single camera.
- **Number of Feeds:** Number of feeds currently running across the SAFR system.
- **Load:** Number of recognition attempts every second across all video feeds that are currently active in your SAFR system.
- **Latency:** Number of milliseconds it takes for your SAFR Server to generate a response after it receives a recognition request from a client.

Configuration

- **Set up Event removal:** Enables removing of events after the specified time interval.
 - **Remove Anonymous Events after:** Determines how many days to wait before removing events triggered by people without a *name* attribute. Floating point numbers are valid. If this value is set to zero, then anonymous events won't be automatically removed.
 - **Remove Known Identity Events after:** Determines how many days to wait before removing non-anonymous events. Floating point numbers are valid. If this value is set to zero, then non-anonymous events won't be automatically removed.
- **Set up Identity removal:** Enables removal of identities after the specified time interval.
 - **Target Directory:** Determines the directory whose identities are to be automatically removed.
 - **Remove Anonymous Identity after:** Determines how many days to wait before removing identities that don't have a *name* attribute. Floating point numbers are valid. If this value is set to zero, then anonymous identities won't be automatically removed.
 - **Remove Identities of person type:** Select the *Person Type* of the identities you'd like removed. If you don't modify this field, then identities of all *Person Types* will be removed.
 - **after:** Determines how many days to wait before removing identities of the specified *Person Type*. Floating point numbers are valid. If this value is set to zero, then identities with *Person Types* won't be automatically removed.
- **Set up Identity synchronization:** Enables the identity synchronization feature. When enabled and configured correctly, your Person Directory will sync with another Person Directory. The Person Directory that you're syncing with can belong to another SAFR system, or it can belong to a different user directory within your own SAFR system.

Selecting the *Set up Identity synchronization* box causes the following pop-up window to appear:

Set up Identity synchronization

Host identity directory

* User directory name:

☒ Only sync identities with the following attributes

Person type:

Id-Classes:

Host connection

Host address:

Host port:


Host User Id:

Host password:

Apply Cancel

- **User directory name:** The name of the user directory that you're trying to sync identities with.
- **Only sync identities with the following attributes:** When selected, it causes only identities with the specified attributes to be synced.
 - **Person type:** The *Person types* that identities must have to be synced.
 - **Id-Classes:** The *Id Classes* that identities must have to be synced.
- **Host address:** The IP address or the hostname of the target host machine.
- **Host port:** The port number that the target machine's CoVi server listens on.
- **Host User Id:** The *User Id* of somebody who has the credentials to log into the host machine.
- **Host password:** The *Password* of somebody who has the credentials to log into the host machine.
- **Set up SMTP Email Service:** Enables SAFR's actions to send emails. Before you can configure SAFR to send emails, make sure you obtain an SMTP server account that you can use to send emails.

When you click on *Set up SMTP Email Service*, a dialogue will pop up requesting configuration information.



Set up SMTP Email Service

* Email Server:

smtp.gmail.com


* Server Port:

587


* Sender Email:

* Password:

••••••••



From Email Address:



Sender Name:

Test Email

To Email:

Subject:

Body:

Apply

Cancel

- **Email Server:** The address of the SMTP email server.
- **Server Port:** The email server port. The default port for SMTP is 587.
- **Sender Email:** The email username of the SMTP account. (e.g. me@gmail.com)
- **Password:** The password for the SMTP account.
- **From Email Address:** The email address that will appear on the “From” line. This feature isn’t supported by all email servers; if this field isn’t used then the *Sender Email* value is used for the “From” line.
- **Test Email:** Configure the test email that will be sent after you finish setting up the SMTP email service.
 - **To Email:** The email address to which the test email will be sent.
 - **Subject:** The test email’s subject.
 - **Body:** The test email’s body.
- **Set up SMS:** Enables SAFR’s actions to send short message service (SMS) messages. Before you can set up SMS, you must first set up an [AWS account](#) which is configured for your region so it can send SMS messages.

When you click on *Set up SMS*, a dialogue will pop up requesting configuration information.

×

Set up SMS

* SMS Provider: Amazon SNS ▼

* Amazon SNS Sender Id:

* Amazon SNS Access Key:

* Amazon SNS Secret Key: 🔑

* Amazon SNS Region:

Test Message

To Phone Number: ?

Message:

Apply Cancel

- **SMS Provider:** The SMS provider that you're using. This value will always be Amazon SNS.
- **Amazon SNS Sender Id:** The name that will be used to send the SMS notifications.
- **Amazon SNS Access Key:** Your Amazon SNS Access Key.
- **Amazon SNS Secret Key:** Your Amazon SNS Secret Key.
- **Amazon SNS Region:** The region of your Amazon SNS.
- **Test Message:** Configure the test message that will be sent after you finish setting up SMS.
 - **To Phone Number:** The phone number to which the test message will be sent.
 - **Message:** The text message that will be sent to the phone number specified above.

License Information

Shows the operating limits of your SAFR license.

- **Expiration date:** The date when the SAFR license expires. After this date, SAFR software discontinues operation.
- **Max Feeds per Hour:** Maximum number of video feeds that can be used at one time by the SAFR system. If you attempt to connect more video feeds than your license allows, the excess video feed connection attempts will all fail. Existing video feeds must be disconnected for a period of 1 hour before new video feeds are allowed to re-use the

license.

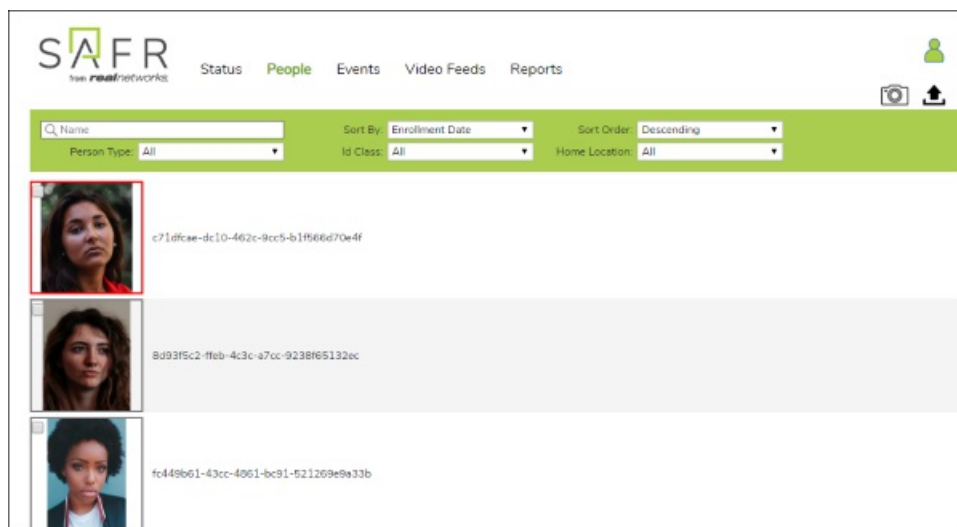
Note: If a single camera is providing video feeds to 2 different Desktop client instances, that counts as 2 video feeds for licensing purposes.

- **Max Faces:** Maximum number of faces that can be stored in SAFR's database. Attempting to save more faces than this limit allows results in an error.
- **Max Days Between Reports:** The maximum elapsed time that can pass before the SAFR system can report its status to a SAFR License Server. SAFR Server discontinues operation if it is unable to reach the SAFR License Server after the specified time has elapsed. If you need to operate your SAFR system on a private network that isn't connected to the Internet, contact your SAFR account manager to acquire a special offline license.

Note: This metric is only applicable for local deployments, and won't appear on the Web Consoles of cloud deployments.

