

# Installation speech analysis software of EML Linux version 2.1



## Installation manual for system providers

11/18/2020

### Product line neo, version 6.x

The described functions can be used with the following ASC products:

EVOIPneo

EVOLUTIONneo / XXL / eco

EVOflex (country-specific)

Please note that you can always find the most up-to-date technical documentation and product updates in the partner area on our website at <http://www.asctechnologies.com>.

Copyright © 2019 ASC Technologies AG. All rights reserved.

Windows is a registered trademark of Microsoft Corporation. VMware® is a registered trademark of VMware, Inc. All other marks and names mentioned herein may be trademarks of their respective companies.



## Contents

<b>1</b>	<b>General information .....</b>	<b>4</b>
<b>2</b>	<b>Introduction .....</b>	<b>5</b>
<b>3</b>	<b>Installation .....</b>	<b>6</b>
3.1	EML Transcription Server Version Linux 2.1.pdf.....	7

## 1 General information

In the context of this document ASC represents ASC Technologies AG, its subsidiaries, branch offices, and distributors. An up-to-date overview of the aforementioned entities can be found at <https://www.asctechnologies.com>

ASC assumes no guarantee for the actuality, correctness, integrity or quality of the information provided in the manuals.

ASC regularly checks the content of the released manuals for consistency with the described hardware and software. Nevertheless, deviations cannot be excluded. Necessary revisions are included in subsequent editions.

Some aspects of the ASC technology are described in general terms to protect the ownership and the confidential information or trade secrets of ASC.

The software programs and the manuals of ASC are protected by copyright law. All rights on the manuals are reserved including the rights of reproduction and multiplication of any kind, be it photo mechanical, typographical or on digital data media. This also applies to translations. Copying the manuals, completely or in parts, is only allowed with written authorization of ASC.

Representative, if not defined otherwise, is the technical status at the time of the delivery of the software, the devices and the manuals of ASC. Technical changes without specified announcements are reserved. Previous manuals lose their validity.

The general conditions of sales and delivery of ASC in their latest version apply.

## 2 Introduction

This manual describes the installation of the audio analysis software EML Transcription Server of the company EML for Linux operating systems to be used with the *neo* recording system.

EML Transcription Server allows transcribing audio into text or detecting keywords which can then be searched for.

Audio analysis jobs are configured and administrated in the Audio Analysis module of the application INSPIRATION*neo*.

### Transcription

Transcription is based on the LVCSR technology (large vocabulary continuous speech recognition).

The transcription converts audio recordings into text which is then available for analysis. To be able to recognize all words, dictionaries are uploaded to look up the audio data in. As each separate word must be recognized and converted into text, this approach initially requires more time than keyword spotting. But on the other hand, transcription makes the entire audio recording available as text so that any word can be found via full-text search.

The result (text) is entered in the INSPIRATION*neo* database via an XML interface.

An advantage of full-text searches is that the search terms can be displayed in context. This excludes misunderstandings, e. g. in the event of ambiguities. The texts are available for additional analyses and can be transferred to other systems to do so.

Transcription is a real-time process (on a core of a default server CPU). The transcription quality depends on the number of channel licenses.

There is no upper limit for the length of the audio, however it must have a minimum length of 200 milliseconds to be processed.

---

ASC recommends using transcription for stereo calls.

Transcription for mono calls is possible but not advisable.



As all call participants are merged in one track in mono calls, the results would be associated with one participant. The audio analysis engine is not able to properly separate cross-talk occurring when participants speak at the same time; as a result, the quality of the transcription decreases.

---

### Keyword spotting

By means of keyword spotting, you can filter for certain topics or categorize the sessions. To this end, you compile all expressions and phrases (keywords) which describe a topic in an analysis list. The defined keywords will then be searched automatically in the sessions.

Since this approach is limited to detecting individual words and phrases, sessions can be searched quickly. Since you have to define the expressions which are supposed to be searched for in advance, this approach especially serves to identify already known topics which frequently come up again.



For further information about the configuration of the audio analysis software EML Transcription Server refer to the administration manual *Configuration speech analysis*.



Additional information about creating audio analysis jobs and about how to use them can be found in the user manual *Usage Audio Analysis module*.

### 3 Installation

Install the speech analysis software according to the manufacturer's instructions following in this chapter.



Please note, that the speech analysis software must be installed on its own server with a separate PostgreSQL database.



Please note that for new installations and updates of language packages a decoder version  $\geq 1.4.3$  and an EML Transcription Server version  $\geq 1.3.2$  is required.



For further information about the update of EML speech analysis software refer to the installation manual for system providers *Software updates*.

