Rosette Server User Guide

1.25.0

Publication date 2023-03-28

Babel Street. Unlock the Most Insights that Matter.

Babel Street provides the most advanced data analytics and intelligence platform for the world's most trusted government and commercial brands. The AI-enabled platform helps them stay informed and improves decision-making for threat intelligence, risk mitigation, identity management, and alerting use cases. Teams are empowered to rapidly detect and collaborate on what matters in seconds by transforming massive amounts of global, multilingual data into actionable and contextual insights so they can act with confidence. Learn more at BabelStreet.com.

Babel Street
1818 Library Street, Suite 500
Reston, VA 20190
703-956-3572
support@rosette.com
http://support.rosette.com

Table of Contents

1. Overview

Rosette Server is a Java-based service that offers Rosette as a locally-deployable package providing access to Rosette's functions as RESTful web service endpoints.

Rosette Server can be installed on Windows, Linux, or macOS. It can also be run as a Docker application.

NOTE
Rosette Server was previously named Rosette Enterprise.

1.1. Minimum system requirements

IMPORTANT
Many installations will require more than 32 GB of disk space [3]. A complete installation may require up to 64GB to install Rosette Server. The exact amount needed will depend on the endpoints and the languages installed.

Any installation including the `/entities`, `/sentiment`, `/topics`, or `/relationships` endpoints will require additional space.

- x86_64 CPU with 4 or more physical cores
- Basic Memory Requirements:
  - Minimum 16GB RAM
  - 50GB of disk space [3] (more may be needed for growing logs)
- 64-bit macOS, Linux, or Windows
- 64-bit JDK 11 or 17 installed (tested with OpenJDK)
- The following commands must be installed on Linux
  - `curl`
  - `netstat`
  - `ps`
  - `gettext`

  - `bash`

- Native OS Libraries, needed for some endpoints
  - **Linux:**
    - glibc-2.17
    - glibc++-3.4.19
    - libgcc_s-3..0
  - **Windows:**
    Microsoft C++ Redistributable for Visual Studio 2017

NOTICE
Endpoints requiring native OS libraries.

- `/name-similarity`, `/name-translation`, and `name-deduplication` when the language is Chinese, Japanese, Korean, Russian, or Arabic.
- `/sentiment` when the option for using DNN model is specified `("options": {"modelType": "dnn"})`
- `/morphology/*`, `/sentences`, `/tokens`, when the endpoint is using a neural model. This includes:
  - When the language of the data is Hebrew and the option for using the DNN models is specified `"options": {"disambiguatorType": "dnn"}`.
  - When the language of the data is Korean and the option for using the DNN models is specified `"options": {"modelType": "dnn"}`.
- `/morphology` when the language of the data is Indonesian, Standard Malay, or Tagalog and the `morphoFeature` is `complete` or `parts-of-speech`.
- `/entities` when the language of the data is English, Arabic, or Korean and the option for using DNN model is specified `("options": {"modelType": "dnn"})`.
- `/relationships`.

1.2. Optimal memory settings

Rosette's memory consumption includes the JVM heap and memory mapping of on-disk files. The size of these vary depending on the endpoint(s) enabled in the instance.

1.2.1. Memory mapped files

Rosette Server's data files are loaded into virtual memory. Some endpoints, such as `/entities`, involve a large amount of data. In order for Rosette to operate at its peak performance, we recommend that you reserve enough free memory to allow memory mapping of all our data files so that page misses are minimized at runtime.

To estimate the size for memory mapping, you can sum up the files in the unpacked `roots` folder in your installation.

Many of Rosette's endpoints organize their data by language. So you can further refine your estimates if you know exactly which languages your input documents are in. Just look for the sub-folders and files under `roots/<component-name>-<version>` that carry a 3-character ISO 693 code and exclude those not applicable to you.

1.3. Disk requirements

You should have sufficient amount of free space to unpack the application and data from your shipment. This could range from 1GB to 90GB. The package size will grow as product updates are released. The amount required is the unpacked directories and files from all `*.tar.gz` in the shipment package.

Rosette Server also requires space to hold logs and other temporary files. Logs can grow depending on the log level and the number of calls. If you have a log rotation mechanism in place, a couple of GBs should be sufficient. Otherwise, experiment with your call patterns and plan for growth accordingly.

1.4. Shipment

You will receive an email containing all the files needed to install Rosette Server, for multiple operating systems and also for using Docker. The files you download depends on your operating system and type of install.

1.4.1. License key file

The Rosette license key file, `rosette-license.xml`, may be sent in the same email or a separate email.

File Name	Function
`rosette-license.xml`	Rosette license key file

TIP

Each endpoint has a **supported-languages** method which returns the languages supported by the endpoint in addition to your license status for the language. The method returns a boolean for the field `licensed` where `true` indicates that you are licensed for the particular language.

1.4.2. Documentation

The release notes (`rosette-server-release-notes-<version>.pdf`) and this user guide (`rosette-server-user-guide-<version>.pdf`) are included with every shipment. Before downloading any other files, you can review the new features and bug fixes included in the release.

File Name	Function
`rosette-server-release-notes-<version>`	Cumulative release notes for Rosette Server
`rosette-server-user-guide-<version>`	This file
`rosette-entity-extractor-appdev-guide-<version>.pdf`	Rosette Entity Extractor Application Developer's Guide

1.4.3. Docker installation

If installing using a Docker container, download only the Docker compose file (`docker_compose.yml`) and the license file (`rosette-license.xml`). Docker will download the remaining files. For offline

installation, [4] you will need to download all the files while connected. Those files will be in the last section of the shipment email.

You must run the installer while connected to the internet to download the files and create the volumes. Once the files are downloaded, you can run `docker-compose up` without be connected to the internet.

File Name	Function
`docker-compose.yml`	Docker compose file

1.4.4. Docker - offline installation

If you are only connected to the internet for the file download, and will be performing the remaining install while not connected to the internet, you will download the Docker yaml file and the all Docker images.

Download all the component tar files and `import_docker_images.sh` to create the docker volumes locally [8].

File Name	
`download.sh`	A helper script to download the components used for the offline Docker installation.
`docker-compose-yaml`	Docker compose file
`import_docker_images.sh`	Script to create Docker volumes locally
`root-<component>-<version>-docker-image.tar`	Root file for <component>. Your shipment will contain one or more root files.
`server-enterprise-<version>-docker-image.tar`	Docker image for Rosette Enterpise server

1.4.5. macOS/Linux

When installing on macOS or Linux, only download the installer (`install_rosette.sh`) and the license file (`rosette-license.xml`). The installer will download the remaining files.

NOTE

You must run the installer while connected to the internet to download the files. Once the files are downloaded, re-run the installer locally, without a connection, to resume the install.

File Name	Function
`install_rosette.sh`	Install script for macOS and Linux

1.4.6. Windows

The Windows install requires manually downloading install files, product files, as well as the component (root) files.

File Name	Function
`package-roots.yaml`	List of the roots for each endpoint
`<component>-root-<version>.tar.gz`	Root file for <component>. Your shipment will contain one or more root files.

File Name	Function
`rosette-enterprise-<version>.tar.gz`	Rosette Server RESTful server
`unpack-roots.bat`	Windows script to unpack and arrange component (root) files

1.4.7. Custom Endpoint Installation

All shipments contain a file for implementing custom endpoints [43]. It contains an installation script and the files for the reverse proxy and application server. The installation script is currently for Linux and macOS only.

File Name	Function
rosette-custom-endpoint-installer-<version>.tar.gz	Custom endpoint installer and archives.

1.5. Root components

The component files (roots) contain the endpoint specific models and data. Each shipment contains one or more root files, based on your license. You may receive roots that you did not order because of Root dependencies [6].

Root Component Files

File Name	Endpoints
`ascent-root-<version>.tar.gz`	Sentiment Analysis
`nlp4j-root-<version>.tar.gz`	Relationship Extraction and Syntactical Dependencies
`rbl-root-<version>.tar.gz`	Morphological Analysis, Tokenization, and Sentence Tagging
`rct-root-<version>.tar.gz`	Transliteration[a.]
`relax-root-<version>.tar.gz`	Relationship Extraction and Syntactical Dependencies
`rex-root-<version>.tar.gz`	Entity Extraction
`rex-root-<version>-<lang>.tar.gz`	
`rli-root-<version>.tar.gz`	Language Identification
`rni-rnt-root-<version>.tar.gz`	Name Similarity, Translation, and Deduplication
`tcat-root-<version>.tar.gz`	Categorization
`topics-root-<version>.tar.gz`	Topics
`tvec-root-<version>-<lang>.tar.gz`	Semantic Vectors and Similar Terms

[a.] Rosette can transliterate Arabizi or Romanized Arabic chat alphabet to native Arabic script and vice versa.

1.5.1. Language-specific files

To minimize the size of your Rosette Server installation, the entity extraction (`rex-root`) and semantic similarity (`tvec-root`) components are shipped by language. The name of the language specific files contain the three letter ISO-639 language code, indicating which language is supported by the file.

- Entity extraction is shipped with one base file and one or more language-specific files. Example:
  - `rex-root-<version>.tar.gz`
  - `rex-root-<version>-eng.tar.gz` for English language files
  - `rex-root-<version>-deu.tar.gz` for German language files
- Semantic Similarity is shipped with one file per language. Example:
  - `tvec-root-<version>-eng.tar.gz` for English language files

- `tvec-root-<version>-deu.tar.gz` for German language files

1.5.2. Root dependencies

If you receive any roots that are not part of your licensed endpoints, it's most likely because that root is a dependency for one of your licensed endpoints. For example, if you license entity extraction, you will also have the root for morphological analysis. The language identification root (`rli-root`) is shipped with many endpoints to determine the language of the request.

Package Contents by Endpoint

Endpoint	Package
categories	`rbl-root-<version>.tar.gz` `tcat-root-<version>.tar.gz`
entities	`rbl-root-<version>.tar.gz` `rex-root-<version>.tar.gz`
language	`rbl-root-<version>.tar.gz` `rli-root-<version>.tar.gz`
morphology	`rbl-root-<version>.tar.gz`
name-deduplication	`rni-rnt-root-<version>.tar.gz`
name-similarity	`rni-rnt-root-<version>.tar.gz`
name-translation	`rni-rnt-root-<version>.tar.gz`

relationships	`nlp4j-root-<version>.tar.gz`
	`rbl-root-<version>.tar.gz`
	`relax-root-<version>.tar.gz`
	`rex-root-<version>.tar.gz`

sentences	`rbl-root-<version>.tar.gz`

sentiment	`ascent-root-<version>.tar.gz`
	`rbl-root-<version>.tar.gz`
	`rex-root-<version>.tar.gz`

syntax/dependencies	`nlp4j-root-<version>.tar.gz`
semantics	`rbl-root-<version>.tar.gz` `tvec-root-<version>.tar.gz`
tokens	`rbl-root-<version>.tar.gz`

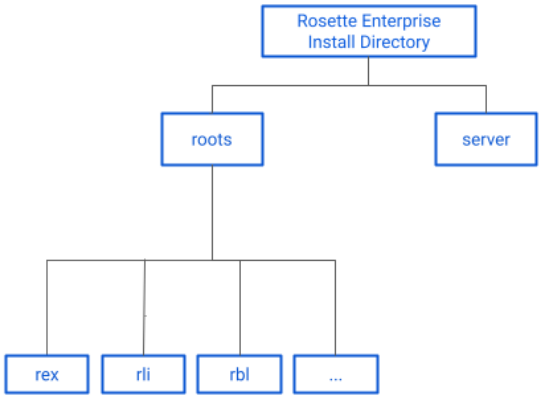
topics	`rbl-root-<version>.tar.gz`
	`rex-root-<version>.tar.gz`
	`topics-root-<version>.tar.gz`

transliteration	`rct-root-<version>.tar.gz`

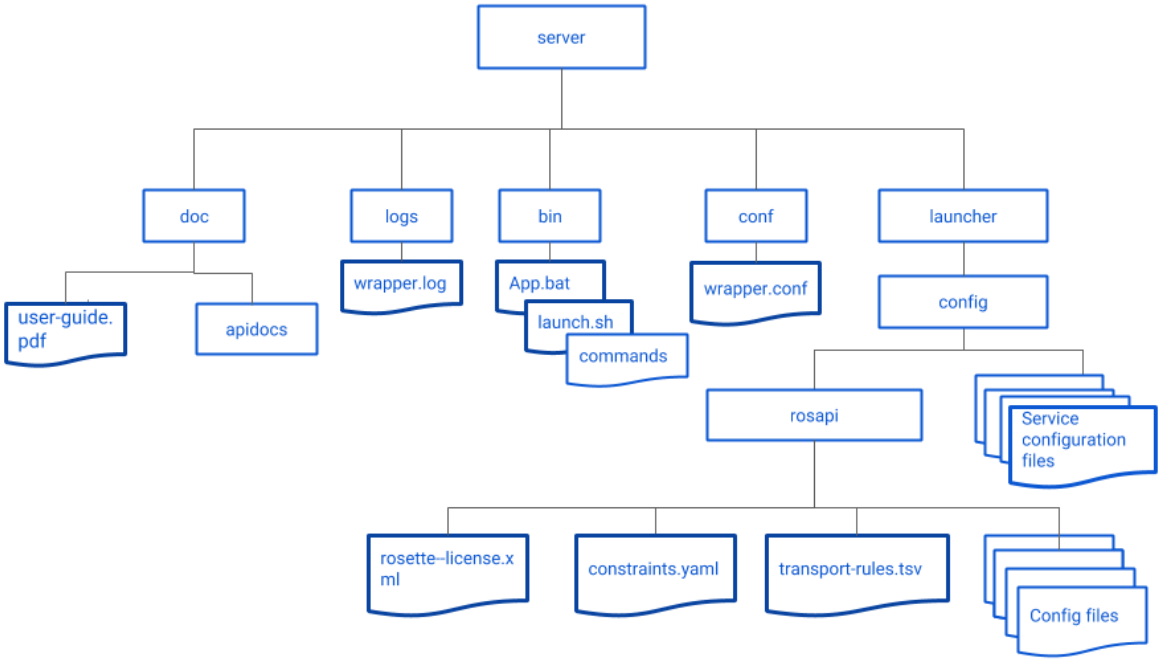
2. Rosette Server file structure

The diagrams below shows the file structure of Rosette Server.

