



SPEECHMATICS

Batch Virtual Appliance 3.3.0

Table of Contents

- [Batch Virtual Appliance](#)
 - [High-Level Summary](#)
 - [Important Notices](#)
 - [What's New](#)
 - [Issues Fixed](#)
 - [Known Limitations](#)
 - [Supported Platforms](#)
 - [Form Factors](#)
 - [Upgrade Path](#)
 - [Installation](#)
- [Batch Virtual Appliance Installation and Admin Guide](#)
- [System requirements](#)
 - [Host requirements](#)
 - [Batch Virtual Appliance requirements](#)
 - [Real-Time Virtual Appliance](#)
 - [Batch Virtual Appliance](#)
- [Downloading the appliance](#)
- [Importing the appliance](#)
 - [VMware ESXi](#)
 - [VMware Workstation Player](#)
 - [VirtualBox](#)
 - [Amazon Web Services](#)
 - [Prerequisites](#)
 - [Uploading the OVA file to S3](#)
 - [Importing the OVA as AMI instance](#)
 - [Creating an Import Service Role](#)
 - [Creating a Role Policy](#)
 - [Importing the OVA](#)
 - [Security](#)
 - [Real-time Virtual Appliance](#)
 - [Batch Virtual Appliance](#)
 - [Launching a Virtual Appliance](#)
- [Network Configuration](#)
 - [Network interface mapping](#)
 - [VMware ESXi](#)
 - [VMware Workstation Player](#)
 - [VirtualBox](#)
 - [IP Configuration](#)
 - [Configure static IP](#)
 - [Configure DHCP IP](#)
- [Licensing](#)
 - [Applying License](#)
- [Verify and Go \(Batch\)](#)
- [SSL Configuration](#)
 - [Default behaviour](#)
 - [Management API Examples](#)
 - [Monitoring API Example](#)
 - [Speech API Example](#)
 - [Using your own SSL certificate and private key](#)
 - [Uploading the certificate and key to the appliance](#)
 - [Disabling HTTP access](#)

- [Enable Basic Authentication for Admin](#)
- [FAQs](#)
 - [How do I reset the SSL settings?](#)
 - [What if I forget the admin password?](#)
 - [What versions of SSL/TLS do you support?](#)
 - [What cipher suites do you support?](#)
- [Networking](#)
 - [Network Requirements](#)
 - [Configure Static IP](#)
 - [Configure DHCP](#)
 - [Firewall Ports](#)
 - [Using Proxies](#)
- [Virtual Appliance Scaling](#)
 - [Real-Time Virtual Appliance Scaling](#)
 - [Worker Limits](#)
 - [View Maximum Workers](#)
 - [Setting Maximum Workers](#)
 - [Batch Virtual Appliance Scaling](#)
 - [Worker Limits](#)
 - [View Maximum Workers](#)
 - [Setting Maximum Workers](#)
- [Monitoring](#)
- [Services](#)
 - [Service status](#)
 - [Service restart](#)
 - [Access Logs](#)
 - [System restart](#)
 - [Troubleshooting](#)
 - [Transcription job failure](#)
 - [Illegal instruction errors](#)
 - [Console for Advanced Troubleshooting](#)
 - [License](#)
 - [Networking](#)
 - [Reboot and Shutdown](#)
 - [Services](#)
 - [Tools](#)
 - [Workers](#)
- [Security](#)
 - [Overview](#)
 - [Firewall Ports](#)
 - [Securing your Deployment](#)
- [Batch Virtual Appliance](#)
 - [Overview](#)
 - [Terms](#)
 - [Getting Started](#)
 - [Audio Formats](#)
 - [API Versions](#)
 - [Transcription Formats](#)
 - [Authentication](#)
 - [Troubleshooting](#)
- [Example Usage](#)
 - [Submitting a Job](#)
 - [Sample Request](#)

- [Example Response](#)
 - [Supplying a Job Configuration](#)
 - [Example Configurations](#)
 - [Speaker Diarization](#)
 - [Channel Diarization](#)
 - [Speaker Change Detection](#)
 - [Speaker Change Detection With Channel Diarization](#)
 - [Custom Dictionary](#)
 - [Output Locale](#)
 - [Advanced Punctuation](#)
 - [Output Formats](#)
 - [Legacy API](#)
 - [Passing Metadata](#)
 - [Callbacks Usage](#)
- [Speechmatics ASR REST API](#)
 - [Overview 3.3.0](#)
 - [Version information](#)
 - [Contact information](#)
 - [License information](#)
 - [URI scheme](#)
 - [Paths](#)
 - [/jobs](#)
 - [**POST**](#)
 - [**GET**](#)
 - [/jobs/{jobid}](#)
 - [**GET**](#)
 - [**DELETE**](#)
 - [/jobs/{jobid}/data](#)
 - [**GET**](#)
 - [/jobs/{jobid}/transcript](#)
 - [**GET**](#)
 - [Models](#)
 - [ErrorResponse](#)
 - [TranscriptionConfig](#)
 - [JobConfig](#)
 - [CreateJobResponse](#)
 - [JobDetails](#)
 - [RetrieveJobsResponse](#)
 - [RetrieveJobResponse](#)
 - [DeleteJobResponse](#)
 - [JobInfo](#)
 - [RecognitionMetadata](#)
 - [RecognitionDisplay](#)
 - [RecognitionAlternative](#)
 - [RecognitionResult](#)
 - [RetrieveTranscriptResponse](#)
 - [Error Codes](#)
 - [4XX Errors](#)
 - [5XX Errors](#)
 - [Formatting Common Entities](#)
 - [Overview](#)
 - [Supported Languages](#)
 - [Using the enable_entities parameter](#)

- [Configuration example](#)
- [Different entity classes](#)
- [Output locale styling](#)
- [Example output](#)

Batch Virtual Appliance

High-Level Summary

This release contains improvements to telephony models for English and Spanish, and a number of bugfixes.

The following languages now support advanced punctuation: English (en), German (de), Spanish (es), French (fr), Dutch (nl) and Danish (da).

It is recommended that customers on previous releases upgrade to this version.

Important Notices

The legacy V1 API that the Batch Virtual Appliance currently supports will be discontinued in a future release as we align the product with the same V2 API used by the Speechmatics SaaS: <https://asr.api.speechmatics.com/v2/docs>. We recommend that customers familiarise themselves with the configuration object used to specify job configurations, and only use form parameters for callback notifications. Future notices will be provided to announce the end of life of the V1 API, and provide detailed instructions on migrations to the V2 API.

What's New

- Improvements to the telephony models for English and Spanish

Issues Fixed

The following issues are addressed since the previous release:

Issue ID	Summary	Resolution Description
REQ-7268	The internal 'reset' method can cause a server panic	This messages will now no longer be seen in the logs when services are restarted.
REQ-7461	Memory leak affecting licensing container	A small memory leak affecting the licensing service has now been addressed.
REQ-9122	Appliance could run into problems with services not being able to restart if it ran out of disk space.	The appliance will now reject jobs if there is < 250MB of free space on the disk.
REQ-9569	Unfriendly error when empty audio file is submitted	It is now possible to submit zero length audio files.
REQ-10095	Stoptleration Exception in post processing can cause a job to fail	The exception is now handled properly.
REQ-11644	Offline activation cannot generate deactivation code	Previously if you submitted the PUT/POST request for offlineactivation, any attempt to deactivate with a DELETE request failed; this has now been fixed.
REQ-11728	Files less than 1 second skip decoding.	Files that are < 0.3s long are considered to be zero length; but files between 0.3 and 1.0s (and above) will now be transcribed (assuming speech is detected in them).

Known Limitations

The following are known issues in this release:

Issue ID	Summary	Detailed Description and Possible Workarounds

REQ-1409	Proteus HCL with <unk> causes out of memory error	A custom dictionary list that contains the word '' causes the worker to crash.
REQ-7549	Memory leak affecting gRPC	There is a small memory leak in the gRPC Python server (https://github.com/grpc/grpc/issues/5913).
REQ-10160	Advanced punctuation for Spanish (es) does not contain inverted marks.	Inverted marks [¿ ;] are not currently available for Spanish advanced punctuation.
REQ-10627	Double full stops when acronym is at the end of the sentence	If there is an acronym at the end of the sentence, then a double full stop will be output, for example: "team G.B.."
REQ-11135	3.2.0 introduced unwanted hesitations in transcripts for English (en).	Due to changes in the way that training data is now ingested to improve the accuracy of spontaneous speech for English (en) there is a greater likelihood that hesitations will be included in the output transcripts. We plan to support a hesitation filtering capability in a future release for customers that do not want to see hesitations on transcripts.

Supported Platforms

Virtual Appliance image (OVA) for installation on:

- VMware ESXi 6.5+ or VMware Workstation Player.
- VirtualBox 5.2+
- Amazon EC2

See the Installation and Admin Guide for details on the minimum specifications for the VM. The maximum number of concurrent jobs (maxworkers) that you can run on a single appliance is 32.

Form Factors

There are four variants of the Batch Virtual Appliance.

Variant	Image Size	Max. Disk Space	Languages
nano	13GB	40GB	en
mini	21GB	40GB	en, de, es
midi	41GB	60GB	en, de, es, fr, ko, ja, nl, pt
maxi	65GB	80GB	en, de, es, fr, ko, ja, nl, pt, it, da, pl, ca, hi, ru, sv

Upgrade Path

Remove the license from your old appliance (see the Admin Guide), then re-import the new OVA and configure networking as per the Installation and Admin guide. You will need to re-apply the license code you have once the OVA has imported.

Installation

Upload the OVA to VMWare ESX, VMWare Workstation Player, or VirtualBox. See the Installation and Admin Guide for more information.

Batch Virtual Appliance Installation and Admin Guide

This guide explains how to install and administer the Batch Virtual Appliance using the Management REST API.

The simplest method of accessing the Management API is with a web browser by connecting to the Batch Virtual Appliance using the following URL:

```
https://${APPLIANCE_HOST}:8080/docs/
```

Where `${APPLIANCE_HOST}` is the the IP address or DNS name of the virtual appliance.

There are two flavors or "modes" of Speechmatics virtual appliance: real-time and batch. For the most part installation and administration are identical for both modes. Where differences exist this is explicitly noted in this guide.

System requirements

The Speechmatics Batch Virtual Appliance operates on a hypervisor host system. For this version of the appliance, the following hypervisors are supported:

- VMware®
- VirtualBox
- AWS EC2

For the virtual appliance to operate as required, the host must meet the requirements and have the resources available as defined below.

Host requirements

The virtual appliance can operate in any VMware supported environment that claims support for VMware virtual hardware specification 9 and above (see <https://kb.vmware.com/s/article/1003746>).

The host machine requires a processor with following minimum specification: Intel® Xeon® CPU E5-2630 v4 (Sandy Bridge) 2.20GHz (or equivalent). This is important because these chipsets (and later ones) support Advanced Vector Extensions (AVX). The machine learning algorithms used by Speechmatics ASR require the performance optimizations that AVX provides. You should also ensure that your hypervisor has AVX enabled.

See below for minimum Batch Virtual Appliance VM (guest) specifications; the host machine must have enough resources (processor, memory and storage) to run the hypervisor, the guest VMs you intend to host on it, plus any other processes you expect to run on it. Vendor guidelines should be followed for other host requirements and installation process.

For VMWare, the document Performance Best Practices for VMware vSphere® 6.0 contains a comprehensive overview of hardware considerations and recommendations on how to optimize your host platform. See <https://www.vmware.com/support.html> for up-to-date technical information on VMWare.

For VirtualBox, please consult the online documentation: <https://www.virtualbox.org/wiki/Documentation>

For Amazon EC2, the following link explains how to setup a VM using an Amazon S3 to store the OVA file: <https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-image-import.html>.

Batch Virtual Appliance requirements

Real-Time Virtual Appliance

The Speechmatics Real-Time Batch Virtual Appliance must be allocated the following *minimum* specification:

- 1 vCPU
- 4GB RAM
- Up to 38GB hard disk space

For each concurrent input stream the appliance requires an additional 1 vCPU and up to 1.5GB RAM. If you are using the custom dictionary (additional words) feature then each concurrent input stream that is configured to use it will require up to 3GB RAM.

Batch Virtual Appliance

For operation in batch mode, the following *minimum* specifications are required:

- 2 vCPUs
- 8GB RAM
- Up to 44GB hard disk space

For more details on how to scale your Batch Virtual Appliance and protect resources by configuring limits on the maximum number of concurrent jobs that can be processed, please refer to the relevant section in the Admin Guide.

Downloading the appliance

A download link will be provided by Speechmatics through the solutions section of the support portal (<https://support.speechmatics.com>). The latest version of the appliance can be located within the solutions section. Select the required version number within the "Batch Virtual Appliance" area (for example {{ book.product.version }}) to view the download link and all associated documentation for the virtual appliance. Once the download link is selected the download will begin, or a save file prompt will appear, enabling the file to be saved (the exact method will depend on the web browser being used). After the download a file with an ".ova" extension will be stored on the computer.

An account is required to access the documents and download link in the support portal. If an account is not available or the "Batch Virtual Appliance" section is not visible in the support portal, please contact Speechmatics Support support@speechmatics.com for help.

Importing the appliance

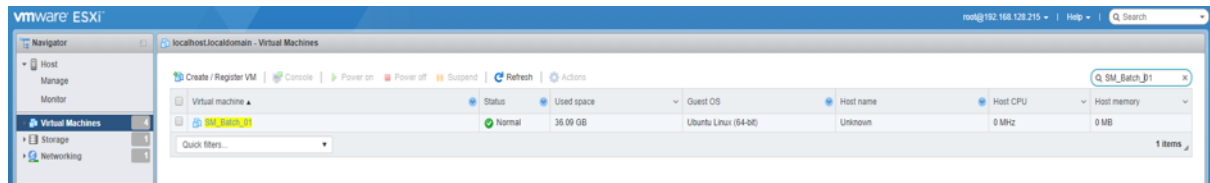
Once the .ova file has been downloaded, it is ready to be imported into the host you have already prepared. Please ensure that the host meets the requirements stated earlier in this guide, then based on the hypervisor environment follow the instructions below.

VMware ESXi

The following steps can be used to import the virtual appliance into VMWare ESXi 6.5:

- Open the vSphere web console on the host
- Choose "Virtual Machines" from the Navigator
- Select "Create/Register VM" option
- A wizard will appear:
 - Choose "Deploy a virtual machine from an OVF or OVA file" and click "Next"
 - Enter a VM name e.g. "SM_Batch_01", and drag the downloaded .ova file onto the window and click "Next"
 - Select a datastore that has enough capacity to store the virtual appliance and click "Next"
 - From the "VM network" dropdown box, select a network
 - Choose Thin or Thick disk provisioning (the Speechmatics Batch Virtual Appliance supports either. Choose the options that is right for the hosting environment refer to VMWare documentation for help and click "Next"
 - Check the details are correct and click "Finish"
- The virtual appliance will import. This can take a few minutes depending on the datastore chosen.