People Page

The People page provides the ability to view and edit information about all the registered people in the Person Directory. For more information, see [Manage People in the Person Directory](#).



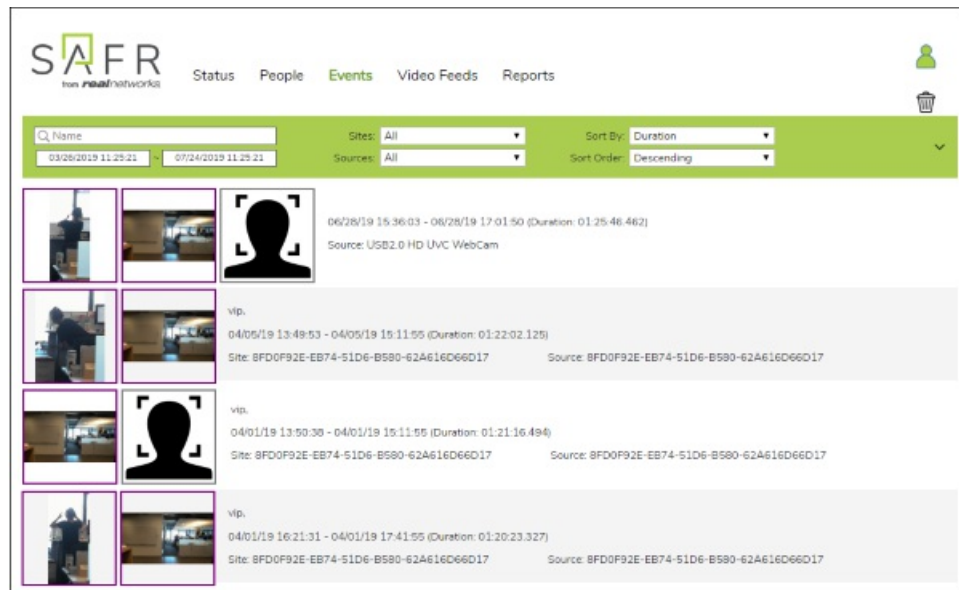
In addition, you can:

- Click the camera icon to take pictures of faces using your integrated camera to register people to the Person Directory.
- Click the upload icon to import images from files. Click the setting icon to adjust the acceptable lower limits of the center pose, contrast, and sharpness image quality metrics.

See [Importing and Registering People](#) for more information.

Events Page

The Events page lists all reported events stored in your Event Archive. For more information see the View Events in the Event Archive section.



Video Feeds Pages

The Video Feeds pages provide processor status and tenant configuration capabilities for all your connected video feeds. Root Config provides a list of all SAFR global default processor and feed properties.

The system is organized as follows:

- Tenants can have directories.
- Users and user IDs are security principals. They have privileges and map to a tenant. They have access to all directories within the tenant.
- If you have super privileges, you're also able to read, write, or config other tenants' properties for APIs that allow for those changes.
- A user ID can be restricted to particular directories within a tenant using white-listing.

Note: For cloud deployments, the Root Config properties are read-only. For local deployments, the Root Config property defaults can be changed by users with super config privileges. However, you are advised to make Root Config changes only when necessary.

The Root and Tenant configs and modes are set on the tabs. Worker config is set by clicking the **Config** button on the **Processor Status** page.

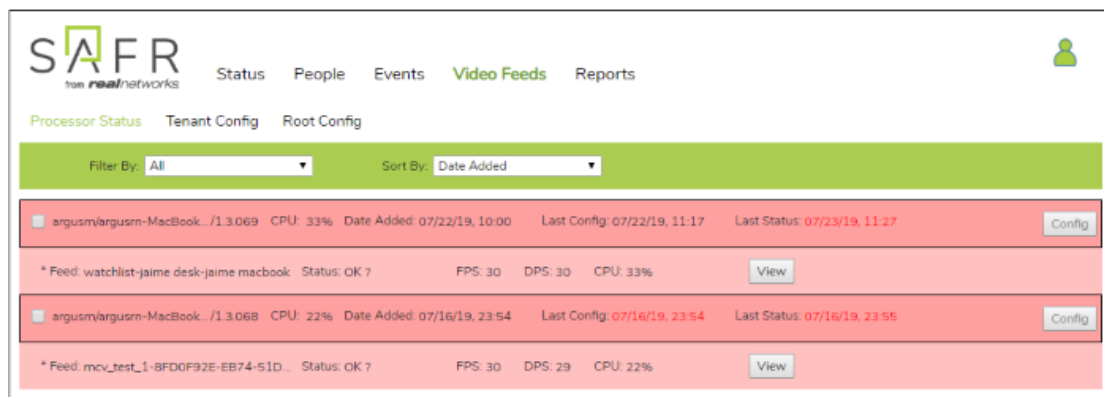
- Tenant Config properties override Root Config properties or Feed properties for your account.
- Root mode overrides settings set on the Root and Tenant Config pages. Tenant mode overrides settings on other pages.

- Like the source URL, the Worker Config sets instance properties, although you can override any settings. This is useful to override settings for an individual device if, for example, there are unique lighting conditions for one feed.

Processor Status Page

This page provides a list of Desktop client instances and video feeds associated with the account. Each row represents a separate computer running the Desktop client that has a video feed associated with it. Inactive video feeds are identified by a red date-time status. Feeds are made inactive by either having status reporting disabled or shutting down the associated Desktop client.

If the video feed is active, click **View** to access a streaming video window. Depending on your privileges, click **Config** to view, edit, or add attributes to override Root and Tenant global configuration settings for a single video feed. To make changes to global account settings, go to the Tenant Config page.



The screenshot shows the SAFR interface with the 'Video Feeds' tab selected. Below the navigation bar, there are filters for 'Filter By: All' and 'Sort By: Date Added'. The main table lists two desktop clients, each with a video feed. The first client has a status of 'OK?' and the second has a status of 'OK?'. Both have 'View' and 'Config' buttons.

Client	CPU	Date Added	Last Config	Last Status	Actions
argusm/argusm-MacBook.../1.3.069	CPU: 33%	Date Added: 07/22/19, 10:00	Last Config: 07/22/19, 11:17	Last Status: 07/23/19, 11:27	Config
* Feed: watchlist-jaima desk-jaima macbook	Status: OK?	FPS: 30	DPS: 30	CPU: 33%	View
argusm/argusm-MacBook.../1.3.068	CPU: 22%	Date Added: 07/16/19, 23:54	Last Config: 07/16/19, 23:54	Last Status: 07/16/19, 23:55	Config
* Feed: mcv_test_1-8FDD0F92E-EB74-51D...	Status: OK?	FPS: 30	DPS: 29	CPU: 22%	View

Tenant Config Page

The Tenant is the primary account. Use this page to add and edit attributes of global settings at the account level to override the Root configurations. Directories can be added at this level by clicking the **Add Item** link. To make changes to individual video feeds, go to the Processor Status page.

SAFR ten realnetworks

Status People Events **Video Feeds** Reports

Processor Status **Tenant Config** Root Config

Tenants [Add item](#)

item 1 ▾ [Delete item](#) [Add Attributes](#)

tenant name

global ▾ [Add Attributes](#)

status-interval

feed ▾ [Add Attributes](#)

mode list ▾ [Add mode](#)

Root Config Page

The Root Config page displays all the properties set in VIRGO by RealNetworks. These global settings are read-only for cloud deployments, but they can be changed for local deployments. To override these settings for your deployment, go to the Tenant Config and Processor Status pages.

SAFR ten realnetworks

Status People Events **Video Feeds** Reports

Processor Status Tenant Config **Root Config**

Root Config

global ▾

status-interval

update list ▾

update 1 ▾

client-type

download-uri

log.enabled

progress-interval

update 2 ▾

client-type

download-uri

log.enabled

progress-interval

Reports Page

Click on the report that you're interested in to set the report's parameters and generate the report.