Rosette Server High Level Structure



Server (RESTful) File Structure



2.1. Log files

Rosette uses Log4j to configure and control logging. The file `<rosette>/server/conf/log4j2.xml` configures logging of the Rosette Server instance. This file can be customized to provide the level of logging preferred by your organization.

By default, Rosette generates the following log files in the `<rosette>/server/logs/` directory:

- `500-exception.log`
- `rosapi.log`
- `wrapper.log`

3. Installing with Docker

TIP

Rosette can be configured and run with the roots hosted on an NFS server. An example Helm deployment can be found at https://github.com/rosette-api/helm.

3.1. Docker requirements

- You must have a recent version of Docker Engine installed
- Docker disk image size must be increased to 120GB (from the default of 60GB) to install the full Rosette Server package.
- If installing on Windows, Docker for Windows must be installed (not Docker Toolbox or Docker Machine) with Hyper-V enabled.

3.1.1. Memory requirements

The Docker memory must be set to at least 16 GB if all endpoints are licensed and activated, and may require more depending on your application.

At a minimum, the Docker maximum memory should be the same or more than the Rosette JVM heap size [22]. Otherwise, when running in a Docker container Rosette Server may get SIGKILL when the JVM asks for more memory the Docker allocates.

3.2. Installing offline

Once the Rosette files are downloaded and installed, Docker can be run without being connected to the internet. To install offline:

1. Download the component file tarballs, along with the Docker files and license, from the email containing the Rosette Server files.
2. Run `import_docker_images.sh` to create the Docker volumes.

3. Run the Docker container (`docker compose`) as described below.

3.3. Install and run Docker container

To download the volumes directly, you must have an internet connection.

1. Download the Docker file `docker-compose.yml` and license file `rosette-license.xml`. Note the location of the license file (path-to-license).
2. To run the Docker container (and download the volumes if they haven't already been downloaded):

```
ROSAPI_LICENSE_PATH=<path-to-license>/rosette-license.xml docker compose up
```

   You can also provide a stack name:

```
ROSAPI_LICENSE_PATH=<path-to-license>/rosette-license.xml docker compose\
    -p <stack-name> up
```

3. The Rosette RESTful server will be accessible on the Docker host on the port defined in the `docker-compose.yml` file.

NOTE

If your installation includes the entity extraction component (`rex-root`), you may see `failed to open ...` warning messages for data files in languages not installed in your system. These can safely be ignored.

TIP

Now you can Try it out [16].

3.4. Modifying Rosette parameters in Docker

IMPORTANT

To modify Rosette parameters, edit the `docker-compose.yaml` file.

The following configuration options can be changed by editing the environment section of the file.

- JVM Heap Size [22]
- Number of worker threads [23]
- Whether Rosette is set to Pre-Warm [24]
- Rosette Documentation host port (necessary if you change the host port)

Uncomment the line for the variable and change the value.

NOTE
To run the entity extraction and linking, sentiment analysis, and topic extraction endpoints, the recommended value for ROSETTE_JVM_MAX_HEAP is 16GB. The default value in the file is 4 GB.

```
environment:
# - ROSETTE_JVM_MAX_HEAP=4  # max Java heap size in GB, default is 4, must be >=4;
                            # to run all endpoints the recommended minimum is 16
# - ROSETTE_WORKER_THREADS=2  # number of worker threads, default is 2, must be >=1
# - ROSETTE_PRE_WARM=false  # pre-warm the server on startup, default is false,
                            # valid values are true|false
# - ROSETTE_DOC_HOST_PORT=localhost:8181  # hostname should be accessible on the network,
                                          # port value should match mapped port above
```

You can specify your own volume, for example, backed by a different volume driver.

```
volumes:
# if a local volume is not desirable, change this to suit your needs
rosette-roots-vol:
```

The default docker configuration uses port 8181 for the Rosette endpoints. To change this, modify the ports section.

```
ports:
- "8181:8181"
```

Only the first value in the port statement should be changed. The port statement and the `ROSETTE_DOC_HOST_PORT` value must match.

```
ports:
- "4444:8181"

environment:
- ROSETTE_DOC_HOST_PORT=localhost:4444
```

If you're accessing the documentation from a different machine, change local host to the documentation machine network accessible host name.

3.5. Adding and modifying Rosette files in Docker

There are times you may need to modify and/or add files to the Rosette Server installation. For example, to add an English gazetteer to entity extraction, you must add the file to the *installDirectory*/roots/rex/<version>/data/gazetteer/eng/accept directory.

To access the installation directories within the Docker volumes:

1. By default, the server volume is mounted as read-only (ro). Before starting the container, edit the docker-compose.yml file to make the Rosette server volume writable. Remove the :ro from the end of the rosette-roots-vol: statement:
   As shipped:

   ```
   volumes:
         - rosette-roots-vol:/rosette/server/roots:ro
         - ${ROSAPI_LICENSE_PATH}:/rosette/server/launcher/config/rosapi/rosette-license.xml:ro
   ```

   Modified:

   ```
   volumes:
         - rosette-roots-vol:/rosette/server/roots
         - ${ROSAPI_LICENSE_PATH}:/rosette/server/launcher/config/rosapi/rosette-license.xml:ro
   ```

2. Start the Docker container:

   ```
   ROSAPI_LICENSE_PATH=<path-to-license>/rosette-license.xml docker-compose up
   ```

3. Determine the name of the Docker container:

   ```
   docker ps
   ```

4. Execute an interactive shell on the container:

   ```
   docker exec -it containerID sh
   ```

5. Once in the shell, you can add and modify files within the container.

6. Edit the docker-compose.yml file to set the server volume back to read-only (ro):

   ```
   volumes:
         - rosette-roots-vol:/rosette/server/roots:ro
   ```

7. Stop and restart the Docker container to include the new and edited files.

4. Installing stand-alone on macOS and Linux

Before installing, you must download the install script (install_rosette.sh) and the license file (rosette-license.xml). The remaining files will be downloaded into the same directory as the install script.

NOTE
You must run the installer while connected to the internet to download the files. Once the files are downloaded, re-run the installer locally, without a connection, to resume the install.

To start the install, execute the install script from the directory in which it resides.

```
bash install_rosette.sh
```

WARNING
Set the `java` executable in your `PATH` before starting the install. `PATH` takes precedence over `JAVA_HOME`.

The install will guide you through downloading the files and configuring your system. Each question has a default; you can accept the default or enter a different value.

1. The Rosette Server minimum system requirements are displayed. You can choose to continue, or if your system does not meet the minimum requirements, exit at this point.
2. If you have already downloaded the install files, there is no need to download them again. Depending on the number of roots included, this step can be very time consuming.
3. If the license file is not in the same directory as the install script, you will be prompted for the location of the license file.
4. Enter an install directory or select `Enter` to use the default install directory. The download and install directories cannot be the same. The install directory must be completely empty before installing.

4.1. Run Rosette Server

To run on the console:

```
cd <rosette>/server/bin
./launch.sh console
```

To stop in the console:

```
CTRL-C
```

To run as a daemon:

```
./launch.sh start
```

To stop the daemon:

```
./launch.sh stop
```

NOTE
Check the logs in `<rosette>/server/logs/` to see any errors from startup or during the server run.

NOTE
If your installation includes the entity extraction component (`rex-root`), you may see `failed to open ...` warning messages for data files in languages not installed in your system. These can safely be ignored.

TIP
Now you can Try it out [16].

5. Installing on Windows

This section contains instructions for installing and deploying Rosette Server as a RESTful server on Windows.

5.1. Additional Windows requirements

The packages are shipped as tar.gz files. To unpack the files you will need a third-party archiving tool. A recommended tool is:

- **7-zip** http://www.7-zip.org/

1. Install 7-zip.
2. Add the 7-zip installation directory to your system path.
3. Reboot.
4. Follow the instructions to Unpack the files from a Windows prompt [13].
5. Install the license file [14] when you have completed the installation and are ready to continue.

5.2. Unpack the files from a Windows prompt

From a Windows Command Prompt:

1. Set the install location for the Rosette Server main package, referenced here as `%ROSAPI%`

```
set ROSAPI=C:\rosette-<version>\server
mkdir %ROSAPI%
```

2. Using 7-Zip, unpack the Rosette Server main package from the current directory into the `%ROSAPI%` directory

```
7z x "rosette-enterprise-<version>.tar.gz" -so | 7z x -aoa -si -ttar -o"%ROSAPI%"
```

Ignore any symbolic link messages from 7-Zip.

3. Set the install location for the component packages, referenced here as `%ROSAPI_ROOTS%`

```
set ROSAPI_ROOTS=C:\rosette-<version>\roots
mkdir %ROSAPI_ROOTS%
```

4. Unpack the Rosette component packages from the current directory into `%ROSAPI_ROOTS%` directory

```
unpack-roots.bat "path to root tar.gz files" %ROSAPI_ROOTS%
```

`unpack-roots.bat` is a Windows-specific batch script that requires 7-Zip

5.3. Install the license file

Copy the license file into the subdirectory of your install directory.

```
copy rosette-license.xml %ROSAPI%\launcher\config\rosapi\
```

5.4. Set the runtime environment variables

1. Set `ROSAPI_ROOTS` in the user profile.

```
set ROSAPI_ROOTS=C:\rosette-<version>\roots
```

2. Set `JAVA_HOME` in the user profile, if not already set. For example:

```
set JAVA_HOME=C:\java\jdk-<version>
```

TIP

Your location of JAVA_HOME may differ based on your installation of Java.

3. If installing the **names roots** (name similarity, translation, or deduplication endpoints), add the roots to the path.

```
set PATH=%PATH%;%ROSAPI_ROOTS%\rni-rnt\<rni-rnt root version>\rlp\bin\amd64-w64-msvc120
```

5.5. Run Rosette Server

To run from the Command Prompt:

```
cd %ROSAPI%\bin
App.bat
```

NOTE

If your installation includes the entity extraction component (`rex-root`), you may see `failed to open ...` warning messages for data files in languages not installed in your system. These can safely be ignored.

NOTE

You can check the logs in `%ROSAPI%\logs\` for any errors that you may encounter at startup or when running the server.

To stop Rosette:

From the `/bin directory`

```
cd %ROSAPI%\bin
launcher stop
```

or

```
cd %ROSAPI%\bin
stop-launcher
```

5.6. Install as a Windows service

Before completing these steps, you must have unpacked the files [13]and installed the license file. [14]

1. Check for existing installations.

   ```
   cd %ROSAPI%\bin
   AppCommand.bat status
   ```

2. Remove any existing installations.

   ```
   cd %ROSAPI%\bin
   AppCommand.bat remove
   ```

3. Install Rosette Server as a Windows service.

```
cd %ROSAPI%\bin
AppCommand.bat install
```

4. (Optional) Add the runtime environment variables [14] (`JAVA_HOME` and `ROSAPI_ROOTS`) to the `%ROSAPI%\conf\wrapper.conf` file.

```
set.JAVA_HOME=C:/Java/jdk-<version>
set.ROSAPI_ROOTS=C:/rosette/roots
```

TIP

Any required runtime environment variables should be accessible to the "Local System Account" or configured in `%ROSAPI%\conf\wrapper.conf`.

It is important to understand how the service wrapper evaluates environment variables. You can read more about the details of variable expansion and definition here.

6. Testing the Install

To test out the install, you will need a way to make an `HTTP` request. Common methods are:

- From the command line using curl
- From Windows Powershell using `Invoke-WebRequest`
- From a browser using the interactive documentation [18]

6.1. Ping

Ping the server to test that Rosette is running and you can connect to it.