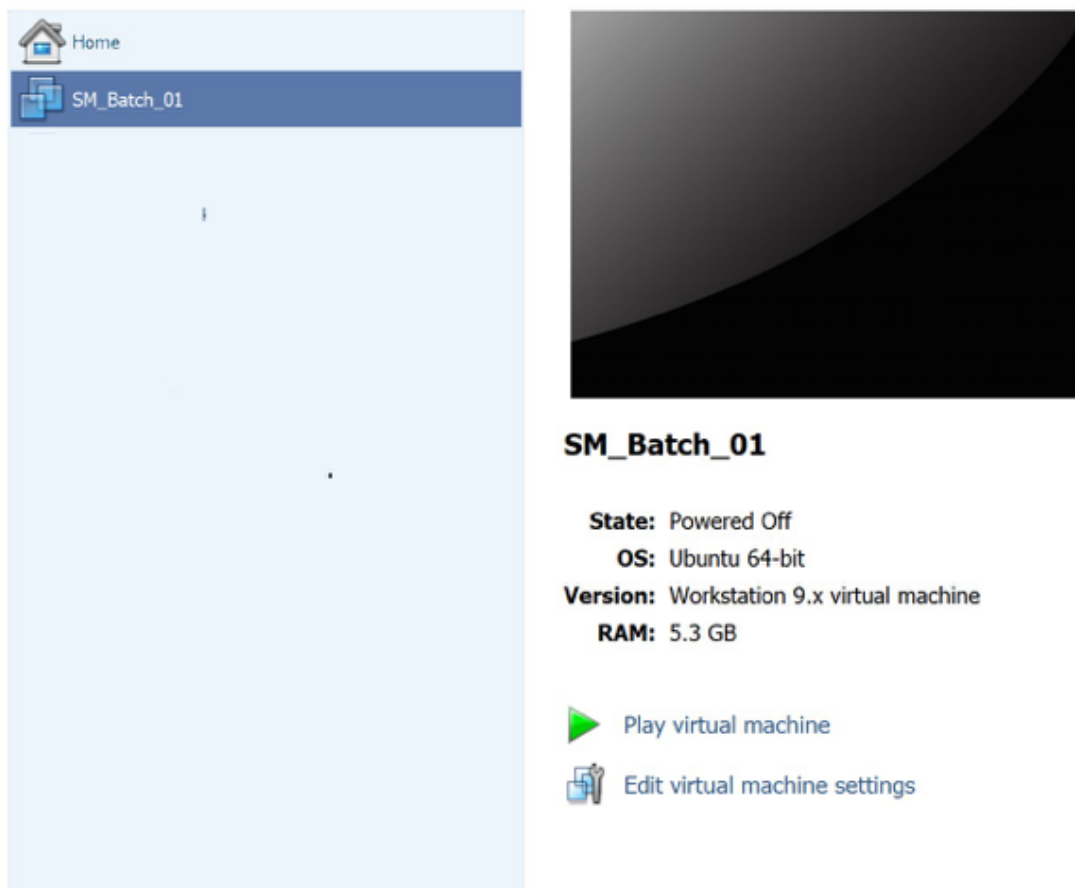
Once the VM has imported it should be visible on the vSphere web console:



VMware Workstation Player

- Open VMware Workstation Player
- From the top options bar select "Player", then "File" and "Open..."
- The "Open Virtual Machine" window will appear. Navigate to the ".ova" file you downloaded earlier, select it, click "Open"
- Enter a VM name e.g. "SM_Batch_01"
- A default storage location for the virtual appliance will be shown, the can be changed if required. Click "Import".
- Dropdown box from the top options bar, click on "File"
- The virtual appliance will import. This can take a few minutes depending on the hard disk chosen

Once the VM has imported it should be visible on the Workstation player:



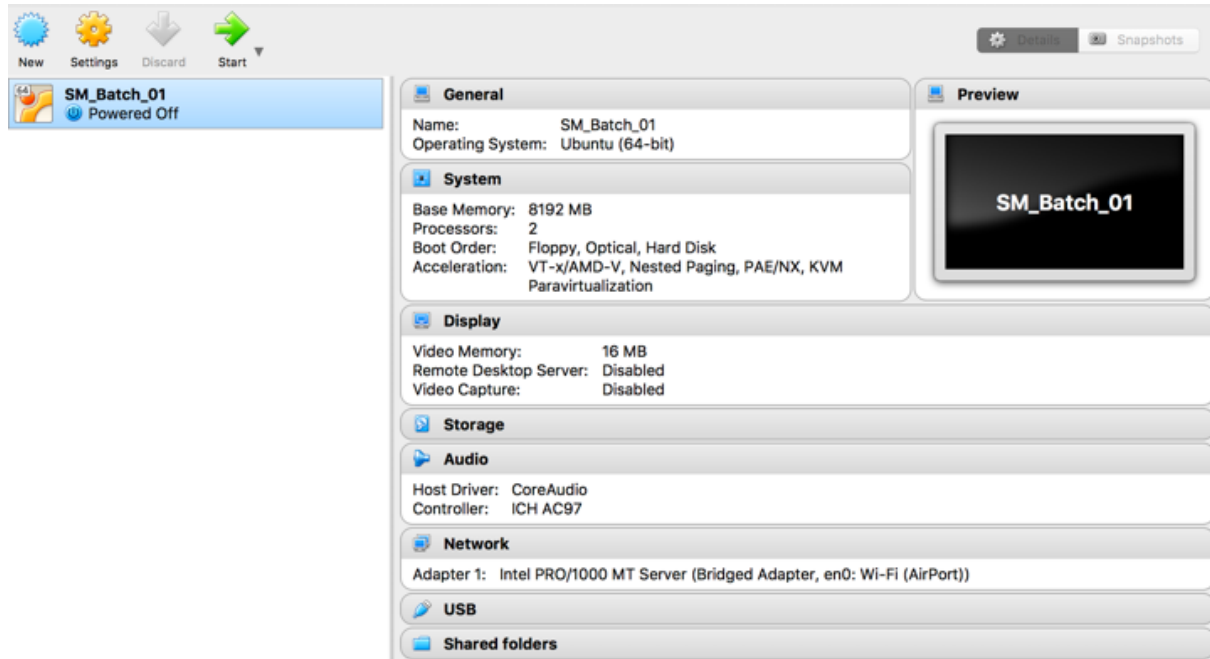
VirtualBox

The following steps can be used to import the virtual appliance into VirtualBox 5.2 or above.

- Open VirtualBox
- From the Manager window select "File", then "Import Appliance..."

- In the Name field, name the Batch Virtual Appliance e.g. "SM_Batch_01"
- Browse to the OVA file and click on the "Import" button

Once the VM has imported it should be visible on the VirtualBox Manager:



Amazon Web Services

This section explains how to create a Batch Virtual Appliance EC2 instance on the Amazon Web Services (AWS) platform by using the AWS VM Import/Export tool. This tool is designed for importing VM images from the OVA file format provided by Speechmatics. You will import the image as an Amazon Machine Image (AMI), from which you can then launch machine instances.

The information in this section is taken from the official AWS documentation and parts of it have been extracted to focus more on the particulars of the Speechmatics Batch Virtual Appliance. For more details of the Amazon VM image import process, please refer to <https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-image-import.html>

Prerequisites

There are a few pre-requisites that you will need to have setup before you can follow the instructions in this section:

- [AWS Command Line Interface](#) (CLI)
- Python 2.6.5 or higher

Please follow the recommendations on configuration of the AWS CLI by referring to the [Getting Started](#) guide.

Uploading the OVA file to S3

This section describes the process of uploading the Speechmatics OVA file to an Amazon S3 bucket from where it can be imported as an AMI instance. We recommend using a bucket in the same region where you want the AMI to be created and made available.

Once you've identified or created the S3 bucket on your account where the Speechmatics Batch Virtual Appliance OVA will be uploaded to, you can use any of the tools below to help with the upload of the OVA file.

- The following [AWS SDK libraries](#) support S3 multipart upload (which is the recommended method given the large size of the OVA file):
 - AWS SDK for Java
 - AWS SDK for .NET
 - AWS SDK for PHP

- AWS SDK for Python (Boto)
- AWS SDK for Ruby
- You can also use the [Multipart Upload API](#) directly
- User interface tools, for instance:
 - [S3 Browser](#)
 - [CloudBerry S3 Explorer](#)

For more information about the multipart uploads, see the [AWS documentation](#).

Importing the OVA as AMI instance

After the Virtual Appliance OVA file has been successfully uploaded to an S3 bucket, it's time to import the image.

See the AWS documentation that covers [uploading an image](#) for full details.

The steps that you will perform in this section include (in order):

- Creating a Service Role on your AWS account
- Assigning a Role Policy to this Service Role
- Importing the OVA for the Batch Virtual Appliance from the S3 bucket file

Creating an Import Service Role

First of all, a **service role** needs to be created on your AWS account. This allows certain operations, including downloading images from an S3 bucket.

Create a file named `trust-policy.json` with the following policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "Service": "vmie.amazonaws.com" },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:Externalid": "vmimport"
        }
      }
    }
  ]
}
```

Then use the `create-role` command from the AWS CLI to create a role named `vmimport`. You need to specify the full path of the `trust-policy.json` file:

```
aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-policy.json
```

You need to ensure that the `file://` prefix is prepended to the filename.

Creating a Role Policy

Create a file named `role-policy.json` with the following policy. Where you see `ova-bucket` it will need to be replaced with the name of the S3 bucket where the OVA file is stored.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

    "Action": [
      "s3:GetBucketLocation",
      "s3:GetObject",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3:::ova-bucket",
      "arn:aws:s3:::ova-bucket/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:ModifySnapshotAttribute",
      "ec2:CopySnapshot",
      "ec2:RegisterImage",
      "ec2:Describe*"
    ],
    "Resource": "*"
  }
]
}

```

Use the `put-role-policy` command to attach the policy to the role. You must specify the full path to the location of the `role-policy.json`:

```
aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document
file://role-policy.json
```

Importing the OVA

Importing the virtual appliance image (OVA) to Amazon EC2 as an Amazon Machine Image (AMI) is the next step.

Create a file named `containers.json` with the following content. Where you see `ova-bucket` it will need to be replaced with the name of the S3 bucket where the OVA file is stored. The below example has a file named `rtappliance-2.0.1-b21086.ova`.

```

[
  {
    "Description": "Real Time Virtual Appliance OVA",
    "Format": "ova",
    "UserBucket": {
      "S3Bucket": "ova-bucket",
      "S3Key": "rtappliance-1.1.0-b21086.ova"
    }
  }
]

```

Use the `import-image` command to create an import task (Specify the full path to the location of the `containers.json`):

```
aws ec2 import-image --description "Real Time Virtual Appliance OVA" --disk-containers
file://containers.json
```

The resulting JSON output will show an `ImportTaskId` which you can use to check the status of the import task. You do this by running the `describe-import-image-tasks` command:

```
aws ec2 describe-import-image-tasks --import-task-ids import-ami-abcd1234
```

You need to replace the task identifier with the `ImportTaskId` for your import task (`import-ami-abcd1234` in this example).

When the status is in the `completed` state the AMI is ready to use.

Security

The following ports should be opened to be able to submit jobs and manage and monitor the Speechmatics Virtual Appliance. For more background on creating security groups refer to the official [AWS documentation](#)

Real-time Virtual Appliance

Port	Description
8080/TCP	Used for the Management API to manage the virtual appliance
9000/TCP	WebSockets Speech API for submitting jobs
3000/TCP	Monitoring (Glances)

Batch Virtual Appliance

Port	Description
8080/TCP	Used for the Management API to manage the virtual appliance
8082/TCP	REST Speech API for submitting jobs
3000/TCP	Monitoring (Glances)

Launching a Virtual Appliance

Now that the Virtual Appliance has been imported, it will be available as an AMI which can be launched as an instance. To launch a Speechmatics Virtual Appliance, do the following:

- Login to the AWS console and find your image under **EC2 Service | Images**
- Right-click the image and choose **Launch**
- Refer to the **System requirements** section of the Speechmatics Quick Start Guide or Admin Guide to identify how much system resources is required for your set up. Choose the instance type that meets your requirement
- Choose **Review and Launch** from the console. Setup the Key Pair if required and choose **Launch** again.

Full instructions for launching instances can be found here:

<https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/launching-instance.html>

Network Configuration

Before starting the virtual appliance for the first time, it is important to consider the network settings that will be used. The section below describes the options.

Network interface mapping

Whilst the virtual appliance is powered off, the virtual network adaptor should be mapped to the correct physical adaptor on the host. The virtual interface must be mapped to a physical adaptor on which the Speechmatics Batch Virtual Appliance will be contacted. Steps are provided below for the supported hypervisors.

VMware ESXi

There is nothing to configure here. The network as specified during the import stage described above will be used.

VMware Workstation Player

Speechmatics recommends using bridged network mode. To ensure bridged networking is selected:

- Open VMware Workstation Player
- Right click on the virtual appliance e.g. "SM_App_01", and select "Settings..."
- Select the "Network Adapter" in the devices list
 - Select "Bridged: Connected directly to the physical network"
- Click "OK"

This will result in the VM using an IP Address for its use that is independent from that of the host.