Traffic Dashboard

The traffic dashboard provides in-depth information about human traffic in a particular area, including:

- Total number of people viewed.
- Sentiment scores.
- Percentage of male and female faces.
- Age and sentiment percentages.

Input Parameters

Parameters

* Directory:

Site:

Source:

☒ Live for last Days

☐ Time Range: ~

Shortest Gap: (seconds)

Coalesce same person appearance count

Count Interval: Minutes

Count event numbers every

Red Alert Count in Interval:

Yellow Alert Count in Interval:

Sub-counts: ☒ New ☒ Return ☒ Person Type

Colors:

Logo Image URL:

- **Directory:** User directory from which to run the dashboard.
- **Site:** Set of cameras to use. This parameter is mutually exclusive with *Source* below.
- **Source:** Single camera to use. This parameter is mutually exclusive with *Site* above.
- **Live for last:** Number of previous days to include in the dashboard. This parameter is mutually exclusive with *Time Range* below.
- **Time Range:** Dates to include in the dashboard. This parameter is mutually exclusive with *Life for last* above.
- **Shortest Gap:** If a person is viewed multiple times within this time period, all those appearances are considered the same event.
 - **Coalesce same person appearance count:** Determines if multiple viewings of a person during the specified time period are coalesced into a single event.
- **Count Interval:** Defines the time interval included in each data bar.
 - **Count event numbers every:** This value will always equal the Count Interval above. (This field is provided to help you select larger time intervals.)
- **Red Alert Count in Interval:** When the count within a count interval is greater than this number, the bar is shown in red. Set this value to zero if you don't want any bars shown in red.
- **Yellow Alert Count in Interval:** When the count within a count interval is greater than this number, the bar is shown in yellow. Set this value to zero if you don't want any bars shown in yellow.

- **Sub-counts:** Specifies which sub-counts, if any, you want displayed on your dashboard. You can choose one or more of the following sub-counts:
 - **New:** Number of registered faces that appear only once.
 - **Return:** Number of registered faces that appear multiple times.
 - **Person Type:** Number of times this *Person Type* appears.
- **Colors:** Allows you to choose which color scheme you prefer.
- **Logo Image URL:** You can use your company's logo on the dashboard, if you want.

Generated Dashboard

Below is a sample traffic dashboard.



Note: The refresh rate for this dashboard is hard-coded to 30 seconds.

Queue Dashboard

The queue dashboard is used to monitor wait times in a queue. In order to use the queue dashboard you'll need 2 cameras: one for the entrance, and one for the exit.

Input Parameters

Parameters

* Directory:

main

Site:

Ignore Person Types:

☒ Live for last

24

Hours

☐ Time Range:

02/12/2020 00:00:00

~

02/20/2020 00:00:00

Queue Name:

Main-Queue

* Entry Source:

* Exit Source:

Count Interval:

60

Minutes

Max wait time:

120

Minutes

Red Alert Wait Time:

30

Minutes

Yellow Alert Wait Time:

15

Minutes

Colors:

Green Theme ▼

Logo Image URL:

https://safr.real.com/console/img/SAFR_TM_color.svg

Refresh Interval:

1

Minutes

View

Cancel

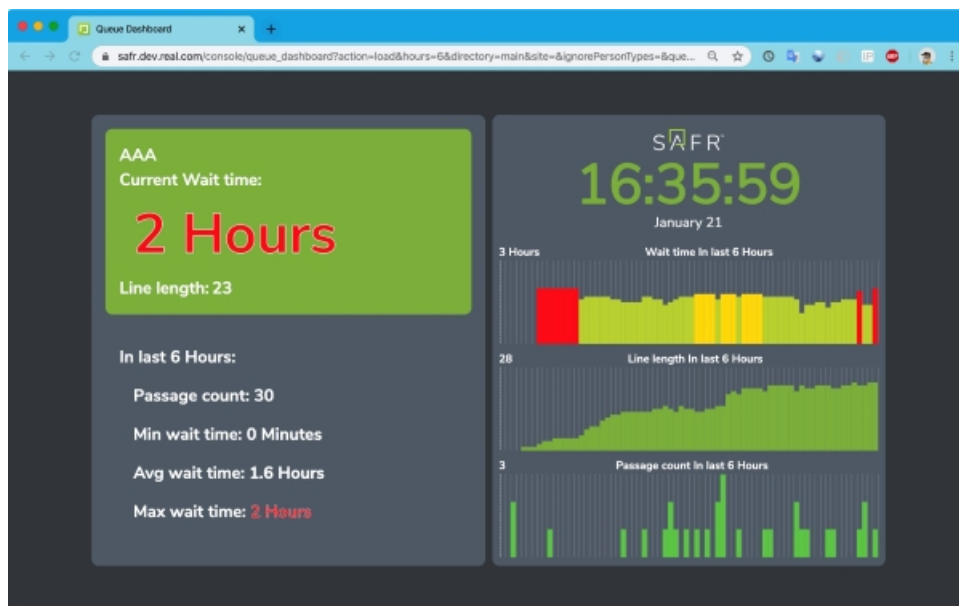
- **Directory:** User directory from which to run the dashboard.
- **Site:** Set of cameras to use.
- **Ignore Person Types:** The Person Types that should not be included in the dashboard, if any.
- **Live for last :** Number of previous hours to include in the dashboard. This parameter is mutually exclusive with *Time Range* below.
- **Time Range:** Time range to include in the dashboard. This parameter is mutually exclusive with *Live for last* above.
- **Queue Name:** - Title of the queue that appears at the top of the dashboard.
 - **Entry Source:** The camera at the beginning of the queue.
 - **Exit Source:** The camera at the exit of the queue.
- **Count Interval:** The amount of time each bar on the traffic dashboard represents.
- **Max wait time:** Any individual whose wait time exceeds this value is assumed to be a false data point and is discarded. It's assumed that the person left the queue without waiting within it to get to the end.
- **Red Alert Wait Time:** When the wait for somebody is greater than this number, the bar is shown in red. Set this parameter to zero if you don't want any bars shown in red.
- **Yellow Alert Wait Time:** When the wait for somebody is greater than this number, the bar is shown in yellow. Set this parameter to zero if you don't want any bars shown in yellow.
- **Colors:** Specifies which color scheme will be used for the dashboard. There are two

options: *Blue Theme* and *Green Theme*.

- **Logo Image URL:** You can use your company's logo on the dashboard, if you want.
- **Refresh Interval:** - Specifies how frequently the data on the dashboard is refreshed. If "0" is entered, the dashboard won't work. If you want a very quick refresh time, enter a very small non-zero number such as 0.1.

Generated Dashboard

Below is a sample queue dashboard.



Attendance Dashboard

The attendance dashboard shows the attendance record for a group of people (e.g. employees or students) on a given day. This dashboard provides:

- Recognized individuals in attendance with an accompanying photo.
- Time interval during which individuals are observed.
- Accumulated time present.
- A snapshot of current attendance. For accuracy, this is best used at the end of the work day or the beginning of the next work day.