- bash:

```
curl http://localhost:8181/rest/v1/ping
```

- Windows Powershell:

```
Invoke-WebRequest -Uri http://localhost:8181/rest/v1/ping
```

- Windows Command Prompt:

```
start "" http://localhost:8181/rest/v1/ping
```

This should return:

```
{"message":"Rosette at your service","time":1467912784915}
```

6.2. Query the version

- bash:

```
curl http://localhost:8181/rest/v1/info
```

- Windows Powershell:

```
Invoke-WebRequest -Uri http://localhost:8181/rest/v1/info
```

This should return:

```
{"name":"Rosette","version":"<current-version>","buildNumber":"","buildTime":""}
```

6.3. Test an endpoint

Test an endpoint that you have a license for. For example, the following code tests the `entities` endpoint.

- bash:

```
curl --request POST \
--url http://localhost:8181/rest/v1/entities \
--header 'accept: application/json' \
--header 'content-type: application/json' \
--data '{"content": "Bill Murray will appear in new Ghostbusters film: Dr. Venkman was spotted filming in Boston."}'
```

- Windows Powershell:

```
Invoke-WebRequest -Uri http://localhost:8181/rest/v1/entities
-Method POST
-Headers @{"accept"="application/json"}
-ContentType "application/json"
-Body '{"content":"Bill Murray will appear in new Ghostbusters film: Dr.  Venkman was spotted filming in Boston."}'
```

This will return a list of extracted entities in JSON format.

NOTE
Calling an endpoint that you are not licensed for will result in an error.

7. Upgrading to a New Release

Each release of Rosette Server is a complete release and should be installed in a new directory. You cannot run multiple versions of Rosette Server on the same machine at the same time.

WARNING

macOS and Linux Users

If you have $ROSAPI_ROOTS set from a previous release, you will need to remove it before starting the install script.

1. Download and install the new release into an empty directory, following the instructions for your operating system.
2. On Windows:
   Update the $ROSAPI_ROOTS and $ROSAPI environment variables to the new locations.
   Copy in the new license file.
3. Ensure that you have stopped the server from the previous release.
4. Start the new server.

Once you have installed the new release, you can delete the previous version. You may choose to keep the old version in case you encounter issues with the new installation.

8. Accessing the deployed documentation

Once the Rosette Server is running, you can access the documentation.

IMPORTANT

The recommended browser for viewing the documentation is Chrome. Edge and IE may not properly display the pages.

- **Features and Functions**
  Provides an overview of each endpoint with actual code and response examples.

```
http://localhost:8181/rest/doc/
```

- **Interactive Documentation**
  Allows you to make calls to Rosette from within the browser.

```
http://localhost:8181/rest/doc/swagger
```

NOTE
If you try to view the documentation from a browser that's not on the server where Rosette Server is installed, you will need to replace `localhost:8181` with the appropriate hostname and ensure that the port is accessible. See Update documentation hostname [29] on how to update the documentation hostname.

9. Configuration files

There are two groupings of configuration files.

- The system configuration [19] files are found in `config`. These are the files for overall service configuration. These are files whose names end in `.cfg`. These files are in Java property file syntax, and define name-value pairs.
- The more complex Endpoint and transport rules configuration files [20] are found in `launcher/config/rosapi`. These are the files you need to edit to change the input parameters, transport rules, and configuration requirements of the individual endpoints.

9.1. System configuration files

These are the configuration files for overall service operation. The perceptive reader may notice that there are endpoint specific configuration files here as well, e.g. dedupe, rni, rnt. The architecture is still evolving.

Location: server/launcher/config

Configuration File	Purpose
`com.basistech.downloadextract.cfg`	Defines the pathnames for the download extractor config file and the constraints file.
`com.basistech.rli.cfg`	Configuration for RLI specifically the short string threshold.
`com.basistech.worker.service.cfg`	Configuration for worker service startup.
`com.basistech.ws.cxf.cf`	Configuration for CXF. Defines urlBase. Works in conjunction with org.apache.cxf.http.jetty-main.cfg.
`com.basistech.ws.dedupe.cfg`	Configuration for name deduplication.
`com.basistech.ws.doc.cfg`	Configuration for the documentation settings.
`com.basistech.ws.fe.health.cfg`	Configuration for frontend health check.
`com.basistech.ws.frontend.cfg`	Configuration for the front end RESTful services parameters.
`com.basistech.ws.local.usage.tracker.cfg`	Configuration file for usage tracking [39].
`com.basistech.ws.metrics.cfg`	Configuration for AWS cloudwatch metrics.
`com.basistech.ws.rni.cfg`	Configuration for name indexing.
`com.basistech.ws.rnt.cfg`	Configuration for name translation.
`com.basistech.ws.transport.http.cfg`	This configures the workers' web server's asynchronous request processing, its queuing, and failure retry.
`com.basistech.ws.worker.cfg`	Configuration for the location of each endpoint root as well as the native code root. It also contains configuration settings for worker threads and CloudWatch metrics.

Configuration File	Purpose
`org.apache.cxf.http.jetty-main.cfg`	Configuration for CXF HTTP Jetty. CXF is Rosette's webservice framework and Jetty is its embedded webserver. Works in conjunction with com.basistech.ws.cxf.cfg.
`org.apache.cxf.osgi.cfg`	Configuration for Apache CXF.

9.2. Endpoint and transport rules configuration files

The following files contain configuration parameters for Rosette Server, individual endpoints, and transport rules.

Location: `/server/launcher/config/rosapi`

Configuration File	Purpose
`constraints.yaml`	Defines input constraints [26] for Rosette: maximums for document and text input sizes as well as names deduplication list size. It is referenced by `com.basistech.downloadextract.cfg`.
`downloaderExtractor.yaml`	Provides the detailed configuration of the download/text-extractor (DTE) component. It is referenced by `com.basistech.downloadextract.cfg`.
`transport-rules.tsv`	Provides a mapping to the worker REST URL. See Understanding the transport rules [30]
`worker-config.yaml`	Configures the pipelines. The entries contained in this file are highly dependent on the backend code. It may be useful for identifying language support for the various endpoints as well as the associated endpoint configuration file, but should be left in its shipped state unless otherwise instructed.
`xxx-factory-config.yaml`	Lists the individual factory configurations, as called for by `worker-config.yaml`. Their contents are defined by the individual endpoint requirements. These files are where you modify the configuration for individual endpoints.

9.3. Factory configuration files

The `worker-config.yaml` file details component factories and the pipelines for each endpoint. A single endpoint may use multiple factories. Use this file to determine which factories you may have to modify to set the configuration values for a task. Some factories, such as `rbl-factory-config.yaml` are used by multiple endpoints.

Factory Files

File Name	Primary Endpoint
`analyze-factory-config.yaml`	topics
`cat-factory-config.yaml`	categories
`dp-factory-config.yaml`	syntax/dependencies
`rbl-factory-config.yaml`	morphology sentences
`rct-factory-config.yaml`	transliteration
`relax-factory-config.yaml`	relationships
`rex-factory-config.yaml`	entities
`rex-no-resolution-factory-config.yaml`	topics entities
`rli-factory-config.yaml`	language
`rni-dedup-factory-config.yaml`	name-deduplication
`rni-factory-config.yaml`[a]	name-similarity
`rnt-factor-config.yaml`	name-translation
`semantic-vectors-factory-config.yaml`	semantics/vector

File Name	Primary Endpoint
sent-factory-config.yaml	sentiment
similar-terms-factory-config.yaml	semantics/similar
tokenization-factory-config.yaml	tokens
topics-factory-config.yaml	topics

[a] To modify the parameters of the name-similarity endpoint, see Name similarity configuration files [22]

9.4. Disabling licensed endpoints

Typically, all endpoints which you have active licenses for will load and run when called. Use these instructions to disable specific endpoints.

WARNING

We recommend that you do not disable the `/language` endpoint as it is used by many endpoints to identify the language of the request.

1. In the `server/launcher/config` directory, create a file named `override-endpoints.yaml` listing only the endpoints that you want enabled.

```
endpoints:
    - /language
    - /entities
    - /categories
    - /semantics/vector
    - /morphology
    - /sentences
    - /tokens
```

2. In the `server/launcher/config` directory, edit the file `com.basistech.ws.worker.config` and specify the `overrideEndpointsPathname` parameter. This will be the full path name of the file you created in the previous step. For example, if you installed Rosette in the directory `/Users/user/rosette-1.20.3`, the parameter would be:

```
overrideEndpointsPathname=/Users/user/rosette-1.20.3/server/launcher/config/override-endpoints.yaml
```

3. Restart Rosette Server for the changes to take effect.

If you call a disabled endpoint, you will receive an `"unknownError"`. For example, if you are licensed for the /topics endpoint, but disable it, a call to the endpoint will return the following JSON:

```
{
  "code": "unknownError",
  "message": "Worker unsupported target Endpoint{path=/topics}/eng",
  "stack": null
}
```

The `/ping` and `/info` endpoints are always enabled. They do not have to be listed in the `override-endpoints.yaml` file.

9.5. Name similarity configuration files

There are two `.yaml` files located in the *`installDirectory`*`/roots/rni-rnt/<version>/rlpnc/data/etc` directory to guide you in configuring the name-similarity endpoint, `parameter_defs.yaml` and `parameter_profiles.yaml`. The `parameter_defs.yaml` file lists the default value for all parameters, along with a short description. This file should not be modified.

To configure the name-similarity results, change the values of the parameters in the `parameter_defs.yaml` file. Parameter values can be for all language pairs, or for a specific language pair. If the change is for all languages, use the `any:` profile. If a parameter change is for a specific language pair, use the appropriate language code pair. The two language codes are always written in alphabetical order, except for `eng`, which always comes last.

Add a parameter for all languages

1. Edit the *`installDirectory`*`/roots/rni-rnt/<version>/rlpnc/data/etc/parameter_defs.yaml` file.
2. Search for `any:`
3. Add *parameterName: parameter value*
4. Save

Add a parameter for Spanish-English matching

1. Edit the *`installDirectory`*`/roots/rni-rnt/<version>/rlpnc/data/etc/parameter_defs.yaml` file.
2. Search for the language combination `spa_eng:`
3. Add *parameterName: parameter value*
4. Save

10. Configuring server performance

10.1. Configuring the JVM heap size

There is not a single one size fits all number here. The best value for max heap size depends on a number of factors:

- activated endpoints and features
- usage pattern
- data characteristics such as size (both character and token lengths), language, and genre
- java garbage collector and its settings

Our recommendation is to follow directions from well-known sources, such as this to experiment with heap settings by testing your usage of Rosette Server in order to identify the ideal settings that suits you the best.

Please note that it's not recommended setting the max heap to the amount of physical RAM in the system. More heap doesn't always translate to better performance, especially depending on your garbage collection settings. Also, we do require sufficient amount of free memory for memory mapped files.

Use this table to estimate the minimum heap required based on your selection of endpoints. Note that endpoints may have implicit code dependencies on other endpoints, so the dependencies' heap needs to be added if they have not been accounted for.

TIP
We recommend setting the initial and max heap to the same value.

Per endpoint JVM heap recommendation

Endpoint	Min Heap	Note
language	0.25GB	
morphology	1.5GB	
transliteration	0.5GB	
entities	1GB	add 1.5GB if morphology is not already enabled
sentiment	1GB	add 1.5GB if morphology is not already enabled; add 1GB if entities is not already enabled
categories	1GB	add 1.5GB if morphology is not already enabled
topics	1.5GB	add 1.5GB if morphology is not already enabled; add 1GB if entities is not already enabled
text-embeddings	1GB	add 1.5GB if morphology is not already enabled
relationships	3GB	add 1.5GB if morphology is not already enabled; add 1GB if entities is not already enabled
dependencies	0.4GB	
name-similarity	2GB	combined with name-translation
name-translation	2GB	combined with name-similarity
name-deduplication	2GB	add 2GB if neither name-similarity or name-translation is on

On macOS/Linux or Windows:

1. Edit the file `server/conf/wrapper.conf`
2. Modify the value of `wrapper.java.maxmemory`

With Docker:

1. Edit the file `docker-compose.yml`
2. Modify the value of `ROSETTE_JVM_MAX_HEAP`

10.2. Configuring worker threads for HTTP transport

Multiple worker threads allow you to implement parallel request processing. Generally, we recommend that the number of threads should be less than the number of physical cores or less than the total number of hyperthreads, if enabled.

You can experiment with 2-4 worker threads per core. More worker threads may improve throughput a bit, but typically won't improve latency. The default value of worker threads is 2.

