Video Recognition Gateway (VIRGO)
### Contents

1 Video Recognition Gateway (VIRGO) .................................................. 3  
2 VIRGO System Requirements ............................................................. 4  
3 VIRGO Installation Guide ................................................................. 6  
4 VIRGO in the Video Feeds Window ..................................................... 9  
5 VIRGO Command Line Interface ....................................................... 14  
6 Video Feeds Properties ................................................................. 19  
7 Processing Video Files .................................................................. 40  
8 VIRGO Tools ................................................................................. 53  
9 Factory Configuration File ............................................................... 54  
10 Docker ......................................................................................... 59  
11 GPU Support .............................................................................. 62  
12 Service Logging .......................................................................... 65  
13 Service Monitoring ........................................................................ 67  
14 VIRGO Architecture .................................................................... 69  
15 Troubleshooting ........................................................................... 72
1 Video Recognition Gateway (VIRGO)

Video Recognition Gateway (VIRGO) is a daemon system which runs on a Portable Operating System Interface (POSIX)-compatible system. It receives video feeds from one or more cameras and it recognizes and tracks faces in those video streams in real time. It generates tracking events and sends those events to an event server. VIRGO daemons can be managed either through the VIRGO command line interface or through the video feeds window of the Desktop Client or the Web Console.
2 VIRGO System Requirements

2.1 Linux Requirements

Video Recognition Gateway (VIRGO) requires at least the following x86_64 CPU features:

- Ivy Bridge or better CPU architecture
- SSE4
- AVX

A Linux distribution must implement at least the following components:

- LSB support
- systemd

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

- Nvidia CUDA 10.1 compatible or newer

2.1.1 Ubuntu 16.04(.5+)

The following additional software components must be installed to allow VIRGO to run successfully:

- libcurl3
- libgomp1
- libatomic1
- libbsd0
- libv4l-0

To install the software components listed above, execute the following commands in a shell:

```bash
sudo apt-get update
sudo apt-get install libcurl3 libatomic1 libgomp1 libv4l-0 libbsd0
```

2.1.2 Ubuntu 18.04(.2+)

The following additional software components must be installed to allow VIRGO to run successfully:

- libcurl4
- libgomp1
- libatomic1
- libbsd0
- libv4l-0

To install the software components listed above, execute the following commands in a shell:

```bash
sudo apt-get update
sudo apt-get install libcurl4 libatomic1 libgomp1 libv4l-0 libbsd0
```

2.2 macOS 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
3 VIRGO Installation Guide

This page describes how to install standalone Video Recognition Gateway (VIRGO).

Note: If you installed VIRGO when you installed SAFR Platform or SAFR Desktop, there’s no need to install standalone VIRGO.

3.1 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.

3.2 Download the VIRGO Installer

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

3.3 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.

3.3.1 macOS

Installer name: Virgo.pkg

Follow these steps to install VIRGO on your macOS machine:

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

3.3.2 Linux

Installer name: virgo_installer.tar.gz

Follow these steps to install VIRGO on your Linux machine:

1. Download the Linux VIRGO installer from the SAFR Download Portal here: https://safr.real.com/developers
2. Decompress the package: tar -xzf virgo_installer.tar.gz
3. Make sure that the necessary third-party library dependencies are installed. For a list of required libraries see the VIRGO system requirements documentation.
4. Run the installer script. The installer script will by default install VIRGO as a system daemon. Although we strongly recommend that you install VIRGO as a system daemon, we do support user daemon installations. You can explicitly specify the desired type of installation by passing the `--user` or `--system` option to the script:

   - `virgo_installer/install.sh --user` installs VIRGO as a user daemon.
   - `virgo_installer/install.sh --system` installs VIRGO as a system daemon.

VIRGO will be installed into the following location:

- System daemon installation: `/opt/RealNetworks`
- User installation: `~/.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
```

### 3.4 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.

### 3.5 FAQ for Linux 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 a shell:

   > sudo /opt/RealNetworks/virgo/uninstall.sh

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

   > ~/RealNetworks/virgo/uninstall.sh

3. I’ve installed VIRGO as a user daemon. How do I stop virgod?

   Execute the following command in a shell:

   > systemctl stop --user com.real.virgod.service

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

One of the primary ways to manage Video Recognition Gateway (VIRGO) feeds is via the Video Feeds window of the Desktop Client or the Web Console. The sections below describe how to do so.

4.1 Create a VIRGO Video Feed

There are two ways to create a VIRGO video feed. Newly created VIRGO video feeds have the following properties:

- The feed is not affected by subsequent changes to preferences made within the Desktop Client. If you change a preference within your Desktop Client, that change is not cloned to the existing VIRGO video feed(s).
- Feeds continue running and processing their video streams regardless of whether or not the Desktop Client is running or not.
- If you shut down your machine, the video feed will try to restart itself whenever your machine is turned on again.

4.1.1 Camera Feed Analyzer Method

The easiest way to create a VIRGO video feed is to do the following:

1. Open the Desktop Client.
2. Connect a camera to the client in the Camera Feed Analyzer window. (i.e. the default window)
3. Press the Add to Video Feeds for continuous processing in the background button as shown below highlighted by the red arrow.

4. You’ll be prompted for the following information:
   - Feed Name: Enter any name for the video feed you wish.
   - Mode: Select the video processing mode from the drop-down menu that you want the VIRGO daemon to operate in. For a description of the video processing modes, see here.
   - Processor: The machine where you want the VIRGO daemon to run. The current machine is selected by default.
   - Apply Mode Customizations from Preferences: Enable if you want the Desktop Client preferences applied to the new VIRGO daemon.
4.1.2 Manual Method

The manual way to create a VIRGO video feed is as follows:

1. Connect a camera to SAFR.
2. Open the Video Feeds window and click the Configure button for your current machine.
3. Hover your mouse over Feeds, then click the + button.
4. You’ll be prompted for the following information:
   - **Feed Name**: Enter any name for the video feed you wish.
   - **Camera**: Select the camera feed you connected in Step #2 from the drop-down menu.
   - **Format**: Select the resolution and frame rate for the new video feed.
   - **Mode**: Select the video processing mode from the drop-down menu 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.
5. Click the **Add** button.
6. Click **Apply** in the bottom right corner of the Video Feeds window.
   **Note**: If you close the Video Feeds window without first clicking Apply, the video feed won’t be created.

4.2 Manage VIRGO Video Feeds

To manage VIRGO video feeds, open the Video Feeds Window within either the Desktop Client or the Web Console, then click the Configure button for your current machine. You’ll see a screen similar to the following: **(Note: You can often expose additional properties by clicking on the arrow next to entries, as shown by the arrow next to the SomeFeed entry below.)**
- **name**: The name of the machine being managed.

- **admin**: Contains admin properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource-type</td>
<td>tenant The tenant being managed.</td>
</tr>
</tbody>
</table>

- **global**: Contains global properties. See Global Properties for more information.

- **monitoring**: Monitoring properties allow you to monitor a video feed’s health and send a notification email if one of the health metrics degrades to a certain level.

To set up notification emails, do the following:

1. Set up an SMTP Email Service on the Status Page of the Web Console.
2. Enable one or more of the alarm conditions of the video feed’s monitoring properties. There are 7 conditions available:
   1. **delinquent**: The video feed has stopped responding/sending status updates.
   2. **feed.error**: The video feed has encountered an error.
   3. **lowRAM**: The host machine has low RAM memory.
   4. **lowDisk**: The host machine has low hard drive storage space.
   5. **lowGPUMemory**: The host machine has low GPU memory.
   6. **lowCPU**: The host machine has low CPU processing power.
   7. **lowGPU**: The host machine has low GPU processing power.
3. Set the subject and message properties for your enabled conditions.
4. Set the threshold property for enabled conditions. (Note: The delinquent and feed.error conditions don’t have threshold properties),
5. Set the alarm.mail.username and alarm.mail.password properties to your email credentials.
6. Set `alarm.mail.enabled` to TRUE to enable the notification emails.
   See Monitoring Properties for a list of all the available monitoring properties.

- **feeds**: Specifies the feed properties. See Feeds Properties for more information.
- **update**: The update properties are not intended for public consumption at this time.

### 4.3 Terminate a VIRGO Video Feed

VIRGO video feeds that haven’t been terminated will continuously run in the background, and will automatically restart themselves after system shutdowns and reboots. Because each video feed consumes a significant amount of CPU resources, you’ll want to terminate video feeds that are no longer of interest to you. Do the following to terminate a video feed:

1. Open the *Video Feeds Window* within either the Desktop Client or the Web Console.
2. Click the *Configure* button.

3. When you hover your mouse over name of the feed that you want to terminate, you’ll see a + button and a - button. Click the - button.
4. Click the **Apply** button in the bottom right of the window. At this point, the feed will be terminated. **Note:** If you click the **Cancel** button, the feed termination is undone and the feed will continue operating.
5 VIRGO Command Line Interface

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

- Video Recognition Gateway (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 Portable Operating System Interface (POSIX) convention. This means that many of those options come in a short (single dash prefix) and a long (double dash prefix) form.

5.1 Command Line Options

Help

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

Shows all available VIRGO command line options.

5.1.1 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.

5.1.2 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.

**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**
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.

5.2 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]

PROD

Returns the currently active environment. This is the environment to which virgod and all of its virgafeedd 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.

5.3 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

```bash
> 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.

5.4 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

```bash
> 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

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

Prints the feed configuration JSON dictionary.

Update/set the configuration information for a feed

```bash
> 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 Video Feeds Properties 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. 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.
6 Video Feeds Properties

6.1 Global Properties

Below is a list of global processor properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status-interval</td>
<td>500</td>
<td>Status reporting time interval in milliseconds.</td>
</tr>
<tr>
<td>avigilon</td>
<td>N/A</td>
<td>Exposes the properties related to the SAFR Avigilon integration.</td>
</tr>
<tr>
<td>genetec</td>
<td>N/A</td>
<td>Exposes the properties related to the SAFR Genetec and SAFR Genetec FaceRec integrations.</td>
</tr>
<tr>
<td>digifort</td>
<td>N/A</td>
<td>Exposes the properties related to the SAFR Digifort integration.</td>
</tr>
<tr>
<td>milestone</td>
<td>N/A</td>
<td>Exposes the properties related to the SAFR Milestone integration.</td>
</tr>
<tr>
<td>geutebrueck</td>
<td>N/A</td>
<td>Exposes the properties related to the SAFR Geutebrueck integration.</td>
</tr>
<tr>
<td>videoInsight</td>
<td>N/A</td>
<td>Exposes the properties related to the SAFR Video Insight integration.</td>
</tr>
</tbody>
</table>