#### See also

 [EML Transcription Server Version Linux 2.1.pdf](#) [▶ 7]



# **EML Transcription Server**

## **Installation Instructions (Linux)**

**Version 2.1**

**January 2019**

---

## Document-History

---

### Revisions of this document

Revision-number	Revision-date	Changes from previous revisions	Marked changes
1.0	2009-03-18	Initial version	N
1.1	2010-08-13	Updates for version 1.1 of the decoder, changed Naming	N
1.2	2011-05-31	Updates for Geronimo 2.2.1	N
1.3	2011-09-30	New Functions added, job expiration, post processing	N
1.4	2012-12-21		
1.5	2014-05-08	Update to new Transcription Server	N
1.6	2015-03-05	Minor Revision	N
1.7	2015-08-24	Update to new docker based installation of Transcription Server	N
1.7.1	2015-10-15	Add disk space requirement	N
1.8	2016-08-15	Update requirements	N
1.9	2017-04-10	Update requirements for KWS	N
1.10	2017-06-08	Added Offline Installation	N
1.11	2017-07-10	Added G2P and Troubleshooting	N
1.12	2017-08-22	Updated Transcription Server requirements with G2P	N
1.13	2018-02-09	Updated Workflow/Architecture pictures	N
1.14	2018-03-19	Minor updates	N
2.0	2018-05-14	Add update notes	N
2.1	2019-01-14	Updated Decoder installation	N





### Contents

1.	Overview.....	5
1.1	Basic Architecture .....	5
1.2	Streaming Setup .....	6
1.3	Ports.....	6
1.4	Workflow .....	7
1.5	Best-Practice Setup .....	7
2.	Transcription Server .....	8
2.1	System Requirements.....	8
2.1.1	Operating System .....	8
2.1.2	Software Dependencies.....	8
2.1.3	Hardware .....	8
2.1.4	Firewall.....	8
2.2	Installation .....	9
2.2.1	Offline installation .....	9
2.2.2	Additional parameter.....	9
2.3	Update.....	9
2.3.1	Offline update.....	10
2.4	Verify Installation/Update .....	10
2.5	Configuration.....	11
2.5.1	Update Configuration .....	11
2.5.2	Populate Model and Project databases .....	11
2.6	Verify Configuration.....	13
2.7	Post-Installation Notes .....	13
3.	Decoder .....	14
3.1	System Requirements.....	14
3.1.1	Operating System .....	14
3.1.2	Software Dependencies.....	14
3.1.3	Hardware .....	14
3.1.4	Firewall.....	14
3.2	Installation .....	15
3.3	Update.....	15
3.4	Verify Installation.....	15
3.5	Configuration.....	15
3.6	Verify Configuration.....	16
3.6.1	Run decoder as a service on Ubuntu .....	16
3.7	Alternative installation with docker.....	17



## EML Transcription Server - Installation Instructions

---

4.	Streaming Service .....	18
4.1	Verify Installation.....	18
4.2	Configuration.....	18
4.3	Verify Configuration.....	18
5.	G2P .....	19
5.1	Verify Configuration.....	19
6.	Troubleshooting.....	20
6.1	EML Decoder Log .....	20
6.2	EML Transcription Server Log .....	20
6.2.1	Increase log level.....	20

# 1. Overview

The EML Transcription Server consists of the following components:

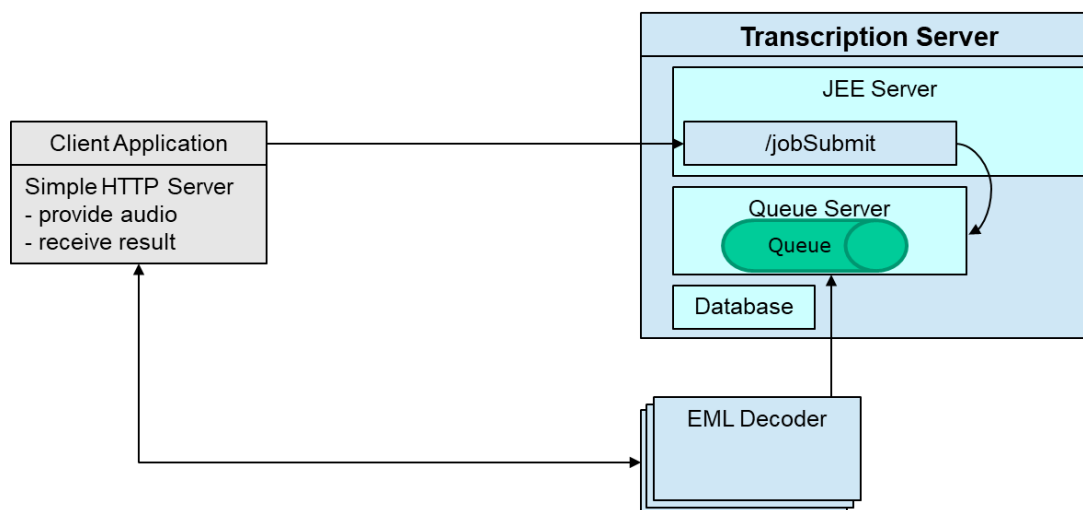
- *Decoder*: The decoder transcribes audio files from a queue server using a model.
- *Transcription Server*: The transcription server puts jobs into the queue the decoder listens to. It also provides a monitoring user interface for the queue server and manages the available models. Clients need to supply an audio URL from which the decoder can download the audio file and a second URL to which the decoder can send the results.

In order to be able to update a model independently of a specific client without the need to change the client, the transcription server keeps track of so called *projects*. A project is simply a mapping from a “project name” to a specific model. The client only provides the project name in the request to the transcription server, and the transcription server will translate it into a proper model name which can be used by the decoder.

- *Additional: Streaming Service*: The streaming server is used to establish a streaming transport of the audio and the result.