Input Parameters

Parameters

* Directory:

main

Site:

Person Type:

☒ Live for current day

☐ For prior day:

02/19/2020

Sort Order:

Alphabetical by name ▼

Refresh Interval:

1

Minutes

View

Cancel





- **Directory:** User directory from which to run the dashboard.
- **Site:** Set of cameras to use.
- **Person Type:** The Person Type(s) to be included in the dashboard. If this parameter is left blank, then all Person Types are included.
- **Live for current day:** Causes the current day to appear in the dashboard. Selecting this parameter is mutually exclusive with the *For prior day* parameter below.
- **For prior day:** The day which you want to appear in the dashboard. Selecting this parameter is mutually exclusive with the *Live for current day* parameter above.
- **Sort Order:** Specifies the criteria by which the people are sorted. There are 4 options:
 - Alphabetical by name - Sorts based on the alphabetical order of their names.
 - In order of arrival - Sorts based on the order of people's arrival times, with people who arrived first being displayed first.
 - Shortest attendance first - Sorts based on how long each person has attended, with the shortest attendances appearing first.
 - Longest attendance first - Sorts based on how long each person has attended, with the longest attendances appearing first.
- **Refresh Interval:** Specifies how frequently the data on the dashboard is refreshed. If "0" is entered, the dashboard won't work. If you want a very quick refresh time, enter a very small non-zero number such as 0.1.

Generated Dashboard

Below is a sample attendance dashboard. Note that you can download the dashboard as an *.xlsx file by clicking on the download symbol in the upper right corner.

01/29/2020



Photo	Name	First Seen	Last Seen	Accu.Time
	Jason Metheny employee	07:03 RNHQ 6015-Door	15:27 RNHQ HR-Door	08:23:52
	Ann Shepard employee	06:57 RNHQ 6100-Door	15:06 RNHQ Cafe-Door	08:08:45
	Alex Gildner employee	08:18 RNHQ Cafe-Door	15:24 RNHQ Cafe-Door	07:05:49
	Dan Grimm employee	08:29 RNHQ 6851-Door	15:34 RNHQ Cafe-Door	07:05:02
	Elaine Eng employee	08:43 RNHQ Cafe-Door	15:44 RNHQ Cafe-Door	07:01:45
	Andrew Grimm employee	08:37 RNHQ Cafe-Door	15:29 RNHQ Cafe-Door	06:52:36

Traversal Dashboard

Displays traversal durations of individuals along a defined set of cameras. This dashboard highlights individuals exceeding expected traversal times and can be used to identify suspicious activity or general slow-downs (i.e. congestion) in real-time or time-frames in the past.

Input Parameters

Parameters

* Directory:

Site:

Ignore Person Types:

☒ Live for last Minutes

☐ Time Range: ~

Path:

Path Sources:

Min Sources Traversed:

Max Traversal Time: Minutes

Red Alert Traversal Time: Minutes

Yellow Alert Traversal Time: Minutes

Sort Order: ▼

Refresh Interval: Minutes

- **Directory:** User directory from which to run the dashboard.
- **Site:** Set of cameras to use.
- **Ignore Person Types:** The Person Types the dashboard should ignore, if any.
- **Live for last:** Number of previous minutes to include in the dashboard. This parameter is mutually exclusive with *Time Range* below.
- **Time Range:** Dates to include in the dashboard. This parameter is mutually exclusive with *Live for last* above.
- **Path:** The name you want to give to this particular Path. **Note:** If you have already defined one or more Paths, then you can select them from a drop down menu rather than defining a new Path.
- **Path Sources:** All the cameras that make up this traversal route. **Note:** The order you add cameras to this field doesn't matter.
- **Min Sources Traversed:** The minimum number of cameras that a person must pass in front of before the traversal dashboard will include them in its data.
- **Max Traversal Time:** Any individual whose traversal time exceeds this value is assumed to be a false data point and is discarded. It's assumed that the person left the traversal area without completing the path.
- **Red Alert Traversal Time:** When a person's traversal time exceeds this value, their data is shown in red. Set this value to zero if you don't want any data shown in red.
- **Yellow Alert Traversal Time:** When a person's traversal time exceeds this value, their data is shown in yellow. Set this value to zero if you don't want any data shown in yellow. The *Red Alert Traversal Time* parameter takes precedence over this parameter.
- **Sort Order:** Specifies the criteria by which the people are sorted. There are three values

you can choose from:

- Traversal Duration - longest first
- Traversal Start - in order of arrival
- Traversal Start - most recent first
- **Refresh Interval:** Specifies how frequently the data on the dashboard is refreshed. If “0” is entered, the dashboard won’t work. If you want a very quick refresh time, enter a very small non-zero number such as 0.1.

Generated Dashboard

Below is a sample traversal dashboard. Note that you can download the dashboard as an *.xlsx file by clicking on the download symbol in the upper right corner.



Photo	Name	6015-Door	6851-Door	HR-Door	Accu.Time
	James Walker admin	13:38 03/11	13:57 03/11		1:37:29 13:38 03/11 - 15:16 03/11
	Gerni Sarkisev IT	13:12 03/11	14:04 03/11	14:37 03/11	1:25:20 13:12 03/11 - 14:38 03/11
	Michael Parham exec	15:12 03/11		14:27 03/11	0:48:15 14:27 03/11 - 15:16 03/11
	Dan Grimm employee		14:22 03/11		0:16:5 14:22 03/11 - 14:38 03/11
	Timothy Lloyd IT		14:06 03/11		0:13:14 14:06 03/11 - 14:19 03/11
	Gary Lewis employee	13:36 03/11			0:0:4 13:36 03/11 - 13:36 03/11

Traffic Report

The traffic report is a pdf that provides in-depth information about human traffic, including:

- Total number of events.
- Counts for unknown and known persons, as well as for unique known and unique unknown persons.
- Gender and age profiles.
- Traffic trends per day.
- Dwell time: The amount of time a person remains on camera per event.

Input Parameters

Parameters

* Directory:

Site:

Time Range: ~

Span Sources: ☒

Shortest Gap: (seconds)

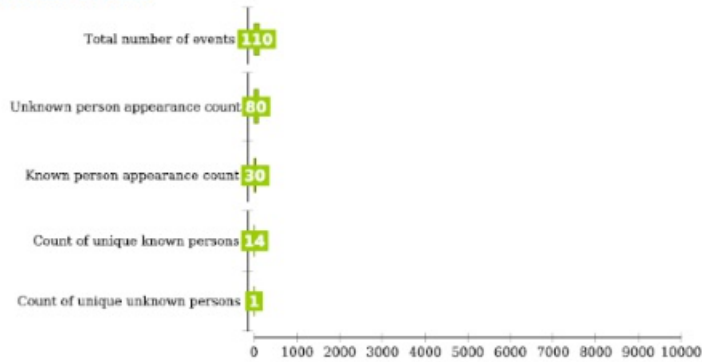
Shortest Gap(Unidentified): (seconds)

- **Directory:** User directory from which to run the report.
- **Site:** Set of cameras to use.
- **Time Range:** Dates and times to include in the report.
- **Span Sources:** Specifies whether or not events triggered in multiple cameras at the same time by the same person should be combined into a single event.
- **Shortest Gap:** If an identified person is viewed by one or more cameras multiple times within this time period, all those appearances are considered the same event.
- **Shortest Gap(Unidentified):** If an unidentified person is viewed by one or more cameras multiple times within this time period, all those appearances are considered the same event.