6.2 Monitoring Properties

Below is a list of all the monitoring properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarm.condition.delinquent.enabled</td>
<td>TRUE</td>
<td>Enables the delinquent alarm condition.</td>
</tr>
<tr>
<td>alarm.condition.delinquent.subject</td>
<td>“SAFR Feed Processor Unresponsive”</td>
<td>Sets the text of the subject line of the delinquent notification mail.</td>
</tr>
<tr>
<td>alarm.condition.delinquent.message</td>
<td>“SAFR Feed Processor %s is not responding.”</td>
<td>Sets the text of the email body of the delinquent notification mail.</td>
</tr>
<tr>
<td>alarm.condition.feed.error.enabled</td>
<td>TRUE</td>
<td>Enables the feed.error alarm condition.</td>
</tr>
<tr>
<td>alarm.condition.feed.error.subject</td>
<td>“SAFR Feed Error”</td>
<td>Sets the text of the subject line of the feed.error notification mail.</td>
</tr>
<tr>
<td>alarm.condition.feed.error.message</td>
<td>“SAFR Feed Processor %s feed %s encountered an error %d: %s.”</td>
<td>Sets the text of the email body of the feed.error notification mail.</td>
</tr>
<tr>
<td>alarm.condition.lowRAM.enabled</td>
<td>TRUE</td>
<td>Enables the lowRAM alarm condition.</td>
</tr>
<tr>
<td>alarm.condition.lowRAM.subject</td>
<td>“SAFR Feed Processor low on RAM”</td>
<td>Sets the text of the subject line of the lowRAM notification mail.</td>
</tr>
<tr>
<td>alarm.condition.lowRAM.threshold</td>
<td>0.5GB</td>
<td>The threshold, in GB, below which the lowRAM alarm condition is triggered.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>alarm.condition.lowRAM.message</td>
<td>“SAFR Feed Processor %s RAM remaining is at %f.1GB which is below healthy threshold of %f.1GB.”</td>
<td>Sets the text of the email body of the lowRAM notification mail.</td>
</tr>
<tr>
<td>alarm.condition.lowDisk.enabled</td>
<td>TRUE</td>
<td>Enables the lowDisk alarm condition.</td>
</tr>
<tr>
<td>alarm.condition.lowDisk.subject</td>
<td>“SAFR Feed Processor low on disk space”</td>
<td>Sets the text of the subject line of the lowDisk notification mail.</td>
</tr>
<tr>
<td>alarm.condition.lowDisk.thresholdGB</td>
<td>0</td>
<td>The threshold, in GB, below which the lowDisk alarm condition is triggered.</td>
</tr>
<tr>
<td>alarm.condition.lowDisk.message</td>
<td>“SAFR Feed Processor %s disk space remaining is at %f.1GB which is below healthy threshold of %f.1GB.”</td>
<td>Sets the text of the email body of the lowDisk notification mail.</td>
</tr>
<tr>
<td>alarm.condition.lowGPUMemory.enabled</td>
<td>TRUE</td>
<td>Enables the lowGPUMemory alarm condition.</td>
</tr>
<tr>
<td>alarm.condition.lowGPUMemory.subject</td>
<td>“SAFR Feed Processor low on GPU memory”</td>
<td>Sets the text of the subject line of the lowGPUMemory notification mail.</td>
</tr>
<tr>
<td>alarm.condition.lowGPUMemory.thresholdGB</td>
<td>0</td>
<td>The threshold, in GB, below which the lowGPUMemory alarm condition is triggered.</td>
</tr>
<tr>
<td>alarm.condition.lowGPUMemory.message</td>
<td>“SAFR Feed Processor %s GPU memory remaining is at %f.1GB which is below healthy threshold of %f.1GB.”</td>
<td>Sets the text of the email body of the lowGPUMemory notification mail.</td>
</tr>
<tr>
<td>alarm.condition.lowCPU.enabled</td>
<td>TRUE</td>
<td>Enables the lowCPU alarm condition.</td>
</tr>
<tr>
<td>alarm.condition.lowCPU.subject</td>
<td>“SAFR Feed Processor low on CPU”</td>
<td>Sets the text of the subject line of the lowCPU notification mail.</td>
</tr>
<tr>
<td>alarm.condition.lowCPU.thresholdPercent</td>
<td>15.0</td>
<td>The threshold, as a percentage, below which the lowCPU alarm condition is triggered.</td>
</tr>
<tr>
<td>alarm.condition.lowCPU.message</td>
<td>“SAFR Feed Processor %s CPU capacity remaining is at %f.1% which is below healthy threshold of %f.1%.”</td>
<td>Sets the text of the email body of the lowCPU notification mail.</td>
</tr>
<tr>
<td>alarm.condition.lowGPU.enabled</td>
<td>TRUE</td>
<td>Enables the lowGPU alarm condition.</td>
</tr>
<tr>
<td>alarm.condition.lowGPU.subject</td>
<td>“SAFR Feed Processor low on GPU”</td>
<td>The threshold, as a percentage, below which the lowGPU alarm condition is triggered.</td>
</tr>
<tr>
<td>alarm.condition.lowGPU.thresholdPercent</td>
<td>15.0</td>
<td>The threshold, as a percentage, below which the lowGPU alarm condition is triggered.</td>
</tr>
<tr>
<td>alarm.condition.lowGPU.message</td>
<td>“SAFR Feed Processor %s GPU capacity remaining is at %f.1% which is below healthy threshold of %f.1%.”</td>
<td>Sets the text of the email body of the lowGPU notification mail.</td>
</tr>
<tr>
<td>alarm.mail.enabled</td>
<td>TRUE</td>
<td>Enables alarm notification mails.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>alarm.mail.recipients</td>
<td>N/A</td>
<td>A comma-separated list of emails that specifies who should receive email notifications.</td>
</tr>
<tr>
<td>alarm.mail.username</td>
<td>N/A</td>
<td>The username for your email service.</td>
</tr>
<tr>
<td>alarm.mail.password</td>
<td>N/A</td>
<td>The password for your email service.</td>
</tr>
</tbody>
</table>