If the URL for all licensed endpoints are set to `local:` (not distributed):

1. Edit the file `/launcher/config/com.basistech.ws.transport.embedded.cfg`.
2. Modify the value of `workerThreadCount`

If using transport rules in a distributed deployment on macOS/Linux or Windows:

1. Edit the file `/launcher/config/com.basistech.ws.transport.embedded.cfg`.
2. Modify the value of `workerThreadCount`.
3. Edit the file `/launcher/config/com.basistech.ws.worker.cfg`
4. Modify the value of `workerThreadCount`

If using Docker, only the `docker-compose.yml` file must be modified:

1. Edit the file `docker-compose.yml`
2. Modify the value of `ROSETTE_WORKER_THREADS`

10.3. Setting Rosette to pre-warm

To speed up first call response time, Rosette can be pre-warmed by loading data files at startup at the cost of a larger memory footprint.

Most components load their data lazily, meaning that the data required for processing will only be loaded into memory when an actual call hits. This is particularly true for language-specific data. The consequence is that when the very first call with text in a given language arrives at a worker, the worker can take a quite a bit of time loading data before it can process the request.

Pre-warming is Rosette's attempt to address the 1st-call penalty by hitting the worker with text in every licensed language it supports at boot time. Then, when an actual customer request comes in, all data will have already been memory mapped and you won't experience a first call delay as the data is loaded. Only languages licensed for your installation will be pre-warmed.

The default is set to `false`, pre-warm is *not* enabled.

To set Rosette to warm up the worker upon activation

On macOS/Linux or Windows:

1. Edit the file `/com.basistech.ws.worker.cfg`
2. set `warmUpWorker=true`

TIP
When installing on macOS or Linux, Rosette can be set to pre-warm in the installation. Select `Y` when asked `Pre-warm Rosette at startup?` You can always change the option by editing the `com.basistech.ws.worker.cfg` file.

With Docker:

1. Edit the file `docker-compose.yml`
2. Set `ROSETTE_PRE_WARM=true`

10.4. Configuring the transport rules

The transport rules provide a means of defining a distributed installation of endpoints. By default this is not enabled and the `transport-rules.tsv` is not in the distribution therefore making all endpoints run locally (in the same address space as the front end).

If using transport rules in a distributed deployment there is a special URL, `local:`, which routes requests within the JVM, bypassing the overhead associated with a network connection. Generally, our recommendation is to use `local:` if the worker resides on the same machine and same JVM as the frontend.

Typically, for large scale deployments, we would recommend having endpoints with a high hit-rate distributed on a machine separate from the server's frontend. The /language endpoint is an exception to this rule. It is called internally on every request that does not have the language preset. However, it does not consume a lot of the server's resources, so we advise keeping it on `local:` to minimize the networking overhead.

TIP
When using the single-box monolith Rosette Server deployment, we recommend setting the URL for *all* licensed endpoints to `local:`.

10.5. Setting the language parameter

If the language of the input text is known, you can add the language parameter to bypass the language identification step in the processing pipeline, speeding up the processing time and increasing throughput.

Each document endpoint accepts an optional language parameter:

```
{"content": "your_text_here", "language":"eng"}
```

10.6. Optimizing the /entities endpoint

If the data consists of many relatively small individual files, concatenating them will improve the throughput. But you must be aware that this can impact the accuracy of the model. The statistical model includes a

consistency feature which reflects a tendency of the model to label recurring tokens with the same type. This may cause entities to be labelled incorrectly when concatenating text samples that don't share the same context.

Regular Expressions

Regular expressions (regexes) are used for finding entities which follow a strict pattern with a rigid form and infinite combinations, such as URLs and credit card numbers. In the default REX installation the regex files are:

- **language specific:** `data/regex/<lang>/accept/regexes.xml` where <lang> is the ISO 693-3 language code
- **cross-language:** `data/regex/xxx/accept/regexes.xml`
- **supplemental:** `data/regex/<lang>/accept/supplemental`

Regular expressions can decrease throughput performance. The /entities endpoint is pre-configured with a set of regular expressions. You can improve performance by removing unused expressions by:

- moving the files with the unused expressions out of the directory, or
- commenting out specific expressions within the file.

The supplemental regular expressions are configured in the `rex-factory-config.yaml` file. Remove or comment out values from the `supplementalRegularExpressionPaths` parameter to remove unused supplemental regex files.

Disable Entity linking [52]. By default, entity linking is disabled, but enabling it can slow down the response time of Rosette Server.

Disable Pronominal resolution [53] By default, pronominal resolution is disabled, but enabling it can slow down the response time of Rosette Server.

Disable In-document Coreference Documents often contain multiple references to a single entity. In-document coreference (indoc coref) chains together all mentions to the same entity. By default, indoc coref is disabled (`NULL`).

11. Advanced configuration options

The following sections describe custom installation configurations and will not apply to all installs.

11.1. Modify the input constraints

The limits for the input parameters are in the file `/rosapi/constraints.yaml`. Modify the values in this file to increase the limits on the maximum input character count and maximum input payload per call. You can also increase the number of names per list for each call to the name deduplication endpoint.

The default values were determined as optimal during early rounds of performance tests targeting < 2 second response times. Larger values may cause degradation of system performance.

constraints.yaml

Parameter	Minimum	Maximum	Default Value	Description
maxInputRawByteSize	1	10,000,000	614400	The maximum number of input bytes per raw doc
maxInputRawTextSize	1	1,000,000	50000	The maximum number of input characters per submission
maxNameDedupeListSize	1	100,000	1000	The maximum number of names to be deduplicated.

To modify the input constraints:

1. Edit the file `/rosapi/constraints.yaml`
2. Modify the value for one or more parameters

11.2. Example

NOTE
These instructions assume all workers are on a single machine. If Rosette Server is installed in an environment with distributed workers, contact Rosette support.

1. Generate an RSA key pair for the server.
   This example is for evaluation purposes only. The generated key is good for seven days. Please work with your appropriate internal group to acquire your keys for production usage.

```
$JAVA_HOME/bin/keytool -genkeypair \
        -validity 7 \
        -alias myservicekey \
        -keystore serviceKeystore.jks \
        -dname "cn=exampleName, ou=exampleGroup, o=exampleCompany, c=us" \
        -keypass skpass \
        -storepass sspass \
        -keyalg RSA \
        -sigalg SHA256withRSA
```

2. Set the permissions for the keystore file to read only

```
chmod 400 serviceKeystore.jks
```

3. Rename the file `launcher/config/rosapi/transport-rules.tsv`. Removing this file forces local transports for all endpoints. We recommend renaming the file, to have the original file as a backup.

```
mv launcher/config/rosapi/transport-rules.tsv launcher/config/rosapi/transport-rules.tsv.original
```

4. Change `http` to `https` in `launcher/config/com.basistech.ws.cxf.cfg`.

```
urlBase=https://0.0.0.0:${rosapi.port}/rest
```

5. Edit the file `launcher/config/org.apache.cxf.http.jetty-main.cfg` and add the following lines to use the generated keystore:

```
tlsServerParameters.keyManagers.keyPassword=skpass
tlsServerParameters.keyManagers.keyStore.file=<path_to_keystore>/serviceKeystore.jks
tlsServerParameters.keyManagers.keyStore.password=sspass
tlsServerParameters.keyManagers.keyStore.type=JKS
```

11.2.1. Optional: SSL with remote workers

To use remote workers, the certificate needs to be trusted.

For testing, import the certificate to the truststore file, `cacerts.jks`, as trusted.

This example is for evaluation purposes only, continuing using the previously generated key. Please work with your appropriate internal group to acquire your keys for production usage. If your key is acquired from a trusted certificate authority, no further configuration may be required. As this example uses self-signed certificates, the following steps are necessary.

1. Export the certificate from the Java KeyStore.

```
keytool -exportcert \
        -alias myservicekey \
        -keystore serviceKeystore.jks \
        -file server.cer \
        -storepass sspass
```

2. Import the certificate into a trust store.

```
keytool -import \
        -v \
        -trustcacerts \
        -alias localhost \
        -file server.cer \
        -keystore cacerts.jks \
        -storepass capass
```

3. Instruct the JRE to trust the self-signed certificate by updating `conf/wrapper.conf`.

```
wrapper.java.additional.201=-Djavax.net.ssl.trustStore=/path-to-cacerts/cacerts.jks
wrapper.java.additional.202=-Djavax.net.ssl.trustStorePassword=capass
```

11.3. Enable passing files to endpoints

Most endpoints can take either a text block, a file, or a link to a webpage as the input text. The webpage link is in the form of a URI. To enable passing a URI to an endpoint, the `enableDTE` flag must be set in the file `com.basistech.ws.worker.cfg`.

By default, the flag is set to `True`; URI passing is enabled.

```
#download and text
extractorenableDte=true
```

11.4. Change Rosette RESTful server port

NOTE
Use this is to change the default port on Windows installations or to change the server port after installation on Linux and macOS. The Linux and macOS install script `install_rosette.sh` appends this line to `conf/wrapper.conf` during install if you override the default.

The default installation uses port `8181` for the Rosette endpoints. To change the default port, edit the file `conf/wrapper.conf`; uncomment and modify the port value.

```
wrapper.java.additional.301=-Drosapi.port=8181
```

For example, change 8181 to 9191.

When changing the port, update the documentation hostname as well:

1. `cd doc/swagger`
2. Edit the file `swagger.yaml`
3. In the `servers` section, replace `localhost:8181` with the correct port.

```
servers:
  - url: 'http://localhost:8181/rest/v1/'
```

11.5. Update documentation hostname

NOTE
Use this is to change the default port on Windows installations or to change the server port after installation on Linux and macOS. The Linux and macOS install script `install_rosette.sh` appends this line to `conf/wrapper.conf` during install if you override the default.

If you want to change the default port to execute the interactive documentation:

1. `cd doc/swagger`
2. Edit the file `swagger.yaml`
3. In the `servers` section, replace `localhost:8181` with the correct port.

```
servers:
  - url: 'http://localhost:8181/rest/v1/'
```

11.6. Understanding the transport rules

The transport rules provide a means of mapping an endpoint along with some defined options (language, linked-entities, length) to a processing URL. Transport rules allow you to define a distributed deployment routing a subset of calls to different machines, balancing the load by routing high demand calls to separate machines.

Location: `<version>/server/launcher/config/rosapi/transport-rules.tsv`

The basic format of an entry is:

```
endpoint [tab] options [tab] URL
```

where:

- `endpoint` is any valid endpoint and may have one or more entries. Multiple rules for the same endpoint are processed in the order listed; if there is a conflict between rules, the first one processed prevails. The most specific rules should be listed first. The last rule should be general enough to match any remaining conditions. Conditions left without a valid routing will fail.
- `options` zero or more of:
  - `lang=a|b|c` provides a list of languages for this particular rule. Zero or more entries are permitted.
  - the wildcard `*` may be used to specify any languages
  - `length > n` and `length < n` where length is the number of UTF-16 characters in the input string. To define a range, you need to create 2 rules, a < rule and a > rule.
- `URL` valid URL for transport. There is a special URL, `local:`. This is used to route `/language` requests to be processed inside the front end, not sent over the network at all. The recommendation is to use `local:` if the worker resides on the same machine and same JVM with the frontend.

An example of multiple endpoint entries:

```
/entities lang=eng|spa, http://localhost:${rosapi.port}/rest/worker/process
/entities linkEntities=false  http://localhost:${rosapi.port}/rest/worker/process
/entities lang=ara|eng|jpn|spa|zho  http://localhost:${rosapi.port}/rest/worker/process
/entities * http://localhost:${rosapi.port}/rest/worker/process
/language       *       local:
```

11.7. Install TensorFlow GPU support

NOTE
TensorFlow GPU is currently only available on the Linux and Windows operating systems.

To improve Rosette's performance, especially when invoking deep neural network based models, you may choose to run on a GPU.

Download TensorFlow

To make use of GPUs on your Linux system, you should download `libtensorflow_jni_gpu-1.14.0.jar` Platform-dependent native code with GPU (CUDA) support for the TensorFlow Java library from http://repo1.maven.org/maven2/org/tensorflow/libtensorflow_jni_gpu/1.14.0/libtensorflow_jni_gpu-1.14.0.jar.