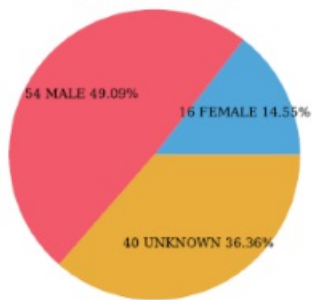
Generated Report

Below are screenshots from a sample traffic report:

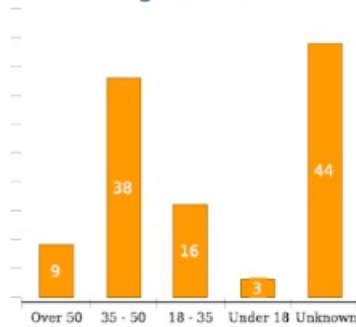
Overall Traffic



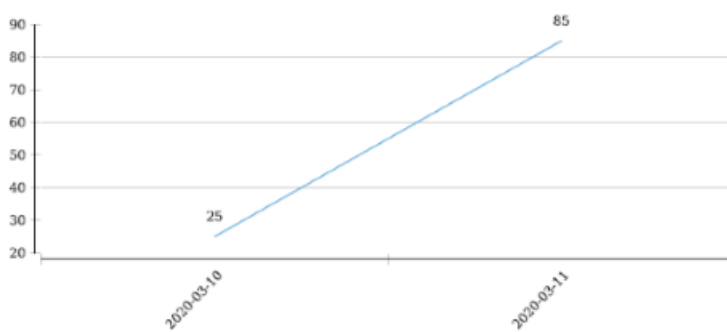
Gender Profile



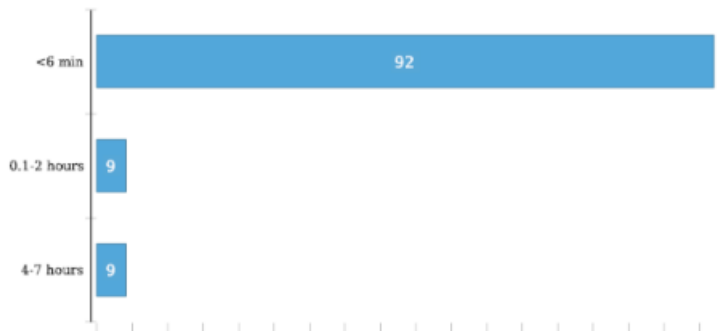
Age Profile



Traffic Trend



Dwell Time



You can download the sample traffic report [here](#).

Pose Liveness Detection

Liveness detection is the process whereby facial recognition software attempts to differentiate between genuine live faces and spoofed fake faces. (such as from a photo of a face) SAFR uses a face's center pose quality to attempt to detect liveness.

Pose liveness detection operates as follows:

1. State A: An unrecognized face needs to be recognized.
2. State B: Proof of liveness will pursued as follows:
 1. The recognized face is tracked at the rate of at least 25 frames per second.
 2. Any loss of tracking (occurrence of lingering for more than 1 frame) or a detection gap > 40 ms in frame capture time results in the need to re-recognize the face and thus a return to State A.
 3. Pose quality must maintain a score of 0.5 or higher for 3 consecutive frames and at least one of the samples must have a profile pose confidence of 35% or less to trigger the transition to the next state, State C.
3. State C: A smooth transition to profile pose will be pursued as follows:
 1. The face is tracked at the rate of at least 25 detections per second.
 2. Any loss of tracking (occurrence of lingering for more than 10 frames) or a detection gap > 40 ms in frame capture time results in the need for re-recognition and thus return to State A.
 3. A momentary loss of tracking (recovered in less than 10 frames) will require a center pose quality difference from the prior frame of no less than 0.15.
 4. If change in identity is detected as part of prescribed re-recognition, State B will restarted.
 5. Pose quality must be observed to transition to score of 0.26 or lower for at least 3 consecutive frames and with 66% of at least 3 images but no more than 30 images immediately proceeding with scores observed > 0.26 and < 0.5 and in decreasing sequence to trigger transition to State D.
 - For example: 0.45, 0.37, 0.23, 0.12, 0.24
 - This algorithm can interpreted as requiring presence of descending strand of samples being at least 66% of the number of samples with min number being specified in preferences and max number being 30 (~1 second).
4. State D: After the profile pose state has changed, a verification call is issued to obtain a similarity score to the identity obtained in State A.
 1. The verification call must indicate at least a 86% match.
 2. A response from recognition must also indicate that the face is in profile pose, based on profile pose confidence returned and threshold set.
 3. If both of above are met, liveness detection will conclude.
 4. If both aren't true, re-recognition will continue immediately for as long as the pose quality score remains at 0.26 or lower until successful confirmation of pose and 86% identity match is confirmed.
 5. If pose score exceeds value of 0.26 for 3 consecutive frames, transition back to

state B will occur.

6. Any loss of tracking (occurrence of lingering) will result in need for re-recognition and thus return to State A.

March 2020 Release Notes

Web Console

- Increased Video Viewer Frame Rate video feed viewer
- Video feed viewer overall support
- Event Archive support for Unauthorized Direction of Travel Detection action events.
- Email and SMS Server Configuration in Status Tab
- Support for Unauthorized Direction of Travel Detection feed configuration attributes
- Support for Cropping Parameters feed configuration attributes
- Support for Contrast Enhancement Integration feed configuration attributes
- Support for person detection input size configuration

Windows

Lite Desktop Client

- SAFR 2.0 UX
 - Live monitoring in single-window app
 - Inline camera settings UX
 - Handling password change for licenser userId
 - Option to disable Operator Console as primary window.
- Increased Video Viewer Frame Rate - support up to 30fps (new platform needed)
 - 30fps, 480p video for local deployments (configurable in VIRGA Tenant Config)
 - 5fps, 480p video for cloud deployments (configurable in VIRGA Tenant Config)
- Video Feed Viewer overlays
 - Right click on feed video to open context menu with overlay options.
- Video contrast enhancement (~20% improvement):
 - Contrast Enhancement Integration
- Unauthorized Direction of Travel Detection configuration and display in Event Archive

Windows Desktop Client

- All the Lite Desktop client changes
- Avigilon Integration
- More efficient face detection on NVIDIA GPUs
- Person detection input size configuration: NORMAL (default), SMALL, and LARGE.
 - SMALL: 26% faster than NORMAL

- LARGE: 66% slower than NORMAL
- VIRGO for Windows updated:
 - VIRGO support for multiple remote video feed viewers
 - VIRGO support for video overlays (shown on remote video feed viewers).
 - VIRGO support for video feed Cropping Parameters
 - VIRGO support for Unauthorized Direction of Travel Detection configuration
 - VIRGO support for person detection input size configuration
 - VIRGO support for Contrast Enhancement Integration configuration.
 - VIRGO stability fixes
- Updated higher accuracy person detection model
 - Max Accuracy and Balanced modes improvement: 3%
 - Max Speed mode improvement: 7.1%
 - Balanced vs. Max Speed accuracy advantage: 36.2%

Windows SAFR Edge

- All the Windows Desktop client changes
- SMS Notifications support in SAFR Actions
 - Support for SMS Server Config in SAFR Actions (AWS SNS)
 - Support for configuring SMS alerts triggered by events in SAFR Actions
- Support for Unauthorized Direction of Travel Detection action events

Windows SAFR Platform

- Age Model update with accuracy age recognition model
 - 15% improvement on Asian faces
 - 9.4% general improvement
- Increased Video Viewer Frame Rate
- Security Patches
- All the Windows SAFR Edge changes

Linux

SAFR Linux Ubuntu and CentOS Platform

- All the Windows SAFR Platform changes

Jetson

- Person detection added
- Higher efficient (faster) face detection
- All the Windows SAFR Platform changes