6.3 Feeds Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accelerator</td>
<td>“auto”</td>
<td>The type of acceleration that a feed should use. See the table “Feed accelerator types” below for a list of the supported acceleration types. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>capture.frame-delay</td>
<td>30</td>
<td>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. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>capture.maximum-frames</td>
<td>3600</td>
<td>If &gt; 0, enables the capture of “max-frames” frames; if 0, disables capture. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>capture.size</td>
<td>480</td>
<td>Specifies size of the smaller dimension of the image that will be sent. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.detect-badges</td>
<td>FALSE</td>
<td>Whether detection of badges should be enabled for this feed. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.detect-faces</td>
<td>TRUE</td>
<td>Whether detection of faces should be enabled for this feed.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>detector.detect-faces-input-size</td>
<td>“normal”</td>
<td>Sets the face detector input size. This property allows you to manage the trade-off between accuracy vs. speed. There are 3 possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>normal</strong> - This is the standard against which the other 2 possible values are measured.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>small</strong> - Decreased accuracy but increased speed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>large</strong> - Increased accuracy but decreased speed.</td>
</tr>
<tr>
<td>detector.detect-faces-service</td>
<td>“auto”</td>
<td>Specifies which face detection service will be used:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>standard</strong> - The standard facial detection service that SAFR uses.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>high-sensitivity</strong> - A high sensitivity facial detection service which has a lower latency and whose performance doesn’t degrade when multiple</td>
</tr>
<tr>
<td></td>
<td></td>
<td>faces are being analyzed simultaneously. The high sensitivity service consumes many more GPU resources than the standard service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>auto</strong> - This value will automatically select the high sensitivity service if sufficient GPU resources are available to run it. If there</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are insufficient GPU resources, then the standard service is used instead.</td>
</tr>
<tr>
<td>detector.detect-people</td>
<td>FALSE</td>
<td>Whether detection of people should be enabled for this feed. This detects any part of a person’s body and not just the face. This property is not</td>
</tr>
<tr>
<td></td>
<td></td>
<td>supported by SAFR Inside.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>detector.detect-people-every-n-frames</td>
<td>1</td>
<td>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. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.detect-people-input-size</td>
<td>&quot;normal&quot;</td>
<td>People detection balances speed and accuracy for best results. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>normal</em> - 416 pixel input. People detection will be the fastest with this input, but least accurate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>small</em> - 320 pixel input. People detection will be the fastest with this input, but least accurate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>large</em> - 608 pixel input. People detection will be the slowest with this input, but most accurate.</td>
</tr>
<tr>
<td>detector.detect-people-model</td>
<td>&quot;balanced&quot;</td>
<td>People detection balances speed and accuracy for best results. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>max-accuracy</em> - Use a larger model for better accuracy, but the speed will be slower.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>max-speed</em> - Use a smaller model for faster speed, but the accuracy will be lower.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>balanced</em> - Use a larger model for better accuracy, but the precision will be slightly lower resulting in faster speeds than the <em>max-accuracy</em> model without sacrificing too much accuracy. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.detect-vehicle</td>
<td>N/A</td>
<td>Internal use only. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.detect-vehicle-every-n-frames</td>
<td>N/A</td>
<td>Internal use only. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.detect-vehicle-input-size</td>
<td>N/A</td>
<td>Internal use only. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.detect-vehicle-model</td>
<td>N/A</td>
<td>Internal use only. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>detector.face-sensitivity-threshold</td>
<td>0</td>
<td>The sensitivity threshold when using the <em>High Sensitivity</em> facial detection service. The lower this value is, the more lenient the facial detection service will be when attempting to recognize a face, which can result in additional false positives. This setting is only available if you selected <em>High Sensitivity</em> for the <strong>Detection service</strong> setting above. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.final-face-selection-threshold</td>
<td>0.9</td>
<td>The final face candidate threshold that is used during face detection. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.initial-face-selection-threshold</td>
<td>0.8</td>
<td>The initial face candidate threshold that is used during face detection. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.maximum-concurrent-detections</td>
<td>0</td>
<td>The maximum number of concurrent detections to allow. 0 means to automatically set this. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.maximum-input-resolution</td>
<td>720</td>
<td>Maximum resolution of the Input image. Bigger images are scaled down (aspect-ratio preserving) to this resolution before detection.</td>
</tr>
<tr>
<td>detector.maximum-input-resolution-badge</td>
<td>4320</td>
<td>Maximum resolution of the Input image. Bigger images are scaled down (aspect-ratio preserving) to this resolution before detection.</td>
</tr>
<tr>
<td>detector.middle-face-selection-threshold</td>
<td>0.85</td>
<td>The middle face candidate threshold that is used during face detection. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.minimum-consecutive-detections-required-person</td>
<td>0</td>
<td>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. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>detector.minimum-consecutive-detections-required-vehicle</td>
<td>N/A</td>
<td>Internal use only. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.minimum-required-badge-size</td>
<td>0</td>
<td>The minimum size of badges to accept from the detector. Only badges with at least this size are eligible for recognition. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.minimum-required-face-size</td>
<td>0</td>
<td>The minimum size of faces to accept from the detector. Only faces with at least this size are eligible for recognition.</td>
</tr>
<tr>
<td>detector.minimum-required-person-to-screen-height-proportion</td>
<td>0</td>
<td>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. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.minimum-required-vehicle-to-screen-proportion</td>
<td>N/A</td>
<td>Internal use only. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.minimum-searched-badge-size</td>
<td>20</td>
<td>The badge detector is advised to search for badges of at least this size. This value is applied while searching the image. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.minimum-searched-face-size</td>
<td>80</td>
<td>The face detector is advised to search for faces of at least this size. This value is applied while searching the image. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>detector.person-detection-threshold</td>
<td>0.4</td>
<td>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. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>detector.person-separation-threshold</td>
<td>0.45</td>
<td>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. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>directory</td>
<td>N/A</td>
<td>Directory name.</td>
</tr>
<tr>
<td>enabled</td>
<td>FALSE</td>
<td>Marks the feed as enabled or disabled.</td>
</tr>
<tr>
<td>input.back-channel.mobotix.cash-point</td>
<td>“None”</td>
<td>When the connected camera is a Mobotix camera, this property must be set to the configured cash point within the Mobotix app for the back-channel to work.</td>
</tr>
<tr>
<td>input.back-channel.type</td>
<td>“None”</td>
<td>When the connected camera is a Mobotix camera, you can set this property to “Mobotix MX” in order to have SAFR report STRANGER and RECOGNIZED event types to the camera. This feature is necessary if you want to make use of the Mobotix app. If the connected camera isn’t a Mobotix camera, this property doesn’t have any effect.</td>
</tr>
<tr>
<td>input.contrast-enhancement.detection-only</td>
<td>FALSE</td>
<td>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. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.contrast-enhancement.enabled</td>
<td>FALSE</td>
<td>Enables contrast enhancement of the input video frame. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.contrast-enhancement.exposure-boost</td>
<td>0</td>
<td>The boost for contrast enhancement. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.contrast-enhancement.low-light-threshold</td>
<td>0.02</td>
<td>Low-light-threshold for contrast enhancement. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>input.crop-rectangle.enabled</td>
<td>FALSE</td>
<td>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) \times (1, 1)). This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.crop-rectangle.height</td>
<td>1</td>
<td>The normalized height value relative to the video of how big the crop rectangle size should be. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.crop-rectangle.left</td>
<td>0</td>
<td>The normalized left coordinate relative to the video of where the crop rectangle origin should be. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.crop-rectangle.top</td>
<td>0</td>
<td>The normalized top coordinate relative to the video of where the crop rectangle origin should be. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.crop-rectangle.width</td>
<td>1</td>
<td>The normalized width value relative to the video of how big the crop rectangle size should be. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.lens-correction.enabled</td>
<td>FALSE</td>
<td>Enables lens correction for the camera. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.lens-correction.k1</td>
<td>0</td>
<td>The “k1” lens correction factor. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.lens-correction.k2</td>
<td>0</td>
<td>The “k2” lens correction factor. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.loop</td>
<td>FALSE</td>
<td>Enables looping of the feed input. Only video file-based feeds support looping. Ignored for cameras. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.mirroring.enabled</td>
<td>FALSE</td>
<td>Whether the video image should be mirrored before detection and recognition operations are executed. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.password</td>
<td>N/A</td>
<td>Password of the person accessing the video stream. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>input.rotation.angle</td>
<td>0</td>
<td>Whether the video should be rotated before detection and recognition operations are executed. Valid values are 0, 90, 180, and 270. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.stream.id</td>
<td>N/A</td>
<td>Identifier used to connect to a stream if the URL is blank. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.stream.name</td>
<td>N/A</td>
<td>A friendly name used for display purposes. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.stream.rtsp.transport</td>
<td>“udp”</td>
<td>The transport protocol that should be used while accessing the RTSP video stream. Must be one of “udp”, “tcp”, or “udp-multicast”. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>input.stream.url</td>
<td>N/A</td>
<td>The video stream URL. The URL must point to a RTSP, HTTP or FILE stream.</td>
</tr>
<tr>
<td>input.type</td>
<td>“stream”</td>
<td>The type of feed input; either “stream” or “file”.</td>
</tr>
<tr>
<td>input.user-name</td>
<td>N/A</td>
<td>Username of the person accessing the video stream. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>mode</td>
<td>“Enrolled and Stranger Monitoring”</td>
<td>Specifies which video processing mode the feed is using.</td>
</tr>
<tr>
<td>recognizer.3d-liveness-threshold</td>
<td>0.6</td>
<td>Windows only. Specifies the 3d liveness threshold. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.detect-3d-liveness</td>
<td>FALSE</td>
<td>Windows only. Enables 3D liveness. 3D liveness is a special feature of certain Intel RealSense camera models that allows them to distinguish flat images from 3 dimensional ones, thus allowing SAFR to tell the difference between a real face and a photo. This feature only works with Intel RealSense D415 and D435 cameras; if you don’t have any cameras of those types connected to SAFR, then this feature will not work. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.detect-age</td>
<td>FALSE</td>
<td>Enables the detection of age information.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>recognizer.detect-gender</td>
<td>FALSE</td>
<td>Enables the detection of gender information.</td>
</tr>
<tr>
<td>recognizer.detect-identity</td>
<td>TRUE</td>
<td>Enables detection of an identity, which matches against the existing database of people (identities).</td>
</tr>
<tr>
<td>recognizer.detect-mask</td>
<td>FALSE</td>
<td>When enabled, SAFR will evaluate all occluded faces to see if they're covered by a mask. If they are, then SAFR will use the mask enhanced model to attempt to recognize the face behind the mask. If the occluded face isn't covered by a mask, then the normal occluded model will be used instead.</td>
</tr>
<tr>
<td>recognizer.detect-mask-model</td>
<td>“precise”</td>
<td>Specifies the model to be used for mask detection. There are 3 possible values: <strong>Precise</strong>: This model produces the least number of false positives (i.e. detecting that a person is wearing a mask but there is no mask), but it suffers from the lowest true positive rate. (i.e. detecting masks that are actually there) <strong>Sensitive</strong>: This model produces the highest true positive rate, but it suffers from the highest number of false positives. <strong>Normal</strong>: This model produces a moderate amount of both false positives and true positives.</td>
</tr>
<tr>
<td>recognizer.detect-mask-threshold</td>
<td>0.5</td>
<td>Specifies the threshold at and above which mask detection will conclude that mask=true.</td>
</tr>
<tr>
<td>recognizer.detect-occlusion</td>
<td>FALSE</td>
<td>Enables occlusion detection during recognition.</td>
</tr>
<tr>
<td>recognizer.detect-pose-action</td>
<td>FALSE</td>
<td>Enables the pose liveness action recognizer. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.detect-sentiment</td>
<td>FALSE</td>
<td>Enables the detection of sentiment information.</td>
</tr>
<tr>
<td>recognizer.detect-smile-action</td>
<td>FALSE</td>
<td>Enables the smile action recognizer. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.identity-masked-threshold-offset</td>
<td>0</td>
<td>Sets the identity threshold when detecting masks.</td>
</tr>
<tr>
<td>recognizer.identity-proximity-threshold-allowance</td>
<td>0.13</td>
<td>A boost value that is added to the Identity Recognition Threshold.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>recognizer.identity-recognition-threshold</td>
<td>0.54</td>
<td>Identity recognition threshold.</td>
</tr>
<tr>
<td>recognizer.learning-enabled</td>
<td>FALSE</td>
<td>Enables the feed to learn new identities.</td>
</tr>
<tr>
<td>recognizer.learn-occluded-faces</td>
<td>FALSE</td>
<td>Enables 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.</td>
</tr>
<tr>
<td>recognizer.max-occlusion</td>
<td>0</td>
<td>The maximum occlusion value that is allowed when adding a new candidate images into the Person Directory. If the face is occluded with a value greater than this then the face will not be added, but if it’s less than or equal to this value then it will be added.</td>
</tr>
<tr>
<td>recognizer.maximum-clip-ratio</td>
<td>0.1</td>
<td>The maximum clip ratio on either side the recognition candidate might have.</td>
</tr>
<tr>
<td>recognizer.maximum-clip-ratio-identification</td>
<td>0</td>
<td>The maximum clip ratio on either side the insertion candidate might have.</td>
</tr>
<tr>
<td>recognizer.maximum-pitch-identification</td>
<td>0.4</td>
<td>The maximum pitch value used to determine if the face is looking straight ahead. The pitch value is the forward/backward movement of the face.</td>
</tr>
<tr>
<td>recognizer.maximum-roll-identification</td>
<td>0.15</td>
<td>The maximum roll value used to determine if the face is looking straight ahead. The roll value is the side to side tilt movement of the face.</td>
</tr>
<tr>
<td>recognizer.maximum-yaw-identification</td>
<td>0.4</td>
<td>The maximum yaw value used to determine if the face is looking straight ahead. The yaw value is the side to side movement of the face.</td>
</tr>
<tr>
<td>recognizer.minimum-center-pose-quality</td>
<td>0.05</td>
<td>The minimum center pose quality that a recognition candidate must have in order to allow the addition of the candidate image into the Person Directory.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>recognizer.minimum-center-pose-quality-identification</td>
<td>0.45</td>
<td>The minimum center pose quality that a recognition candidate must have in order to allow the addition of the candidate image into the Person Directory.</td>
</tr>
<tr>
<td>recognizer.minimum-center-pose-quality-merging</td>
<td>0.59</td>
<td>The minimum center pose quality that a recognition candidate must have in order to allow merging.</td>
</tr>
<tr>
<td>recognizer.minimum-face-contrast-quality</td>
<td>0.1</td>
<td>The minimum face contrast quality that a face image must have before recognition is attempted.</td>
</tr>
<tr>
<td>recognizer.minimum-face-contrast-quality-identification</td>
<td>0.3</td>
<td>The minimum face contrast quality that a recognition candidate must have in order to allow the addition of the candidate image into the Person Directory.</td>
</tr>
<tr>
<td>recognizer.minimum-face-contrast-quality-merging</td>
<td>0.45</td>
<td>The minimum face contrast quality that a recognition candidate must have in order to allow merging.</td>
</tr>
<tr>
<td>recognizer.minimum-face-sharpness-quality</td>
<td>0.1</td>
<td>The minimum face sharpness quality that a face image must have before recognition is attempted.</td>
</tr>
<tr>
<td>recognizer.minimum-face-sharpness-quality-identification</td>
<td>0.3</td>
<td>The minimum face sharpness quality that a recognition candidate must have in order to allow the addition of the candidate image into the Person Directory.</td>
</tr>
<tr>
<td>recognizer.minimum-face-sharpness-quality-merging</td>
<td>0.45</td>
<td>The minimum face sharpness quality that a recognition candidate must have in order to allow merging.</td>
</tr>
<tr>
<td>recognizer.minimum-face-size</td>
<td>80</td>
<td>The minimum size of faces to detect. This value is applied after searching the image.</td>
</tr>
<tr>
<td>recognizer.minimum-face-size-identification</td>
<td>120</td>
<td>The minimum resolution that a recognition candidate image must have in order to allow the addition of the candidate image into the Person Directory.</td>
</tr>
<tr>
<td>recognizer.minimum-face-size-merging</td>
<td>220</td>
<td>The minimum resolution a recognition candidate must have in order to allow merging.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>recognizer.pose-action-max-cpq-jump-after-discontinuity</td>
<td>0.15</td>
<td>The maximum change between samples while the pose is changing from center to profile if lingering. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-action-max-cpq-jump-in-continuity</td>
<td>0.18</td>
<td>The maximum change between samples while the pose is changing from center to profile. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-action-max-profile-confidence-end</td>
<td>0.60</td>
<td>The maximum profile pose confidence to allow during the final profile pose detection phase. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-action-max-profile-pose-quality</td>
<td>0.26</td>
<td>The maximum center pose quality to use when detecting the final profile pose. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-action-max-profile-pose-roll</td>
<td>0.3</td>
<td>The maximum roll threshold in either direction in which the face can rotate when determining whether the face is in profile pose. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-action-min-center-pose-quality</td>
<td>0.5</td>
<td>The minimum center pose quality to use when detecting the initial center pose. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-action-min-detections-per-second</td>
<td>15</td>
<td>The minimum number of frames per second that is required during the process. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-action-min-profile-confidence-start</td>
<td>0.35</td>
<td>The minimum profile pose confidence to allow during the initial center pose detection phase. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-action-min-profile-pose-yaw</td>
<td>0.81</td>
<td>The minimum profile pose yaw value that is required during the final profile pose detection phase. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-action-min-profile-similarity</td>
<td>0.86</td>
<td>The minimum similarity score required when verifying the final profile pose. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>recognizer.pose-action-min-transition-poses</td>
<td>2</td>
<td>The minimum number of required center pose samples during the transition from center to profile pose. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-action-profile-pose-required-confirmations</td>
<td>1</td>
<td>The number of consecutive confirmations required to enter the final profile pose detection phase. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-action-required-confirmations</td>
<td>3</td>
<td>The number of consecutive confirmations required to enter the initial center pose detection phase. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.pose-configuration-identification-enabled</td>
<td>FALSE</td>
<td>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. When pose configuration is enabled, then recognizer.minimum-center-pose-quality is ignored and the following 3 properties are used instead: recognizer.maximum-yaw-identification, recognizer.maximum-pitch-identification, and recognizer.maximum-roll-identification.</td>
</tr>
<tr>
<td>recognizer.smile-duration</td>
<td>0</td>
<td>The amount of time that the smile should last. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.smile-identity-threshold-boost</td>
<td>0.13</td>
<td>The smile threshold to boost temporarily during the smile action. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.smile-pre-delay</td>
<td>100</td>
<td>The amount of time that there should be no smile. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.smile-thresholds-enabled</td>
<td>FALSE</td>
<td>Enables the smile threshold values. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>recognizer.smile-threshold-neutral</td>
<td>-0.1</td>
<td>The threshold in which there is no smile. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>recognizer.smile-threshold-smiling</td>
<td>0.7</td>
<td>The threshold in which there is a smile. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>reporter.delay</td>
<td>0</td>
<td>Delay the event reporting to the server by this amount in seconds.</td>
</tr>
<tr>
<td>reporter.enabled</td>
<td>TRUE</td>
<td>Enables or disables event reporting.</td>
</tr>
<tr>
<td>reporter.minimum-event-duration-identified</td>
<td>0</td>
<td>The minimum allowed recognized person event duration in seconds. Events below this value will not be reported.</td>
</tr>
<tr>
<td>reporter.minimum-event-duration-unidentified</td>
<td>1500</td>
<td>The minimum allowed unrecognized person event duration in seconds. Events below this value will not be reported.</td>
</tr>
<tr>
<td>reporter.report-event-face</td>
<td>TRUE</td>
<td>Enables the inclusion of face thumbnails in event reports.</td>
</tr>
<tr>
<td>reporter.report-event-scene</td>
<td>FALSE</td>
<td>Enables the inclusion of scene images in event reports.</td>
</tr>
<tr>
<td>reporter.report-secondary-events</td>
<td>FALSE</td>
<td>Reports secondary events. Secondary events are events that are associated with a primary event via the rootEventId property in the event. 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.</td>
</tr>
<tr>
<td>reporter.report-speculated-events</td>
<td>TRUE</td>
<td>Reports events for speculated faces. Speculated faces are faces that aren’t a 100% match, but are close.</td>
</tr>
<tr>
<td>reporter.report-stranger-events</td>
<td>TRUE</td>
<td>Reports events for people that are strangers. These are people not registered by the system after running facial recognition on the face.</td>
</tr>
<tr>
<td>reporter.report-unrecognizable-events</td>
<td>TRUE</td>
<td>Reports events for people that are not recognizable.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>reporter.stranger-events.only-if-occluded</td>
<td>FALSE</td>
<td>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.</td>
</tr>
<tr>
<td>reporter.stranger-maximum-age</td>
<td>0</td>
<td>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.</td>
</tr>
<tr>
<td>reporter.stranger-minimum-age</td>
<td>0</td>
<td>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.</td>
</tr>
<tr>
<td>reporter.update-images</td>
<td>FALSE</td>
<td>Updates the thumbnail images with higher quality images during the course of the event if possible.</td>
</tr>
<tr>
<td>site</td>
<td>N/A</td>
<td>Site name, if any.</td>
</tr>
<tr>
<td>source</td>
<td>N/A</td>
<td>Source name.</td>
</tr>
<tr>
<td>statistics.enabled</td>
<td>FALSE</td>
<td>Specifies whether VIRGO should record and report statistics for this feed.</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.bottom-boundary</td>
<td>0</td>
<td>The percentage of the bottom side of the camera view field to exclude from direction of travel event reporting. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.down</td>
<td>FALSE</td>
<td>Enables direction of travel detection in the downward direction. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.down-distance</td>
<td>0.1</td>
<td>The percentage of the camera view that a tracked person can travel in a downward direction before a direction of travel event is generated. This property doesn’t have any effect if tracker.detect-direction-of-travel.person.down is set to FALSE. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.left</td>
<td>FALSE</td>
<td>Enables direction of travel detection in the leftward direction. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.left-boundary</td>
<td>0</td>
<td>The percentage of the left side of the camera view field to exclude from direction of travel event reporting. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.left-distance</td>
<td>0.1</td>
<td>The percentage of the camera view that a tracked person can travel in a leftward direction before a direction of travel event is generated. This property doesn’t have any effect if tracker.detect-direction-of-travel.person.left is set to FALSE. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.right</td>
<td>FALSE</td>
<td>Enables direction of travel detection in the rightward direction. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.right-boundary</td>
<td>0</td>
<td>The percentage of the right side of the camera view field to exclude from direction of travel event reporting. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.right-distance</td>
<td>0.1</td>
<td>The percentage of the camera view that a tracked person can travel in a rightward direction before a direction of travel event is generated. This property doesn’t have any effect if tracker.detect-direction-of-travel.person.right is set to FALSE. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.top-boundary</td>
<td>0</td>
<td>The percentage of the top side of the camera view field to exclude from direction of travel event reporting. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.up</td>
<td>FALSE</td>
<td>Enables direction of travel detection in the upward direction. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tracker.detect-direction-of-travel.person.up-distance</td>
<td>0.1</td>
<td>The percentage of the camera view that a tracked person can travel in an upward direction before a direction of travel event is generated. This property doesn’t have any effect if tracker.detect-direction-of-travel.person.up is set to FALSE. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.enable-face-bounds-prediction</td>
<td>TRUE</td>
<td>Enables face bounds prediction, which predicts which direction the face is moving to maintain tracking.</td>
</tr>
<tr>
<td>tracker.enable-face-size-correlation</td>
<td>TRUE</td>
<td>Enables face correlation of tracked faces, which compares detected faces looking for a change in area.</td>
</tr>
<tr>
<td>tracker.enable-high-precision</td>
<td>FALSE</td>
<td>Enables high precision tracking, which decreases event fragmentation and increases the stickiness of SAFR’s tracking algorithm at the cost of computer processing power. This property should be enabled if you are experiencing duplicate or missing Direction of Travel events. See Camera Preferences for information about the Direction of Travel feature. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.identity-relearn-interval-days</td>
<td>0</td>
<td>Updates the identity only when the currently saved identity is older than the updated identity.</td>
</tr>
<tr>
<td>tracker.identity-update-better-image</td>
<td>FALSE</td>
<td>Updates the identity in the case where the identity currently saved is of lower quality (in all aspects) than the updated identity.</td>
</tr>
<tr>
<td>tracker.initial-recognition-attempts</td>
<td>3</td>
<td>The number of initial recognition attempts to make on an unrecognized person as fast as possible.</td>
</tr>
<tr>
<td>tracker.maximum-linger-frames</td>
<td>30</td>
<td>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.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tracker.max-position-change-relative-to-face</td>
<td>115</td>
<td>The maximum position change, specified in percentage relative to the face size, to continue tracking.</td>
</tr>
<tr>
<td>tracker.max-size-change-relative-to-face</td>
<td>50</td>
<td>The maximum size change, specified in percentage relative to the object size, to continue tracking.</td>
</tr>
<tr>
<td>tracker.min-failed-recognitions-to-stop-tracking-identity</td>
<td>3</td>
<td>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.</td>
</tr>
<tr>
<td>tracker.minimum-number-identical-recognitions-learn</td>
<td>2</td>
<td>The number of consecutive recognitions that need to occur before adding a new identity to the system.</td>
</tr>
<tr>
<td>tracker.minimum-number-identical-recognitions-lock</td>
<td>1</td>
<td>The number of consecutive recognition attempts that we must run and produce the same person identity before we lock onto this identity.</td>
</tr>
<tr>
<td>tracker.minimum-required-consecutive-badge-detections</td>
<td>0</td>
<td>The number of consecutive detections that are required before reporting that the object (based on object id) was actually detected. This property can be used to filter out false positives. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.minimum-required-consecutive-mask-detections</td>
<td>1</td>
<td>The number of consecutive detections that are required before reporting that the masked person was detected. This property can be used to filter out false positives. This property is not supported by SAFR Inside.</td>
</tr>
<tr>
<td>tracker.reconfirmation-interval</td>
<td>1000</td>
<td>Identity reconfirmation time interval in ms.</td>
</tr>
<tr>
<td>tracker.stop-tracking-on-failed-re-recognition</td>
<td>FALSE</td>
<td>If recognition fails when re-recognizing a person then delete the identity that was created.</td>
</tr>
</tbody>
</table>
### 6.4 Update Properties