VirtualBox

Speechmatics recommends using bridged network mode. To ensure bridged networking is selected:

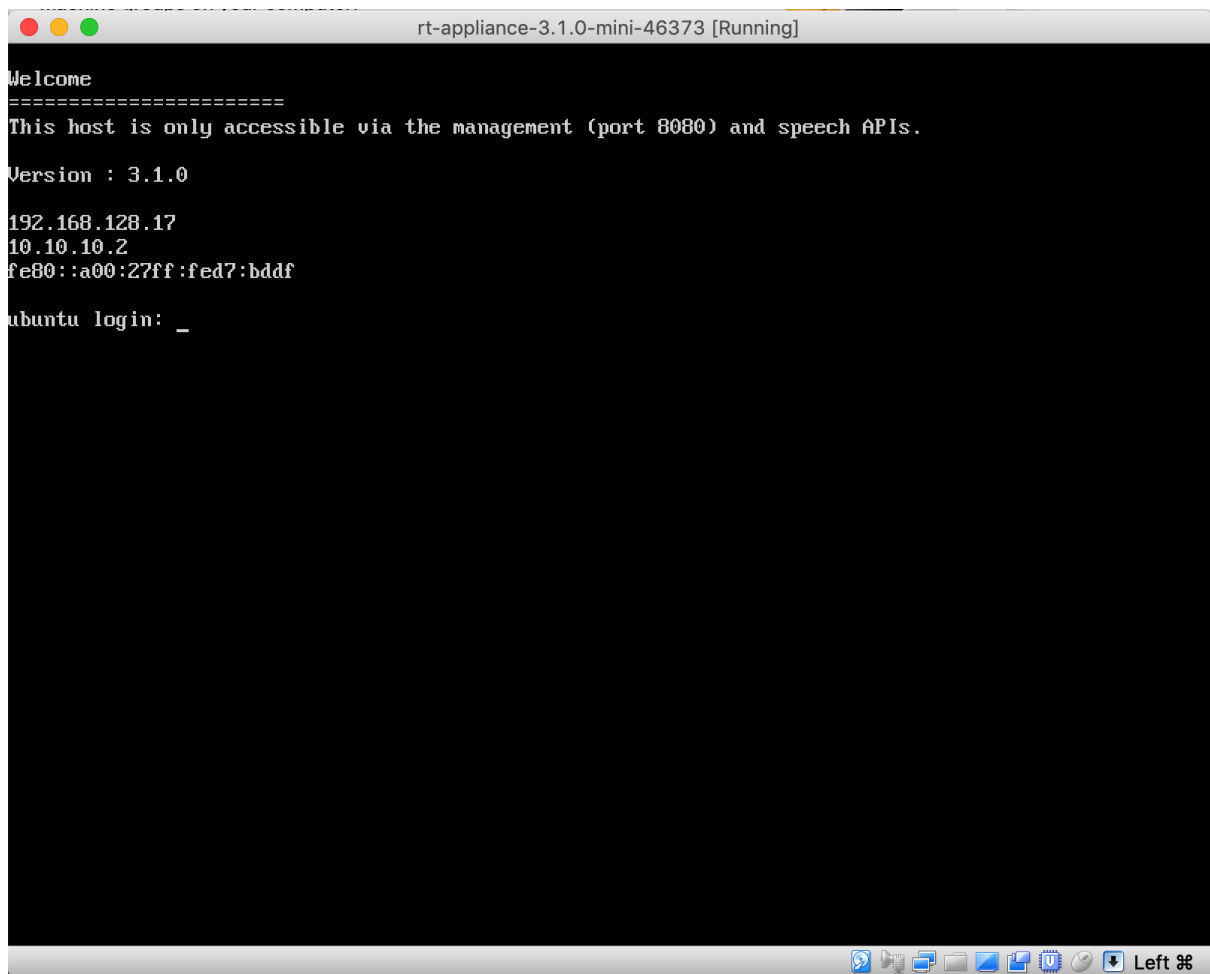
- Open VirtualBox
- Right click on the virtual appliance and select "Settings..."
- Select "Network" and from the "Attached to:" dropdown box, select "Bridged Adaptor"
- Click "OK"

This will result in the VM using an IP Address for its use that is independent from that of the host.

IP Configuration

When the Speechmatics Batch Virtual Appliance is started, the default behavior will be to dynamically acquire an IP address. If there is no DHCP service available on the network, it will fall back to an IP address automatically assigned.

The IP address information can be viewed by opening the virtual appliance console once it has booted, as shown below.



The screenshot shows a terminal window titled "rt-appliance-3.1.0-mini-46373 [Running]". The terminal output is as follows:

```

Welcome
=====
This host is only accessible via the management (port 8080) and speech APIs.

Version : 3.1.0

192.168.128.17
10.10.10.2
fe80::a00:27ff:fed7:bddf

ubuntu login: _
  
```

The terminal window also shows a standard Ubuntu desktop environment at the bottom with various system icons and a taskbar.

The screen shot above shows the 10.10.10.2 IP address as the fallback address. The other address shown was allocated by DHCP and should be used for all communication.

If DHCP cannot be used, a static IP address can be configured as described below.

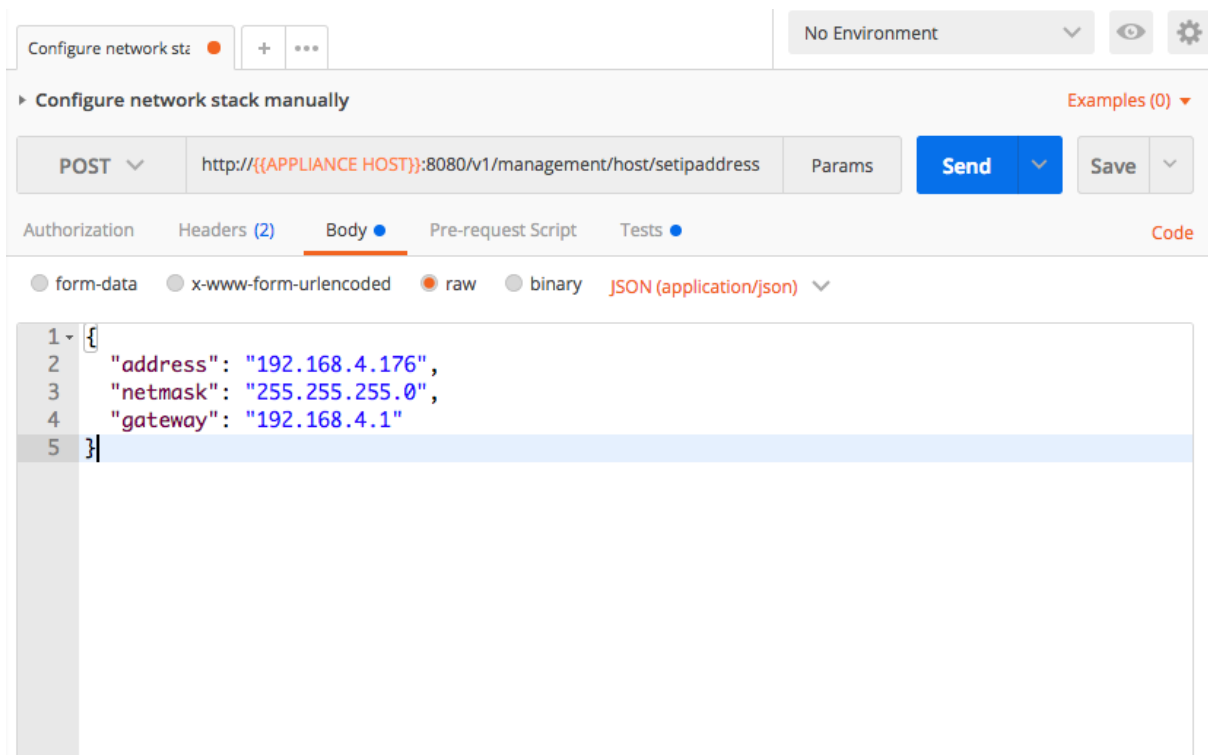
Configure static IP

To configure a static IP address, the Management REST API for the virtual appliance is used. The following information is required:

- **Method:** POST
- **URL:**
http://\${APPLIANCE_HOST}:8080/v1/management/host/setipaddress
- **Body Format:** JSON
- **Body:** address, netmask, gateway, nameservers

Where \${APPLIANCE_HOST} is the hostname or IP address of your Batch Virtual Appliance.

The example below shows use of [Postman](#) (available for free from the Chrome web store) to POST new IP settings.



You can optionally specify a list of nameservers to use (if none are specified then, 8.8.8.8 is used), for example this time using [curl](#) from the command-line to make the POST request:

```
curl -L -X POST 'http://${APPLIANCE_HOST}:8080/v1/management/host/setipaddress' \
-H 'Accept: application/json' \
-H 'Content-Type: application/json' \
-d '@network-config.json'
```

In this example, a local file network-config.json is used for the JSON configuration:

```
{
  "address": "192.168.128.96",
  "netmask": "255.255.255.0",
  "gateway": "192.168.4.1",
  "nameservers": ["208.67.222.222", "208.67.220.220"]
}
```

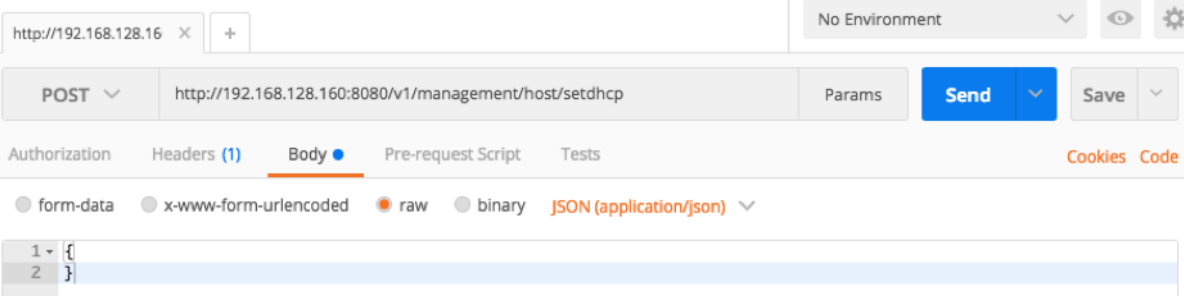

NOTE: once the POST is sent, the virtual appliance will automatically reboot. Check the console to verify the new IP address has been applied.

Configure DHCP IP

To configure a dynamic IP address using DHCP, the admin REST API is used as follows:

- **Method:** POST
- **URL:**
http://\${APPLIANCE_HOST}:8080/v1/management/host/setdhcp
- **Body format:** JSON

The example below shows how to use Postman to POST to the REST API in order to configure a DHCP address.



The screenshot shows the Postman interface for a POST request. The URL is `http://192.168.128.160:8080/v1/management/host/setdhcp`. The request body is in JSON format and contains a single line of code: `curl -L -X POST 'http://${APPLIANCE_HOST}:8080/v1/management/host/setdhcp' \ -H 'Accept: application/json'`.

NOTE: once submitted, the virtual appliance will automatically reboot. Check the console to verify the new IP address has taken affect.

Licensing

The Speechmatics Batch Virtual Appliance uses a cloud-based licensing mechanism, meaning that the under normal circumstances the appliance must be connected to the Internet, at least when the license is activated.

Note: For deployments where this is not possible, and where an offline license has been provided, it is possible to license the appliance without an Internet connection. Consult the Admin Guide for details on how to apply an offline license.

Your appliance must have been activated with a valid license before the Speech API can be used. Use of the Management API does not require a license. Please contact Speechmatics support support@speechmatics.com if you do not have a license.

Applying License

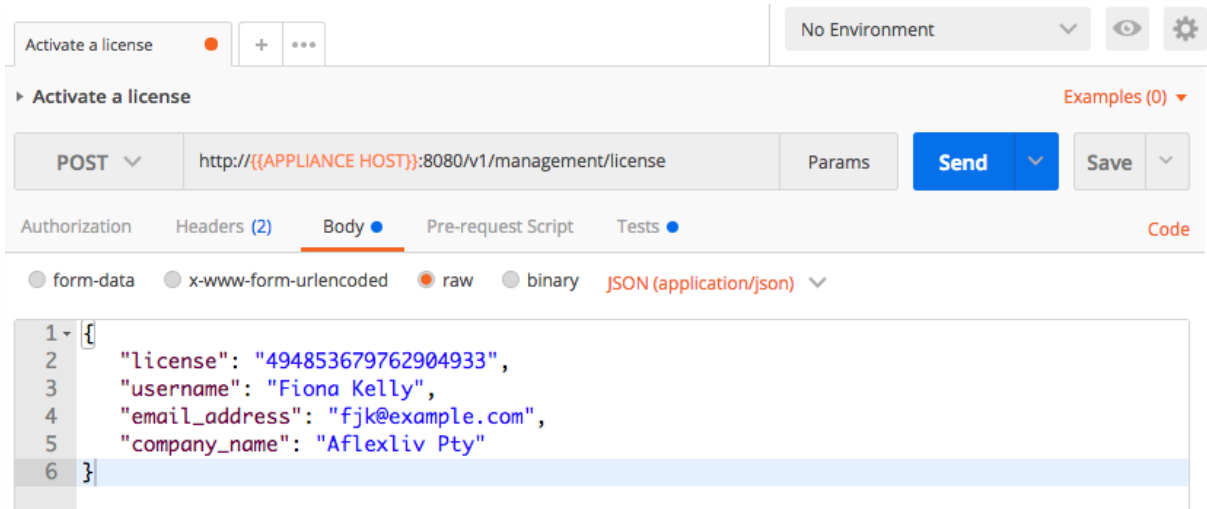
To apply the license that you have received from Speechmatics you use the Management API. The following information will be required:

- **Method:** POST
- **URL:**
http://\${APPLIANCE_HOST}:8080/v1/management/license
- **Body format:** JSON
- **Body:** license, username, email_address, company_name

Where `${LICENSE_CODE}` is the license code you've been provided with. The other fields (username, email_address and company_name) are optional, but we recommend that you fill them in with your details to help with support.

Note: Make sure when applying the license, that all the appliance services are running and that you have a route to the Internet; otherwise the activation will fail.

The example below shows how to use [Postman](#) to POST to the REST API to apply (activate) the license.



Or, the same request from the command-line:

```
curl -L -X POST 'http://${APPLIANCE_HOST}:8080/v1/management/license' \  
-H 'Accept: application/json' \  
-H 'Content-Type: application/json' \  
-d '{  
  "license": "494853679762904933",  
  "username": "Fiona Kelly",  
  "email_address": "fjk@example.com",  
  "company_name": "Aflexliv Pty"  
}' \  
| jq
```

The response should indicate that the licensed status is true. The licensing activation requires a connection to the Internet (using port 80). If you are behind a corporate firewall that does not allow a direct connection to the Internet then you can configure the appliance to use a network proxy or relay server to allow you to license the appliance. Full instructions on how to set this up are to be found later on in this guide.

Verify and Go (Batch)

This section explains how to verify the correct operation of the Batch Virtual Appliance using the REST Speech API.

Check that all the Speechmatics services within the appliance are up and running before passing the audio file. The Management REST API can be used for this.

- **Method:** GET
- **URL:**
`http://${APPLIANCE_HOST}:8080/v1/management/services`

To run a simple transcription job to test that everything is working use the Batch Virtual Appliance Speech API (on port 8082)

- **Method:** POST
- **URL:**
`http://${APPLIANCE_HOST}:8082/v1/user/1/jobs`

For example, you can use the following Speech API request using the curl command-line tool to transcribe an audio file 'sample.wav' and return the Job ID:

```
curl -s -L -X POST 'http://${APPLIANCE_HOST}:8082/v1/user/1/jobs/' \
  -H 'Accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F data_file=@sample.wav \
  -d 'config={ "type": "transcription",
              "transcription_config": { "language": "en" }
    }' \
  | jq '.id'
```

Where `${APPLIANCE_HOST}` is the hostname or IP address of your virtual appliance. The above assumes that `sample.wav` contains English speech; modify the language identifier in the job config to match the language you want to transcribe.

You can use the Job ID to get the status of the job:

```
curl -s -L -X GET 'http://${APPLIANCE_HOST}:8082/v1/user/1/jobs/${JOB_ID}/' \
  -H 'Accept: application/json' \
  | jq
```

Where `${JOB_ID}` is the Job ID (id field) that was returned when you submitted the job. Once the job is done, you use the Job ID to return the transcription:

```
curl -s -L -X GET 'http://${APPLIANCE_HOST}:8082/v1/user/1/jobs/${JOB_ID}/transcript?format=json-v2' \
  -H 'Accept: application/vnd.speechmatics.v2+json' \
  | jq
```

Under normal conditions, the job should take less than half the duration of the media file to process. So for example if you submit a MP3 file that is 60 minutes long, its transcription should be processed in less than 30 minutes. See the REST Speech API Guide for the list of language codes, how to use features of the API, the output formats that are supported, as well as more usage examples.