- All the Windows SAFR Platform changes

macOS

macOS Desktop Client

- Increased Video Viewer Frame Rate
 - support up to 30fps (new platform needed)
 - 30fps, 480p video for local SAFR Platform (configurable in VIRGA Tenant Config)
 - 5fps, 480p video for Cloud SAFR Platform (configurable in VIRGA Tenant Config)
- Video Feed Viewer overlays
 - Right click on feed video to open context menu with overlay options.
- Unauthorized Direction of Travel Detection configuration and display in Event Archive

macOS SAFR Edge

- All the macOS Desktop client changes

macOS SAFR Platform

- All the SAFR Window Platform changes

Android Mobile Client

- Addition of Android Events:
 - New side-menu navigation
 - Recent Matches view
 - Watchlist person view
 - Profile
 - Timeline
 - Deep-links from SMS or email
- Bug Fixes

iOS Mobile Client

- Bug fixes

SAFR SDK

- Windows:
 - Contrast Enhancement Integration

- More efficient face detection on NVIDIA GPUs
 - Updated higher accuracy person detection model
- Android:
 - Bug fixes
- Linux:
 - Contrast Enhancement Integration
 - Updated higher accuracy person detection model
- Jetson:
 - Contrast Enhancement Integration
 - More efficient face detection on NVIDIA GPUs
 - Updated higher accuracy person detection model

Embedded SDK

- Platforms being released:
 - Windows:
 - eSDK-full (includes NVIDIA GPU support, includes auto-selection between non-AVX and AVX2 support)
 - Person detection
 - Bug fixes
 - eSDK-lite (no NVIDIA GPU support, includes auto-selection between non-AVX and AVX2 support)
 - Bug fixes
 - Linux x86 Ubuntu 16.04:
 - eSDK-full (includes NVIDIA GPU support, includes auto-selection between non-AVX and AVX2 support)
 - Person detection
 - Bug fixes
 - eSDK-lite (no NVIDIA GPU support, includes auto-selection between non-AVX and AVX2 support)
 - Bug fixes
 - Linux ARM Ubuntu 18.04:
 - eSDK-lite (no GPU support)
 - Bug fixes
 - Jetson - Linux ARM Ubuntu 18.04:
 - eSDK-Jetson (NVIDIA GPU support)
 - Person detection
 - More efficient face detection
 - Bug fixes
 - Android ARM - Android 5.0 or later:
 - eSDK-lite (no GPU support)
 - Bug fixes
 - eSDK-lite 64 bit (no GPU support)
 - Bug fixes

January 2020 Release Notes

Web Console

- New report: Queue Dashboard.
- Traversal Dashboard improvements.
- Traffic Dashboard optimizations.
- Attendance Dashboard enhancement.

Windows

Lite Desktop Client

- Sign-in UX changes to support operator workflows.
- Option to require sign-in on every start.
- User Administration.
- Option to disable Windows auto-update when in SAFR Kiosk Mode.
- Video Feed Viewer hides stats by default. Right click to display stats.
- Video Feed Viewer supports 10fps, 720p video (new platform needed).
- Genetec FR Plugin Improved SSL error handling and GUI option to turn off SSL.

Windows Desktop Client

- All the Lite Desktop client changes.
- VIRGO for Windows stability fixes.

Windows SAFR Edge

- All the Windows Desktop client changes.

Windows SAFR Platform

- All the Windows Desktop client changes.
- Installer options to install without SAFR Desktop and to customized path.
- Returned installer option to force CPU Face Recognition service.
- Initiated model initialization during installation to reduce initialization time upon launch.

Linux

Linux Ubuntu and CentOS SAFR Platform

- VIRGO enhancements.

Jetson

- SAFR Jetson Ubuntu 18.04 Platform has been implemented.

macOS

macOS Desktop Client

- Bug fixes.

macOS SAFR Edge

- Bug fixes.

macOS SAFR Platform

- Bug fixes.

Android Mobile Client

- Added support for arm64-v8 architecture.
- Bug Fixes.

iOS Mobile Client

- Bug fixes.

SAFR SDK

- Windows:
 - Added Image analyzer support for person (object) and badge detections.
- Android:
 - Added support for arm64-v8 architecture.
 - Bug fixes.
- Linux:
 - Added Image analyzer support for person (object) and badge detections.
- Jetson:

- Initial release

Embedded SDK

- Platforms being released:
 - Windows:
 - eSDK-full (includes NVIDIA GPU support, includes auto-selection between non-AVX and AVX2 support)
 - eSDK-lite (no NVIDIA GPU support, includes auto-selection between non-AVX and AVX2 support)
 - Changes:
 - Bug fixes.

December 2019 Release Notes

Web Console

- New Traversal Dashboard report

Windows

Lite Desktop Client

- Enhanced Event Archive GUI
- Person activity view
- CBP Face Acquisition System
- Advanced configuration of Center Pose Quality for strangers/learning: pitch, roll, and yaw
- Identity retention configuration
- Full Screen, Locked Screen, Auto-restart, Auto-logon, and Kiosk mode for Windows

Windows Desktop Client

- All the Lite Desktop client changes
- Enhanced Person Detection Accuracy - especially in crowded scenes
- Ximea Camera Integration

Windows SAFR Edge

- All the Windows Desktop client changes

Windows SAFR Platform

- All the Windows Desktop client changes
- All the System Console changes
- Concurrent face matching (3.5X lower matching latency on 8 core processor)
- Higher face-recognition throughput on non-GPU machines
- SAFR offline licensing

Linux

Linux Ubuntu VIRGO

- Person-face consolidated tracking enhancements
- Advanced configuration of Center Pose Quality for strangers/learning: pitch, roll, and yaw
- Enhanced Person Detection Accuracy - especially in crowded scenes

Linux Ubuntu and CentOS SAFR Platform

- All the Linux VIRGO changes
- All the System Console changes
- Concurrent face matching (3.5X lower matching latency on 8 core processor)
- Higher face-recognition throughput on non-GPU machines
- Identity retention configuration
- SAFR offline licensing

macOS

macOS Desktop Client

- Person-face consolidated tracking enhancements
- Event retention configuration GUI revision
- Identity retention configuration GUI
- Advanced configuration of Center Pose Quality for strangers/learning: pitch, roll, and yaw

macOS SAFR Edge

- All the macOS Desktop client changes

macOS SAFR Platform

- All the macOS Desktop client changes
- SAFR offline licensing

Cloud

- Concurrent face matching (3.5X lower matching latency on 8 core processor)
- Identity retention configuration

Android Mobile Client

- Hardware Video Decode
- Active Camera Connect
- Bug Fixes

iOS Mobile Client

- Dark mode bug fixes

SAFR SDK

- Windows:
 - Enhanced Person Detection Accuracy - especially in crowded scenes
- Linux:
 - Bug fixes
- macOS:
 - Bug fixes
- Android:
 - Bug fixes
- iOS:
 - Bug fixes

Embedded SDK

- Addition of new models: Age, Gender, Sentiment, Occlusion, Composite Signatures, Pose Profile, and Face/No-Face
- Platforms being released:
 - Windows:
 - eSDK-full (includes NVIDIA GPU support, includes auto-selection between non-AVX and AVX2 support)
 - eSDK-lite (no NVIDIA GPU support, includes auto-selection between non-AVX and AVX2 support)
 - Linux x86 Ubuntu 16.04:

- eSDK-full (includes NVIDIA GPU support, includes auto-selection between non-AVX and AVX2 support)
- eSDK-lite (no NVIDIA GPU support, includes auto-selection between non-AVX and AVX2 support)
- Linux ARM Ubuntu 18.04:
 - eSDK-lite (no GPU support)
- Jetson - Linux ARM Ubuntu 18.04:
 - eSDK-Jetson (NVIDIA GPU support)
 - Include model compilation/optimization tool
- Android ARM - Android 5.0 or later:
 - eSDK-lite (no GPU support)

November 2019 Release Notes

Web Console

- Support for Anonymous vs. Known identity event retention configuration
- Support for Face-Person enhanced tracking
- Video feed occlusion detection config support
- Video feed config support to limit stranger reporting only to occluded strangers