You can also compile your own version of `libtensorflow_jni`, which may provide better performance than the pre-compiled version.

```
cd $ROSAPI
mkdir tf_jni_gpu
jar xf libtensorflow_jni_gpu-1.14.0.jar -C tf_jni_gpu
```

Edit `conf/wrapper.conf` to modify `java.library.path` to the following:

```
wrapper.java.library.path.1=../tf_jni_gpu/org/tensorflow/native/linux-x86_64
wrapper.java.library.path.2=../lib
```

Verify TensorFlow GPU Support

To verify TensorFlow GPU support, run Rosette

```
cd $ROSAPI/bin
./launch.sh console
```

And in an entity request with `DNN modelType` options:

```
curl --request POST \
--url http://localhost:8181/rest/v1/entities \
--header 'accept: application/json' \
--header 'content-type: application/json' \
--data '{"content": "Barack Obama was born in Hawaii", \
  "options": { "modelType": "DNN" }}'
```

Verify that you see Tensorflow found and create a GPU in your console output

```
jvm 1    | 2018-04-18 18:38:15.346273: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1344] Found device 0 with prope
jvm 1    | name: Tesla K80 major: 3 minor: 7 memoryClockRate(GHz): 0.8235
jvm 1    | pciBusID: 0000:00:1e.0
jvm 1    | totalMemory: 11.17GiB freeMemory: 11.10GiB
jvm 1    | 2018-04-18 18:38:15.346291: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1423] Adding visible gpu device
jvm 1    | 2018-04-18 18:38:15.617319: I tensorflow/core/common_runtime/gpu/gpu_device.cc:911] Device interconnect Stream
 1 edge matrix:
jvm 1    | 2018-04-18 18:38:15.617346: I tensorflow/core/common_runtime/gpu/gpu_device.cc:917]      0
jvm 1    | 2018-04-18 18:38:15.617351: I tensorflow/core/common_runtime/gpu/gpu_device.cc:930] 0:   N
jvm 1    | 2018-04-18 18:38:15.617609: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Created TensorFlow device
ca:0/task:0/device:GPU:0 with 10764 MB memory) -> physical GPU (device: 0, name: Tesla K80, pci bus id: 0000:00:1e.0, com
jvm 1    | 2018-04-18 18:38:15.941174: I tensorflow/cc/saved_model/loader.cc:161] Restoring SavedModel bundle.
jvm 1    | 2018-04-18 18:38:16.094135: I tensorflow/cc/saved_model/loader.cc:196] Running LegacyInitOp on SavedModel bund
jvm 1    | 2018-04-18 18:38:16.131329: I tensorflow/cc/saved_model/loader.cc:291] SavedModel load for tags { serve }; Sta
397 microseconds.
```

Known Limitations

Tensorflow does not provide GPU support for Java macOS as of Tensorflow 1.14

12. Custom profiles

Rosette Server can support multiple profiles, each with different data domains (such as user dictionaries, regular expressions files, and custom models) as well as different parameter and configuration settings. Each profile is defined by its own root directory, thus any data or configuration files that live in the root directory of an endpoint can be part of a custom profile.

Using custom profiles, a single endpoint can simultaneously support users with different processing requirements within a single instance of Rosette Server. For example, one user may work with product reviews and have a custom sentiment analysis model they want to use, while another user works with news articles and wants to use the default sentiment analysis model.

Each unique profile in Rosette Server is identified by a string, `profileId`. The profile is specified when calling the API, by adding the `profileId` parameter, indicating the set of configuration and data files to be used for that call.

Custom profiles and their associated data are contained in a `<profile-data-root>` directory. This directory can be anywhere in your environment; it does not have to be in the Rosette Server install directory.

Examples of types of customizable data by endpoint

Endpoint	Applicable data files for custom profile
/categories	Custom models
/entities	Gazetteers, regular expression files, custom models, linking knowledge base
/morphology	User dictionaries
/sentiment	Custom models
/tokens	Custom tokenization dictionaries

NOTE
Custom profiles are not currently supported for the `address-similarity`, `name-deduplication`, `name-similarity`, and `name-translation` endpoints.

12.1. Setting up custom profiles

1. Create a directory to contain the configuration and data files for the custom profile.
   The directory name must be 1 or more characters consisting of `0-9`, `A-Z`, `a-z`, underscore or hyphen and no more than 80 characters long. It cannot contain spaces. It can be anywhere on your server; it does not have to be in the Rosette Server directory structure. This is the `profile-data-root`.

2. Create a subdirectory for each profile, identified by a **profileId**.
   For each profile, create a subdirectory named **profileID** in the **profile-data-root**. The **profile-path** for a project is `profile-data-root/profileId`.

For example, let's assume our `profile-data-root` is **rosette-users**, and we have two profiles: **group1** and **group2**. We would have the following `profile-paths`:

```
rosette-users/group1
rosette-users/group2
```

3. Edit the Rosette Server configuration files to look for the profile directories.
   The configuration files are in the `launcher/config/` directory. Set the `profile-data-root` value in these files:

   - `com.basistech.ws.worker.cfg`
   - `com.basistech.ws.frontend.cfg`

   ```
   # profile data root folder that may contain profile-id/{rex,tcat} etc
   profile-data-root=file:///Users/rosette-users
   ```

4. Add the customization files for each profile. They may be configuration and/or data files.

   - Custom configuration [34]
   - Custom data sets [34]
   - Custom models [35]

When you call the API, add `"profileId" = "myProfileId"` to the body of the call.

```
{"content": "The black bear fought the white tiger at London Zoo.",
 "profileId": "group1"
}
```

12.2. custom-profiles

`https://localhost:8181/rest/v1/custom-profiles`

The `/custom-profiles` endpoint returns a list of all custom profiles on the server.

```
curl -s http://localhost:8181/rest/v1/custom-profiles
```

If the call includes an `app-id` in the request header, the custom-profiles endpoint returns all profiles under the specified `app-id`.

```
curl -s http://localhost:8181/rest/v1/custom-profiles -H "X-RosetteAPI-App-Id: app-id"
```

12.3. Updating custom profiles

New profiles are automatically loaded in Rosette Server. You do not have to bring down or restart the instance to add new models or data to Rosette Server.

When editing an existing profile, you may need to restart Rosette Server. If the profile has been called since Rosette Server was started, the Server must be restarted for the changes to take effect. If the profile has not been called since Rosette Server was started, there is no need to restart.

To add or update models or data, assuming the custom profile root `rosette-users` and profiles `group1` and `group2`.

1. Add a new profile with the new models or new data, for example `group3`.
2. Delete the profile and re-add it. Delete `group1` and then recreate the `group1`directory with the new models and/or data.

12.4. Custom configuration

The configurations for each endpoint are contained in the factory configuration files [20]. The `worker-config.yaml` file describes which factory configuration files are used by each endpoint as well as the pipelines for each endpoint. To modify parameter values or any other configuration values, copy the factory configuration file into the profile path and modify the values.

Modifying entities parameters default values

Let's go back to our example with profile-ids of group1 and group2. Group1 wants to modify the default entities parameters, setting entity linking to `true` and case sensitivity to `false`. These parameters are set in the `rex-factory-config.yaml` file.

1. Copy the file `/launcher/config/rosapi/rex-factory-config.yaml` to `rosette-users/group1/config/rosapi/rex-factory-config.yaml`.
2. Edit the new `rex-factory-config.yaml` file as needed. This is an excerpt from a sample file.

```
# rootDirectory is the location of the rex root
rootDirectory: ${rex-root}

# startingWithDefaultConfigurations sets whether to fill in the defaults with CreateDefaultExtrator
startingWithDefaultConfiguration: true

# calculateConfidence turns on confidence calculation
# values: true | false
calculateConfidence: true

# resolvePronouns turns on pronoun resolution
# values: true | false
resolvePronouns: true

# rblRootDirectory is the location of the rbl root
rblRootDirectory: ${rex-root}/rbl-je

# case sensitivity model defaults to auto
caseSensitivity: false

# linkEntities is default true for the Cloud
linkEntities: true
```

12.5. Custom data sets

Each profile can include custom data sets. For example, the entities endpoint includes multiple types of data files, including regex and gazetteers. These files can be put into their own directory for entities, known as an overlay directory [51]. This is an additional data directory which takes priority over the default entities data directory.

NOTE

If the data overlay directory is named **rex**, the contents of the overlay directory will completely replace all supplied REX data files, including models, regex, and gazetteer files.

- If your custom data sets are intended to supplement the shipped files, the directory name must not be `rex`.
- If your custom data sets are intended to completely replace the shipped files, use the directory name `rex`.

Custom Gazetteer for the Entities Endpoint

We will create a custom gazetteer file called `custom_gaz.txt` specifying "John Doe" as an ENGINEER entity type. Full details on how to create custom gazetteer files are in the section **Creating a Custom Gazetteer** in the *Rosette Entity Extractor Application Developer Guide*.

1. Create the custom gazetteer file in `/Users/rosette-users/group1/custom-rex/data/gazetteer/eng/accept/custom_gaz.txt`. It should consist of just two lines:

```
ENGINEER
John Doe
```

2. Copy the file `/launcher/config/rosapi/rex-factory-config.yaml` to `/Users/rosette-users/group1/config/rosapi/rex-factory-config.yaml`.

3. Edit the new `rex-factory-config.yaml` file, setting the `dataOverlayDirectory`.

```
# rootDirectory is the location of the rex root
rootDirectory: ${rex-root}
dataOverlayDirectory: "/Users/rosette-users/group1/custom-rex/data"
```

4. Call the entities endpoint with the `profileId` set to `group1`:

```
curl -s -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "Cache-Control: no-cache" \
    -d '{"content": "John Doe is employed by Basis Technology", "profileId": "group1"}' \
    "http://localhost:8181/rest/v1/entities"
```

You will see "John Doe" extracted as type ENGINEER from the custom gazetteer.

12.6. Custom models

You can train and deploy a custom model to the entities endpoint for entity extraction. You can either:

- Copy the model file to the default data directory in the REX root folder. `<RosetteServerInstallDir>/roots/rex/<version>/data/statistical/<lang>/<modelfile>` where <lang> is the 3 letter language code for the model.

- Copy the model to the data directory of a custom profile.
  `<profile-data-root>/<profileId>/data/statistical/<lang>/<modelfile>` where <lang> is the 3 letter language code for the model.
  The custom profile must be set up as described in Setting up custom profiles [32]

TIP
Model Naming Convention

The prefix must be `model.` and the suffix must be `-LE.bin`. Any alphanumeric ASCII characters are allowed in between.

Example valid model names:

- model.fruit-LE.bin
- model.customer4-LE.bin

12.7. Example

In this example, we're going to add the entity types **COLORS** and **ANIMALS** to the entities endpoint, using a regex file.

1. Create a `profile-data-root`, called **rosette-users** in the `Users` directory.
2. Create a user with the `profileId` of **group1**. The new `profile-path` is:

   ```
   /Users/rosette-users/group1
   ```

3. Edit the Rosette Server configuration files:

   - `/launcher/config/com.basistech.ws.worker.cfg`
   - `/launcher/config/com.basistech.ws.frontend.cfg`

   adding the profile-data-root.

   ```
   # profile data root folder that may contain app-id/profile-id/{rex,tcat} etc
   profile-data-root=file:///Users/rosette-users
   ```

4. Copy the `rex-factory-config.yaml` file from `/launcher/config/rosapi` into the new directory:

   ```
   /Users/rosette-users/group1/config/rosapi/rex-factory-config.yaml
   ```

5. Edit the copied file, setting the `dataOverlayDirectory` parameter and adding the path for the new regex file. The overlay directory is a directory shaped like the `data` directory. The entities endpoint will look for files in both locations, preferring the version in the overlap directory.

```
dataOverlayDirectory: "/Users/rosette-users/group1/custom-rex/data"

supplementalRegularExpressionPaths:
- "/Users/rosette-users/group1/custom-rex/data/regex/eng/accept/supplemental/custom-regexes.xml"
```

6. Create the file `custom-regexes.xml` in the `/Users/rosette-users/group1/custom-rex/data/regex/eng/accept/supplemental` directory.

```
<regexps>
  <regexp type="COLOR">(?i)red|white|blue|black</regexp>
  <regexp type="ANIMAL">(?i)bear|tiger|whale</regexp>
</regexps>
```

7. Call the entities endpoint without using the custom profile:

```
 curl -s -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "Cache-Control: no-cache" \
    -d '{"content": "The black bear fought the white tiger at London Zoo." }' \
    "http://localhost:8181/rest/v1/entities"
```

The only entity returned is London Zoo:

```
{
  "entities": [
    {
      "type": "LOCATION",
      "mention": "London Zoo",
      "normalized": "London Zoo",
      "count": 1,
      "mentionOffsets": [
        {
          "startOffset": 41,
          "endOffset": 51
        }
      ],
      "entityId": "T0"
    }
  ]
}
```

8. Call the entities endpoint, adding the **profileId** to the call:

```
curl -s -X POST \ -H "Content-Type: application/json" \
-H "Accept: application/json" \
-H "Cache-Control: no-cache" \
-d '{"content": "The black bear fought the white tiger at London Zoo.",
     "profileId": "group1"}' \
"http://localhost:8181/rest/v1/entities"
```

The new colors and animals are also returned:

```
"entities": [
    {
        "type": "COLOR",
        "mention": "black",
        "normalized": "black",
        "count": 1,
        "mentionOffsets": [
            {
                "startOffset": 4,
                "endOffset": 9
            }
        ],
        "entityId": "T0"
    },
    {
        "type": "ANIMAL",
        "mention": "bear",
        "normalized": "bear",
        "count": 1,
        "mentionOffsets": [
            {
                "startOffset": 10,
                "endOffset": 14
            }
        ],
        "entityId": "T1"
    },
    {
        "type": "COLOR",
        "mention": "white",
        "normalized": "white",
        "count": 1,
        "mentionOffsets": [
            {
                "startOffset": 26,
                "endOffset": 31
            }
        ],
        "entityId": "T2"
    },
    {
        "type": "ANIMAL",
        "mention": "tiger",
        "normalized": "tiger",
        "count": 1,
        "mentionOffsets": [
            {
                "startOffset": 32,
                "endOffset": 37
            }
        ],
        "entityId": "T3"
    },
    {
        "type": "LOCATION",
        "mention": "London Zoo",
        "normalized": "London Zoo",
        "count": 1,
        "mentionOffsets": [
            {
                "startOffset": 41,
                "endOffset": 51
            }
        ],
        "entityId": "T4"
    }
```

13. Usage tracking

Usage tracking provides metrics on all Rosette server calls. Call counts are provided by app-id, profileId, endpoint, and language.