The streaming service introduces to notion of a language identifier (e.g. “en”) which is mapped in the streaming service to a project name of the transcription service, thus allowing changing the model or project independently of the clients (i.e. without needing to update the clients). Besides a streaming interface, the streaming service offers an easy to use HTTP interface to submit jobs (it allows to use a simple HTTP POST request to start transcription of an audio file).

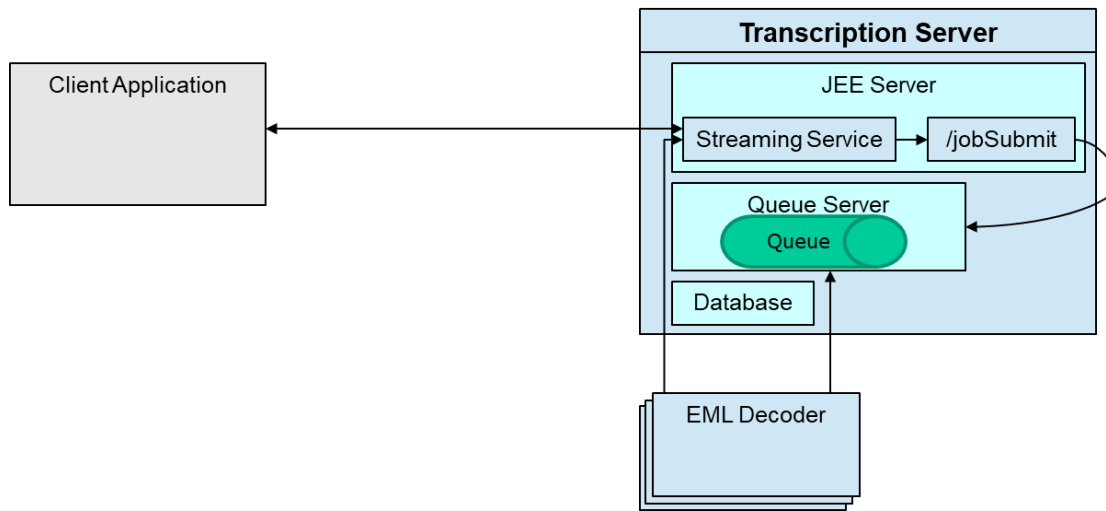
## 1.1 Basic Architecture



The basic architecture consists of the Transcription Server, the Decoder and a Client application. The Client sends a job description document to the transcription server. The job is put into a queue, where it is picked up by a decoder. The decoder requests the audio from the client, transcribes it and sends the result to the client. In the above picture, Transcription Server and Decoder boxes represents a different host, although it is possible to all of the components to run on one machine (given that the system requirements are met).

There can also be run multiple decoders which connect to the same queue to scale vertically. Multiple decoders can be run on one or different machines

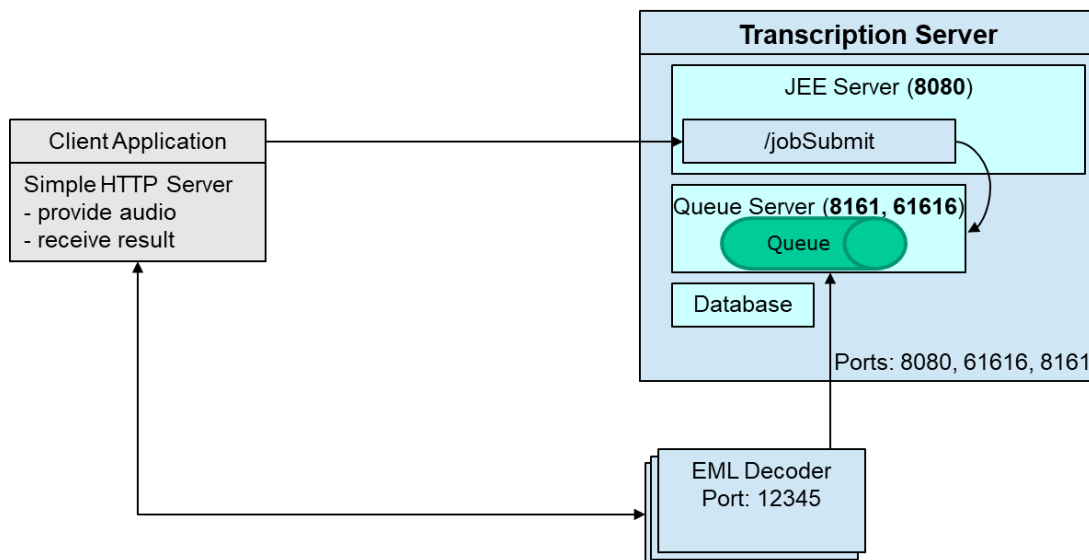
### 1.2 Streaming Setup



The client in this case holds a bi-directional connection to the streaming service, which in turn will post a corresponding job to the transcription server and will receive the results from the decoder. It can process the results and transmit only the requested information to the client, minimizing the amount of transferred data.

Since the streaming service is just an addition to the basic setup, both setups can be used in parallel.