The Speechmatics Batch Virtual Appliance is now ready to use.

SSL Configuration

When the appliance is imported it contains a default self-signed certificate, so you can use HTTPS to access the appliance via the Management, Monitoring and Speech APIs. However, we recommend replacing this default SSL certificate with your own certificate, signed by your organisation or a trusted third-party certificate authority (CA).

Default behaviour

By default, our appliances allow connections over HTTP. The services on the appliance expose several ports for HTTP access, such as 8080 for the management API and 3000 for the monitoring API.

Since version 3.4.0 of the appliances, we also support HTTPS access to these services over port 443. To use HTTPS simply change the protocol used for API calls from `'http'` to `'https'`, and remove the port from the URL. If you are copying the examples from this document you can set the `$APPLIANCE_HOST` environment variable like this: `export APPLIANCE_HOST=localhost`.

Management API Examples

```
curl -L -X GET "http://${APPLIANCE_HOST}:8080/v1/management/services" \
  -H 'Accept: application/json'
```

To modify this to use a secure API call, change `http://` to `https://` and remove the port number `:8080` from the URL:

```
curl -L -X GET "https://${APPLIANCE_HOST}/v1/management/services" \  
-H 'Accept: application/json'
```

Note: If you are using a self-signed certificate (your own, or the Speechmatics certificate that is used by default), then you will see a warning like this when using the above curl command:

```
curl: (60) SSL certificate problem: self signed certificate
```

Warning: The default SSL certificate on the appliance is a self-signed certificate created by Speechmatics, which is not signed by any certificate authority. Your HTTP client or web browser may warn that this is insecure. This warning can be suppressed, for example with cURL by adding the `--insecure` flag, however customers who are serious about security should not be using the self-signed certificate. We recommend uploading your own SSL certificate to the appliance. Instructions for doing this can be found below.

Important: We have added `--insecure` to some of them cURL examples in this document so that the command trusts the self signed certificate. You won't need this option once you've uploaded your own certificate and configured your own system to trust it.

Monitoring API Example

With access to the Monitoring API (available on port 3000 if you are using HTTP) you will need to prefix the endpoint with `/monitor`. For example:

```
curl --insecure -L -X GET "https://${APPLIANCE_HOST}/monitor/api/3/mem"
```

Speech API Example

Access to the REST Speech API (available on port 8082 using HTTP), is also possible via HTTPS:

```
curl --insecure -L -X GET "https://${APPLIANCE_HOST}/v1.0/user/1/jobs/"
```

Using your own SSL certificate and private key

To use your own SSL certificate you'll need to upload your *certificate* file as well as the associated *private key* file.

- The **private key** file normally has a '.key' extension and should look similar to the example below.

```
-----BEGIN RSA PRIVATE KEY-----  
xqgLwi4gJ9+9Qkavpk3WpPFTTYUfVrCJNviKEn5wAltutqLQkRTcxJtrEk8trKI  
fCxeZo35yVhYmDGUIuAdAcPRTPj0XZkXQRhkITmD8TYMc/sVlJpFr+TAssGzute8  
... 21 lines redacted ...  
+bLv4aqI9tZrwpyeziaOuyQRhYodpAjhCyCFMkJjY59BKv/cqMHx8FPDQmaZ9Xs0  
SmE9JAKnDgF5yLHm1Q6WZ1/L/M4SkgIqEglF7ifLd5M3wskpmHia6/f8Fa2KwbBJ  
-----END RSA PRIVATE KEY-----
```

Note: We do not currently support encrypted/password protected private key files.

- The **certificate** file should be PEM encoded and normally has a '.crt' or '.pem' extension. It should look similar to this:

```
-----BEGIN CERTIFICATE-----  
MIIGuzCCBaOgAwIBAgIIHlfyznYUA8wDQYJKoZIhvcNAQELBQAwgbcQxhZG9udmVz  
BAYTA1VTMRAwDgYDVQQIEwdmcm9udmVzBjE6MDYwMDYwMDYwMDYwMDYwMDYwMDYw  
... 32 lines redacted ...  
P4LMbjCA4mqQvlipeSAn1E40rFL47zLcy+H9M0+Rw2CUiwL8QZFq+TAiIZ34tC  
UVCh52xpB9/BhO++QbGd1zObqDhcGEg8pJpJIycej9t4GN1eqNSudn0ibsQWew8=  
-----END CERTIFICATE-----
```

Both files should be in [PKCS8](#) format. If you have to upload a certificate chain, then the file you upload should contain the individual certificates concatenated, with your organisation's certificate first.

Uploading the certificate and key to the appliance

To upload your own certificate to the appliance you will need to make a POST request to the `/v1/security/sslcertificate` endpoint. This can be done using an HTTP client on the command line or with the management interface in a browser.

With the example shown here set `APPLIANCE_HOST` as appropriate (e.g. `export APPLIANCE_HOST=localhost` if your appliance is running locally):

```
curl --insecure -X POST "https://${APPLIANCE_HOST}/v1/security/sslcertificate" \
-F "keyfile=@appliance.key" -F "certfile=@appliance.crt"
```

Warning: Do not upload these files over HTTP, or you risk leaking the private key for your certificate.

If the upload succeeds then you should receive an HTTP 200 response with a success message:

```
{
  "success": true,
  "message": "certificate and private_key applied successfully"
}
```

Be aware that setting a new certificate will cause the web server in the appliance to restart which can take around five seconds. During this period, requests will still be served, however the old certificate will be used. Existing connections such as job uploads or WebSocket streams will not be interrupted.

You can check the certificate on the appliance by using the `openssl` tool:

```
$ openssl s_client -connect ${APPLIANCE_HOST}:443
```

Disabling HTTP access

If desired, HTTP access may be disabled, which will cause any requests to the appliance using HTTP to fail. To do this, make a POST request to the `/v1/security/insecureports` endpoint, with a JSON body containing

```
{"enable_insecure_ports": false}:
```

```
curl -X POST "https://${APPLIANCE_HOST}/v1/security/insecureports" \
-H "Content-Type: application/json" \
-d '{"enable_insecure_ports": false}'
```

If the request succeeded then you should receive an HTTP 200 response. The web server in the appliance will take around five seconds to restart. Now, when attempting to make an HTTP request to the appliance you should see that no response is returned:

```
curl -X GET "http://${APPLIANCE_HOST}:8080/v1/management/services"

curl: (52) Empty reply from server
```

Enable Basic Authentication for Admin

An admin password can be set to enable [HTTP basic authentication](#) for an `admin` user. Note that **authentication is only enforced when using HTTPS**. If you set an admin password then you **must** also disable HTTP access as described in the previous section. If you do not do this then it will be possible for someone else to override the admin password by making an unauthorized HTTP request.

To set a password, make a POST request to the `/v1/security/adminpassword` endpoint. The username for basic auth is always `admin`.

```
curl -X POST "https://${APPLIANCE_HOST}/v1/security/adminpassword" \  
  -H "Content-Type: application/json" \  
  -d "{ \"password\": \"example\" }" \  
  
{ "success":true, "message":"nginx_restart"}
```

If this request was successful then you should receive an HTTP 200 response with a success message. The web server in the appliance will take around five seconds to restart. All requests to HTTPS endpoints will now require a valid `Authorization` header as specified by [RFC7617](#). Unauthenticated requests will fail, for example:

```
$ curl -X GET "https://${APPLIANCE_HOST}/v1/management/services" \  
<html> \  
<head><title>401 Authorization Required</title></head> \  
<body> \  
<center><h1>401 Authorization Required</h1></center> \  
<hr><center>nginx/1.17.6</center> \  
</body> \  
</html>
```

Authenticated requests should succeed. If you are using curl then the `--user` flag can be used to set the username and password (separated with a colon). If you're using the Management UI in a browser than a prompt will appear for a username and password.

```
$ curl --insecure -X GET --user "admin:example" \  
"https://${APPLIANCE_HOST}/v1/management/services"
```

If you have disabled HTTP access then it should now be impossible to make requests to the appliance without knowing the admin password. Please be aware that plain HTTP access does **not** require the admin password, and should be disabled if you are using a password.

FAQs

How do I reset the SSL settings?

If you have made a mistake in your SSL configuration, it is possible to reset the appliance to it's default settings. This will return it to using the self-signed certificate from Speechmatics, and will delete any configured admin password. If you have disabled HTTP access then you need to know the existing admin password in order to do this.

To do this, make a DELETE request to the `/v1/security/reset` endpoint:

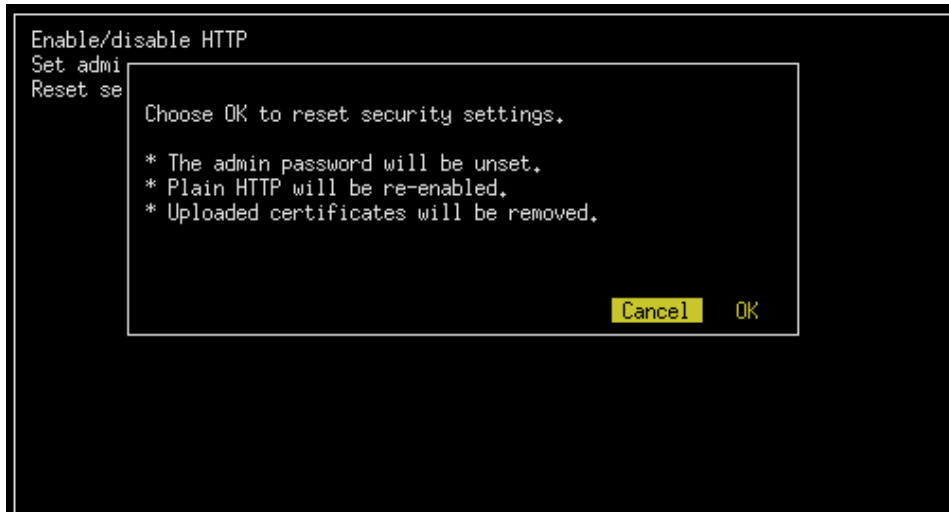
```
$ curl -X DELETE --user "admin:$PWD" "https://${APPLIANCE_HOST}/v1/security/reset" \  
{ "success": true, "message": "nginx_restart"}
```

What if I forget the admin password?

If you have forgotten the admin password you have set, and have disabled HTTP access to the appliance then it will not be possible to interact with the appliance over HTTP/HTTPS. Fortunately there is a way to reset the SSL configuration if you have direct access to the appliance's console (through the hypervisor that you use).

See the 'Administration -> Services -> Console for Advanced Troubleshooting' section for instructions on how to access the console.

Once you have opened the console open the 'Security' menu and select the 'Reset security' option to reset all security settings. It is also possible to toggle HTTP access and set the admin password using this interface.



What versions of SSL/TLS do you support?

We support TLS 1.2 and TLS 1.3. We do not support earlier versions of TLS/SSL as these are considered weak. In general we would recommend you keep your client frameworks up to date with the latest security patches and try to use the strictest TLS configuration that you can.

What cipher suites do you support?

For TLS 1.3 we support the following cipher suites that are considered strong (in server-preferred order):

- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_GCM_SHA256

For TLS 1.2 we support the following cipher suites that are considered strong (in server-preferred order):

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256

Other cipher suites are available for TLS 1.2, but they are considered to be weak. Our recommendation is that you select one of the above cipher suites.

Networking

Network Requirements

When the virtual appliance is started for the first time it will automatically try to acquire an IP address using DHCP. If it is able to successfully acquire an address, it will be displayed on the VM console along with the fallback IP address: 10.10.10.2. However, if there is no DHCP server available on the network only the 10.10.10.2 IP address will be displayed.

The 10.10.10.2 address is a fallback address enabling communication with the virtual appliance when no DHCP services are available. This address should be used temporarily to set a static IP address if no DHCP is available. To do this, ensure that the client connecting to this address is on the same network by assigning it a suitable IP address (e.g. 10.10.10.3/24).

Note: The appliance uses an internal network of **10.254.0.0/22**. You need to ensure that any network you use does not have an IP address conflict with anything in this range: 10.254.0.0 - 10.254.3.255.

Configure Static IP

The virtual appliance can be configured to work on any IP network.

Setting a static IP requires three parameters: the IP address, subnet mask and default gateway. You set the static IP address like this:

```
curl -L -X POST 'http://${APPLIANCE_HOST}:8080/v1/management/host/setipaddress' \
-H 'Accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
  "address": "192.168.128.160",
  "netmask": "255.255.255.0",
  "gateway": "192.168.128.1"
}' \
| jq
```

Note: Once the POST is sent, the virtual appliance will automatically reboot. Check the console (or make an API call) to verify the new IP address has taken affect.

Configure DHCP

You can also change back to using DHCP. Before undertaking this, ensure the network the virtual appliance is on has DHCP enabled.

```
curl -L -X POST 'http://${APPLIANCE_HOST}:8080/v1/management/host/setdhcp' \
-H 'Accept: application/json'
```

NOTE: once submitted, the virtual appliance will automatically reboot. Check the console to verify the new IP address has taken affect.

Firewall Ports

There are several firewall rules that may need to be enabled to ensure the communication can be made to the virtual appliance:

- 8080/TCP - Used for the Management API to manage the virtual appliance
- 8082/TCP - Speech API for submitting jobs
- 3000/TCP - Monitoring

Using Proxies

If the network that you are deploying your appliance into does not have a direct route to the Internet, you may need to use a proxy server in order to talk to the cloud-based license service. See the relevant section in Licensing (below) for details on how to set this up.

Virtual Appliance Scaling

Real-Time Virtual Appliance Scaling

This section explains how to scale the Real-Time Virtual Appliance, and gives advice on how to make sure you've allocated enough resources for your workload.

Worker Limits

The number of concurrent workers can be restricted using the Management API. This can be used to ensure that the system resources do not get exhausted by clients starting more sessions than expected. The maximum number of

concurrent workers is set for the entire system, irrespective of which language packs are being used. The default number of maximum concurrent workers is 1.

View Maximum Workers

Use a GET request to the `maxworkers` endpoint to view the maximum number of workers:

```
curl -L -X GET 'http://${APPLIANCE_HOST}:8080/v1/management/maxworkers' \  
-H 'Accept: application/json' \  
| jq
```

This shows the maximum number of workers that can run concurrently on the appliance. If more sessions are opened by clients using the Speech API then you will receive the job error: `No worker can be scheduled because the service is at capacity.`

Setting Maximum Workers

Before changing the maximum number of concurrent workers for real-time transcription, it is important that the virtual appliance has enough system resources (CPU and RAM) to support the new requirement (see the Batch Virtual Appliance system requirements). This example shows how to set the maximum number of concurrent workers to 5:

```
curl -L -X POST 'http://${APPLIANCE_HOST}:8080/v1/management/maxworkers' \  
-H 'Accept: application/json' \  
-H 'Content-Type: application/json' \  
-d '{ "count": "5" }'
```

As a rule of thumb, each concurrent worker will require 1 vCPU and up to 2GB RAM.