- Application ids [41] (`app-id`) are an optional way to identify the application or group making the call. The `app-id` is the value of `X-RosetteAPI-App-ID` in the call header. If no application id is provided in the call header, the calls are allocated to the `no-app-id` group.
- Profile ids (`profileId`) are an optional way of identifying a custom profile [32]. Each profile can have its own data domain, parameter, and configuration settings. If no profile id is provided in the call, the calls are allocated to the `no-profile-id` group.
- Language is identified by the 3-letter ISO 639-3 language code. `xxx` indicates the language was unknown.
- Calls made to the endpoints `/rest/v1/info`, `/rest/v1/ping`, and `/rest/v1/custom` are not included in the statistics.

Call statistics are kept in the file `launcher/config/rosette-usage.yaml`. The statistics are cumulative from the file creation date. The file is created when the server is started. If the file already exists when the server is started, new statistics are added to the existing file. The file is not deleted when the server is stopped.

13.1. Usage

To access the statistics, call the `usage` endpoint:

```
curl http://localhost:8181/rest/usage
```

where `localhost:8181` is the location of the Rosette installation.

Sample Response

```
{"no-app-id": {
   "no-profile-id": {
      "/rest/v1/tokens": {
        "eng": {
            "calls": 1
        },
        "zho": {
            "calls": 1
        }
      },
      "/rest/v1/categories": {
        "eng": {
            "calls": 1
        }
      },
      "/rest/v1/language": {
        "xxx": {
            "calls": 1
        }
      }
    }
  }
}
```

You can also aggregate usage data by having Prometheus pull metrics from multiple instances using the `usage/metrics` endpoint. A single call returns all endpoints.

```
curl http://localhost:8181/rest/usage/metrics
```

where `localhost:8181` is the location of the Rosette installation.

Sample Response

```
# HELP rosette_http_requests_total Total number of Rosette Enterprise requests processed.
# TYPE rosette_http_requests_total counter
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/tokens",lang="zho",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/semantics/vector",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/morphology/compound-components",lang="deu",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/syntax/dependencies",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/morphology/lemmas",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/topics",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/transliteration",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/sentences",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/address-similarity",lang="xxx",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/name-deduplication",lang="xxx",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/morphology/complete",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/entities",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/name-translation",lang="xxx",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/morphology/parts-of-speech",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/semantics/similar",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/language",lang="xxx",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/sentiment",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/categories",lang="eng",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/name-similarity",lang="xxx",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/morphology/han-readings",lang="zho",} 1.0
rosette_http_requests_total{app_id="no-app-id",profile_id="no-profile-id",
    endpoint="/rest/v1/relationships",lang="eng",} 1.0
```

13.2. Configuration parameters

The configuration parameters for usage tracking is in the file `launcher/config/com.basistech.ws.local.usage.tracker.cfg`.

- **Disable Tracking** By default, usage tracking is turned on. To disable tracking, uncomment the `enabled` parameter and change the value to false:

```
enabled: false
```

- **Report interval** To set the reporting interval in minutes, change the `reportInterval` parameter. The default is 1 minute.

```
reportInterval: 2
```

- **File Location** To set the location for the `rosette-usage.yaml` file set the `usage-tracker-root` parameter. The default location is `<rosette>/server/launch/config`. Uncomment the line and change it to your preferred location. This example changes it to the `/var/log` directory:

```
usage-tracker-root: /var/log
```

13.3. Resetting the counter

To reset the counter:

1. Stop the server
2. Remove the following files:
   - `launcher/config/rosette-usage.yaml`
   - `launcher/config/rosette-usage.yaml.backup`
3. Restart the server

13.4. Identifying an application

No authorization is required when using an on-premise installation of Rosette. You may, however, want to track Rosette calls by groups within your organization. To do this, include an application id (`app-id`) in the request header of all Rosette calls. This allows Rosette to track usage [39] by `app-id`.

An application id is:

- A user-defined string.
- It is defined in the call.
- There is no validation or authorization on the value.
- Used for usage tracking only.
- If no `app-id` is included in the header, calls are allocated to the `no-app-id` group.

Example:

```
curl -s -X POST \
    -H "X-RosetteAPI-App-Id: usergroup1" \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "Cache-Control: no-cache" \
    -d '{"content": "Por favor Señorita, says the man." }' \
    "https://localhost:8181/rest/v1/language"
```

13.5. Associating custom profiles with application ids

NOTE

As of the 1.20.0 release of Rosette Server, custom profiles are no longer associated with application ids. All custom profiles are now available to all application ids without requiring file duplication.

To require custom profiles to be installed under the application group, add the following line to the `wrapper.conf` file:

```
wrapper.java.additional.250=-Drosapi.feature.CUSTOM_PROFILE_UNDER_APP_ID
```

When tracking usage in installations which utilize custom profiles, you can require that the custom profiles must be installed in subdirectories under each application group.

Example:

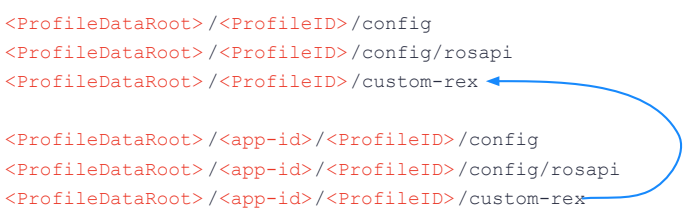
The `app-id` identifies the group making the call. The `app-id` is the value of `X-RosetteAPI-App-ID` in the call header. If no application id is provided in the call header, the calls are allocated to the `no-app-id` group.

The `ProfileId` identifies the customized components

```
<app-id>/<ProfileID>/config
<app-id>/<ProfileID>/config/rosapi
<app-id>/<ProfileID>/custom-rex
```

If you have multiple groups (`app-id`) using the same set of configuration and customizations, you can create symbolic links (`ln -s <source> <link>` to save disk space. Otherwise, you will have to install the customizations under each `app-id`.

Example directories and links



14. Custom endpoints

A custom endpoint combines business logic and calls to Rosette endpoints into a new endpoint that is fully integrated in Rosette Server. With a single call, you can make calls to multiple Rosette endpoints, follow specific workflows, and execute custom logic.

14.1. Architecture

The custom endpoint architecture includes two platform-independent components:

- A reverse proxy that is placed in front of the normal entry point. The shipment includes a Spring Boot application that utilizes Spring Cloud Gateway, an open source routing framework, in order to provide a reverse proxy capability.
- The custom endpoint is a Spring Boot application and can either be deployed as a web archive (WAR) using Tomcat or as a stand-alone Spring Boot application.

NOTE
Spring Cloud Gateway and Tomcat are provided as an example of one way to implement a custom endpoint. You can substitute your preferred reverse proxy or application server to achieve the same results.

The reverse proxy's configuration file routes all incoming requests:

- Calls to standard Rosette endpoints are routed to the Rosette server as usual.
- Calls to custom endpoints are routed to the application server, which makes calls to standard Rosette endpoints and applies custom logic.

14.2. Installing

All shipments contain the file `rosent-custom-endpoint-installer-<version>.tar.gz` containing an installation script and the files for the reverse proxy and application server. The installation script is currently for Linux and macOS only.

You can install the custom endpoints while Rosette Server is running, or start it after you complete the custom endpoints installation.

1. Install Rosette Server as usual. Ensure `JAVA_HOME` is set.
2. Download and extract `rosent-custom-endpoint-installer-<version>.tar.gz` from your shipment email. The files will be extracted into a directory named `rosent-custom-endpoint-installer-<version>`.
3. Run `rosent-custom-endpoint-installer.sh`.

```
./rosent-custom-endpoint-installer.sh
```

4. The installer will guide you through the installation and customization process.
   a. You will have the option of installing tomcat to host the custom endpoints.
   b. You will have the option to install the stand-alone sample and source code.
   c. The sample can be deployed as a standalone Spring Boot application or as a ware deployed in tomcat.

14.3. Configuring and running the services

1. Start the application server: `tomcat/apache-tomcat-<version>/bin/rosent-tomcat.sh start`
   - To start the service:

     ```
     tomcat/apache-tomcat-<version>/bin/rosent-tomcat.sh start
     ```

     The following messages are displayed when the application server starts up:

     ```
     Starting RosetteEnterprise Tomcat...
     Waiting for RosetteEnterprise Tomcat......
     running: PID:76987
     ```

   - To stop the service:

     ```
     tomcat/apache-tomcat-<version>/bin/rosent-tomcat.sh stop
     ```

   - To get the status of the service:

     ```
     tomcat/apache-tomcat-<version>/bin/rosent-tomcat.sh status
     ```

   - To view the console logs:

     ```
     tomcat/apache-tomcat-<version>/bin/rosent-tomcat.sh console
     ```

2. Start the proxy server: `proxy/bin/rosent-proxy.sh start`
   - To start the service:

     ```
     proxy/bin/rosent-proxy.sh start
     ```

     The following messages are displayed when the proxy server starts up:

     ```
     Starting RosetteEnterprise Proxy...
     Waiting for RosetteEnterprise Proxy......
     running: PID:77560
     ```

   - To stop the service:

     ```
      proxy/bin/rosent-proxy.sh stop
     ```

- To get the status of the service:

```
proxy/bin/rosent-proxy.sh status
```

- To view the console logs:

```
proxy/bin/rosent-proxy.sh console
```

3. Start Rosette Server [12]

14.4. Configuring a custom endpoint

The `application.yml` file configures most aspects of the proxy including ports, certificates, routing, and timeouts. If you move the `application.yml` file, the `./conf/proxy-wrapper.conf` must be updated with the new location.

```
wrapper.app.parameter.2=--spring.config.location=<new file location>
```

The proxy can be configured using the values in the `application.yml` file directly or by setting the environment variables in `./conf/proxy-wrapper.conf`.

- To set the port the proxy will use: s

```
set.PROXY_PORT=8182
```

- To set the location logs are written to: set.PROXY_LOG_DIR=./logs

```
set.PROXY_LOG_DIR=./logs
```

- To set the location of the custom application: set.SAMPLE_ENDPOINT_APP_HOST=http:/localhost:8183

```
SAMPLE_ENDPOINT_APP_HOST=http:/localhost:8183
```

- The location of Rosette Server:

```
set.ROSETTE_SERVER_HOST=http://localhost:8181
```

If your custom code is in Java, put your WAR file into the `webapps` directory of the Tomcat installation directory. To use other languages, see Adding non-Java custom code [48].

Your Java web application should define a URI to access the code. Create a rule in the proxy definition to adjust the final path to access the custom code.

14.5. Resiliency

Each component, including Rosette Server, has its own, independent, Tanuki Java Wrapper application. Tanuki monitors the health of the application and relaunches if the Java application crashes or gets into a bad state.

In addition, the proxy application exposes two endpoints that can be used to access the proxy health:

- `/info`: Returns a version of the running proxy and a short description. Example:

```
curl http://localhost:8182/info
```

Returns:

```
{
  "app": {
    "name": "rosent-proxy",
    "description": "Rosette Custom Endpoint Proxy",
    "version": "0.3.0",
  }
}
```

- `/health`: Indicates if the proxy is UP or Down. Example:

```
curl http://localhost:8182/health
```

Returns:

```
{
"status": "UP"
}
```

NOTE
Access to these endpoints can be restricted to network interface if required. Details on restricting access can be obtained from BasisTech support or Spring Boot Actuator documentation, by setting the management server address and port.

14.6. Example

NOTE
The source code for the example application is located in the directory `/rosette/custom-endpoint`. You do not need to extract the file to run the example application, as there is a compiled and packaged version of the code deployed with the Tomcat application server. You only need to access the source code if you want to use it as a template for your own endpoint.

The custom endpoint application includes an example application, **matchSentences** , which calculates the relevance of documents and sentences against a list of keywords. It combines calls to the `/sentences` and `/semantics/vector` endpoints with custom code. The custom code uses the cosine similarity function to calculate similarity scores between the sentences and the keywords, as well as comparing the scores to the input threshold value.