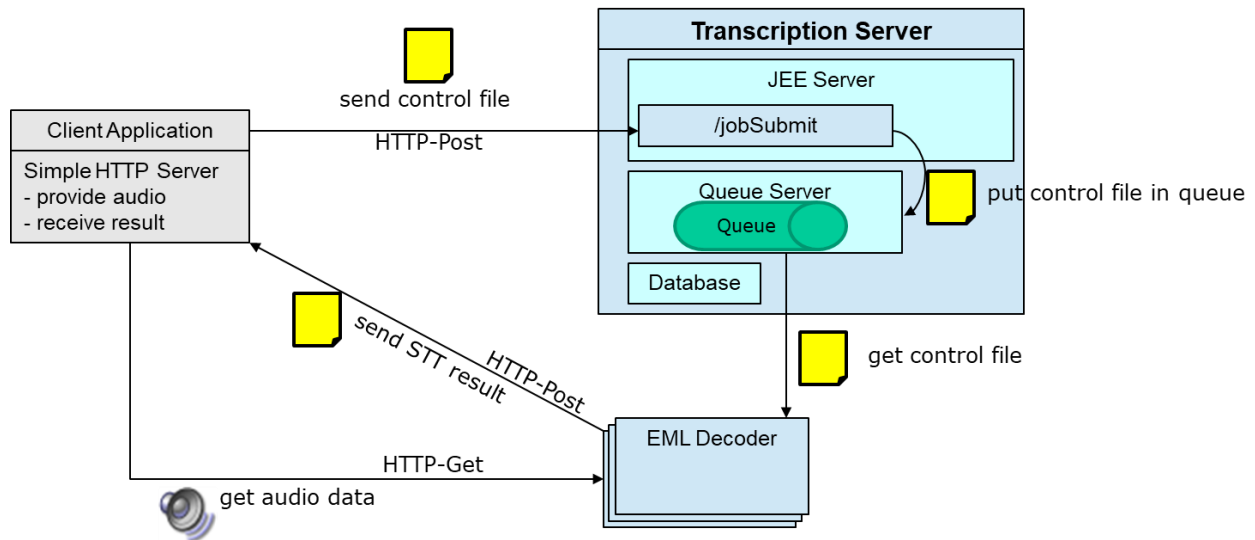
### 1.3 Ports



The Transcription Server is starting the services on ports: 8080/TCP, 61616/TCP and 8161/TCP. The Transcription Decoder will listen to port 12345/TCP when management console is activated (active by default).

Note: All ports used in Transcription Server and Decoder are configurable!

### 1.4 Workflow



An application that uses EML Speech Transcription Server sends the control-file to Transcription Server with an HTTP POST request. The control file is fetched by one of decoders which are subscribed to the queue. In the control file is a pointer to an URI where the audio data can be downloaded. The decoding is then done on a local copy of the audio data.

The control-file is enriched with the result of the speech-to-text processing and the resulting file is sent back to the application using a call-back URI which is provided in the control-file. After the successful processing and the transfer of the result the processing is acknowledged to the queue.

### 1.5 Best-Practice Setup

For large scale setups we recommend installing the EML Transcription Server and the Queuing Server on one host and out the decoders on separate hosts. For smaller setups, where only a few channels are needed the EML Transcription Server, the Queuing Server and the decoders may run on the same host.

---

## 2. Transcription Server

The EML Transcription Server is the service responsible for the automatic transcription of audio. It is a highly scalable JEE Application running in a JEE application server (e.g. JBoss Wildfly) together with a messaging server (e.g. Apache ActiveMQ). It is responsible for the automatic model deployment to the registered EML Transcription Decoders as well as the logging of the job status.

As part of the EML Transcription Server the EML Transcription Server Monitor is deployed. This is a browser based graphical management tool.

---

### 2.1 System Requirements

Note: On Software Dependencies and Operating Systems we expect 64bit versions.

The *EML Transcription Server* comes with the following system requirements:

#### 2.1.1 Operating System

- **Ubuntu 18.04**  
No special requirements
- **Ubuntu 16.04**  
No special requirements
- **Ubuntu 14.04**  
No special requirements

#### 2.1.2 Software Dependencies

- **Java 8** (Oracle JRE or OpenJDK JRE)

#### 2.1.3 Hardware

- **CPU:** No special requirements
- **RAM:** 8-16GB
  - 8GB without using G2P
  - up to 16GB when using G2P (necessary for dynamic lexicon update and dynamic KWS entries)
- **Disk:** 10GB + 10 GB per language

#### 2.1.4 Firewall

- **Mandatory open:**
  - 8080/TCP (JEE Application server e.g. Wildfly)
  - 61616/TCP (ActiveMQ OpenWire protocol)
  - 8161/TCP (ActiveMQ admin and REST interface)

## 2.2 Installation

The installation package contains a fully automatic installer of the EML Transcription Server. It will automatically download and install the JEE application server, a messaging server and a database system as docker<sup>1</sup> images via the EML Docker Hub.

The script will automatically install the required system packages (docker, curl, pwgen, unzip) thus you have to run the installer as root.

During installation dialog you will be asked if the printed "HOSTNAME" is correct. You can specify the hostname by "export HOSTNAME=<myHost>" before executing the installer. Please ensure that the name or ip is valid and accessible from other hosts.