ARES

- hasRootEventId filter was added

Windows

SAFR Windows Desktop Lite

- Support for Occlusion Detection configuration in Video Feeds
- Support for Anonymous vs. Known identity event retention configuration
- Preferences to limit stranger reporting only to occluded strangers
- Improved person import GUI
- Enabled person import directly from Event Archive
- Support for Mobotix Camera Events
- Support for Event time offset for offline videos
- Image quality metrics in Person Details dialog

Desktop Client

- All the SAFR Desktop Lite changes

- Enhanced person-face tracking and reporting
 - Consolidated person/face event reporting
 - Consolidated person/face event display
- Support for false face detection filtering
- Support for Genetec FR Plugin Integration
- Updated Virgo for Windows

SAFR Windows Edge

- Updated SAFR Desktop
- Updated ARES

SAFR Windows Platform

- Higher accuracy Face Recognition Model (v5 signatures)
- Faster (2x) DB Matching
- Redundant CVOS support
- Support for Anonymous vs. Known Identity event retention
- Support for Face No-Face Classification
- Updated System Console
- Updated ARES
- Filtering of secondary faces on import via REST API

Linux

SAFR Linux Ubuntu VIRGO

- Occlusion Detection
- Enhanced person-face tracking and reporting
 - Consolidated person/face event reporting
 - Consolidated person/face event display
- Face No-Face Classification Integration

SAFR Linux Ubuntu and CentOS Platform

- Higher accuracy Face Recognition Model (v5 signatures)
- Faster (2x) DB Matching
- Redundant CVOS support
- Port conflict resolution at install time
- Support for Anonymous vs. Known Identity event retention
- Support for Face No-Face Classification
- Updated VIRGO

- Updated System Console
- Updated ARES
- Filtering of secondary faces on import via REST API

macOS

macOS Desktop Client

- Occlusion Detection
- Enhanced person-face tracking and reporting
 - Consolidated person/face event reporting
 - Consolidated person/face event display
- Face No-Face Classification Integration
- Support for Anonymous vs. Known identity event retention configuration
- Model Upgrade GUI

SAFR macOS Edge

- Updated SAFR Desktop
- Updated ARES

SAFR macOS Platform:

- Higher accuracy Face Recognition Model (v5 signatures)
- Faster (10x) DB Matching
- Support for Anonymous vs. Known Identity event retention
- Support for Face No-Face Classification
- Updated System Console
- Updated ARES
- Filtering of secondary faces on import via REST API

Cloud:

SAFR Platform

- Higher accuracy Face Recognition Model (v5 signatures)
- Faster (2x) DB Matching on Windows
- Redundant CVOS support via NFS
- Support for Anonymous vs. Known Identity event retention
- Support for Face No-Face Classification
- Updated System Console

- Filtering of secondary faces on import via REST API

Download Portal

- Updated Android System Requirements
- Removed Create new account link

Android SAFR App and SDK:

- Bug fixes

Embedded SDK (Windows and Android)

- Composite signature support
- Faster multi-core face signature matching

September 2019 Release Notes

SAFR Windows

1. Central Video Feed Management

Video feeds on Windows can now be configured and managed centrally for the entire cluster of SAFR Windows (and Linux) Platform machines. This means that a large deployment can be configured from a single machine using the Desktop client (preferred) or the System Console. SAFR no longer requires video feed windows to remain open, nor do Windows users need to remain logged. SAFR will now also automatically resume processing on system reboot. This makes SAFR on Windows a fully resilient service that can handle power outages and be easily managed even when distributed on many machines.

To enable this, SAFR Windows Platform now comes with Virgo for Windows which performs video feed processing in the background. Windows Virgo supports Genetec, Milestone and Digifort VMS feeds as well as ONVIF, direct RTSP URL, and USB camera feeds. You can configure Windows Virgo via the Windows Desktop client or the System Console. The Windows Desktop client is recommended as a configuration tool in all cases and is required if configuring VMS feeds. When adding a feed simply select an auto-detected camera and choose operation mode.

2. Redundant DB Configuration

As was already available on Linux, SAFR Windows Platform can now be configured for

redundant DB operation. This means that all DB information (this includes face signatures, person meta-data and events but does not yet include images) will be stored in two or more separate machines and loss of one DB machine will automatically fail-over to another. Redundant DB operation also enables horizontal scalability of the face-matching operation which is distributed across all participating DB machines thus increasing size of deployment achievable (hardware estimator provides number of DB machines needed).

Keep in mind that you must have an odd number of DB machines for automatic failover to function and that the maximum number of redundant DB machines is 50.

3. Watchlist Synchronization across SAFR Platforms and Accounts

SAFR can now be configured to synchronize watchlists from one SAFR Platform or Account to any number of other SAFR Platforms or Accounts. This means that SAFR Platform can now be deployed in a distributed manner with many independent SAFR Platforms at different locations and yet be kept updated with a watchlist maintained centrally (e.g. in Cloud).

You can configure SAFR Platform to synchronize one directory per account (tenant) from the System Console Status tab. Max latency for synchronization is 10 minutes and max throughput is ~20 records per second per sync connection. It might thus take up to 10 minutes to perform initial sync of 10K records.

4. 5X Faster DB Matching Speed

DB matching speed and efficiency have been improved 5x. This means that matches are 5x faster and require 5x less processing power. This translates to significant TCO savings for deployments requiring large watchlists.

On single CPU core, 1 million faces can now be matched in 350-400ms.

5. SAFR Actions for Occlusion

SAFR Actions and Action Relay Event Service (ARES) now supports occlusion event attributes. This means you can configure actions to trigger specifically on occluded faces. For more information, search on "occlusion" in *Action Relay Event Service - ARES manual*.

6. Person (Body) Detection Balanced Mode

Person detection balanced mode delivers 50% more throughput than max accuracy mode with only slight degradation in accuracy. This is now the default mode for person detection and is recommended for all cases when high accuracy of person body detection is needed (e.g. tracking in visually complex environments with several persons present).

In comparison, max speed person detection mode delivers 300% more throughput than

balanced mode but with significant reduction in accuracy. However, this mode is commonly adequate for low complexity tracking such as casino tables or teleconferencing rooms.

SAFR Linux

1. Multi-GPU Scalability

SAFR Linux Platform now offers enhanced scalability across multiple NVIDIA GPUs. SAFR Linux VIRGO has been optimized to be even less reliant on CPU and to maximize use of NVIDIA GPUs. This means that a single large machine can support 6 NVIDIA T4 processors which amounts to a SAFR recognition payload of 90 1080p@15fps feeds or 75 4K@15fps feeds (inclusive of recognition).

This capability is also available in standalone VIRGO Ubuntu download from Developers page.

2. Person Body to Face Recognition Linkage

Person body detection and tracking is now enhanced with face recognition and thus takes on identity established through face recognition. As person body detection is more accurate than face (due to size and being detectable in nearly any orientation) this means that identity tracking with combined person body and face detection is more accurate than face alone. When more accurate account of identity presence before the camera is needed, person events can now be used which are augmented with associated face attributes.

This function is automatically enabled when both person (body) detection and face recognition are enabled.

3. The Following New SAFR Windows features are also now available on Linux

- Watchlist synchronization across SAFR Platforms and Accounts
- 5X faster DB Matching speed
- Person (body) detection balanced mode

macOS Desktop Client

1. Pose Based Liveness Detection

This features previously introduced on Linux is now also available on macOS. It enables liveness detection based on consistent change in face orientation (pose) as an alternative to smile action. It can be used for walk-up and walk-through secure access scenarios that require liveness confirmation when paired with well positioned cameras.