The input is a list of keywords, a document or the URL to a document, and a threshold.

The response is the cosine similarity scores and a true/false value indicating if the score is above the threshold for each sentence, and for the entire document.

14.6.1. Call the endpoint

Example payload

File: `matchSentences.json`

```
{
    "keywords": [
       "USA",
       "Iran",
       "attack",
       "protest"
    ],
    "document": "Iran was planning attacks on    four US embassies
     when its top general was killed, President Donald Trump says.
     When asked what threat led to last Friday's US drone strike,
     he told Fox News: \"I can reveal that I believe it probably
     would've been four embassies.\" \"The killing of Gen Qasem Soleimani,
     a national hero, came after days of protests at the US embassy in Baghdad.",
   "threshold": 0.25
}
```

Call `/matchSentences`

```
curl -s http://localhost:8182/rest/v1/match-sentences -XPOST \
-H 'Content-Type: application/json; charset=utf-8' -d \
@matchSentences.json
```

14.6.2. Example configuration

Spring Cloud Gateway is used for proxy routing. There are 2 routes defined by `/rest/v1/match-sentences` and `/rest/v1/**`. All traffic sent to `/rest/v1/match-sentences` will be sent to the custom application.

The custom application will change depending on how the application is being run.

- When run as a Spring Boot application the path is `/rest/v1/match-sentences -> /service/matchSentences`
- When deployed as a war the path is `/rest/v1/match-sentences -> /custom-endpoint/matchSentences`.
- The paths to `/info` endpoints is either `/service/info` or `/custom-endpoint/info`.

```
spring:
  cloud:
    gateway:
      routes:
        # Route to the custom endpoint
        - id: sample_app_route
          uri: ${SAMPLE_ENDPOINT_APP_HOST:http://localhost:8183}
          predicates:
            - Path=/rest/v1/match-sentences/**,matchTrailingSlash=false
          filters:
# When running the custom endpoint as a stand-alone spring boot application
#           - RewritePath=/rest/v1/match-sentences/info(?<segment>.*), /service/info$\{segment}
#           - RewritePath=/rest/v1/match-sentences/health(?<segment>.*), /service/health$\{segment}
#           - RewritePath=/rest/v1/match-sentences/prometheus(?<segment>.*), /service/prometheus$\{segment}
#           - RewritePath=/rest/v1/match-sentences/env(?<segment>.*), /service/env$\{segment}
#           - RewritePath=/rest/v1/match-sentences(?<segment>.*), /service/matchSentences/$\{segment}
# When running the custom endpoint as a war file in tomcat
#           - RewritePath=/rest/v1/match-sentences/info(?<segment>.*), /custom-endpoint/info$\{segment}
#           - RewritePath=/rest/v1/match-sentences/health(?<segment>.*), /custom-endpoint/health$\{segment}
#           - RewritePath=/rest/v1/match-sentences/prometheus(?<segment>.*), /custom-endpoint/prometheus$\{segment}
#           - RewritePath=/rest/v1/match-sentences/env(?<segment>.*), /custom-endpoint/env$\{segment}
#           - RewritePath=/rest/v1/match-sentences(?<segment>.*), /custom-endpoint/matchSentences/$\{segment}
      # Calls to Rosette Server Endpoints
      - id: rosette_pass_thru
        uri: ${ROSETTE_SERVER_HOST:http://localhost:8181}
        predicates:
          - Path=/rest/v1/**,matchTrailingSlash=false
          - Path=/rest/**,matchTrailingSlash=false
```

14.7. Adding non-Java custom code

The examples and instructions here assume you are writing your custom code in Java. If you are using a different language, you will need to set up your own application server that is specific for that language. Then you add your own routing rules to the proxy so that the traffic is directed to the correct place.

Let's assume you'd like to add a python file named `python_example.py,and` you want to execute it from the url `http://localhost:8182/rest/v1/python_example`.

1. Stand up a python application server.
2. Configure the application server to work with the `python_example.py` file.
3. Create a rule in the proxy's `application.yml` file that routes requests to `rest/v1/python_example` to the python application server such that it calls the `python_example.py` file.

15. Customizing entity extraction in Rosette Server

The Rosette Server endpoints are configured by the files found in the `/launcher/config/rosapi` directory. Be careful when editing any of these files as the endpoints will not work if not configured properly.

The entity extraction and linking parameters are in the file `rex-factory-config.yaml`.

For details on how Rosette entity extraction works and how to customize it, refer to the *Rosette Entity Extractor Application Developers Guide* included in the Rosette Server package.

15.1. Entity Extraction Parameters

To modify the default configuration of the **/entities** endpoint, edit the file `/launcher/config/rosapi/rex-factory-config.yaml`. The file contains the following parameters. To change the value of a parameter, uncomment the parameter and set the new value.

Parameter	Description	Default
`rootDirectory`	A REX root directory contains language models and necessary configuration files.	`${rex-root}`
`rblRootDirectory`	The directory containing the RBL root for REX to use.	`${rex-root}/rbl-je`
`rejectGazetteers`	Additional gazetter files used to reject entities for the given language.	`null`
`snapToTokenBoundaries`	Regular expressions and gazetteers may be configured to match tokens partially independent from token boundaries. If true, reported offsets correspond to token boundaries.	`true`
`rejectRegularExpressionSets`	Additional regex files used to reject entities.	`null`
`allowPartialGazetteerMatches`	The option to allow partial gazetteer matches. For the purposes of this setting, a partial match is one that does not line up with token boundaries as determined by the internal tokenizer. This only applies to accept gazetteers.	`false`
`kbs`	Custom list of Knowledge Bases for the linker, in order of priority	`null`
`maxEntityTokens`	The maximum number of tokens allowed in an entity returned by Statistical Entity Extractor. Entity Redactor discards entities from Statistical Entity Extractor with more than this number of tokens.	`8`
`customProcessors`	Custom processors to add to annotators.	`null`
`acceptRegularExpressionSets`	Additional files used to produce regex entities.	`null`
`calculateConfidence`	If true, entity confidence values are calculated. Can be overridden by specifying `calculateConfidence` in the API call.	`false`
`joinerRuleFiles`	File containing additional joiner rules.	`null`
`excludedEntityTypes`	Entity types to be excluded from extraction.	`null`
`dataOverlayDirectory`	An overlay directory [51] is a directory shaped like the `data` directory. REX will look for files in both the overlay directory and the root directory, using files from both locations. However, if a file exists in both places (as identified by its path relative to the overlay or root data directory), REX prefers the version in the overlay directory. If REX finds a zero-length file in the overlay directory, it ignores both that file and any corresponding file in the root data directory.	`null`
`confidenceThreshold`	The confidence value threshold below which entities extracted by the statistical processor are ignored.	`-1.0`
`resolvePronouns`	When true, resolve pronouns to person entities.	`false`
`statisticalModels`	Additional files used to produce statistical entities for the given language. You may pass multiple statistical models. The parameter should be formatted in trios of values specfying language, case-sensitivity and the model file, separated by commas. Case-sensitivity can be `automatic`, `caseInsensitive` or `caseSensitive`. For example, setting two models for case-sensitive English and Japanese might look like : `eng,caseSensitive,english-model.bin,jpn,automatic,japanese-model.bin`	`null`
`acceptGazetteers`	Additional gazetteer files used to produce entities for the given language.	`null`

Parameter	Description	Default
`linkEntities`	The option to link mentions to knowledge base entities with disambiguation model. Enabling this option also enables `calculateConfidence`.	`false`
`caseSensitivity`	The capitalization (aka 'case') used in the input texts. Processing standard documents requires caseSensitive, which is the default. Documents with all-caps, no-caps or headline capitalization may yield higher accuracy if processed with the caseInsensitive value. Can be `automatic`, `caseSensitive` or `caseInsensitive`	`caseSensitive`
`maxResolvedEntities`	The maximum number of entities for in-document coreference resolution (a.k.a. chaining).	`2000`
`calculateSalience`	If true, entity chain salience values are calculated. Can be overridden by specifying `calculateSalience` in the API call.	`false`
`retainSocialMediaSymbols`	The option to retain social media symbols ('@' and '#') in normalized output	`false`
`statSalienceMode`	An option to calculate entity-chain salience with statistical-based calculation (returns 0 or 1) or simple calculation (returns score between 0 and 1)	`true`
`customProcessorClasses`	Register a custom processor class.	`null`
`keepEntitiesInInput`	The option to keep existing annotated text entities.	`false`
`redactorPreferLength`	The option to prefer length over weights during redaction. If true, the redactor will always choose a longer entity over a shorter one if the two overlap, regardless of their user-defined weights. In this case, if the lengths are the same, then weight is used to disambiguate the entities. If false, the redactor will choose the higher weighted entity when two overlap, regardless of the length of the entity string. In this case, if the weights are the same, then the redactor will choose the longer of the two entities.	`true`
`useDefaultConfidence`	The option to assign default confidence value 1.0 to non-statistical entities instead of null.	`false`
`linkingConfidenceThreshold`	The confidence value threshold below which linking results by the kbLinker processor are ignored.	`-1.0`
`indocType`	An option for document entity resolution (also known as entity chaining). Valid values are: `HIGH`, `STANDARD`, `STANDARD_MINUS` or `NULL`	`STANDARD`
Default processors: `acceptGazetteer`, `acceptRegex`, `rejectGazetteer`, `rejectRegex`, `statistical` `indocCoref`, `redactor`, `joiner` `processors`	List the set of active processors for an entity extraction run. All processors are active by default. This method provides a way to turn off selected processors. The order of the processors cannot be changed. Note that turning off redactor can cause overlapping and unsorted entities to be returned.	`null`
`supplementalRegularExpressionPaths`	The option to add supplemental regex files, usually for entity types that are excluded by #default. The supplemental regex files are located at `data/regex/<lang>/accept/supplemental` and are not used unless specified.	`null`
`structuredRegionsProcessingType`	Configures how structured regions will be processed. It has three values: `none`, `nerModel`, and `nameClassifier`.	`none`
`regexCurrencySplit`	Determines if money values should be extracted as `MONEY` or `CURRENCY_AMT` and `CURRENCY_TYPE`. If true, REX tries to extract `CURRENCY` instead of `MONEY`.	`false`

15.2. Overlay Data Directory

If your project has a set of unique data files that you would like to keep separate from other data files, you can put them in their own directory, also known as an overlay directory. This is an additional data directory, which takes priority over the default REX data directory.

The overlay directory must have the same directory tree as the provided `data` directory. If an overlay directory is set, REX searches both it and the default `data` directory.

- If a file exists in both places, the version in the overlay directory is used.
- If there is an empty file in the overlay directory, REX will ignore the corresponding file in the default `data` directory.
- If there is no file in the overlay directory, REX will use the file in the default directory.

To specify the overlay directory use:

1. Create an overlay directory:

```
<install-directory>/my-data
```

2. Add the overlay directory to the `rex-factory-config.yaml` file:

```
dataOverlayDirectory:
  <install-directory>/my-data
```

Turn Off a Specific Language Gazetteer

1. Create an overlay directory:
2. Add an empty file (`gaz-LE.bin`) to the overlay directory:

```
my-data/gazetteer/eng/accept/gaz-LE.bin
```

3. Add the overlay directory to the `rex-factory-config.yaml` file:

```
dataOverlayDirectory:
  <install-directory>/my-data
```

The default English gazetteer will not be used in calls.

Use a Custom German Reject Gazetteer

In the above example, add a reject gazetteer file:

```
my-data/gazetter/deu/reject/reject-names.txt
```

15.3. Default configuration

The default configuration of the **/entities** endpoint uses the same default values as the REX Java SDK. It is optimized to be more performance-oriented, with fewer options enabled, than the configuration of Rosette Cloud. Rosette Cloud is configured to provide a fully-functional demonstration environment. [1]

Server and Cloud Default Parameters

Feature	Rosette Server	Parameter Setting in Rosette Server	Rosette Cloud
Entity Linking	false (disabled)	`linkEntities:false`	true (enabled)
Regular Expression Files	no files are loaded	`supplementalRegularExpressionPaths null`	all files are loaded
Pronominal Resolution	false (disabled)	`resolvePronouns: false`	true (enabled)
Case Sensitivity	case-sensitive	`caseSensitivity: caseSensitive`	automatic
Structured Regions	none	`structuredRegionsProcessingType: none`	`nerModel`

15.3.1. Entity linking

Text that refers to an entity is called an entity mention, such as “Bill Clinton” and “William Jefferson Clinton”. Rosette connects these two entity mentions with entity linking, since they refer to the same real-world PERSON entity. Linking helps establish the identity of the entity by disambiguating common names and matching a variety of names, such as nicknames and formal titles, with an entity ID.

Rosette uses the Wikidata knowledge base as a base to link Person, Location, and Organization entities. If the entity exists in Wikidata, then Rosette returns the Wikidata QID, such as Q1 for the Universe. If Rosette cannot link the entity, then it creates a placeholder temporary (“T”) entity ID to link mentions of the same entity in the document. However, the TID may be different across documents for the same entity.