To start installation issue

```
unzip <eml-transcription-server-[version].zip> && cd docker && ./install.sh
```

### 2.2.1 Offline installation

If the server cannot access the Docker Repositories you can download the images before and put them on the destination server and issue:

```
./install.sh OFFLINE /tmp/dockeri_server.tar /tmp/dockeri_amq.tar  
/tmp/dockeri_postgres.tar /tmp/dockeri_g2p.tar
```

**NOTE:** The order of the image.tar files matters!

In case the installation failed, you need to clean up the data before executing the installation again. Delete Docker container: "docker rm -f <container\_name>" and also the DATADIR (default: /opt/eml)

### 2.2.2 Additional parameter

There are a few options to modify the installation process:

#### **DEBUG**

When enabled it will show more information during installation on console. (default: 0) 1 = enabled

#### **DATADIR**

This parameter will define the installation target directory. (default: "/opt/eml")

#### **INSTALL\_G2P**

Whether to install the G2P Service or not (default: 1). 0 = do not install

To make use of the additional parameter just put them in front of your command like:

```
DEBUG=1 DATADIR=/opt/other/eml ./install.sh
```

---

## 2.3 Update

Since version 2.7 of eml-transcription-server.zip the package contains a script to easily update your installation by issue

```
./update.sh
```

During update process the old docker container will be removed, the new image will be downloaded and a new docker container created.

---

<sup>1</sup> <https://www.docker.com/>



### 2.3.1 Offline update

If your server cannot access the Docker Repositories you can download the images before and put them on the destination server and issue:

```
./update.sh OFFLINE /tmp/dockeri_server.tar /tmp/(optional)dockeri_g2p.tar
```

NOTE: The order of the image.tar files matters!

---

## 2.4 Verify Installation/Update

You should check the EMLConfiguration table in the database. After deployment, it should already be populated with default values. You can use the REST service to check this. Use the following curl (or any other tool to perform HTTP requests) to query the REST service to see whether it is up and running.

```
curl -u user:password http://x.x.x.x:8080/eml-stt/config/sttweb.current-context
```

This should return “eml-stt”. User and password are the same as for the Transcription Server.

Note: The credentials to Transcription Server and the other applications were printed during installation and were also saved in the working directory in *install.notes*

You can connect to the Active MQ admin interface (on <http://x.x.x.x:8161/> with admin/admin by default). You should see at least one queue created (eml-transcribe-stats) with one listener. The queues are listed in “Manage ActiveMQ broker” -> “Queues”



## 2.5 Configuration

### 2.5.1 Update Configuration

You can use curl and the REST interface to change configuration easily. The command to set a specific configuration value is:

```
curl -X POST -u user:password http://x.x.x.x:8080/eml-stt/config/<key>?value=<value>
```

You can also change configuration with the Transcription Server Monitor (on <http://x.x.x.x:8080/TranscriptionWebMonitor/> by default) on “configuration” panel.

The installer will create an initial configuration with default values:

<i>Key</i>	<i>Description</i>	<i>Default</i>
Host		<i>&lt;hostname&gt;</i>
	The hostname or IP-address of the server the Transcription Server is running on and could be reached from the Transcription Decoders	
Port		<i>8080</i>
	The port on which the Application server is running	
repository.base		<i>/opt/modelRepository</i>
	The directory on the server in which uploaded models will be stored.	
sttweb.default-queue		<i>eml-transcribe</i>
	The default queue to which jobs should be posted.	
monitor.archiveFolder		<i>/opt/reportArchive</i>
	The directory on the server in which generated reports will be stored.	
sttweb.queue-server		<i>default</i>
	The queue server identifier to use to post jobs. This will be appropriately set when using the <code>add-queue-server.sh</code> script.	
sttweb.current-context		<i>/eml-stt</i>
	The web context in which the sttweb component is running. This will be automatically set. Do not change this value.	
monitor.current-context		<i>/TranscriptionWebMonitor</i>
	The web context in which the EML Transcription Server Monitor component is running. This will be automatically set. Do not change this value.	

### 2.5.2 Populate Model and Project databases

You need to populate the model and project database with entries in order to post jobs.

#### 2.5.2.1 Deploy models via script

There is a script part of the installation image to allow you uploading of acoustic model, language model and image files and creating a project conveniently located in `eml-tools`.

```
./upload-models.sh
```

uploads the model files to the EML Transcription Server and create/update a project entry which can be directly used.

### 2.5.2.2 Deploy models via GUI

To upload and create entries manually, use the Transcription Server Monitor (on <http://x.x.x.x:8080/TranscriptionWebMonitor/> by default):

