TikaServer

Introduction to Tika server

This page is documentation on accessing Tika as a RESTful API via the Tika server (tika-server module). See TikaServer in Tika 2.x for how to configure `tika-server`. See TikaServerEndpointsCompared for a summary of differences across the endpoints.

Installation of Tika Server

The current installation process for Tika server post 1.23 and prior to 1.24 is a bit in flux. Read on below for some options:

Building from source

If you need to customize Tika server in some way, and/or need the very latest version to try out a fix, then to build from source:

1. Checkout the source from SVN as detailed on the Apache Tika contributions page or retrieve the latest code from Github,
2. Build source using Maven
3. Run the Apache Tika JAXRS server runnable jar.

```
git clone https://github.com/apache/tika.git tika-trunk
cd ./tika-trunk/
mvn install
cd ./tika-server/target/
java -jar tika-server-x.x.jar
```

Remember to replace x.x with the version you have built.

Running the Tika Server as a Jar file

The Tika Server binary is a standalone runnable jar. Download the latest stable release binary from the Apache Tika downloads page, via your favorite local mirror. You want the *tika-server-1.x.jar* file, e.g. *tika-server-1.24.jar*

You can start it by calling java with the `-jar` option, eg something like `java -jar tika-server-1.24.jar`

You will then see a message such as the following:

```
$ java -jar tika-server-1.24-SNAPSHOT.jar
19-Jan-2015 14:23:36 org.apache.tika.server.TikaServerCli main
INFO: Starting Apache Tika 1.8-SNAPSHOT server
19-Jan-2015 14:23:36 org.apache.cxf.endpoint.ServerImpl initDestination
INFO: Setting the server's publish address to be http://localhost:9998/
19-Jan-2015 14:23:36 org.slf4j.impl.JCLLoggerAdapter info
INFO: jetty-8.y.z-SNAPSHOT
19-Jan-2015 14:23:36 org.slf4j.impl.JCLLoggerAdapter info
INFO: Started SelectChannelConnector@localhost:9998
19-Jan-2015 14:23:36 org.apache.tika.server.TikaServerCli main
INFO: Started
```

Which lets you know that it started correctly.

You can specify additional information to change the host name and port number:

```
java -jar tika-server-x.x.jar --host=intranet.local --port=12345
```

Once the server is running, you can visit the server's URL in your browser (eg `http://localhost:9998/`), and the basic welcome page will confirm that the Server is running, and give links to the various endpoints available.

Below is some basic documentation on how to interact with the services using cURL and HTTP.

Using prebuilt Docker image

Official image for Tika can be found at DockerHub. You can download and start it with:

```
docker run -d -p 9998:9998 apache/tika:<version>
```

Full set of documentation can be found at Github.

Running Tika Server as Unix Service

Shipping in Tika 1.24 is a new Service Installation Script that lets you install Tika server as a service on Linux. This script was heavily influenced by the Apache Solr project's script, so read up on that documentation if you want to customize the script.

Currently the script only supports CentOS, Debian, Red Hat, Suse and Ubuntu Linxu distributions. Before running the script, you need to determine a few parameters about your setup. Specifically, you need to decide where to install Tika server and which system user should be the owner of the Tika files and process

To run the scripts, you'll need the 1.24 (or later) Tika distribution. It will have a `-bin` suffix e.g. `tika-server-1.24-bin.tgz`. Extract the installation script from the distribution via:

```
tar xzf tika-server-1.24-bin.tgz tika-server-1.24-bin/bin/install_tika_service.sh --strip-components=2
```

This will extract the install_tika_service.sh script from the archive into the current directory. If installing on Red Hat, please make sure **lsof** is installed before running the Tika installation script (`sudo yum install lsof`). The installation script must be run as root:

```
sudo bash ./install_tika_service.sh tika-server-1.24-bin.tgz
```

By default, the script extracts the distribution archive into `/opt/tika`, configures Tika to write files into `/var/tika`, and runs Tika as the `tika` user on the default port. Consequently, the following command produces the same result as the previous command:

```
sudo bash ./install_tika_service.sh tika-server-1.24-bin -i /opt -d /var/tika -u tika -s tika -p 9998
```

You can customize the service name, installation directories, port, and owner using options passed to the installation script. To see available options, simply do:

```
sudo bash ./install_tika_service.sh -help
```

Once the script completes, Tika server will be installed as a service and running in the background on your server (on port 9998). To verify, you can do:

```
sudo service tika status
```

Your specific customization to Tika server setup are stored in the `/etc/init.d/tika` file.

Configuring Tika server in 2.x

See TikaServer in Tika 2.x for the details of configuring tika-server in 2.x.

Tika Server Services

All services that take files use HTTP "PUT" requests. When "PUT" is used, the original file must be sent in request body without any additional encoding (do not use multipart/form-data or other containers).

Additionally, TikaResource, Metadata and RecursiveMetadata Services accept POST multipart/form-data requests, where the original file is sent as a single attachment.

Information services (e.g. defined mimetypes, defined parsers etc) work with HTML "GET" requests.

You may optionally specify content type in "Content-Type" header. If you do not specify mime type, Tika will use its detectors to guess it.