The update properties are not intended for public consumption at this time.

<table>
<thead>
<tr>
<th>Property</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client-type</td>
<td>OS-defined client type</td>
<td>Internal use only.</td>
</tr>
<tr>
<td>version</td>
<td>N/A</td>
<td>Internal use only.</td>
</tr>
<tr>
<td>enabled</td>
<td>FALSE</td>
<td>Internal use only.</td>
</tr>
<tr>
<td>progress-status</td>
<td>N/A</td>
<td>Internal use only.</td>
</tr>
<tr>
<td>progress-url</td>
<td>N/A</td>
<td>Internal use only.</td>
</tr>
<tr>
<td>download-url</td>
<td>N/A</td>
<td>Internal use only.</td>
</tr>
<tr>
<td>log-enabled</td>
<td>FALSE</td>
<td>Internal use only.</td>
</tr>
<tr>
<td>progress-interval</td>
<td>1000</td>
<td>Internal use only.</td>
</tr>
</tbody>
</table>
7 Processing Video Files

Both the Video Feeds Window (located in the Desktop Client and the Web Console) and the Command Line Interface can be used to process video files for person detection and/or recognition. This can be useful to recognize faces for purposes of identifying people or generating events on pre-recorded videos. For example, consider the scenario where you have video footage from cameras throughout a facility and you want to determine where and when a person of interest appears in those videos. To do this, you would register the people of interest to the Person Directory (if they weren’t already registered) and you could then process the videos in Enrolled Monitoring video processing mode in order to identify when and where that individual appears.