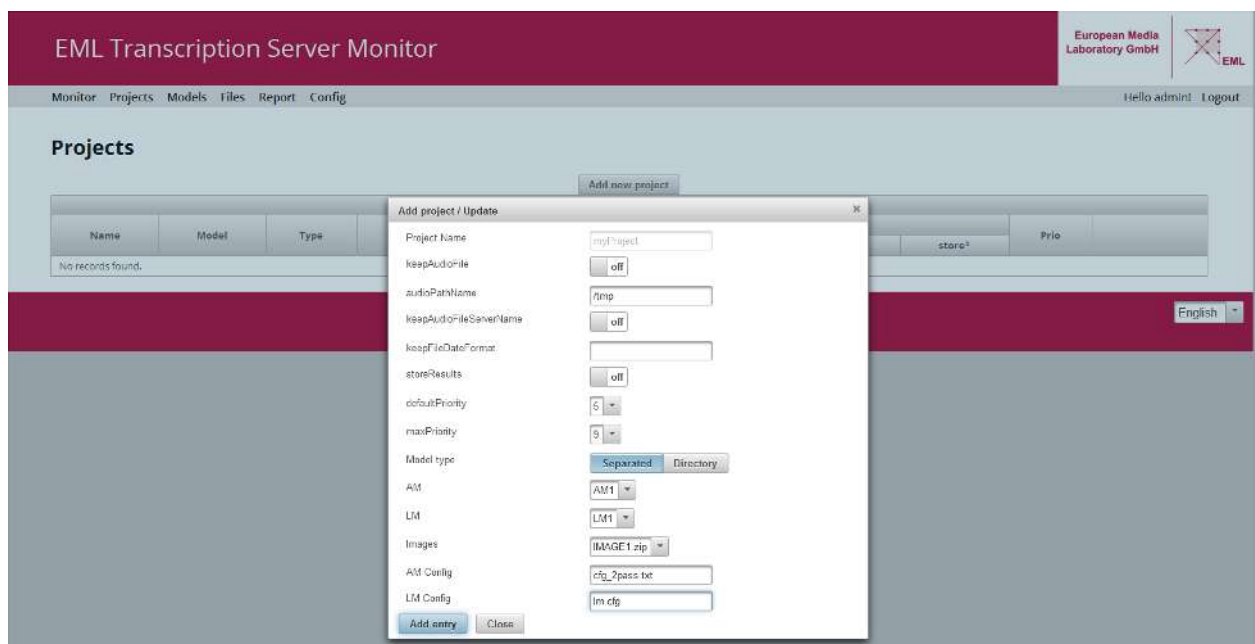
1. Go to the “Files” panel and upload your models files. Make sure to select the right type in the drop down menu. Alternatively, if the model files are already present at the repository location set up above, you can simply import them.
2. Go to “Models”. Select the correct model type (“Directory” for complete models, “Acoustic” for Acoustic Models and “Language” for Language Models). Click the “Add new Model” button and fill out the form properly.

Note that all of these operations also can be executed using the REST interface.

### 2.5.2.3 Create Project

Open Transcription Server Monitor and go to “Projects” in top menu navigation. Click the “Add new project” button. Select a project name and afterwards select the created model entries. There is a switch button “Separated | Directory” to get a list of either complete models (directory) or separated models (acoustic, language and images).

In the separated view there are two additional parameters: AM config and LM config. The AM config is usually also listed in the filename of the images archive (e.g. if the image file is named IMAGES-bcn06.hbn.b\_BCN\_DeDE\_10\_161108\_142740--2\_cfg\_stream.txt--1705.zip the AM config that has to be referenced is `cfg_stream.txt`). The LM config is usually named `lm.cfg`.



**Add new project dialog with separated model**

## 2.6 Verify Configuration

After you have created the project entry, you can submit your first job to the Transcription Server. This job should appear in the proper queue (see the EML Transcription Server Monitor). Since no decoder is running yet, of course it will not get dispatched.

In order to post a job, the easiest way is to use the EML command-line tool `queue-files.sh` which is located in the `eml-tools` directory in the distribution package.

```
./queue-files.sh -if y.y.y.y -host x.x.x.x -queue <queue> -project <project> -  
files <audio.wav>
```

Where `y.y.y.y` is the IP address of the computer on which the script runs and `x.x.x.x` is the host on which the EML Transcription Server runs.

Since no decoder is yet running, the script will block indefinitely and you should abort it by pressing `Ctrl+C`.

If the job appears in the queue the installation is complete and working. Purge the queue in the EML Transcription Server Monitor before proceeding to set up the decoders.

---

## 2.7 Post-Installation Notes

You can access the EML Transcription Server Monitor at `http://x.x.x.x:8080/TranscriptionWebMonitor/`

This installation uses docker. To monitor the logs of the services use `docker logs <container>`. To see all running containers use `docker ps`. It is strongly advised to find out more about docker at <https://docs.docker.com>

The installer has started 3 docker containers. Those will be restarted automatically upon reboot or any error.

If you need to recreate the containers, the installer has created 3 scripts in the current directory to do that for you. You can also find them in the file `install.notes`.

## 3. Decoder

---

### 3.1 System Requirements

---

The Decoder internally consists of several decoding threads, each acting as a full decoder instance. Since each thread loads a model into memory, the requirements depend on how many threads are configured to run.

Note: On Software Dependencies and Operating Systems we expect 64bit versions.