Batch Virtual Appliance Scaling

This section explains how to scale the Batch Virtual Appliance, and gives advice on how to make sure you've allocated enough resources for your workload.

Worker Limits

The number of concurrent workers (jobs) can be restricted using the Management API. This can be used to ensure that the system resources do not get exhausted by clients starting more transcriptions than expected. The maximum number of concurrent workers is set for the entire system, irrespective of which language packs are being used. The default number of maximum concurrent workers is 1.

View Maximum Workers

Use a GET request to the `maxworkers` endpoint to view the maximum number of workers:

```
curl -L -X GET 'http://${APPLIANCE_HOST}:8080/v1/management/maxworkers' \  
-H 'Accept: application/json' \  
| jq
```

The response will indicate the maximum number of workers that can run concurrently on the appliance. If more jobs are submitted by clients using the Speech API then these will be queued up and processed once there is spare capacity on the appliance.

Setting Maximum Workers

Before changing the maximum number of concurrent workers, it is important that the virtual appliance has enough system resources (CPU and RAM) to support the new requirement (see the Batch Virtual Appliance system requirements).

This example shows how to set the maximum number of concurrent workers to 5:

```
curl -L -X POST 'http://${APPLIANCE_HOST}:8080/v1/management/maxworkers' \
-H 'Accept: application/json' \
-H 'Content-Type: application/json' \
-d '{ "count": "5" }'
```

As a rule of thumb, each concurrent worker will require 1 vCPU and up to 5GB of RAM (depending on the quality of the audio).

If the number of jobs submitted exceeds the maximum number of concurrent workers then jobs will start to be queued, and the real-time factor (RTF) will increase, meaning you will wait longer for your transcripts to be made available.

Monitoring

Appliance resources can be monitored at a system-wide level. Exhaustion of any of the resources can have a negative impact on the speed of the transcription.

The following resources that can be monitored:

Resource ID (rId)	Description
cpu	Provides the CPU usage across all the vCPU assigned
mem	Provides the total RAM usage of the appliance

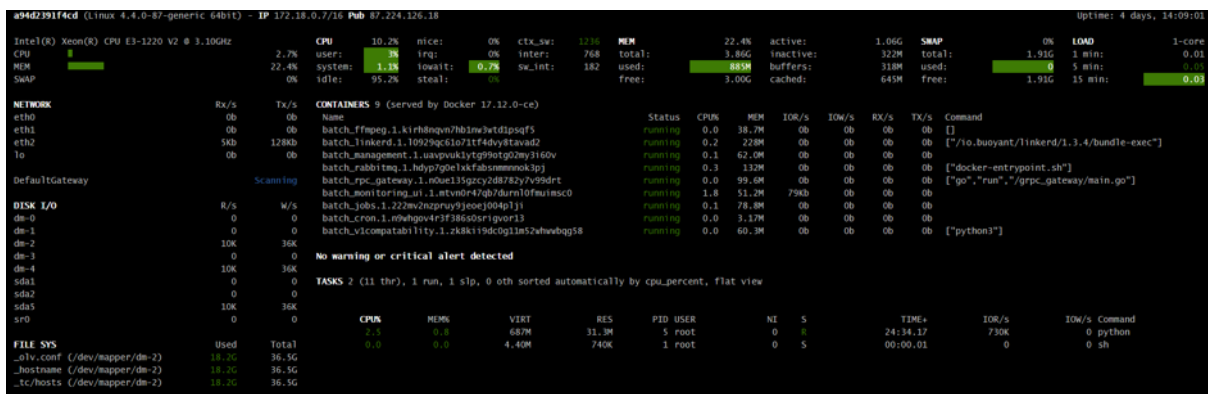
Here is an example GET request for the `mem` (RAM) resource:

```
curl -L -X GET 'http://${APPLIANCE_HOST}:8080/v1/management/resource/mem' \
-H 'Accept: application/json' \
| jq
```

Here is an example response:

```
{
  "rId": "mem",
  "percentage": 10.9,
  "value": 0,
  "intValue": 0
}
```

For advanced monitoring, a utility called [Glances](#) is available that runs on TCP port 3000. It allows real-time resource stats to be monitored on the Batch Virtual Appliance. The easiest way to access this is via a web browser using the link `http://${APPLIANCE_HOST}:3000/` in the address bar.



It is also possible to access the Glances API using XML-RPC or HTTP REST (for JSON output), for example:

```
curl -L -X GET 'http://${APPLIANCE_HOST}:3000/api/2/mem/percent' \
  -H 'Accept: application/json' \
  | jq
```

For more information on the HTTP REST interface, consult the [Glances documentation](#).

Services

The virtual appliance has internal services that are required for operation.

There are system-wide services, and services specific to transcription workers for a given language.

For the Batch Virtual Appliance, this table lists the services:

Service Name (Begins with)	Description	Required Status
batch_ffmpeg...	Conversion of audio	Running
batch_rpc_gateway...	RPC endpoint	Running
batch_license...	Licensing service	Running
batch_linkerd...	Internal Networking	Running
batch_notifier...	Callback function	Running
batch_management...	Management functions	Running
batch_rabbitmq...	Job Queue	Running
batch_monitoring_ui...	Monitoring Web GUI	Running
batch_cron...	Completed job clean-up	Running
batch_v1compatibility...	V1 REST API	Running
batch-jobs...	Used to perform ASR and transcription	Running
batch_nginxlb...	HTTP gateway	Running

The service will always have a current state, these states include:

Service Status	Description
running	Service has started and is running
created	Service is in the process of starting
exited	Service has stopped and is no longer running

Service status

This can be used to ensure all services have the required status to operate (see table above). Example: GET to list services and corresponding status:

```
curl -L -X GET 'http://${APPLIANCE_HOST}:8080/v1/management/services' \
  -H 'Accept: application/json' \
  | jq
```

If the appliance has been licensed then you will see a return like this (for the Batch Virtual Appliance):

```

{
  "service_status": [
    {
      "service": "batch_management.1.iu9c86olweubdmi99wddlcklr",
      "status": "running"
    },
    {
      "service": "batch_ffmpeg.1.jre8d5lempzmsqfki9o871a62",
      "status": "running"
    },
    {
      "service": "batch_notifier.1.idqxysv0srzeg2vkkorh1zjfh",
      "status": "running"
    },
    {
      "service": "batch_linkerd.1.af4t03setx3m64s15s9yawysl",
      "status": "running"
    },
    {
      "service": "batch_batch-cron.1.y5ql8ryyqlxwlyx84q9q3lfrn",
      "status": "running"
    },
    {
      "service": "batch_license.1.7ecytnlzd6hso3jxauvbyvfyi",
      "status": "running"
    },
    {
      "service": "batch_rabbitmq.1.l8ny2q6b2xhz0yr5bxwodbtog",
      "status": "running"
    },
    {
      "service": "batch_monitoring_ui.1.v180r4tq7dlcbfhc3vxyukpdo",
      "status": "running"
    },
    {
      "service": "batch_rpc_gateway.1.fb3ryh2a4d41sy628bhiogyx4",
      "status": "running"
    },
    {
      "service": "batch_postgres.1.wfy284tvznpnmgj22xaw11b55",
      "status": "running"
    },
    {
      "service": "batch_jobs.1.z4vftf8vv42uzmxd4ra3235ao",
      "status": "running"
    },
    {
      "service": "batch_v1compatibility.1.j5rvj3wpqwdfnl848ci42iix2",
      "status": "running"
    }
  ]
}

```

Note: After a service is restarted it will have a random string identifier post fixed to its name.

Service restart

If required for troubleshooting you may need to restart all the services. During the restart, all transcription will stop. The following command performs a service restart:

```
$ curl -X DELETE 'http://<APPLIANCE_HOST>:8080/v1/management/services' \
-H 'Accept: application/json'
```

Access Logs

The individual services on the system provide log files that can be collected to help with troubleshooting. The service name will need to be provided when retrieving logs. See above for instructions on how to view the names of the running services

The following parameters are available when accessing logs:

Name	Description	Required Status
name	Name of the service to collect the logs for	Required
count	Number of log lines wanted, defaults to 100; if all lines are to be returned set to -1	Optional

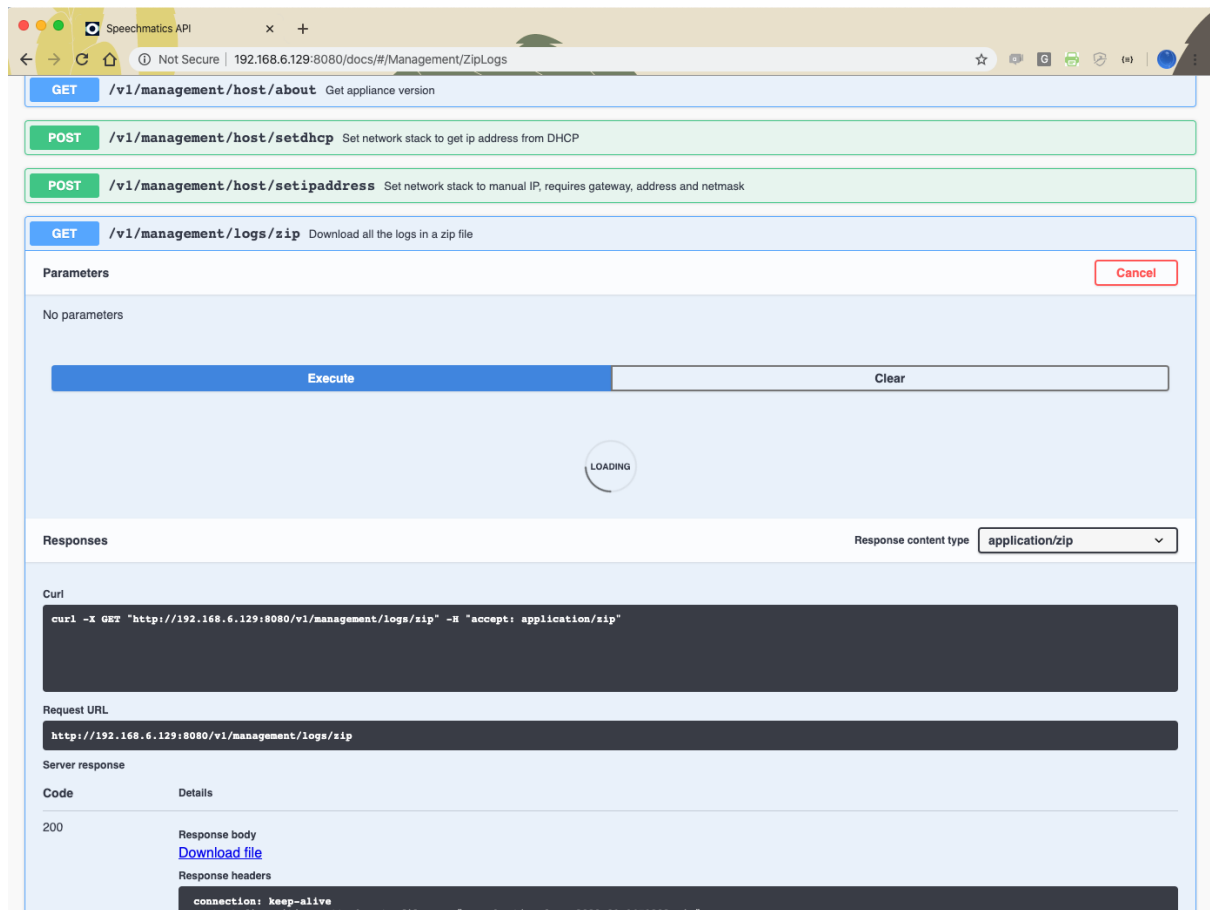
Example: GET to retrieve logs for batch_monitoring_ui service:

```
curl -L -X GET
'http://${APPLIANCE_HOST}:8080/v1/management/logs/batch_monitoring_ui.1.mtvn0r47qb7durnl0fmuimsc0'
\
-H 'Accept: application/json' \
| jq -r '.log_lines'
```

If you want to download *all* the logs (in order to provide information for a support ticket for instance) as a ZIP file, then it is possible to do this using the following command:

```
curl -L -X GET 'http://${APPLIANCE_HOST}:8080/v1/management/logs/zip' \
-H 'Accept: application/json' \
-o ./speechmatics.zip
```

It is also possible to do this directly from the Swagger UI by entering in the following URL to your browser: <http://192.168.6.129:8080/docs/#/Management/ZipLogs>, and then clicking on the download link when the ZIP file is ready.



System restart

If the virtual appliance becomes unresponsive, there might be a need to restart it. If this is the case, it's recommended that the system is restarted through the management API, like this:

```
curl -L -X DELETE 'http://${APPLIANCE_HOST}:8080/v1/management/reboot'
```

If the Management API is not available, then you should reboot the appliance from the hypervisor console. For further information on how to restart the virtual machine via the console, please follow the manufacturers advice.

Troubleshooting

There may be times unexpected behavior is observed with the Batch Virtual Appliance. If this is the case the following should be performed/checked:

- Check the license is valid (see licensing)
- Check the worker services are running
- Check the resources (CPU, memory & disk) to ensure they are not exhausted
- Restart all the services
- Restart the virtual appliance
- Collect logs and contact Speechmatics support: support@speechmatics.com.

Transcription job failure

If your transcription job fails with an `error` job status, more information can be found by looking at the logs from the `jobs` container (using the Management API, as previously described). Search the logs for the job id corresponding with your failure. If you see a `SoftTimeLimitExceeded` exception, this indicates that the job took longer than anticipated and as such was terminated. This is typically caused by poor VM performance, in particular slow disk IO operations

(IOPS). If issues persist it may be necessary to improve the disk IO performance on the underlying host, or you may need to increase the RAM available to the VM such that memory caches can be taken advantage of. Please consult the section above on Host requirements, and the optimization advice specific to your hypervisor to ensure that you are not over-committing your compute resources.

Illegal instruction errors

If jobs fail repeatedly and you see `Illegal instruction` errors in the log information for these jobs then it is likely that the host hardware you are running on does not support AVX. The host machine requirements for the Batch Virtual Appliance must meet the following minimum specification: Intel® Xeon® CPU E5-2630 v4 (Sandy Bridge) 2.20GHz (or equivalent). This is important because these chipsets (and later ones) support Advanced Vector Extensions (AVX). The machine learning algorithms used by Speechmatics ASR require the performance optimizations that AVX provides.