The Video Feeds Window is the best video file processing option when you only have 1 video to process. When you want to process 2 or more video files, the Command Line Interface is the better choice.

**Note** When you switch control from the Video Feeds Window to the Command Line Interface, any feeds that you configured in the Video Feeds Window will be stopped and will not be visible from the Command Line Interface. The feeds still exist in the Video Feeds Window, but you cannot start or manage those feeds from the Command Line Interface. You must re-create new feeds for the Command Line Interface.

### 7.1 Process Files with the Video Feeds Window

To process video files, the Video Feeds Window must have access to those files on the local file system. The window can work with files on an internal hard drive, an external drive attached to the machine, or a network share mounted to the file system. In this example we will use the following directories.

```
mkdir -p /files/videos
mkdir -p /files/feeds
```

- The **videos** directory will be used to store the video files that VIRGO will process.
- The **feeds** directory will be used to store feed configuration files used by VIRGO. Feed configuration files store information necessary to process video from a file or camera.
- **videos** and **feeds** can be either an alias or a directory mounted to an external file system.

#### 7.1.1 Mount Host Filesystem to VIRGO Container

In order to process files that are on a Linux machine, you must mount a file system on the host drive to the Docker container running Video Recognition Gateway (VIRGO).

You can mount a host drive onto the Docker instance by doing the following:

1. Add the following lines to the end of the file “/opt/RealNetworks/SAFR/virgo/app/docker-compose.yml”

```
vi /opt/RealNetworks/SAFR/virgo/app/docker-compose.yml

- /files:/files
```

The hyphen should be included.
The resulting file should look something like this. (DON’T USE THE EXAMPLE BELOW - Your system will have different version info.)

```
version: "3.6"
services:  
virgo:  
  image: safr_virgo:1.2.22  
  container_name: safr_virgo  
  restart: on-failure  
  pid: "host"  
volumes:
```
- /opt/RealNetworks/SAFR/virgo/config/:/etc/virgo
- /files/:/files

- The mount format is `<path on host>`:<path on VIRGO Docker container>.
  - This is equivalent to creating an alias on the Docker container pointing to a actual directory on the host.
  - `<path on VIRGO Docker container>` can be any path. It should not exist already.
  - `<path on host>` should be a real directory. It can be located anywhere on the host drive.
  - Using the same path for both makes it less confusing when typing filepaths because there will be no need to remember the Docker path and to append the relative host path. Just use the fully qualified host path when defining the location of a video in the VIRGO config.

2. Run the following commands to re-mount all paths on the Docker container:

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

3. Check to see if the folder is mounted correctly in the Docker container by doing the following:
   1. Run the following command to sign into the Docker container:

      ```bash
docker-exec -it safr_virgo bash
```

   2. Try to list the directory of the location mounted above:

      ```bash
ls /files
```

You should see the two folders (feeds and videos) that you created listed under this location.

7.1.2 Using the Video Feeds Window

To use the Video Feeds Window, do the following:

1. Connect to the Web Console or open the Desktop Client. See here for details about the Web Console and how to connect to it.
2. Navigate to the **Video Feeds** tab and select **Processor Status**. Assuming you have no other video clients connected to the server (i.e. from additional connected VIRGO feeds or Desktop Clients), you will see a single entry representing VIRGO with no feeds.

3. Click **Config** to add feeds.
4. Enter values for the following fields:
   - **name** - User-defined name. We recommended that you only use ASCII letters and numbers (i.e. no spaces) since this is used to reference the feed on the command line.
   - **mode** - Defines the default settings for SAFR detection and recognition.
   - **input.stream.url** - Source camera URL or filename of the video to use as input. In this case we pre-pend the value with "file://" to indicate that we’re processing a file. The path portion is the path to the video file relative to the Docker container, as explained above.
   - In the screenshot below, the **enabled** field still has its default value of “false”. You should set it to “true” when you are ready to start processing the file.
5. Before clicking **Apply**, add an additional field to specify the start time of the video. Do this by doing the following:
   1. Click **Add Attributes** under the feed.
   2. Search for and select the “reporter.events-initial-date-offset” property.

6. Once added set a date value as Epoch in milliseconds. For example, for Aug 1, 2019 at 11 AM, the Epoch value is 1564682400000.
   - This causes generated events to be recorded at the correct point in time rather than being recorded at the default Epoch value of “0” or January 1, 1970.
   - Try the Chrome Extension ‘utime’ to create Epoch times. This extension allows you to type any date/time and get Epoch values as well as natural language strings such as ‘1 hour ago’.

   **Note**: Be sure to set the time format to “milliseconds” or else the time will be off by many years.

7. Change the **enabled** field to “true” and click **Apply** to save the feed and start processing the video.

8. If all goes well you should see the following:

9. For a few moments you may see the **Last Config** date in red which means that changes have not yet been applied to VIRGO. Once the changes have been applied, VIRGO should start processing and you should see the **Status** reported as “ok”. You can click **View** to see the current frame of video being processed. This will be updated every second or so.

10. See Troubleshooting Feeds below if the video does not start processing successfully.

11. You can move to the **Events** tab to view the events being processed. You may need to change the search criteria to include August 1, 2019 for the events to show up.

### 7.1.3 Troubleshooting Feeds

If there is a problem, first check the **More** button on the VIRGO feed as shown below:
Additional possible troubleshooting steps:

- If you see status as “inactive”, check the enabled option in the queue file. It should be “true”, as in:
  
  "enabled" : true

- To start the feed, you can run the VIRGO command: `docker exec -it safr_virgo ./virgo feed start queue1`

- To avoid this problem with subsequent files, you may want to edit the feed configuration file to set `enabled` to “true”.

If the steps above do not fix the problem, then it may help to look at the service monitor or service log. Both must be started before running VIRGO in order to capture the error output.

### 7.1.3.1 View the VIRGO Service Monitor

To view the service monitor, log into SAFR Server and run the following command:

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

or

```
docker exec -it safr_virgo bash ./virgo service monitor
```

You should see the following:
7.1.3.2 View the VIRGO Service Log

To view the service log, run the following command:

```bash
sudo docker exec -it safr_virgo /opt/RealNetworks/virgo/virgo service log d/feed d/http-cop
```

or

```bash
docker exec -it safr_virgo bash
virgo service log d/feed d/http-cop
```

With the above running, restart the feed. In the Video Feeds Window you may need to do the following to successfully restart the feed:

1. Set the feed `enabled` flag to “false”.
2. Save.
3. Change the `enabled` flag back to “true”.

Additional Notes:

- Information will be printed to the screen. Use Unix Redirect (>) or “tee -a” command to write output to file.
- Additional options for VIRGO logging can be found here.
- It is useful to have both the service monitor and service log open in separate windows as you start the video feed.

7.2 Process Files with the Command Line Interface

7.2.1 Create a Feed Template

To run VIRGO, you need a feed configuration file. You can use the configuration file we created above in VIRGA as a starting point. This will include most of the common configuration properties needed.

1. Export a feed configuration file from VIRGO to use as a template.
2. Get a list of VIRGO feeds.

```bash
> sudo docker exec -it safr_virgo ./virgo feed list
EnrolledAndStrangerMonitoring
```

- EnrolledAndStrangerMonitoring should be the only feed listed.
3. Save that configuration to a file called `template.json`. 
> sudo docker exec -it safr_virgo ./virgo feed get

EnrolledAnd StrangerMonitoring
/files/feeds/template.json

- The folder /files/feeds here is relative to the VIRGO Docker container. This path is mounted to the same path (/files/feeds) on the host filesystem.

4. Ensure that the file was written.

> cd /files/feeds
> ls


5. Edit the file ‘template.json’ if desired. Generally it’s easier to perform editing in the Video Feeds Window before exporting.

- The Video Feeds Window unnecessarily adds extra escapes to forward slashes in the JSON. This makes the string much harder to read. When using the Video Feeds Window to generate configuration files, the file path string can be simplified as follows:
  - Change this: file://opt/RealNetworks/virgo/files/videos/cam4Aug1_output000.mp4
  - To this: file:///opt/RealNetworks/virgo/files/videos/cam4Aug1_output000.mp4

6. Create a copy of the template file and name it ‘queue1.json’. You’re making a copy because you’ll maintain one file for each feed.

> cd /files/feeds
> cp template.json queue1.json

7. Update the feed file by setting values for site, source, start time, and filename. Use the utility script update_virgo_feed.sh documented below for this purpose.

> update_virgo_feed.sh queue1.json Site01 Camera10 1564682400000 /quotesingle.ts1 /quotesingle.ts1 /quotesingle.ts1 /quotesingle.ts1

- See the script’s documentation below for a description of its command line arguments.
- The file path assumes you have mounted the VIRGO Docker container to /files in the host drive as explained above.

This will set the values for the respective fields in queue1.json. View the file in an editor to confirm edits were made.

7.2.2 Create Feed

You can assign the feed configuration file created in the previous section to a feed. Before doing that, you need to first switch control from the Video Feeds Window to the Command Line Interface. As explained above, the Video Feeds Window and the Command Line Interface do not share the same configuration files. That is why you first exported a configuration file above from the Video Feeds Window. Now that you have a configuration file, you’re ready to switch control to the Command Line Interface. The Command Line Interface will start out with no feeds defined. You will then create feeds from the configuration file saved above.

1. Switch control from the Video Feeds Window to the Command Line Interface

> sudo docker exec -it safr_virgo ./virgo administrator set virgo

Note: This changes control of VIRGO from the Video Feeds Window to the Command Line Interface. The Video Feeds Window will no longer be able to start or stop VIRGO feeds until control is restored

45
back to it. Note that all feeds added to the Video Feeds Window are not available to the Command Line Interface. You will re-create feeds as described below.

2. Create a new feed ‘queue1’ from the queue1.json feed configuration file created above.

```bash
> sudo docker exec -it safr_virgo ./virgo feed add queue1 /files/feeds/queue1.json
```

3. Confirm that the feed was added to VIRGO.

```bash
> sudo docker exec -it safr_virgo ./virgo feed list
queue1
```

4. VIRGO will attempt to start processing the feed right away. Run VIRGO Service Status or VIRGO Service Monitor/

```bash
> sudo docker exec -it safr_virgo ./virgo service status
queue1 ok
```

See the Service Monitor section above for information on how to run VIRGO Service Monitor. Initially you may see a status reported as “prerolling” which means the feed is starting up.

After processing is done, the status should change from “ok” to “eos” (meaning End of Stream). If any other status is shown, see Troubleshooting Feeds above.

### 7.2.3 Edit Feed and Reload

To edit a feed and re-load, do the following.

1. Stop the feed with following command:

```bash
docker exec -it safr_virgo ./virgo feed stop queue1
```

2. Edit the file and correct problems. Re-load the file from disk using following command:

```bash
docker exec -it safr_virgo ./virgo feed set queue1 /files/feeds/queue1.json
```

3. Restart the feed with following command:

```bash
docker exec -it safr_virgo ./virgo feed start queue1
```

### 7.3 Batch Processing Files with the Command Line

When batch processing multiple files, it’s most efficient to process two or more files in parallel. The optimal number of parallel processes is a function of processor speed and disk/network speeds and should be determined by experimentation on your hardware. VIRGO will try to process files as fast as possible and leverage multiple CPU cores and GPUs but there is generally a limit to how many CPU cores it uses. Processing more files in parallel will allow you to fully utilize the machine resources.