Rosette supports linking to other knowledge bases, specifically the DBpedia ontology and the Thomson Reuters PermID. You can also link to one or more custom knowledge bases. Contact BasisTech Support for more information on using the Field Training Kit (FTK) to add your own linking knowledge base.

Entity linking in Rosette Server is off by default, to improve call speed. When entity linking is turned off, Rosette returns the entities with a TID.

You can enable entity linking in Rosette Server for a single call or as a system default in your environment.

- Per-call: add `{"options": {"linkEntities": true}}` to your call.
- Default: edit the `/launcher/config/rosapi/rex-factory-config.yaml` file as shown below:

```
#The option to link mentions to knowledge base entities with #disambiguation model.
#Enabling this option also enables calculateConfidence.
linkEntities: true
```

By default, linking to DBpedia is turned off. To turn it on:

- Enable entity linking
- Add `{"options": {"includeDBpediaTypes": true}}` to the call

The list of knowledge bases can be customized in the `rex-factory-config.yaml` file with the `kbs` parameter, which takes a `List` of `Path`s to knowledge bases.

```
kbs:
    - /customKBs/kb1
    - /customKBs/kb2
    - /rosette/server/roots/rex/7.44.1.c62.2/data/flinx/data/kb/basis
```

[1]This change became effective in the 1.14.0 (August 2019) version of Rosette Server.

NOTE
Setting the list of knowledge bases completely overwrites the list of knowledge bases the linker uses. If you want the default Wikidata knowledge base to be included, it must be on the list of knowledge bases.

15.3.2. Loading regex files

By default, the supplemental regex files are not loaded when the entity extraction endpoint is loaded. To load either the provided supplemental files or new files, edit the `/launcher/config/rosapi/rex-factory-config.yaml` file adding the `supplementalRegularExpressionPaths` statements to the file, as show below.

TIP
The files are named following the pattern: `data/regex/<lang>/accept/regexes.xml`, where `<lang>` is either the ISO 639-3 language code for the supported language, or `xxx` for all or any languages.

```
#The option to add supplemental regex files, usually for entity types that are excluded by
#default. The supplemental regex files are located at data/regex/<lang>/accept/supplemental and
#are not used unless specified.
supplementalRegularExpressionPaths:
- "${rex-root}/data/regex/eng/accept/supplemental/date-regexes.xml"
- "${rex-root}/data/regex/eng/accept/supplemental/geo-regexes.xml"
```

15.3.3. Pronominal resolution

Entity extraction can try to resolve pronouns with their antecedent entities. For example, in the sentences:

`John Smith lives in Boston. He is originally from New York`

pronominal resolution would resolve `he` with `John Smith`. By default, pronominal resolution is disabled.

To enable it, edit the `/launcher/config/rosapi/rex-factory-config.yaml` file as shown below:

```
#The option to resolve pronouns to person entities.
resolvePronouns: true
```

15.3.4. Case sensitivity

Case sensitivity refers to the capitalization (aka 'case') used in the input texts. Entity extraction can use case to help identify named entities (such as proper nouns) in documents.

Valid values for `caseSensitivity`

- `caseSensitive`: (default) Case found in standard documents, those in which case follows grammar for the most part.

- `caseInsensitive`: Used for documents with all-caps, no-caps, or headline capitalization. These are documents in which capitalization is not a good indicator for named entities.
- `automatic`: Rosette detects the case from the input model and chooses an appropriate model to use.

To change the default case sensitivity, edit the `/launcher/config/rosapi/rex-factory-config.yaml` file as shown below:

```
#The capitalization (aka 'case') used in the input texts. Processing standard documents
#requires caseSensitive, which is the default. Documents with all-caps, no-caps or headline
#capitalization may yield higher accuracy if processed with the caseInsensitive value.
caseSensitivity: automatic
```

16. Customizing the morphology and sentences endpoints

The Rosette Server endpoints are configured by the files found in the `/launcher/config/rosapi` directory. Be careful when editing any of these files as the endpoints will not work if not configured properly.

The morphology-specific parameters and settings are in the file: `rbl-factory-config.yaml`. The sentences endpoint uses the same configuration file.

16.1. Fragment boundary detection

In cases where a document or part of a document contains tables and lists, instead of sentences, the /sentences endpoint can detect fragment boundaries as sentence boundaries. One way fragment boundaries are identified is by encountering fragment delimiters. A delimiter is restricted to one character and the default delimiters are U+0009 (tab), U+000B (vertical tab), and U+000C (form feed).

You can modify the set of recognized delimiters:

- Edit the file`/launcher/config/rosapi/rbl-factor-config.yaml`
- Remove the comment from the `fragmentBoundaryDelimiters` parameter
- Edit the parameter values string to contain all values to be recognized as fragment boundaries, including any of the default values you want to keep

In addition to the fragment delimiters, the fragment boundary detector automatically inserts a break:

- After 3+ consecutive spaces
- After 2 new lines
- At the end of the line, when the line has less than 7 tokens
- At the end of the line which contains a previous fragment boundary
- After every newline in a list. A list is defined as 3 or more lines containing the same punctuation mark within the first 5 characters of the line.

By default, fragment boundary detection is turned **on**. To turn off fragment boundary detection:

- Edit the file `/launcher/config/rosapi/rbl-factor-config.yaml`
- Remove the comment from the `fragmentBoundaryDetection` parameter
- Set `fragmentBoundaryDetection: false`

17. Customizing the language identification endpoint

The `/language` endpoint provides an additional, rule- and model-based algorithm that is more accurate than the regular algorithm for short inputs. By default, the short-string threshold is 0 and short-string language detection is inactive. To turn it on, set the threshold to a non-negative integer, such as 20. If the string contains fewer characters than this threshold, the `/language` endpoint will perform short-string language detection.

To enable the short-string algorithm, edit the `rli-factory-config.yaml` file and set `shortStringThreshold` to your preferred value.

```
shortStringThreshold: 20
```

18. Customizing categorization and sentiment endpoints

The Rosette Server endpoints are configured by the files found in the `/launcher/config/rosapi` directory. Be careful when editing any of these files as the endpoints will not work if not configured properly.

The endpoint-specific parameters and settings are in the files of the format: `cat-factory-config.yaml` and `sent-factory-config.yaml`.

The `worker-config.yaml` file configures the pipeline for each endpoint. The entries in this file are highly dependent on the backend code.

18.1. Adding new models for categorization and sentiment

The Rosette Classification Field Training Kit allows user to train their own classification models for the `/categories` and `/sentiment` endpoints. Reasons for training a new model include:

- Supporting a language that Rosette does not currently support
- Increasing accuracy on your particular input data
- Supporting a specific categorization taxonomy for your data or task.

See the *Training Classification Models with Rosette* publication for more information.

18.1.1. Integrating Your Custom Model with Rosette Server

To deploy your custom-trained model, integrate it into Rosette Server as follows:

- Ensure that, for the language your are targeting, the following directory exists: `${tcat-root}/models/<lang>/combined-iab-qag`

- Move any existing model files in the target directory to an `unused` directory, e.g.

```
> mkdir ${tcat-root}/models/<lang>/unused
> mv ${tcat-root}/models/<lang>/combined-iab-qag/* ${tcat-root}/models/<lang>/unused
```

- Copy all the model files from your newly trained model into your target directory, `${tcat-root}/models/<lang>/combined-iab-qag`
- Relaunch the Rosette Server server

After relaunching the Rosette Server, the `categorization` endpoint will use the models in the `combined-iab-qag` directory, therefore using your new model for the language of the newmodel.

For sentiment model integration, place your model files into `${sentiment-root}/data/svm/<lang>/` and use the `sentiment` endpoint. Similarly, move all existing files for that model to a backup directory before copying over the new files.

NOTE

Note that depending on your specific FTK version, your newly created model may have a `lexicon_filtered` file while the existing model has `lexicon.filtered` instead. Rosette supports both naming schemes for backwards compatibility. Regardless of which naming scheme you see, you should remove the existing filtered lexicon file before adding the one from your new model. If both `lexicon.filtered` and `lexicon_filtered` files are in the same model directory, `lexicon.filtered` will take precedence.

18.1.2. Adding new language models

Out of the box, the `/sentiment` and `/categories` endpoints only support the languages of the models that ship with the distribution. Once you have trained a model in a new language, you must add the new languages to the `transport-rules.tsv` and `worker-config.yaml` files in Rosette Enteprise.

- For both endpoints, edit the `transport-rules.tsv` file. Each endpoint is listed, with a `lang=` statement listing the supported languages for the endpoint. Add the three letter ISO 693-3 language code for the new model languages.

```
/categories lang=eng
/sentiment  lang=ara|eng|fas|fra|jpn|spa
```

- For the `/sentiment` endpoint only, edit the `worker-config.yaml` file. Go to the section labeled `textPipelines`. Each endpoint is listed with a `languages:` statement listing the supported languages for the endpoint. Add the three letter ISO 693-3 language code for the new model languages.

```
# sentiment
- endpoint: /sentiment
  languages: [ 'ara', 'eng', 'fas', 'fra', 'jpn', 'spa' ]
  steps:
  - componentName: entity-extraction
  - componentName: sentiment
```

18.2. Configuring the sentiment endpoint for document-level analysis

The sentiment analysis endpoint can be configured to return document-level sentiment analysis only, by turning off entity-level sentiment analysis. This requires modifying the `worker-config.yaml` file to remove the entity extraction step from the process. This will speed up document-level sentiment analysis.

The following edits are made to the `worker-config.yaml` file. The shipped version of the file is:

```
# sentiment
- endpoint: /sentiment
  languages: [ 'ara', 'eng', 'fas', 'fra', 'jpn', 'spa' ]
  steps:
  - componentName: entity-extraction
  - componentName: sentiment
```

Change the above block to:

```
# sentiment
- endpoint: /sentiment
  languages: [ 'ara', 'eng', 'fas', 'fra', 'jpn', 'spa' ]
  steps:
  - componentName: base-linguistics
    factoryName: tokenize
  - componentName: sentiment
```

The entity extraction endpoint must be replaced by the tokenization endpoint in the pipeline.

19. Troubleshooting

19.1. Common Rosette Server error codes

The following error codes are returned by Rosette when accessing an endpoint.

Error Code	Meaning	Description
400	Bad Request	There was something wrong with your request, or the language was not supported by the endpoint.
403	Forbidden	Access denied. This could be based on plan, limits, or profiles. [a]
404	Not Found	The specified URI could not be found
405	Method Not Allowed	You tried to access Rosette with an invalid method
409	Incompatible Client Version	Your binding was out of date. Please update to the latest version.
413	Too Much Data	Your payload was bigger than the size limit of 600KB, or 50K characters.
429	Too Many Requests	Slow down! Rosette can only process one call at a time. [a]
500	Internal Server Error	There is a problem with your server.

[a] applies to cloud instances only

19.2. Troubleshooting 400 errors

There can be multiple causes of a 400 error. Listed here are some common issues customers have encountered.

400 Bad Request Format Error possible causes:

- You are passing in text in a language that the endpoint doesn't support. Try setting the `language` JSON request parameter to the three-letter code of the language of your text, and make sure the endpoint supports it by checking the documentation. You can also check the `X-RosetteAPI-ProcessedLanguage` header in the response to see what language Rosette thinks your text is in.
- You've passed in an empty string or an invalid character.
- You didn't pass in required parameters, like a `targetLanguage` to the /name-translation endpoint, or either `content` or `contentUri` to others.

400 NonWorkingURI Returned from Valid URL

Sometimes sophisticated sites block our URL text extractor, thinking it's a bot. The workaround is to copy the text from the webpage and pass it as text in the `'content'` parameter.

19.3. "Language `xxx` not supported"

This error is raised If no language is specified in a call, and Rosette cannot detect the language. `xxx` is the language code returned when an endpoint cannot determine the language.

20. Customizing the language identification endpoint

The `/language` endpoint provides an additional, rule- and model-based algorithm that is more accurate than the regular algorithm for short inputs. By default, the short-string threshold is 0 and short-string language detection is inactive. To turn it on, set the threshold to a non-negative integer, such as 20. If the string contains fewer characters than this threshold, the `/language` endpoint will perform short-string language detection.

To enable the short-string algorithm, edit the `rli-factory-config.yaml` file and set `shortStringThreshold` to your preferred value.

```
shortStringThreshold: 20
```

21. FAQs

21.1. How can I increase the maximum character count and maximum payload per call?

The limits for the input parameters are in the file `/rosapi/constraints.yaml`. See Modify the input constraints [26] for more information.

21.2. How do I find out what's been fixed in newer releases?

Release notes are posted on our support site at https://support.rosette.com/hc/en-us/articles/360018354971-Release-Notes .

21.3. How do I get the latest version?

Requests for the latest version should be sent to support@rosette.com.