You can check this by looking in the management log when the appliance starts up. If you see a message like this:

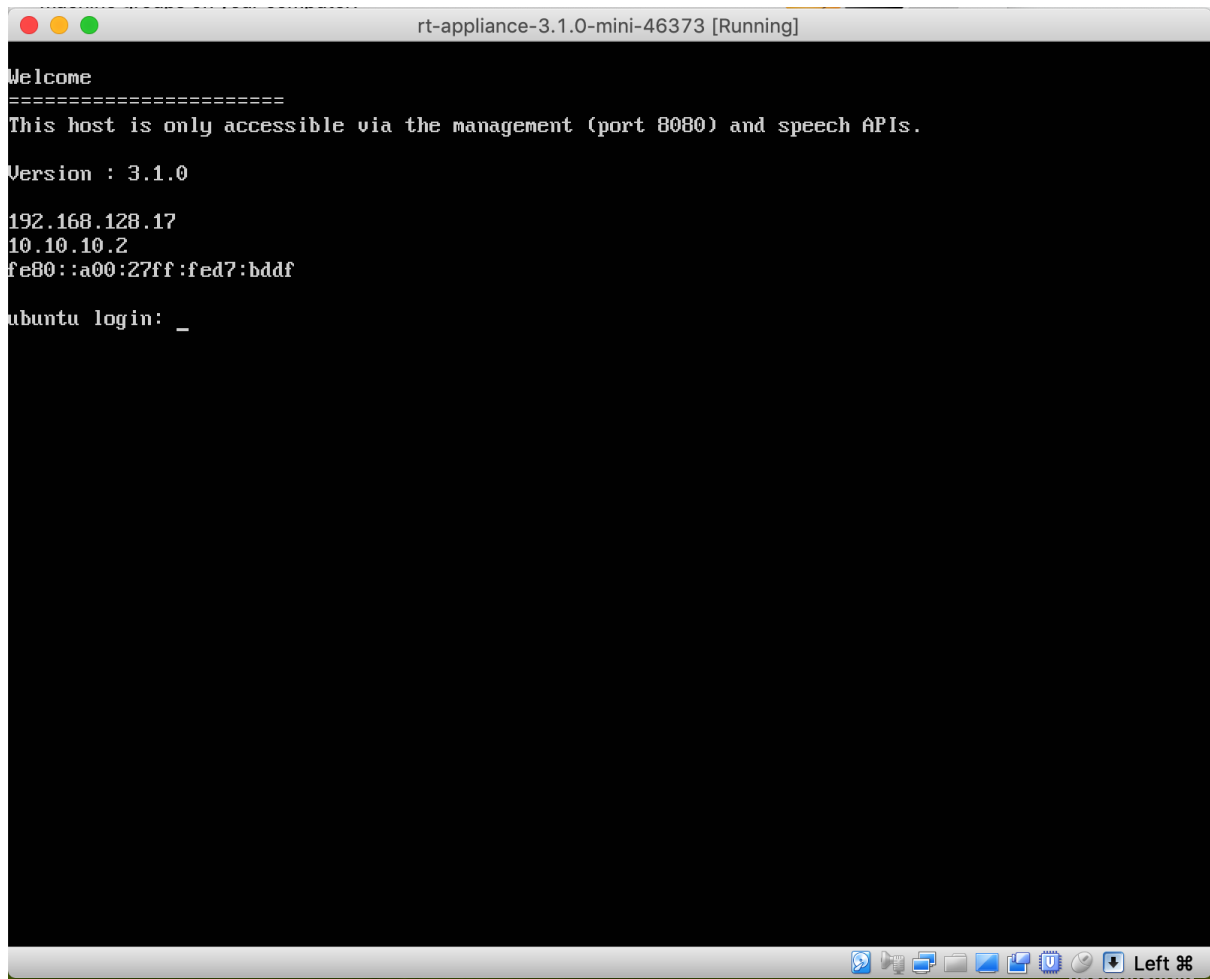
```
2019-03-26 16:53:07,136    sm_management.app    ERROR    Processor not AVX capable. Tensorflow
language models cannot run.
```

Then it means that your host's CPU does not support AVX, or that your hypervisor does not have AVX support.

A console is available to help with advanced troubleshooting in the event that the Management API is unavailable. It is described in the next section.

Console for Advanced Troubleshooting

In the event that the Management API is unavailable (it is unresponsive, or there is no network connectivity) you can use the console to restore network connectivity, restart the appliance, or view information about services. To use this you need to use your hypervisor's GUI to access the logon screen for the appliance.



From this screen use the CTRL+ALT+F5 key combination to get to the console. Once you are in the console you have the following menu options available:

- License
- Networking
- Reboot
- Services
- Shutdown
- Tools
- Workers



The home screen shows high-level information about the appliance: IP addressing, software version and license status.

In the **System status** panel the **API responding** indicator shows the state of the Management API. **Network status** shows the IP address the appliance is currently configured with, and **ASR status** shows the license state and available storage space on the appliance.

In the event that you need to provide information to Speechmatics support you may be asked to connect to the console and provide this information. This section provides some tips on how to use the console to perform basic troubleshooting yourself.

Note: We recommend that you use the Management API for most troubleshooting tasks as it is easier to use. The console can be used in the event that the Management API is unavailable, but it does not provide all the features of the Management API.

License

The [Licensing Troubleshooting](#) section provides detailed instructions on how to use the Management API to resolve common licensing issues. If you cannot use the Management API then you can still use console to check the license status and perform basic licensing steps.

Networking

You can use the networking option to configure a static IP address, or use DHCP.

Reboot and Shutdown

Reboot and Shutdown options exist to allow you to restart or shutdown the appliance from the console. You will be asked to select OK to confirm.

Services

From this menu you can access the list of services that are running on the appliance. Selecting a service shows the log entries for that service.

Tools

This menu allows you to access a number of useful Unix utilities that can be used for advanced troubleshooting. In order to help progress a support ticket you may be asked to provide the output (ie. a screenshot) from running one of these commands.

Workers

This allows you to view and change the maximum number of workers allowed to run concurrently.

Security

The appliance is designed to be installed within your own security perimeter. It has its own firewall installed to only allow ingress to ports that are required for its management, monitoring and Speech APIs.

Overview

The appliance uses a microservices architecture running on a customized Ubuntu machine. [AppArmor](#) default security policies are used to protect the OS and running applications on the appliance.

Data on the appliance (including audio and video data that is submitted via the Speech API, logs, and output transcripts) are encrypted on disk.

Firewall Ports

There are several firewall rules that may need to be enabled to ensure the communication can be made to the virtual appliance:

- 8080/TCP - Used for the Management API to manage the virtual appliance
- 3000/TCP - Monitoring (Glances)
- 8082/TCP - REST Speech API for submitting jobs (batch ASR)
- 9000/TCP - WebSocket Speech API for real-time ASR

Securing your Deployment

The WebSocket Speech API for real-time uses the secure `wss` protocol (using a self-signed certificate). However, access to the Management API, Monitoring API (Glances) and Speech API is not secured (`http` only), and no authorization tokens or passwords are required for access to the APIs. It is therefore up to the customer to deploy the appliance behind a load balancer or gateway that can provide those features if you need them. This is especially important if you are intending to deploy your appliances onto a public cloud (for example as an Amazon EC2 instance).

Batch Virtual Appliance

Overview

The Speechmatics Batch Virtual Appliance exposes a REST Speech API to enable communication between a client application and the appliance over a HTTP connection. This provides the ability to convert a media file into a text transcript, providing words, speaker, and timing information.

Terms

For the purposes of this guide the following terms are used.

Term	Description
Client	An application connecting to the Batch Virtual Appliance using the Transcription API. The client will provide audio containing speech, and process the transcripts received as a result.
Management API	The REST API that allows administrators to manage the virtual appliance over port 8080. To access the documentation you can use the following URI: <code>https://{APPLIANCE_HOST}:8080/docs/</code> , where

	<code>\${APPLIANCE_HOST}</code> is the IP address or hostname of your appliance.
Speech API	The REST API that allows users of the appliance to submit ASR jobs over port 8082. The endpoint <code>http://\${APPLIANCE_HOST}:8082/v1.0/</code> is used. This is the API that is described in this document.
Batch Virtual Appliance	The appliance (VM) that provides ASR transcription capability.

Getting Started

In order to use the REST Speech API you need access to a Batch Virtual Appliance. See the Speechmatics Virtual Appliance Installation and Admin Guide on how to install and configure the appliance.

You do not need user credentials (such as an authorization token) to use the Speech API with the Batch Virtual Appliance. Otherwise the Speech API is very similar to the Speechmatics API V2 used to transcribe speech to text on the Speechmatics Speech as a Service (SaaS) platform (available from <https://asr.api.speechmatics.com/v2/>).

Audio Formats

A variety of audio formats for input are supported; there is no need to specify the audio format when it is submitted for transcription; the Batch Virtual Appliance automatically detects the format and handles it using the correct decoder. The following formats have been tested: WAV, MP3, M4A.

Note: the native formats are 16KHz or 8KHz (PCM32 LE) WAV; for the best results and performance we recommend that you submit files in that format.

API Versions

The legacy V1 API that the Batch Virtual Appliance currently supports will be discontinued in a future release as we align the product with the same V2 API used by the Speechmatics SaaS: <https://asr.api.speechmatics.com/v2/docs>. We recommend that customers familiarise themselves with the configuration object used to specify job configurations, and only use form parameters for callback notifications. Future notices will be provided to announce the end of life of the V1 API, and provide detailed instructions on migrations to the V2 API.

[NOTE] Endpoint for Speech API

The same endpoint is used for both legacy V1 and V2 APIs: `http://${APPLIANCE_HOST}:8082/v1.0/`

There are three things to note here:

- The scheme used is http. The https scheme is not currently supported; if you want to use https then you will need to place your appliance behind a device or service capable of providing SSL offload.
- The port used is 8082.
- The endpoint `/v1.0` is used, even when using the newer V2 features.

We recommend that you use the config object to access the appliance; the older V1 API will be retired in future. For full details of the new V2 API, see the [Speechmatics ASR REST API](#) section.

Transcription Formats

Three output formats for transcription are available: `json` (the default), `json-v2`, and `txt`. The default format (`json`) is the base output format similar to the Speechmatics V1 SaaS. The `json-v2` format is a richer format that fully supports new features such as channel diarization, custom dictionary and advanced punctuation. If the output format is set to `txt`, the file is returned in plain text rather than JSON format.

Authentication

No authentication is required in order to call the API. SSL is not currently used for this API; all calls must use HTTP on port 8082.

Troubleshooting

If you have problems making a call, ensure that you are using exactly the same URI format as shown in this document. For instance, not including the trailing '/' character on the URIs will cause a 302 redirect response to be sent – if your client does not handle redirects then this may cause problems.

Example Usage

The Speech API is a REST API that enables you to create and manage transcription jobs by uploading audio files to the Speechmatics Batch Virtual Appliance, and downloading the resulting transcriptions.

If you are familiar with the Speechmatics SaaS then you will see that the Speech API for the Batch Virtual Appliance is very similar. With the Speech API on the Batch Virtual Appliance however, there is no need to supply an API Authentication Token.

[warning] User ID in request URL

Although you do not need an API auth token, you do need to supply a User ID in the URL, although this is a dummy value to maintain compatibility with the V1 SaaS API. It can be any positive integer, and can be used to track transcription requests since the ID that you use will be returned in any job response.

The base URI for the Speech API requests looks like this:

```
http://${APPLIANCE_HOST}:8082/v1/user/1/jobs/
```

Where `${APPLIANCE_HOST}` is the IP address or hostname of the appliance you want to use. You must use HTTP (HTTPS is currently not supported directly by the appliance), on port 8082. A user ID of 1 is used for all the examples in this document (you can use any positive integer value however).

You can access a dashboard on the appliance from any browser by navigating to the following URL:

```
http://${APPLIANCE_HOST}:8080/help
```

From here you can get to documentation links, and to Swagger UI pages which let you access the Management API and the Speech API. The Speech API can be used to easily submit and view jobs from a browser:

```
http://${APPLIANCE_HOST}:8082/v1
```

[warning] Unsupported parameters in Swagger UI

The Batch Virtual Appliance supports many of the same job settings as the V1 SaaS; however the following fields are not supported and will produce a '501 Not Implemented' error if you attempt to use them: `data_url`, `text_url`, `text_file` and `notification_email_address`.

All examples in this document use [curl](#) to make the REST API call from a command line. We recommend using retry parameters, so that retry attempts can be made for at least one minute. With the curl command this is done with the `--retry 5 --retry-delay 10` parameters. This has been omitted from the examples in this document for brevity.

All successful API calls return the HTTP body in JSON format along with status code 200 OK. Users of curl will see this displayed in their terminal; other interfaces may need a JSON parser.

Submitting a Job

Sample Request

The simplest example to get going is to submit an audio file for transcription. This is done by making a POST request with the audio file and the language model you want to use:

```
curl -X POST 'http://192.168.128.60:8082/v1/user/1/jobs/' \  
  -H 'Content-Type: multipart/form-data' \  
  -H 'Accept: application/json' \  
  -F 'audio=@audio.mp3;lang=en'
```

```
-F data_file=@example.wav \  
-F 'config={ "type": "transcription", "transcription_config": { "language": "en" } }'
```

In this example, the Batch Virtual Appliance in use has the IP address 192.168.128.60; you will change this to the IP address or hostname of your appliance. The `data_file` form field is used to submit the audio, and a `config` object is passed in with details of how to transcribe the file (in this case, using the global English language model).

Example Response

On successful submission of the job a 200 OK status will be returned by the appliance, along with JSON output showing the id of the job, and an indication in the `check_wait` property of how many seconds to wait before checking that the transcription is done:

The response headers returned will look like this:

```
{  
  "date": "Wed, 24 Jul 2019 16:35:25 GMT",  
  "server": "nginx/1.15.10",  
  "connection": "keep-alive",  
  "content-length": "1479",  
  "content-type": "application/json"  
}
```

And the body will comprise a JSON object like this:

```
{  
  "balance": 0,  
  "check_wait": 30,  
  "cost": 0,  
  "id": 111  
}
```

Note: The balance and cost properties are not used by the Batch Virtual Appliance. They will always return zero values.

Supplying a Job Configuration

The `config` object parameter is used to pass information about these features into the appliance. This is the recommended approach for passing information about the transcription job to the appliance. The simplest config object is where just the transcription language is specified, for example:

```
{  
  "type": "transcription",  
  "transcription_config": { "language": "en" }  
}
```

Example Configurations

Examples of configurations for specific features are shown in this section. See the [complete reference section](#) for more detail.