In general the process is as follows:

- N queues are created. Each queue will be able to process 1 file in parallel.
- 1 file is set to be processed on each queue.
- Each of the queues is started. (As long as enabled=true is set, processing will start as soon as the feed file is assigned to the queue.)
- As each queue is started, its status will be “ok”.
- As each queue completes, the status will change to ‘eos’. 

46
• Any feed with status ‘eos’ is assigned the next file in line to be processed.
• Files continue to be added to queues until all files are processed.

7.3.1 Create Feed Queues

In order to create a feed queue, create an additional feed called queue2.

```bash
> cp template.json queue2.json
> update_virgo_feed.sh queue2.json Site06 Camera04
  'file://files/videos/test/vid01.mp4'
> sudo docker exec -it safr_virgo ./virgo feed add queue2
  /files/feeds/queue2.json
```

You’ll maintain different queue files (queue1.json and queue2.json) in order to allow queue1 and queue2 to be processed independently. Each queue file serves as a record of what is running on the current queue. If an error occurs on one of the queues, you can use the `get_input_stream.sh` script to identify the file that was not processed for the queue and write it to an error log file.

The 2nd feed should start automatically. If you check the VIRGO Service Monitor, you should see something like the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Feed</th>
<th>Epoch</th>
<th>P-Time</th>
<th>Resolution</th>
<th>FPS</th>
<th>DPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>eos</td>
<td>queue1</td>
<td>04:45:50</td>
<td>00:00:04.343</td>
<td>1920x1080</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ok</td>
<td>queue2</td>
<td>06:28:36</td>
<td>00:00:04.413</td>
<td>1920x1080</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

7.3.2 Create Feed File List

The feed file list is a CSV file with following columns:

• **Video File Relative Path** - The path of the video file. It is relative to the base path set in the `process_files.sh` script.
• **Site** - Specifies site (building, GPS location, etc) name for the purpose of identifying location.
• **Source** - Specifies the camera name for purposes of identifying location.
• **Date** - Internal use only.
• **EpochDate** - Start time of the video file in Unix Epoch date format. This is used to ensure events are created at the proper point in the timeline.
  • This value must be in Epoch milliseconds.
  • See the Process Video Files Script section below for information on creating Epoch values using the `process_files.sh` script.
  • Epoch millisecond values can be created from human readable dates in Excel using this formula:
    `=(D1-DATE(1970,1,1))*86400/1000`
    Where “D1” is the cell containing the start date of the video.

Below is an example of the contents of the feed file list file:

```plaintext
vid001.mp4,Site05,Camera025,8/20/19 7:12 AM,1566285120000
vid002.mp4,Site03,Camera024,8/20/19 7:12 AM,1566285120000
xvid003.mp4,Site01,Camera003,8/20/19 4:48 AM,1566276480000
vid004.mp4,Site01,Camera030,8/20/19 12:00 PM,1566302400000
vid005.mp4,Site04,Camera016,8/20/19 12:00 PM,1566302400000
vid006.mp4,Site04,Camera002,8/20/19 7:12 AM,1566285120000
vid007.mp4,Site03,Camera016,8/20/19 12:00 PM,1566302400000
xvid008.mp4,Site02,Camera020,8/20/19 12:00 PM,1566302400000
vid009.mp4,Site05,Camera011,8/20/19 9:36 AM,1566293760000
```
7.3.3 Edit the Process Files Script

Edit the configurable parameters in the `process_files.sh` script. Below are the default values in the script.

```bash
## USER CONFIGURABLE PARAMETERS ##
##################################
# User Directory
user_dir = main1
# Set this to the location of the feed configuration files.
feeds_dir = /files/feeds
# If different than above, Set to the path to feeds from inside docker container
docker_feeds_dir = $feeds_dir
# Set this to the location of the video files
video_files_dir = /files/videos/1min_segments
##################################
```

- `user_dir` is the user directory where identities and events are stored. It can be useful to edit this parameter when experimenting with different settings.
- The default values for `feeds_dir` and `docker_feeds_dir` are the same because of the most common ways that host file systems are mounted in the docker container. This may not necessarily be the case so the two attribute are separated in the script.
- The script assumes video_files path is identical in the host as well as the docker container. Script would need to be modified if this was not the case (one part of the script checks the existence of the file from the host OS while in another place, the path to the video file is passed to VIRGO running in the docker container via the feed configuration file.

7.4 Reference

This section includes reference files and example scripts useful in processing video files with the Command Line Interface.

7.4.1 VIRGO Command Line Help

Command line interface to the virgo daemon

Syntax:

'virgo' followed by one of the following:
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>administrator set &lt;admin&gt;</td>
<td>sets the administrator. Either 'virgo' or 'virga'</td>
</tr>
<tr>
<td>administrator get</td>
<td>shows the current administrator</td>
</tr>
<tr>
<td>environment get</td>
<td>shows the current environment</td>
</tr>
<tr>
<td>environment set &lt;env&gt;</td>
<td>sets the current environment</td>
</tr>
<tr>
<td>environment list</td>
<td>shows all supported environments</td>
</tr>
<tr>
<td>feed list</td>
<td>shows all known feeds</td>
</tr>
<tr>
<td>feed get &lt;feed&gt; [path]</td>
<td>shows the feed configuration</td>
</tr>
<tr>
<td></td>
<td>and optionally saves it as a feed configuration file</td>
</tr>
<tr>
<td>feed status &lt;feed&gt;</td>
<td>shows the current feed status</td>
</tr>
<tr>
<td>feed start &lt;feed&gt;</td>
<td>marks the feed as enabled and starts it running</td>
</tr>
<tr>
<td>feed stop &lt;feed&gt;</td>
<td>stops the feed and marks it as disabled</td>
</tr>
<tr>
<td>feed remove &lt;feed&gt;</td>
<td>stops the feed and removes it</td>
</tr>
<tr>
<td>feed add &lt;feed&gt; &lt;path to config&gt;</td>
<td>reads the feed configuration file and adds the new feed to the known feeds</td>
</tr>
<tr>
<td>feed set &lt;feed&gt; &lt;path to config&gt;</td>
<td>reads the feed configuration file and updates the feed with the new configuration</td>
</tr>
<tr>
<td>feed capture-image &lt;feed&gt; &lt;url or path&gt;</td>
<td>captures one or more images from the feed and stores them in the directory &lt;url&gt;</td>
</tr>
<tr>
<td>service info</td>
<td>shows the service information</td>
</tr>
<tr>
<td>service status</td>
<td>shows the current status of all feeds</td>
</tr>
<tr>
<td>service reset</td>
<td>resets the persistent virgo state back to the factory defaults</td>
</tr>
<tr>
<td>service log</td>
<td>continuously shows the current service log information</td>
</tr>
<tr>
<td>service monitor</td>
<td>continuously shows the current service status and statistical information</td>
</tr>
<tr>
<td>service update &lt;version&gt; [&lt;url or path&gt;]</td>
<td>upgrades or downgrades virgo to the specified version</td>
</tr>
<tr>
<td>service versions</td>
<td>shows all installed versions</td>
</tr>
<tr>
<td>user get</td>
<td>shows the current user cloud identity</td>
</tr>
<tr>
<td>user set [user id]</td>
<td>sets the current user cloud identity</td>
</tr>
</tbody>
</table>

<environment> is an environment name.
<feed> is a feed name.
<version> is a semantic version number (e.g. 1.0.3).

--help | -h       shows this help message.
--verbose | -v enables the display of more detailed information.
--max-frames the maximum number of frames to capture from a feed
--frame-delay the delay between capturing frames. This is in milliseconds
--size the size to which a captured image should be scaled
7.4.2 Process Video Files Script

process_files.sh

This is the primary commands that loops through all files and adds them to VIRGO queues for processing.

```bash
#!/bin/bash

script_dir=$(dirname "$0")

## USER CONFIGURABLE PARAMETERS ##
###################################
# Set this to the location of the feed configuration files.
feeds_dir=/files/feeds
# If different than above, Set to the path to feeds from inside docker container
docker_feeds_dir=$feeds_dir
# Set this to location of the video files
video_files_dir=/files/videos/1min_segments
# User Directory
user_dir=main1
###################################

# Make sure we got correct number of args.
if [ "#$" -lt 1 ]; then
  echo "Usage: $0 <job list file>
  Where <job list file> csv file with video_file,site,source"; exit;
fi
# Check if video file list exists
[ ! -f $1 ] && { echo "$1 video list file not found"; exit 99; }

### Loop thru each line in input file
#########################################
OLDIFS=$IFS; IFS=,
cat $1 | tr -d \'\r\' | while read fname site source date epoch; do
echo "Placing file $fname for site: $site, source: $source and time: $epoch"
  [ ! -f "$video_files_dir/$fname" ] && {
    echo "$video_files_dir/$fname not found. Skipping" | tee -a error.log; continue;
  }
### Loop Feeds - Outer loop
## Get list of queues and pass this into inner loop
### Break once inner loop exits with successful assignment
############################################################################
assigned=false; echo "processing file $fname"
while true; do
  ### Loop Feeds - Inner loop
  ### Loop thru each queue - check to see if any are ready for next job
  ###
  while read feed_line; do
    TEMP=$IFS; IFS=': ' read queue_id queue_status <<< $feed_line;
    IFS=$TEMP
```

50
if [ "$queue_status" == "failed" ]; then
  # Something went wrong. Log file being processed by this queue
to look at later
  echo $(($script_dir/get_input_stream.sh $queue_id.json)) >>
  error.log
fi
if [ "$queue_status" == "ok" ] || [ "$queue_status" == "prerolling"
]; then
  # Skip and go to next feed
  echo " Skipping queue $queue_id with status $queue_status"
elif [ "$queue_status" == "eos" ] || [ "$queue_status" ==
  "inactive" ]; then
  $script_dir/update_virgo_feed.sh $feeds_dir/$queue_id.json $site
  $source $epoch "file://$video_files_dir/$fname"
  $script_dir/set_virgo_feed_attr.sh $feeds_dir/$queue_id.json
  directory $user_dir
  echo "Set file $fname on queue $queue_id" | tee -a processed.log;
  docker exec safr_virgo ./virgo feed set $queue_id
  $feeds_dir/$queue_id.json
  if [ "$queue_status" == "inactive" ]; then
    docker exec safr_virgo ./virgo feed start $queue_id
  fi
  # Skip all other feeds and go to next file
  assigned=true; break 1;
else
  # Unexpected processing status. Report in error log
  echo "Unexpected virgo feed status $queue_status" >> error.log
fi
sleep 1
done <<< "$(docker exec safr_virgo ./virgo service status)"
if $assigned; then echo "Assigned $fname to $queue_id"; break 1; fi
done
IFS=$OLDIFS

7.4.3 Utility Scripts

The following scripts are used by the processing_files.sh script.

update_virgo_feed.sh

#!/bin/bash
if [ "$#" -lt 4 ]; then
  echo "Usage: $0 <feed filename> <sitename> <sourcename> <starttime>
  '<file_path>'":exit;
fi
sed -i -e "s|"site" *=.*"[^"]*"|"site" : "$2"|g" $1
sed -i -e "s|"source" *=.*"[^"]*"|"source" : "$3"|g" $1
sed -i -e "s|"reporter.events-initial-date-offset" *=.*"[^"]*"|"reporter.events-initial-date-offset" : "$4"|g" $1
sed -i -e "s|"input.stream.url" *=.*"[^"]*"|"input.stream.url" : "$5"|g" $1

Where:
• `<feed filename>` - Name of the file that contains feed settings.
• `<sitename>` - Specifies the site (building, GPS location, etc) name for the purpose of identifying the location.
• `<sourcename>` - Specifies the camera name for the purpose of identifying the location.
• `<starttime>` - Start time/date of the video. The value given as Epoch in milliseconds. For example, for Aug 1, 2019 at 11 AM, the epoch value is 1564682400000.
  • Try the Chrome extension ‘utime’ to create epoch times. This extension allows you to type any date/time and get epoch values as well as natural language strings such as ‘1 hour ago’. Be sure to set the time format to “milliseconds” or else the time will be off by many years.
• `<file path>` - The fully qualified path to the file relative to the Docker container.