#### 3.1.1 Operating System

- **Ubuntu Linux 18.04**

```
apt-get install libgomp1 libsndfile1 libxml2
```

- **Ubuntu Linux 16.04**

```
apt-get install libgomp1 libsndfile1 libxml2
```

- **Ubuntu Linux 14.04**

```
apt-get install libgomp1 libsndfile1 libxml2
```

- **CentOS 7**

```
yum install libgomp libsndfile
```

- **CentOS 6**

```
yum install libgomp libsndfile
```

#### 3.1.2 Software Dependencies

- **Java 8** (Oracle JRE or OpenJDK JRE)
- Libraries (see 3.1.1)
- Transcription Server installed and configured

#### 3.1.3 Hardware

- **CPU:** One physical core per decoding thread
- **For Transcription:**
  - **RAM:** Depending on model, typically 4-8 GB per decoding thread
  - **Disk:** 20GB + 10 GB per language
- **For Transcription with dynamic Lexicon:**
  - **RAM:** Depending on model, typically 5-10 GB per decoding thread
  - **Disk:** 20GB + 10 GB per language
- **For KWS**
  - **RAM:** 512MB per decoding thread
  - **Disk:** 5GB per language

#### 3.1.4 Firewall

- *12345/TCP (default management port used by EML Transcription Server Monitor)*

## 3.2 Installation

Just unpack the received package as the user the decoder should eventually run as. The directory where it is put generally does not matter, by default we assume it is in `/opt/speech/emlDecoder`

```
mkdir -p /opt/speech/emlDecoder && tar -xjf <eml-decoder.tar.bz2> -C
/opt/speech/emlDecoder
```

---

## 3.3 Update

Stop the current Decoder instance and delete the `/opt/speech/emlDecoder/lib64/` folder. Then just unpack the new package into the previously installed directory, by default we assume it is in `/opt/speech/emlDecoder`

```
tar -xjf <eml-decoder.tar.bz2> -C /opt/speech/emlDecoder
```

---

## 3.4 Verify Installation

Check that the directory contains all the files from the archive, most prominently check that `emlDecoder.sh`, `emltd`, and the `lib64` directory are present.

To check whether all software dependencies are met, please try to run the `emltd` program by executing

```
./emltd -p 18081 -n test
```

Note that 12345 denotes a port number. Simply choose a port number not currently in use by the system. You should see output like:

```
Starting EML Transcription Daemon
listening port: 18081 name: test
```

If the program does not crash, all dependencies are met. If the program complains about missing libraries use the following command to determine which libraries cannot be found and use the package manager to install them.

```
ldd emltd
```

Use `Ctrl+C` to stop the program.

---

## 3.5 Configuration

To configure the decoder, please rename “`decoder.properties.template`” to “`decoder.properties`” and make the changes required for your environment. Having a proper configuration is crucial. The default configuration assumes the EML Transcription Server has been installed on the same host as the decoder. If this is not the case, change the `decoderServer` parameter accordingly. The “`defaultQueue`” parameter sets the name of queue to which to connect. Note that the queue will be created automatically if it is not existing in the queue server.

The “`numInstances`” parameter controls the number of decoding threads. The “`modelRepositoryPath`” parameter denotes the directory where the decoder will unpack the model files downloaded from the transcription server. Make sure the location exists and is writeable by the user the decoder is running as. Uncomment the “`managementConsole`” parameters to enable a REST service to query the state of the decoders and visualization in the EML Transcription Server Monitor.

To configure the logging system, rename “`log4j.properties.template`” to “`log4j.properties`”. The default settings enable most of the debug logs. Have a look at the `log4j` project on what this configuration file may contain.

Finally rename “`env.sh.template`” to “`env.sh`” which may define some bash variables in order to properly start up the decoders. Possible variables and their implications:

## EML Transcription Server - Installation Instructions

---

- **JAVA\_HOME:** Sets the path to the JRE. Needs only to be set if JAVA\_HOME is not set by the system or another JRE then the default one shall be used.
- **SUFFIX:** If you installed your decoder to `/opt/speech/emlDecoder-test` the SUFFIX needs to be set to `-test`. The log and pid files will also be suffixed by this variable. If you have extracted the decoder to `/opt/speech/emlDecoder` you do not need to set this variable.
- **EML\_HOME:** If you have installed the decoder not to `/opt/speech/emlDecoder${SUFFIX}` this variable needs to point to the directory in which the decoder is installed.
- **PIDFILE:** By default the pid file will be `/opt/speech/log/emlDecoder${SUFFIX}.pid`
- **API\_LOG\_DIR, DATA\_DIR:** These specify directories where the decoder will create additional log files and some temporary data files. By default, the locations of these are located in `/tmp`. If you need to run several decoder instances with different users on the same machine, you might want to change these variables. Also running the decoder not as root might require these.

Make sure to “export” the variables.

---

### 3.6 Verify Configuration

After setting up the needed configuration files, you can try starting the decoder by issuing `./emlDecoder.sh start` in the command line. You can then have a look at the log file (configured in `log4j.properties`, by default `/opt/speech/log/emlDecoder.log`) in order to check whether it starts.

For each instance you should see eventually a line saying “Waiting for a new message now...” You can use the EML Transcription Server Monitor on the EML Transcription Server to check whether the decoder is connected to the right queue. You should see for each decoding thread 1 consumer on the queue. This means the queue configuration on both ends is correct and working.