Speaker Diarization

There are two modes of diarization that can be used. The simplest, speaker diarization, aggregates all input channels into a single stream for processing, and picks out different speakers based on acoustic matching.

```
{  
  "type": "transcription",  
  "transcription_config": {  
    "diarization": "speaker",  
    "language": "en"  
  }  
}
```

Channel Diarization

The other diarization mode, channel diarization, processes multiple input channels or streams individually and applies a label to each (through use of the `channel_diarization_labels` list).

```
{ "type": "transcription",
  "transcription_config": {
    "diarization": "channel",
    "channel_diarization_labels": ["Agent", "Caller"],
    "language": "en"
  }
}
```

Speaker Change Detection

This feature allows changes in the speaker to be detected and then marked in the transcript. Typically it is used to make some changes in the user interface to indicate to the reader that someone else is talking. Detection of speaker change is done without detecting which segments were spoken by the same speaker. The config used to request speaker change detection looks like this:

```
{ "type": "transcription",
  "transcription_config": {
    "diarization": "speaker_change",
    "speaker_change_sensitivity": 0.8
  }
}
```

Note: Speaker change is only recorded as JSON V2 output, so make sure you use the `json-v2` format when you retrieve the transcript.

The `speaker_change_sensitivity` property, if used, must be a numeric value between 0 and 1. It indicates to the algorithm how sensitive to speaker change events you want to make it. A low value will mean that very few changes will be signalled (with higher possibility of false negatives), whilst a high value will mean you will see more changes in the output (with higher possibility of false positives). If this property is not specified, a default of 0.4 is used.

Speaker change elements in the `results` array appear like this:

```
{
  "type": "speaker_change",
  "start_time": 0.55,
  "end_time": 0.55,
  "alternatives": []
}
```

Note: Although there is an `alternatives` property in the speaker change element it is always empty, and can be ignored. The `start_time` and `end_time` properties are always identical, and provide the time when the change was detected.

A speaker change indicates where we think a different person has started talking. For example, if one person says "Hello James" and the other responds with "Hi", there should be a `speaker_change` element between "James" and "Hi", for example:

```
{
  "format": "2.4",
  "job": {
    ....
    "results": [
      {
        "start_time": 0.1,
```

```

    "end_time": 0.22,
    "type": "word",
    "alternatives": [
      {
        "confidence": 0.71,
        "content": "Hello",
        "language": "en",
        "speaker": "UU"
      }
    ]
  },
  {
    "start_time": 0.22,
    "end_time": 0.55,
    "type": "word",
    "alternatives": [
      {
        "confidence": 0.71,
        "content": "James",
        "language": "en",
        "speaker": "UU"
      }
    ]
  },
  {
    "start_time": 0.55,
    "end_time": 0.55,
    "type": "speaker_change",
    "alternatives": []
  },
  {
    "start_time": 0.56,
    "end_time": 0.61,
    "type": "word",
    "alternatives": [
      {
        "confidence": 0.71,
        "content": "Hi",
        "language": "en",
        "speaker": "UU"
      }
    ]
  }
]
}

```

Speaker Change Detection With Channel Diarization

Speaker change can be combined with channel diarization. It will process channels separately and indicate in the output both the channels and the speaker changes. For example, if a two-channel audio contains two people greeting each other (both recorded over the same channel), the config submitted with the audio can request the speaker change detection:

```

{"type": "transcription",
 "transcription_config": {
   "diarization": "channel_and_speaker_change",
   "speaker_change_sensitivity": 0.8
 }
}

```

```
}  
}
```

The output will have special elements in the `results` array between two words where a different person starts talking. For example, if one person says "Hello James" and the other responds with "Hi", there will a `speaker_change` json element between "James" and "Hi".

```
{  
  "format": "2.4",  
  "job": {  
    ....  
  },  
  "metadata": {  
    ....  
  },  
  "results": [  
    {  
      "channel": "channel_1",  
      "start_time": 0.1,  
      "end_time": 0.22,  
      "type": "word",  
      "alternatives": [  
        {  
          "confidence": 0.71,  
          "content": "Hello",  
          "language": "en",  
          "speaker": "UU"  
        }  
      ]  
    },  
    {  
      "channel": "channel_1",  
      "start_time": 0.22,  
      "end_time": 0.55,  
      "type": "word",  
      "alternatives": [  
        {  
          "confidence": 0.71,  
          "content": "James",  
          "language": "en",  
          "speaker": "UU"  
        }  
      ]  
    },  
    {  
      "channel": "channel_1",  
      "start_time": 0.55,  
      "end_time": 0.55,  
      "type": "speaker_change",  
      "alternatives": []  
    },  
    {  
      "channel": "channel_1",  
      "start_time": 0.56,  
      "end_time": 0.61,  
      "type": "word",  
      "alternatives": [  

```



```

    {
      "confidence": 0.71,
      "content": "Hi",
      "language": "en",
      "speaker": "UU"
    }
  ]
}
]
}

```

Custom Dictionary

The Custom Dictionary feature can be accessed through the `additional_vocab` property. This is a list of custom words or phrases that should be recognized. Custom Dictionary Sounds is an extension to this to allow alternative pronunciations to be specified in order to aid recognition, or provide for alternative transcriptions.

```

{ "type": "transcription",
  "transcription_config": {
    "language": "en",
    "additional_vocab": [
      { "content": "Ciarán", "sounds_like": [ "Kieran" ] },
      { "content": "FA", "sounds_like": [ "football association" ] },
      "speechmatics"
    ]
  }
}

```

You can specify up to 1000 words or phrases per job in your custom dictionary.

Note: For the `additional_vocab` property you can provide the list of alternate pronunciations using an array of `sounds_like` words or phrases. In the simple case where there is no difference of pronunciation, you can conflate the `content` and `sounds_like` fields into a single bareword or phrase (for example "speechmatics"). The `additional_vocab` property should be used for words that do not exist in the dictionary (in other words they are 'out of vocabulary'). The Custom Dictionary feature is designed to be used in environments where there is contextual information available (proper names, technical terms or other unusual words) that are likely to be out-of-vocabulary.

Output Locale

It is possible to specify the spelling rules to be used when generating the transcription, based on locale. The `output_locale` configuration setting is used for this. As an example, the following configuration uses the Global English (en) language pack with an output locale of British English (en-GB):

```

{ "type": "transcription",
  "transcription_config": {
    "language": "en",
    "output_locale": "en-GB"
  }
}

```

Currently, Global English is the only language pack that supports different output locales. The three locales that are available in this release are:

- British English (en-GB)
- US English (en-US)
- Australian English (en-AU)

If no locale is specified then the ASR engine will use whatever spelling it has learnt as part of our language model training (in other words it will be based on the training data used).

Advanced Punctuation

Some language models (English, French, German and Spanish currently) now support advanced punctuation. This uses machine learning techniques to add in more naturalistic punctuation to make the transcript more readable. As well as putting punctuation marks in more naturalistic positions in the output, additional punctuation marks such as commas (,) and exclamation and question marks (!, ?) will also appear.

There is no need to explicitly enable this in the job configuration; languages that support advanced punctuation will automatically output these marks. If you do not want to see these punctuation marks in the output, then you can explicitly control this through the `punctuation_overrides` settings in the config.json file, for example:

```
"transcription_config": {
  "language": "en",
  "punctuation_overrides": {
    "permitted_marks": [ ".", ", " ]
  }
}
```

Both plain text and JSON output supports punctuation. JSON output places punctuation marks in the results list marked with a `type` of `"punctuation"`. So you can also filter on the output if you want to modify or remove punctuation.

Sample JSON output (`json-v2` only) containing punctuation looks like this:

```
{
  "alternatives": [
    {
      "confidence": 1,
      "content": ", ",
      "language": "en",
      "speaker": "UU"
    }
  ],
  "attaches_to": "previous",
  "end_time": 10.15,
  "is_eos": false,
  "start_time": 10.15,
  "type": "punctuation"
}
```

If you specify the `punctuation_overrides` element for languages that do not yet support advanced punctuation then it is ignored.

Output Formats

The default job transcription output format is `json`, which is the same format used by the Speechmatics SaaS. The `json-v2` output format is also available with a richer set of information: you should use this format when you want to use newer features such as Channel Diarization. It is also possible to use a plain text transcription output if you do not need timing information.

Legacy API

If you want to re-use a client implementation that already uses the Speechmatics V1 SaaS API (app.speechmatics.com), then you can use the legacy request parameters to do this. The sample request shown above would look like this using the legacy API parameters:

```
curl -X POST 'http://192.168.128.60:8082/v1/user/1/jobs/' \
-H 'Content-Type: multipart/form-data' \
-H 'Accept: application/json' \
```

```
-F data_file=@example.wav \  
-F model=en
```

The following legacy parameters are available with the Batch Virtual Appliance:

Parameter	Description	Notes
model	Language model used to process the job.	Replaced by the <code>language</code> property in the config object.
notification	How you would like to be notified of your job finishing.	The <code>email</code> notification type is not supported by the Batch Virtual Appliance.
callback	If set, and notification is set to 'callback', the appliance will make a POST request to this URL when the job completes.	
callback_format	The format to be used by the appliance for the callback POST request.	Available formats are: 'txt', 'json' and 'json-v2'.
meta	Metadata about the job you would like to be able to view later.	
diarization	Controls whether speaker diarization is used.	Replaced by the <code>"diarization": "speaker"</code> , property in the config object.
diarisation	A synonym for 'diarization'.	See above.

It's recommended that you use the config object to pass the job configuration; the other methods of specifying job configuration will be deprecated at some point in the future. However, if you want to pass metadata with the job, or use a notification callback then you can do so using the legacy API parameters.

[NOTE] Passing metadata and using callback notifications

The next sections describe how to pass metadata and use notification callbacks using the legacy API in the event you need to use this functionality. These features will be covered by additional parameters in the config object in a future release.

Passing Metadata

You can use the `meta` parameter in the legacy API to associate metadata to the job, and use this for tracking the job through your workflow. For instance, you can use this to associate your own asset tag or job number, and retrieve it later on when you process the JSON transcript.

```
curl -X POST "http://${APPLIANCE_HOST}:8082/v1/user/1/jobs/" \  
-H 'Content-Type: multipart/form-data' \  
-H 'Accept: application/json' \  
-F data_file=@example.wav \  
-F 'meta'='asset-id=29309231123' \  
-F 'config={ "type": "transcription", "transcription_config": { "language": "en" } }'
```

You'll then see meta information when you query the job, or retrieve the (JSON) transcript:

```
{  
  "format": "2.4",  
  "job": {  
    "created_at": "Tue Nov 19 17:34:41 2019",  
    "duration": 383,  
    "id": 4,  
  }  
}
```

```

    "lang": "en",
    "meta": "asset-id=29309231123",
    "name": "en.mp3",
    "user_id": 1
  },
  "metadata": {
    "created_at": "2019-11-19T17:36:04.525Z",
    "transcription_config": {
      "language": "en"
    },
    "type": "transcription"
  },
},

```

Callbacks Usage

If you want to trigger a callback, so that you don't have to keep polling the jobs endpoint, you can do so by using the notification and callback parameters in the legacy API. This ensures that the Batch Appliance will send a POST to an HTTP server once the job is complete; typically, you would maintain a service running on that HTTP server that listens for these POST events and then performs some action to process the transcription (for example by writing it into a database, or copying the transcription to a file for further processing). Here is an example of how to setup a callback:

```

curl -X POST "http://${APPLIANCE_HOST}:8082/v1/user/1/jobs/" \
  -H 'Content-Type: multipart/form-data' \
  -H 'Accept: application/json' \
  -F data_file=@example.wav \
  -F model=en \
  -F notification=callback \
  -F callback=http://www.example.com/transcript_callback \
  -F callback_format=txt

```

The callback appends the job ID as a query string parameter with name `id`. As an example, if the job ID is 546, you'd see the following POST request:

```

POST /transcript_callback?id=546 HTTP/1.1
Host: www.example.com

```

The user agent is `Speechmatics-API/1.0`.

Note: For future compatibility we recommend that you use the config object, as new features will require it.

Speechmatics ASR REST API

Overview 3.3.0

The Speechmatics Automatic Speech Recognition REST API is used to submit ASR jobs and receive the results. The supported job types are transcription of audio files, and alignment of audio files with existing transcripts to add word or line timings.

Version information

Version: 1.1.0

Contact information

Contact Email: support@speechmatics.com

License information

Terms of service : <https://www.speechmatics.com/terms-and-conditions/>

URI scheme

BasePath : /v1 Schemes : HTTP

Paths

The base URL `http://${APPLIANCE_HOST}:8082/v1/user/1/` is used for REST Speech API requests. The user ID component can be any positive integer; by convention we use 1 for all requests.

/jobs

Requests without a job ID component are used to create a new job, or to return a list of all submitted jobs.

POST

Summary: Create a new job.

Parameters

Name	Located in	Description	Required	Schema
config	formData	JSON containing a <code>JobConfig</code> model indicating the type and parameters for the recognition job.	Yes	string
data_file	formData	The data file to be processed. Alternatively the data file can be fetched from a url specified in <code>JobConfig</code> .	No	file

Responses

Code	Description	Schema
201	OK	CreateJobResponse
400	Bad request	ErrorResponse
401	Unauthorized	ErrorResponse
403	Forbidden	ErrorResponse
500	Internal Server Error	ErrorResponse

GET

Summary: List all jobs.

Responses

Code	Description	Schema
200	OK	RetrieveJobsResponse
401	Unauthorized	ErrorResponse
500	Internal Server Error	ErrorResponse

/jobs/{jobid}

Requests with a job ID component are used to view the status, transcript or audio data for a job, or remove a given job from the system.

GET

Summary: Get job details, including progress and any error reports.

Parameters

Name	Located in	Description	Required	Schema
jobid	path	ID of the job.	Yes	string

Responses

Code	Description	Schema
200	OK	RetrieveJobResponse
401	Unauthorized	ErrorResponse
404	Not found	ErrorResponse
500	Internal Server Error	ErrorResponse

DELETE

Summary: Delete a job and remove all associated resources.

Parameters

Name	Located in	Description	Required	Schema
jobid	path	ID of the job to delete.	Yes	string

Responses

Code	Description	Schema
200	The job that was deleted.	DeleteJobResponse
401	Unauthorized	ErrorResponse
404	Not found	ErrorResponse
500	Internal Server Error	ErrorResponse

/jobs/{jobid}/data

GET

Summary: Get the data file used as input to a job.

Parameters

Name	Located in	Description	Required	Schema
jobid	path	ID of the job.	Yes	string

Responses

Code	Description	Schema
200	OK	file
401	Unauthorized	ErrorResponse
404	Not found	ErrorResponse
500	Internal Server Error	ErrorResponse

/jobs/{jobid}/transcript

GET

Summary: Get the transcript for a transcription job.