`set_virgo_feed_attr.sh`

```bash
#!/bin/bash
if [ "$#" -lt 3 ]; then
  echo "Usage: $0 <feed filename> <attr name> <attr value>"; exit;
fi
sed -i -e "s |" $2 "*: *"[^"]*"|" $2 " : " $3 "| g" $1
```

Where:

• `<feed filename>` - Name of the file that contains feed settings.
• `<attr name>` - The JSON attribute name from the feed configuration file.
• `<attr value>` - The value to be assigned to the JSON attribute from the feed configuration file.

`get_input_stream.sh`

```bash
#!/bin/bash
if [ "$#" -lt 1 ]; then
  echo "Usage: $0 <feed filename>"
fi
grep input.stream.url $1 | sed -e 's:\input.stream.url":*:\1:g'
```

Where:

• `<feed filename>` - Name of the file that contains feed settings.
8 VIRGO Tools

The Video Recognition Gateway (VIRGO) installation includes a couple scripts to manage VIRGO. They are located in the app folder under the VIRGO installation location.

- On macOS: /Library/RealNetworks/SAFR/virgo/app
- On Linux or Jetson: /opt/RealNetworks/SAFR/virgo/app
- On Windows: C:\Program Files\RealNetworks\SAFR\virgo\app

8.1 Backup virgo-factory.conf

Before you use virgo_tools you’d better backup current virgo-factory.conf, you will find this configure file in:

/etc
/opt/RealNetworks/SAFR/virgo/config

8.2 virgo_updateip.sh

This script will use current local IP to upade virgo configure file (virgo-factory.conf)

1. go to /opt/RealNetworks/SAFR/virgo/app
2. run virgo_updateip.sh root@real:/opt/RealNetworks/SAFR/virgo/app# ./virgo_updateip.sh IT will display below message

```
11:22:38 CST - Collect local ip
11:22:38 CST - Local IP 10.10.51.189
11:22:38 CST - Updating configuration files
11:22:38 CST - Applying new virgo configuration
```

you will find IP is been changed in virgo-factory.conf

8.3 virgo_configure.sh

This script will reset the VIRGO service.

1. go to /opt/RealNetworks/SAFR/virgo/app
2. run virgo_configure.sh root@real:/opt/RealNetworks/SAFR/virgo/app# ./virgo_configure.sh Username Password You can find username and password from virgo-factory.conf username= user-id (from virgo-factory.conf ) password= user-encrypted-password (from virgo-factory.conf ) IT will display below message

```
11:54:21 CST - Updating virgo factory configuration
11:54:21 CST - Username realnetworksbei13
11:54:21 CST - Password %pHUQfSS4mk7Urh5au0aZ\+qshbJPGIx4rEw\'/RpCgGpUrYCiWrzc2uh8g9HsxJHF
11:54:21 CST - Cleaning out old virgo configuration
11:54:21 CST - Cloning template file
11:54:21 CST - Collect local ip
11:54:21 CST - Local IP 10-10-51-189
11:54:21 CST - Copy config to /etc/ folder
11:54:21 CST - Reset Virgo to reload configuration
11:54:21 CST - Finished
```
9 Factory Configuration File

Every Video Recognition Gateway (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.

9.1 Factory Configuration File Format

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

```json
{
   "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.

9.1.1 The Global Section

The following properties are supported in the global section:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status-interval</td>
<td>Int?</td>
<td>5000</td>
<td>Status reporting time interval in ms.</td>
</tr>
<tr>
<td>environment</td>
<td>String?</td>
<td>PROD</td>
<td>The name of the environment which should be used by virgod. See the “Environments Section” below for a list of pre-defined environment names.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>machine-id-prefix</td>
<td>String?</td>
<td>empty string</td>
<td>The machine ID prefix. The default machine ID prefix is the empty string.</td>
</tr>
<tr>
<td>machine-id</td>
<td>String?</td>
<td>OS defined machine ID</td>
<td>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.</td>
</tr>
<tr>
<td>client-type</td>
<td>String?</td>
<td>OS defined client type</td>
<td>The client type. This value is sent to the cloud in the X-CLIENT-TYPE header.</td>
</tr>
<tr>
<td>user-id</td>
<td>String</td>
<td></td>
<td>The user ID for the cloud account.</td>
</tr>
<tr>
<td>user-password</td>
<td>String</td>
<td></td>
<td>The password for the cloud account. Note that the password is stored in clear text. Use user-encrypted-password whenever possible instead.</td>
</tr>
<tr>
<td>user-encrypted-password</td>
<td>String</td>
<td></td>
<td>The encrypted password for the cloud account.</td>
</tr>
<tr>
<td>administrator</td>
<td>String?</td>
<td>cloud</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

9.2 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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>covi-server-url</td>
<td>URL</td>
<td>none</td>
<td>The face recognition service.</td>
</tr>
<tr>
<td>rncv-server-url</td>
<td>URL?</td>
<td>none</td>
<td>The face detection service.</td>
</tr>
<tr>
<td>event-server-url</td>
<td>URL</td>
<td>none</td>
<td>The detection and recognition event recording service.</td>
</tr>
<tr>
<td>object-server-url</td>
<td>URL</td>
<td>none</td>
<td>The service which stores objects such as images and logs.</td>
</tr>
<tr>
<td>admin-server-url</td>
<td>URL</td>
<td>none</td>
<td>The VIRGO administration service.</td>
</tr>
</tbody>
</table>

The following table lists the pre-defined environments:

<table>
<thead>
<tr>
<th>Name</th>
<th>Alternative name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAFR Local</td>
<td>LOCAL</td>
</tr>
<tr>
<td>SAFR Developer Cloud</td>
<td>DEV</td>
</tr>
<tr>
<td>SAFR Partner Cloud</td>
<td>INT2</td>
</tr>
<tr>
<td>SAFR Cloud</td>
<td>PROD</td>
</tr>
</tbody>
</table>
You can use the alternative environment name in place of the full environment name.

9.2.1 The Client ID
VIRGO computes the client ID by concatenating the machine-id-prefix and the machine-id properties.

9.3 Example Configuration Files
The following subsections show some typical factory configuration files.

9.3.1 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>
  }
}
```

9.3.2 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
    }
  }
}
```

9.3.3 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.

```json
{
    "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"
        }
    }
}
```
10 Docker

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

10.1 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.

10.2 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.

10.3 Service Status

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

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

<table>
<thead>
<tr>
<th>CONTAINER ID</th>
<th>IMAGE</th>
<th>COMMAND</th>
<th>CREATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>cf2a2dd33875</td>
<td>safr_virgo:1.1.38</td>
<td>&quot;/bin/sh -c &quot;$VIRGO_AaAe&quot; &quot;</td>
<td>18 hours ago</td>
</tr>
<tr>
<td>Up 18 hours</td>
<td>safr_virgo</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If there is no output check the following:

- Run the same command again with the -a switch to determine if the container is stopped or restarting (i.e. failing).
- Verify there is a valid virgo-factory.conf file 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

10.4 Execution

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

The container is automatically started by the SAFR Platform Installer and stopped by the SAFR Platform Uninstaller.
10.5 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.

10.6 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.

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

10.7.1 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
  ```

10.8 Add Volume Mount to Existing Container
1. Update the compose file to add the additional volume instructions.
   - The format is `<local folder>:`<docker folder>
   - The <docker folder> will be created if not already existing.

   **Example:** (Your folder names might be different.)
   ```
   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 the local folder to mount into the container.

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

3. Stop and delete the container.

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

4. Create a container instance with new volume mount.

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

5. Create a test file in local mount point.

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

6. Check that the test file exists inside the container’s mount location.