To check whether decoding is working, make sure that you have a project, which references to a model for which you have uploaded the files (AM, LM, Images) to the transcription server. If this is the case, submit a job for that project to the queue either by using your own client application or by using the EML command line tool mentioned in 2.6. You should see the job appearing in EML Transcription Server Monitor. The job will disappear from the queue *after* the job has been completely processed by the decoder. In the decoder’s log file, you should see several things happen: First of all, the message that a job was received, next details of the job will be printed (e.g. which model to use). After that, it is checked whether the model already exists in the model repository path, and if not it will be downloaded and extracted. If that is done the following workflow will take place:

1. **Initialization of Engines:** For each decoding there might be multiple engine processes that need to be spawned and initialized with the model. For future decodings with the same model, the already initialized engines will be re-used. So this is a one-time setup time.
2. **Download of the audio:** The audio is downloaded from the given URL.
3. **Decoding**
4. **Sending the result(s):** The results are posted to the given call back URL.

Please note, that the first few decodings will usually take longer (due to the initialization steps) then subsequent ones. Since the initialization time is a one-time setup, it is typically not included in real-time factor calculations.

#### 3.6.1 Run decoder as a service on Ubuntu

Since Ubuntu 14.10 systemd is a new init-system. Instead of running the Decoder by using the `./emlDecoder.sh`-Script you can also run it as a service:

1. `sudo cp /opt/speech/emlDecoder/eml-decoder.service /etc/systemd/system/`
2. `sudo systemctl daemon-reload`
3. `sudo systemctl enable eml-decoder.service`



4. `sudo systemctl start eml-decoder`
5. `sudo systemctl status eml-decoder`

Start/Stop log information could be printed by

```
journalctl -n 1000 -u eml-decoder.service
```

---

### 3.7 Alternative installation with docker

We recommend to install the Decoder with the native installation package mentioned in 3.2 but if requested EML also provides a docker-based installation.

The docker-based installation package contains a fully automatic installer of the EML Transcription Decoder. It will automatically download and install the Decoder as docker image via the EML Docker Hub.

The script will automatically install the required system packages (docker, curl) thus you have to run the installer as root.

To start installation issue

```
unzip <eml-transcription-decoder-[version].zip> && cd docker && ./install.sh
```

---

## 4. Streaming Service

As the streaming service is part of the EML Transcription Server installation package no further installation is required. You can skip this part if it's not required to use streaming API.

---

### 4.1 Verify Installation

Point your browser to `http://x.x.x.x:8080/webSocket` and try to login with the Transcription Server credentials.

---

### 4.2 Configuration

After installation the Streaming Service needs a bit configuration. By default there is a configuration file stored on disk.

```
/opt/eml/websocket-config/configuration.properties
```

As the first setting, check that you setup the URL and credentials for the transcription server correctly.

After finishing the configuration open /webSocket UI and create a new authentication key. After that, create a new language identifier. See the manual for the streaming service if you run into problems with that. Make sure you have setup the transcription server configuration items correctly before creating a language entry. Also make sure you have created at least one project in the transcription server. When creating a language id, you have to assign a project to it. Therefore, the streaming service requests a list of projects from the transcription server.

---

### 4.3 Verify Configuration

To verify that the configuration is working, use the REST interface with a WAV audio file with the proper sampling rate:

```
curl -H "Content-Type: application/octet-stream" --data-binary @<file>
"http://x.x.x.x:port/webSocket/rest/batch/<language-identifier>/<authentication-key>/?encoding=wav"
```

The result of this command should be the following actions:

- In the Transcription Server docker log file you see a new request (look for RESTART messages)
- When the file has been transmitted, you will see a "Posting job to ..." message. If there is no exception, the message has been posted to the transcription server. The processing on the transcription server and on the decoder eventually starts (check those logs)
- In the docker log you will see more messages when the results arrive from the decoder.
- It is quickly processed and should be printed as result of the curl call on your command line.

After this step was successful, use our command line tools, to stream the audio file using the Websocket connection. The processing steps detailed above should be the same.



---

## 5. G2P

As the grapheme-to-phoneme (G2P) is part of the Transcription Server installation package no further installation is required. This is an optional tool to create pronunciations (e.g. used in KWS).

---

### 5.1 Verify Configuration

Point your browser to `http://x.x.x.x:8120/g2p/phonetize/DeDE_2/Hallo/5` and wait a few seconds until you see the pronunciations. The first request may take 15-20 seconds until the engine is initialized.

The result of this command should be:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><phonetization-Result><phonetizedWords wordClass=""><word>Hallo</word><phonetizations><phonetization soundslike="Hallo" language="DeDE_2">h a l o:</phonetization></phonetizations></phonetizedWords></phonetizationResult>
```

---

## 6. Troubleshooting

In case you have any problems you cannot solve yourself and you want to contact the support make sure to include the relevant parts of the EML Decoder Log and the EML Transcription Server Log.

---

### 6.1 EML Decoder Log

By default, the log file is located at

```
/opt/speech/log/emlDecoder.log
```

---

### 6.2 EML Transcription Server Log

To save the log into a file run:

```
docker logs transcription-server > /tmp/ts.log
```

To follow the log at runtime:

```
docker logs -tail=300 -f transcription-server
```

#### 6.2.1 Increase log level

The log level is on INFO by default. When you experience some issues and want to provide logs please enable DEBUG log and try to reproduce the issue.

```
docker rm -f transcription-server && ./create-transcription-server-container-debug.sh
```