Parameters

Name	Located in	Description	Required	Schema
jobid	path	ID of the job.	Yes	string
format	query	The transcription format (by default the <code>json-v2</code> format is returned).	No	string

Responses

Code	Description	Schema
200	OK	RetrieveTranscriptResponse
401	Unauthorized	ErrorResponse
404	Not found	ErrorResponse
500	Internal Server Error	ErrorResponse

Models

ErrorResponse

Name	Type	Description	Required
code	integer	The HTTP status code.	Yes
error	string	The error message.	Yes
detail	string	The details of the error.	No

TranscriptionConfig

Name	Type	Description	Required
language	string	Language model to process the audio input, normally specified as an ISO language code	Yes
additional_vocab	[object]	List of custom words or phrases that should be recognized. Alternative pronunciations can be specified to aid recognition.	No
punctuation_overrides	[object]	Control punctuation settings.	No
diarization	string	Specify whether speaker or channel labels are added to the transcript. The default is <code>none</code> .	No
channel_diarization_labels	[string]	Transcript labels to use when using collating separate input channels.	No

For the diarization parameter, the following values are valid:

Value	Description
none	no speaker or channel labels are added.
speaker	speaker attribution is performed based on acoustic matching; all input channels are mixed into a single stream for processing.

channel	multiple input channels are processed individually and collated into a single transcript.
speaker_change	the output indicates when the speaker in the audio changes. No speaker attribution is performed. This is a faster method than speaker. The reported speaker changes may not agree with speaker.
channel_and_speaker_change	both channel and speaker_change are switched on. The speaker change is indicated if more than one speaker are recorded in one channel.

JobConfig

JSON object that contains various groups of job configuration parameters. Based on the value of `type`, a type-specific object such as `transcription_config` is required to be present to specify all configuration settings or parameters needed to process the job inputs as expected.

Name	Type	Description	Required
type	string		Yes
transcription_config	TranscriptionConfig		Yes

CreateJobResponse

In the job response you will see `balance` and `cost` values returned, but these are not used by the appliance; they are only maintained for backwards compatibility with the legacy V1 SaaS, and should be ignored by clients.

Name	Type	Description	Required
id	string	The unique ID assigned to the job. Keep a record of this for later retrieval of your completed job.	Yes
balance	integer	Not used	No
cost	integer	Not used	No

JobDetails

Name	Type	Description	Required
created_at	dateTime	The UTC date time the job was created.	Yes
data_name	string	Name of the data file submitted for job.	Yes
duration	integer	The file duration (in seconds). May be missing for fetch URL jobs.	No
id	string	The unique id assigned to the job.	Yes
status	string	The status of the job. * <code>running</code> - The job is actively running. * <code>done</code> - The job completed successfully. * <code>rejected</code> - The job was accepted at first, but later could not be processed by the transcriber. * <code>deleted</code> - The user deleted the job. * <code>expired</code> - The system deleted the job. Usually because the job was in the <code>done</code> state for a very long time.	Yes
config	JobConfig		Yes

RetrieveJobsResponse

Name	Type	Description	Required
jobs	[JobDetails]		Yes

RetrieveJobResponse

Name	Type	Description	Required
job	JobDetails		Yes

DeleteJobResponse

Name	Type	Description	Required
job	JobDetails		Yes

JobInfo

Summary information about an ASR job, to support identification and tracking.

Name	Type	Description	Required
created_at	dateTime	The UTC date time the job was created.	Yes
data_name	string	Name of data file submitted for job.	Yes
duration	integer	The data file audio duration (in seconds).	Yes
id	string	The unique id assigned to the job.	Yes

RecognitionMetadata

Summary information about the output from an ASR job, comprising the job type and configuration parameters used when generating the output.

Name	Type	Description	Required
created_at	dateTime	The UTC date time the transcription output was created.	Yes
type	string		Yes
transcription_config	TranscriptionConfig		No

RecognitionDisplay

Name	Type	Description	Required
direction	string		Yes

RecognitionAlternative

List of possible job output item values, ordered by likelihood.

Name	Type	Description	Required
content	string		Yes
confidence	float		Yes
language	string		Yes
display	RecognitionDisplay		No
speaker	string		No

RecognitionResult

An ASR job output item. The primary item types are `word` and `punctuation`. Other item types may be present, for example to provide semantic information of different forms.

Name	Type	Description	Required
channel	string		No
start_time	float		Yes
end_time	float		Yes
type	string	New types of items may appear without being requested; unrecognized item types can be ignored.	Yes
alternatives	[RecognitionAlternative]		Yes

RetrieveTranscriptResponse

Name	Type	Description	Required
format	string	Speechmatics JSON transcript format version number.	Yes
job	JobInfo		Yes
metadata	RecognitionMetadata		Yes
results	[RecognitionResult]		Yes

Error Codes

If the Batch Appliance is unable to process a request, then it will typically return one of the error codes listed below. We suggest that your API integration be created to robustly handle these errors.

4XX Errors

The 4xx class of status codes is intended for situations in which the request seems is in error.

All 4xx HTTP errors return in JSON format. Users of curl will see this displayed in their terminal, other interfaces may need a JSON parser. The JSON object contains two components: a return code and an error message. More details on these common errors for each endpoint are below. An example error object (from curl) would look like this:

```
curl "http://${APPLIANCE_HOST}:8082/v1.0/user/1/jobs/4087/transcript/"
{
  "detail": "The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.",
  "status": 404,
  "title": "Not Found",
  "type": "about:blank"
}
```

In this example, the URL was malformed (inclusion of trailing '/' character where none is required), giving a 404 status code.

Status Code	Status Message	Detailed Notes

400	Malformed request	Your request contained something unexpected - perhaps a string as a parameter where an integer was expected.
400	Missing data_file	Your request did not contain a data file in the data_file field.
400	No language selected	You did not supply a language code value in the model field.
400	Requested product not available	You requested an unsupported language.
403	Job rejected due to invalid audio	The audio file you submitted was in a format we do not support.
404	Job not found	We could not find a job with the specified id associated.
404	Job was rejected on submission	The job with this id was rejected on submission, probably due to an unsupported file format.
404	Job In Progress	The requested job is still being processed - please wait.
404	Output format Not Supported	You have requested the output in an unsupported format.
429	Too Many requests	You are trying to make too many POST requests in too short a period of time. Reduce your POST send rate.

5XX Errors

The 5xx class of status codes is intended for situations in which your request appears valid but nonetheless the server failed to fulfil an apparently valid request.

All 5xx HTTP errors return a HTML snippet.

Status Code	Status Message	Detailed Notes
500	Internal Server Error	Our service suffered an internal error. Please please contact us with as much information as possible for us to debug the problem.
502	Bad Gateway	The service is not active at present.
503	Service Temporarily Unavailable	Our service is temporarily overloaded and unable to process your request.

Formatting Common Entities

Overview

Entities are commonly recognisable classes of information that appear in languages, for example numbers and dates. Formatting these entities is commonly referred to as Inverse Text Normalisation (ITN). Speechmatics will output entities in a predictable, consistent written form, reducing post-processing work required aiming to make the transcript more readable.

The language pack will use these formatted entities by default in the transcription for all outputs (JSON, text and srt). Additional metadata about these entities can be requested via the API including the spoken words without formatting and the entity class that was used to format it.

Supported Languages

Entities are supported in the following languages:

- Cantonese
- Chinese Mandarin (Simplified and Traditional)
- English
- French
- German
- Hindi
- Italian
- Japanese
- Portuguese
- Russian
- Spanish

Using the `enable_entities` parameter

Speechmatics now includes an `enable_entities` parameter. This can be requested via the API. By default this is `false`.

Changing `enable_entities` to `true` will enable a richer set of metadata in the JSON output only. Customers can choose between the default written form, spoken form, or a mixture, for their own workflows.

The changes are as following:

- A new `type - entity` in the JSON output in addition to `word` and `punctuation`. For example: "1.99" would have a `type` of `entity` and a corresponding `entity_class` of `decimal`
- The `entity` will contain the formatted text in the `content` section, like other words and punctuation
 - The `content` can include spaces, non-breaking spaces, and symbols (e.g. \$/£/%)
- A new output element, `entity_class` has been introduced. This provides more detail about how the entity has been formatted. A full list of entity classes is provided below.
- The start and end time of the entity will span all the words that make up that entity
- The entity also contains two ways that the content will be output:
 - `spoken_form` - Each individual `word` within the entity, written out in words as it was spoken. Each individual word has its own start time, end time, and confidence score. For example: "one", "million", "dollars"
 - `written_form` - The same output as within `entity` content, with a `type` of `word` instead. If there are spaces in the content it will be split into individual words. For example: "\$1", "million"

Configuration example

Please see an example configuration file that would request entities:

```
{
  "type": "transcription",
  "transcription_config": {
    "language": "en",
    "enable_entities": true
  }
}
```

Different entity classes

The following `entity_classes` can be returned. Entity classes indicate how the numerals are formatted. In some cases, the choice of class can be contextual and the class may not be what was expected (for example "2001" may be a "cardinal" instead of "date"). The number of `entity_classes` may grow or shrink in the future.

N.B. Please note existing behaviour for English where numbers from zero to 10 (excluding where they are output as a decimal/money/percentage) are output as **words** is unchanged.

Entity Class	Formatting Behaviour	Spoken Word Form Example	Written Form Example
alphanum	A series of three or more alphanumerics, where an alphanumeric is a digit less than 10, a character or symbol	triple seven five four	77754
cardinal	Any number greater than ten is converted to numbers. Numbers ten or below remain as words. Includes negative numbers	nineteen	19
credit card	A long series of spoken digits less than 10 are converted to numbers. Support for common credit cards	one one one one two two two two three three three three four four four four	1111222233334444
date	Day, month and year, or a year on its own. Any words spoken in the date are maintained (including "the" and "of")	fifteenth of January twenty twenty two	15th of January 2022
decimal	A series of numbers divided by a separator	eighteen point one two	18.12
fraction	Small fractions are kept as words ("half"), complex fractions are converted to numbers separated by "/"	three sixteenths	3/16
money	Currency words are converted to symbols before or after the number (depending on the language)	twenty dollars	\$20
ordinal	Ordinals greater than 10 are output as numbers	forty second	42nd
percentage	Numbers with a per cent have the per cent converted to a % symbol	duecento percento	200%
span	A range expressed as "x to y" where x and y correspond to another entity class	one hundred to two hundred million pounds	100 to £200 million
time	Times are converted to numbers	eleven forty a m	11:40 a.m.
word	Entities that do not match a specific class	hundreds	hundreds

Output locale styling

Each language has a specific style applied to it for thousands, decimals and where the symbol is positioned for money or percentages.

For example

- English contains commas as separators for numbers above 9999 (example: "20,000"), the money symbol at the start (example: "\$10") and full stops for decimals (example: "10.5")
- German contains full stops as separators for numbers above 9999 (example: "20.000"), the money symbol comes after with a non-breaking space (example: "10 \$") and commas for decimals (example: "10,5")
- French contains non-breaking spaces as separators for numbers above 9999 (example: "20 000"), the money symbol comes after with a non-breaking space (example: "10 \$") and commas for decimals (example: "10,5")

Example output

Here is an example of a transcript requested with `enable_entities` set to true:

- An `entity` that is "17th of January 2022", including spaces
 - The start and end times span the entire entity
 - An `entity_class` of `date`
 - The `spoken_form` is split into the following individual words: "seventeenth", "of", "January", "twenty", "twenty", "two". Each word has its own start and end time
 - the `written_form` split into the following individual words: "17th", "of", "January", "2022". Each word has its own start and end time

Note:

- By default and when speaker diarization is enabled, `speaker` parameter is added per word within the entity, spoken and written form
- When channel diarization is enabled, `channel` parameter is only added on the `results` parent within the entity and not included in spoken and written form

```
"results": [
  {
    "alternatives": [
      {
        "confidence": 0.99,
        "content": "17th of January 2022",
        "language": "en",
        "speaker": "UU"
      }
    ],
    "end_time": 3.14,
    "entity_class": "date",
    "spoken_form": [
      {
        "alternatives": [
          {
            "confidence": 1.0,
            "content": "seventeenth",
            "language": "en",
            "speaker": "UU"
          }
        ],
        "end_time": 1.41,
        "start_time": 0.72,
        "type": "word"
      },
      {
        "alternatives": [
          {
            "confidence": 1.0,
            "content": "of",
            "language": "en",
            "speaker": "UU"
          }
        ],
        "end_time": 1.53,
        "start_time": 1.41,
        "type": "word"
      },
      {
        "alternatives": [
          {
```

```

        "confidence": 1.0,
        "content": "January",
        "language": "en",
        "speaker": "UU"
    }
],
"end_time": 2.04,
"start_time": 1.53,
"type": "word"
},
{
    "alternatives": [
        {
            "confidence": 1.0,
            "content": "twenty",
            "language": "en",
            "speaker": "UU"
        }
    ],
    "end_time": 2.46,
    "start_time": 2.04,
    "type": "word"
},
{
    "alternatives": [
        {
            "confidence": 1.0,
            "content": "twenty",
            "language": "en",
            "speaker": "UU"
        }
    ],
    "end_time": 2.79,
    "start_time": 2.46,
    "type": "word"
},
{
    "alternatives": [
        {
            "confidence": 0.97,
            "content": "two",
            "language": "en",
            "speaker": "UU"
        }
    ],
    "end_time": 3.14,
    "start_time": 2.79,
    "type": "word"
}
],
"start_time": 0.72,
"type": "entity",
"written_form": [
    {
        "alternatives": [
            {
                "confidence": 0.99,

```

```

        "content": "17th",
        "language": "en",
        "speaker": "UU"
      }
    ],
    "end_time": 1.33,
    "start_time": 0.72,
    "type": "word"
  },
  {
    "alternatives": [
      {
        "confidence": 0.99,
        "content": "of",
        "language": "en",
        "speaker": "UU"
      }
    ],
    "end_time": 1.93,
    "start_time": 1.33,
    "type": "word"
  },
  {
    "alternatives": [
      {
        "confidence": 0.99,
        "content": "January",
        "language": "en",
        "speaker": "UU"
      }
    ],
    "end_time": 2.54,
    "start_time": 1.93,
    "type": "word"
  },
  {
    "alternatives": [
      {
        "confidence": 0.99,
        "content": "2022",
        "language": "en",
        "speaker": "UU"
      }
    ],
    "end_time": 3.14,
    "start_time": 2.54,
    "type": "word"
  }
]
}
]

```

If `enable_entities` is set to `false`, the output is as below:

```

"results": [
  {
    "alternatives": [
      {

```



```
        "confidence": 0.99,
        "content": "17th",
        "language": "en",
        "speaker": "UU"
      }
    ],
    "end_time": 1.33,
    "start_time": 0.72,
    "type": "word"
  },
  {
    "alternatives": [
      {
        "confidence": 0.99,
        "content": "of",
        "language": "en",
        "speaker": "UU"
      }
    ],
    "end_time": 1.93,
    "start_time": 1.33,
    "type": "word"
  },
  {
    "alternatives": [
      {
        "confidence": 0.99,
        "content": "January",
        "language": "en",
        "speaker": "UU"
      }
    ],
    "end_time": 2.54,
    "start_time": 1.93,
    "type": "word"
  },
  {
    "alternatives": [
      {
        "confidence": 0.99,
        "content": "2022",
        "language": "en",
        "speaker": "UU"
      }
    ],
    "end_time": 3.14,
    "start_time": 2.54,
    "type": "word"
  }
]
}
```