```bash
# docker exec -it safr_virgo ls -l /opt/RealNetworks/virgo/files
```
11 GPU Support

Starting with version 1.1.16, Video Recognition Gateway (VIRGO) on Linux supports acceleration of video decoding, graphics processing, and face detection functions via one or more GPUs. VIRGO automatically detects the presence of a compatible graphics card and will use it. On systems without a GPU VIRGO falls back to doing everything on the CPU.

Only Nvidia Compute Unified Device Architecture (CUDA) GPUs are currently supported.

11.1 GPU Requirements and Installation

NVIDIA drivers version 418.67 or greater are required. The CUDA toolkit is not required.

11.1.1 Installation

1. Install dependencies.
   • For Ubuntu: Run `DEBIAN_FRONTEND=noninteractive apt-get update -y && apt-get install -y gcc make`
   • For Centos: Run `yum install -y gcc make kernel-devel`
   • For Amazon: Run `yum install -y gcc make "kernel-devel-uname-r == $(uname -r)"`

2. Download the most recent NVIDIA Linux drivers from https://www.nvidia.com/object/unix.html.
   • Example: `curl -LO http://us.download.nvidia.com/tesla/418.67/NVIDIA-Linux-x86_64-418.67.run`

3. Stop x-windows, if running:
   • For Ubuntu: Run `service lightdm stop`

4. Run driver installer:
   • Run `sudo bash NVIDIA-Linux-x86_64-418.67.run --silent`

5. Verify if your installation was successful:
   • Run `nvidia-smi`
6. If your installation was unsuccessful, view the log:
   - Run `less /var/log/nvidia-installer.log`

11.2 Enable a Feed to Run on a GPU

There’s nothing you need to do to make this happen; VIRGO automatically detects the presence of a suitable GPU and assigns a feed to it. A feed will automatically fall back to the CPU if there’s a problem with the GPU or all GPU resources have been exhausted.

VIRGO also takes advantage of multiple GPUs installed in the system. It automatically distributes feeds across all available GPUs. This enables you to easily scale up a system which allows you to run more feeds on a single VIRGO host.

VIRGO returns comprehensive statistical information about a feed. This includes information about which GPU a feed is running on as well as how much of its processing power it’s using per second.

### 11.2.1 Manual Feed Assignment

Sometimes more control over which feed is assigned to the CPU vs a GPU is desired. VIRGO allows you to individually specify for each feed whether it should exclusively run on a GPU or the CPU. This allows you to maximize the use of all available GPUs and the CPU by assigning some feeds exclusively to the GPU and some exclusively to the CPU. The following table shows the available feed accelerator configurations:
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>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.</td>
</tr>
<tr>
<td>cpu</td>
<td>The feed will exclusively run on the CPU and not use any GPU even if a GPU is available.</td>
</tr>
<tr>
<td>gpu</td>
<td>The feed will 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.</td>
</tr>
</tbody>
</table>
12 Service Logging

The Video Recognition Gateway (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**:

Enable DEBUG-level logging for the ‘tracking’ package in all feeds:

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

Enable DEBUG-level logging for the ‘capture’ and the ‘cop-http’ packages in all feeds:

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

Enable DEBUG-level logging for the ‘tracking’ package in the feed ‘foo’: (This does not change the current log configuration for any other feed.)

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

The following log levels are supported:

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>Verbose</td>
</tr>
<tr>
<td>D</td>
<td>Debug</td>
</tr>
<tr>
<td>I</td>
<td>Info</td>
</tr>
<tr>
<td>W</td>
<td>Warn</td>
</tr>
<tr>
<td>E</td>
<td>Error</td>
</tr>
<tr>
<td>O</td>
<td>Off</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Package</th>
<th>Supports feed name?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>detection</td>
<td>yes</td>
<td>Object detector related messages</td>
</tr>
<tr>
<td>recognition</td>
<td>yes</td>
<td>Face recognizer related messages</td>
</tr>
<tr>
<td>tracking</td>
<td>yes</td>
<td>Object tracker messages</td>
</tr>
<tr>
<td>capture</td>
<td>yes</td>
<td>Image capture related messages</td>
</tr>
<tr>
<td>events</td>
<td>yes</td>
<td>Event reporting related messages</td>
</tr>
<tr>
<td>pose-liveness</td>
<td>yes</td>
<td>Pose Liveness Action Recognizer related messages</td>
</tr>
<tr>
<td>Package</td>
<td>Supports feed name?</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>feed</td>
<td>yes</td>
<td>Feed life cycle related messages</td>
</tr>
<tr>
<td>cop-http</td>
<td>no</td>
<td>COP over HTTP related messages</td>
</tr>
<tr>
<td>config</td>
<td>no</td>
<td>Virgod configuration management related messages</td>
</tr>
<tr>
<td>updates</td>
<td>no</td>
<td>Virgod update initiation mechanism related messages</td>
</tr>
</tbody>
</table>
13 Service Monitoring

The Video Recognition Gateway (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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Feed</th>
<th>PID</th>
<th>Epoch</th>
<th>P-Time</th>
<th>Resolution</th>
<th>FPS</th>
<th>DPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ok</td>
<td>camera_1</td>
<td>14536</td>
<td>12/06/17 00:24:13.450</td>
<td>1280x720</td>
<td>120</td>
<td>8ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>250ms</td>
<td></td>
<td>250 ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>18</td>
<td></td>
<td>120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td></td>
<td>8ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ok</td>
<td>camera_2</td>
<td>67289</td>
<td>13:07:12 80:10:00.000</td>
<td>1920x1080</td>
<td>29.97</td>
<td>8ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>250ms</td>
<td></td>
<td>250 ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1920</td>
<td></td>
<td>1400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>1000</td>
<td></td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>4%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inactive</td>
<td>camera_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>The feed status. This is one of ok, inactive, eos, error, or failure.</td>
</tr>
<tr>
<td>Feed</td>
<td>The feed name.</td>
</tr>
<tr>
<td>PID</td>
<td>The PID of the feed daemon if the daemon is running</td>
</tr>
<tr>
<td>Epoch</td>
<td>The time when the feed processed the first frame in the video stream.</td>
</tr>
<tr>
<td>P-Time</td>
<td>The amount of time that the feed has spent on processing the video stream. This is in terms of milliseconds.</td>
</tr>
<tr>
<td>Resolution</td>
<td>The width and height of a video frame in pixels</td>
</tr>
<tr>
<td>FPS</td>
<td>The frames per second of the input video.</td>
</tr>
<tr>
<td>DPS</td>
<td>The number of detections per second.</td>
</tr>
<tr>
<td>dDt</td>
<td>The latency of a single detection operation in milliseconds.</td>
</tr>
<tr>
<td>dRt</td>
<td>The latency of a single recognition operation in milliseconds.</td>
</tr>
<tr>
<td>#D</td>
<td>The number of detection operations that have been triggered.</td>
</tr>
<tr>
<td>#D-Badge</td>
<td>The number of badges that have been detected.</td>
</tr>
<tr>
<td>#D-Face</td>
<td>The number of faces that have been detected.</td>
</tr>
<tr>
<td>#D-Skip</td>
<td>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.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>#R</td>
<td>The number of face recognition or reconfirmation operations that have been triggered.</td>
</tr>
<tr>
<td>#R-Face</td>
<td>The number of successful face recognition or reconfirmation operations that have been run.</td>
</tr>
<tr>
<td>#R-Err</td>
<td>The number of face recognition or reconfirmation operations that have failed for some reason.</td>
</tr>
<tr>
<td>#R-Skip</td>
<td>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.</td>
</tr>
<tr>
<td>#Evt</td>
<td>The number of events that have been reported.</td>
</tr>
<tr>
<td>%CPU</td>
<td>How CPU is used by the feed. Note that this number is in the range 0% to CPU_COUNT * 100%.</td>
</tr>
<tr>
<td>GPU#</td>
<td>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.</td>
</tr>
<tr>
<td>GPU</td>
<td>A string which indicates which modules in the feed are using the GPU: V -&gt; video decoder F -&gt; face detector B -&gt; badge detector O -&gt; object detector An empty/non-existing string indicates that the feed is not using the GPU at all.</td>
</tr>
<tr>
<td>GPU-Name</td>
<td>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.</td>
</tr>
</tbody>
</table>

### 13.1 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:

```bash
> 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.
14 VIRGO Architecture

A single Video Recognition Gateway (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 caring 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 the Video Recognition Gateway Administrator (VIRGA) command and control server about the current status of `virgod`. 

69
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.
- After the update has finished, the updater exits.

virgo:
- Implements the local (CLI-based) user interface to virgod.
- Offers commands to show the current status, selects the cloud environment, gets a screen capture from a feed, etc.

14.1 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
```

14.2 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 can be managed through the VIRGO command line tool or through the video feeds window of the Desktop Client or the Web Console.
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 in the video feeds windows by changing the enabled setting. 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 two types of input currently supported are “stream” and “file”. A stream input is specified by a URL which may point to a publicly accessible RTSP or HTTP 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.
15 Troubleshooting

15.1 Linux

15.1.1 Which Linux distributions are supported?

- Ubuntu 16.04(.5+) is known to work and has seen extensive testing.
- Ubuntu 18.04(.2+) appears to work but has not seen extensive testing.
- All other Linux distribution may or may not work; they have not seen any testing.

15.1.2 I just want to do a quick experiment with Video Recognition Gateway (VIRGO). Do I really have to do a full installation?

Actually no. If you just want to run VIRGO temporarily (e.g. to do testing) then there is no need to do a full installation. Do this instead:

1. Create a `virgo-factory.conf` file in your home directory which contains the necessary account, environment, and feed information.
2. Open a shell window and run `virgo/versions/current/virgod -l` in it.
3. Open a second shell window and use it to control VIRGO from there. For example, type `virgo/virgo service monitor` to see the current status of VIRGO.

Once you’re done with your work you should terminate VIRGO by typing Control-C in the shell window in which you started `virgod`.

Here is a small example `virgo-factory.conf` file:

```json
{
  "global":{
    "environment": "INT2",
    "machine-id-prefix": "vRGo-Real18L-X-",
    "user-id": "<Your SAFR cloud account ID here>",
    "user-password": "<Your SAFR cloud account password here>",
    "remote-control-enabled":false
  },

  "feeds":{
    "Axis Q6128-E": {
      "directory":"testy",
      "input.type": "stream",
      "input.stream.url":"rtsp://user:password@101.102.103.104/axis-media/media.
      "enabled":true
    }
  }
}
```

Note that this quick & dirty way of running VIRGO is not suitable for a production system.

For example, VIRGO will stop running as soon as you log out of the system and the VIRGO factory configuration file is not secured which means that passwords (SAFR cloud account, camera IP passwords, etc) may be exposed to 3rd parties.

15.1.3 I’ve installed VIRGO but all my feeds die with an “Unexpected termination” error. What is wrong?

Your Linux installation is most likely missing a required APT package/library. Please make sure that you follow the installation instructions for Linux precisely. See this page for the list of required APT packages.
To find out which library is exactly missing, invoke the VIRGO feed daemon directly like this:

```
> virgo/versions/current/virgofeedd
```

This will cause the operating system to print the name of the missing library (.so file). Note that this command will print an error message about a missing/broken pipe if no library is missing. This later error is expected but any complaint about a missing dependency/library is not expected and points to a problem you need to fix.

If you see the following, it means that all dependencies are satisfied:

```
> virgo/versions/current/virgofeedd
```

```
Fatal error: 'try!' expression unexpectedly raised an error:
  virgofeedd.DTPError.io(message: "Bad file descriptor (9)"): file
/var/lib/jenkins/workspace/ubuntu_16_04_virgo_trunk_daily/build/virgo-build-x86_64-linux/virgofeedd/Sources/main.swift,
line 31
```

If, on the other hand, you see the following, it means that a library is missing:

```
> virgo/versions/current/virgofeedd
```

```
virgo_installer/virgo/versions/current/virgofeedd: error while loading shared libraries: libcuda.so.1: cannot open shared object file: No such file or directory
```

15.1.4 I’ve connected a camera to VIRGO and it is perpetually stuck in prerolling mode with the error Codec parameters not found. What’s going on?

Some cameras have buggy firmware which fail to generate a correct H264 PPS packet if the RTSP transport protocol is set to UDP. Note that VIRGO connects to RTSP cameras via UDP by default because UDP requires less networking resources and has lower latency compared to TCP.

However in this case and to fix this problem you need to tell VIRGO to connect to the camera using TCP instead. Do this by adding the following property to the feed dictionary for the camera:

```
"input.stream.rtsp.transport":"tcp"
```

15.1.5 I’ve just installed VIRGO, changed some things in the virgo-factory.conf file, and now virgod seems to crash all the time?!

Most likely there’s a syntax error in the virgo-factory.conf file now. For example, you may have forgotten to add a comma at the end of a property. You can run virgod like this to see the actual error message:

```
> virgo/versions/current/virgod -l
```

```
Factory config error:
  dataCorrupted(Swift.DecodingError.Context(codingPath: [],
    debugDescription: "The given data was not valid JSON.",
    underlyingError: Optional(Error Domain=NSCocoaErrorDomain Code=3840
      "Badly formed object around character 54."
    UserInfo={NSDebugDescription=Badly formed object around character
      54.}}))
```

You can also check the virgod exit code. It will be 78 (POSIX EX_CONFIG) if there is a syntax error in the factory configuration file.

Note that this kind of error can not be captured by the VIRGO logging system because it happens at the very startup of virgod and before the logging system has been initialized.
15.2 macOS

15.2.1 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.

15.3 Docker

15.3.1 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 thus time out.