2. Person Body to Face Recognition Linkage (Same as Linux)

SAFR Android

Faster SAFR Native Face Detector

SAFR native face detector is now multi-threaded on Android and offers higher frame-rate and accuracy than Google Vision face detector (available when Google Play is present on the device). The Android Mobile client now delivers excellent face detection performance at ~15fps while utilizing 35% CPU and Google Pixel phone.

Frame Skipping Logic to Maintain Low Latency of Detection and Recognition

When video frame rate is higher than detection rate device can deliver, video frames will be appropriately skipped for analysis in order to not cause backlog of processing that would increase latency in detection and recognition.

SAFR Embedded SDK (Windows and Android)

1. Person record export / import API
2. Face landmark coordinates (eyes, nose, mouth)
3. Face signature export / import API

SAFR SDK

Windows:

- Bug Fixes

Android:

- Multi-threaded face detector with higher face detection throughput.
- Frame skipping logic to maintain low latency of detection and recognition.

August 2019 Release Notes

SAFR Windows

- Occlusion Detection:

SAFR now has the ability to detect faces that are occluded. Occlusion constitutes any obstruction of the key facial features such as from a scarf, hand, glasses, hair draping

over the face, etc... This capability is currently integrated to accomplish two features:

1. To filter out any occluded faces while learning them in the wild and thus prevent storing ambiguous face references in the SAFR person database.

For example, such a feature is used when learning and memorizing players sitting at the casino table to prevent learning them with an occlusion feature such as a wineglass in front of their face which may later create recognition inaccuracies.

2. To update occurrence event records with better face images without the occlusion and thus increase the value of the image stored with the event for presentation and investigation purposes.

You will find the occlusion recognition switch in the **Recognition** tab under SAFR Preferences as well as max tolerable occlusion level adjustment for newly learned faces.

- Core Face Recognition Optimizations for NVIDIA GPUs:

These optimizations enable up to 463 recognitions per second on NVIDIA GTX 1080Ti graphics cards. This is 14x more recognition throughput in comparison to the maximum achievable on 4 Core 3.4GHz Intel Xeon Skylake-SP processor. The improvement is even more pronounced when all face attributes are computed together (identity, age, gender, sentiment). In such case optimization delivers 320 combined recognitions per second which is 40x more throughput in comparison to maximum achievable on 4 Core 3.4GHz Intel Xeon Skylake-SP processor. These optimizations also reduce recognition latency by 50% and thus enable even faster and more reliable recognition. All this results in cost reductions for on-premise core recognition subsystem deployments from \$2,477 to \$518 per 100 recognitions per second and from \$10,667 to \$797 for 100 all-attributes recognitions per second.

Note that these optimizations introduced a necessary one-time GPU calibration step which is performed when the system is started for the first time with GPU(s) present. It takes about 3 minutes per recognition model (15 minutes total) and per GPU for the system to be properly calibrated. Until this is completed, you will see System Initializing message in video view and recognition will not be operational.

- Person Body Detection NVIDIA GPU Optimizations

Person detection speed was improved by 30% and throughput by 50%. This means person detection is faster and more fluid than before. Maximum person detection throughput for our max accuracy model is 115 frames per second on NVIDIA GTX 1080Ti and 329 frames per second on NVIDIA Quadro RTX 6000. Maximum person detection throughput for our max speed model is 625 frames per second on NVIDIA GTX 1080Ti and 1052 frames per second on NVIDIA Quadro RTX 6000.

- Customizable options were added to our popular traffic dashboard (available from the Reports tab in the System Console). These options enable traffic dashboard to be

customized in color, logo, language, and time-range. The traffic dashboard can now also be linked directly from another web site and all customization options are available as URL query parameters. This feature enables easy integration of the dashboard into customers' portals who may wish to display the dashboard in colors and logos of their brand.

- A new attendance dashboard was added to the Report tab in the System Console. For a specified time range and location, it displays all recognized individuals in attendance along with the time interval they were observed present. This dashboard can be used as a replacement of punch-card system that tracks employee attendance when properly combined with entry and exit camera monitoring ingress and egress at the work site.
- Installer has been equipped with more customizable options to allow SAFR Logs to be removed from deployment and heap auto-configure behavior to automatically scale memory allocation for SAFR based on system memory available. These options enable SAFR Platform to be deployed on very small PCs (8GB RAM, 32GB Disk, \$550) that can independently monitor 2 1080p video feeds. For example, such a small configuration could be used for a small SAFR Platform deployed at a casino table. The heap auto-config also enables SAFR to scale up on larger system and thus reliably handle higher recognition throughput and event traffic.
- To further protect privacy, SAFR now also limits retention of system logs associated with events to the same time frame as configured for events retention in the SAFR database. This means that no trace of individual whereabouts is kept beyond the configured retention time. Recognition logs have also been reduced in their default logging level so as not to include any personally identifiable information (PII).

SAFR Linux

- The Linux release inherited the following improvements introduced above for Windows:
 - Customizable options for Traffic Dashboard.
 - New Attendance Dashboard.
 - Log retention and log content changes to protect privacy.
- Database fail-over is now enabled on Linux. This means when SAFR is deployed on multiple machines with Database redundancy enabled, failure of the primary machine (containing primary Database) will not degrade secondary nodes that are running redundant Database from full functionality.

SAFR SDK

- RTSP support has been added to iOS and Android SAFR SDK. This means that SAFR SDK can now process video feeds delivered via rtsp protocol widely supported by IP cameras and can be thus used to process video feeds from a detached camera. For example, iOS or Android device can be used to process video feed from body camera

connected to the device via WiFi.

- iOS SAFR SDK is available in our Partner Cloud and Production environment.
- Android SAFR SDK is available in our Partner Cloud environment and will be further validated and pushed to production next week.
- Windows SAFR SDK has person body detection added to its capabilities which enables developers to implement alerts based on body detection and traffic counting. Also new in Windows SAFR SDK is availability of pitch, roll and yaw face attributes which describe orientation of the face around all three axis.

Mobile Clients

- iOS and Android Mobile clients have been equipped with same RTSP support described above for SAFR SDK. To connect an RTSP feed, press-and-hold camera selection button in bottom right corner. You will be able to register several RTSP feeds that will be stored and made available for selection.
- iOS SAFR application is awaiting review by Apple and will be available next week in the app-store.
- Android SAFR application will also be available next week on SAFR Partner and Production Cloud portal.

SAFR Cloud

- Occlusion detection is now available in SAFR Cloud and can be utilized by developers via SAFR REST APIs or be used through the Desktop client for Windows.
- Customizable Traffic Dashboard and Attendance Dashboard described above are also available in SAFR Cloud.

SAFR Stability

- 67 defects were fixed for this release.

Follow-up Update

A small follow-up update was released later in August.

1. The Mobile client for Android was released with the following new capabilities:
 - RTSP video feeds are now supported. This means that Mobile clients can now process video feeds delivered via RTSP protocol widely supported by IP cameras and can be thus used to process video feeds from a detached camera. For example, Android devices can be used to process video feeds from body cameras connected to the device via WiFi. To connect an RTSP feed, long-press camera selection button in bottom right corner. You will be able to register several RTSP feeds that will be stored and made available for selection.

- Google Play Services are no longer required on Android device. SAFR now includes own SAFR face detector. You can switch between Google and SAFR detectors for integrated camera use. SAFR face detector provides higher detection accuracy but is slightly slower when processing feeds from devices integrated camera due image conversion overhead which we will look to eliminate in the future. RTSP feeds are always processed via SAFR face detector which offers higher detection accuracy and speed over Google supplied face detector.
2. SAFR Cloud, SAFR Windows Platform, SAFR Windows SDK, and SAFR Android SDK were released with a few more bug fixes.