You may specify additional identifier in URL after resource name, like "/tika/my-file-i-sent-to-tika-resource" for "/tika" resource. Tikaserver uses this name only for logging, so you may put there file name, UUID or any other identifier (do not forget to url-encode any special characters).

Resources may return following HTTP codes:

- 200 Ok - request completed sucessfully
- 204 No content - request completed sucessfully, result is empty
- 422 Unprocessable Entity - Unsupported mime-type, encrypted document & etc
- 500 Error - Error while processing document

NOTE: Please see TikaServerEndpointsCompared for a quick comparison of the features of some of these endpoints.

Metadata Resource

```
/meta
```

HTTP PUTs a document to the /meta service and you get back "text/csv" of the metadata.

Some Example calls with cURL:

```
$ curl -X PUT --data-ascii @zipcode.csv http://localhost:9998/meta --header "Content-Type: text/csv"
$ curl -T price.xls http://localhost:9998/meta
```

Returns:

```
"Content-Encoding","ISO-8859-2"
"Content-Type","text/plain"
```

Get metadata as JSON:

```
$ curl -T test_recursive_embedded.docx http://localhost:9998/meta --header "Accept: application/json"
```

Or XMP:

```
$ curl -T test_recursive_embedded.docx http://localhost:9998/meta --header "Accept: application/rdf+xml"
```

Get specific metadata key's value as simple text string:

```
$ curl -T test_recursive_embedded.docx http://localhost:9998/meta/Content-Type --header "Accept: text/plain"
```

Returns:

```
application/vnd.openxmlformats-officedocument.wordprocessingml.document
```

Get specific metadata key's value(s) as CSV:

```
$ curl -T test_recursive_embedded.docx http://localhost:9998/meta/Content-Type --header "Accept: text/csv"
```

Or JSON:

```
$ curl -T test_recursive_embedded.docx http://localhost:9998/meta/Content-Type --header "Accept: application
/json"
```

Or XMP:

```
$ curl -T test_recursive_embedded.docx http://localhost:9998/meta/Content-Type --header "Accept: application
/rdf+xml"
```

Note: when requesting specific metadata keys value(s) in XMP, make sure to request the XMP name, e.g. "dc:creator" vs. "Author"

Multipart Support

Metadata Resource also accepts the files as multipart/form-data attachments with POST. Posting files as multipart attachments may be beneficial in cases when the files are too big for them to be PUT directly in the request body. Note that Tika JAX-RS server makes the best effort at storing some of the multipart content to the disk while still supporting the streaming:

```
curl -F upload=@price.xls URL http://localhost:9998/meta/form
```

Note that the address has an extra "/form" path segment.

Tika Resource

```
/tika
```

HTTP PUTs a document to the /tika service and you get back the extracted text in text, html or "body" format (see below). See also the `/rmeta` endpoint for text and metadata of embedded objects.

HTTP GET prints a greeting stating the server is up.

Some Example calls with cURL:

Get HELLO message back

```
$ curl -X GET http://localhost:9998/tika
This is Tika Server. Please PUT
```

Get the Text of a Document

```
$ curl -X PUT --data-binary @GeoSPARQL.pdf http://localhost:9998/tika --header "Content-type: application/pdf"
$ curl -T price.xls http://localhost:9998/tika --header "Accept: text/html"
$ curl -T price.xls http://localhost:9998/tika --header "Accept: text/plain"
```

Use the Boilerpipe handler (equivalent to tika-app's `--text-main`) with text output:

```
$ curl -T price.xls http://localhost:9998/tika/main --header "Accept: text/plain"
```

With Tika 1.27 and greater, you can get the text and metadata of a file in json format with:

```
$ curl -T price.xls http://localhost:9998/tika --header "Accept: application/json"
```

To specify whether you want the content to be text (vs. html) specify the handler type after `/tika`:

```
$ curl -T price.xls http://localhost:9998/tika/text --header "Accept: application/json"
```

Skip Embedded Files/Attachments

```
$ curl -T test_recursive_embedded.docx http://localhost:9998/tika --header "Accept: text/plain" --header"X-Tika-Skip-Embedded: true"
```

Multipart Support

Tika Resource also accepts the files as multipart/form-data attachments with POST. Posting files as multipart attachments may be beneficial in cases when the files are too big for them to be PUT directly in the request body. Note that Tika JAX-RS server makes the best effort at storing some of the multipart content to the disk while still supporting the streaming:

```
curl -F upload=@price.xls URL http://localhost:9998/tika/form
```

Note that the address has an extra "/form" path segment.

Detector Resource

```
/detect/stream
```

HTTP PUTs a document and uses the Default Detector from Tika to identify its MIME/media type. The caveat here is that providing a hint for the filename can increase the quality of detection.

Default return is a string of the Media type name.

Some Example calls with cURL:

PUT an RTF file and get back RTF

```
$ curl -X PUT --data-binary @TODO.rtf http://localhost:9998/detect/stream
```

PUT a CSV file without filename hint and get back text/plain

```
$ curl -X PUT --upload-file foo.csv http://localhost:9998/detect/stream
```

PUT a CSV file with filename hint and get back text/csv

```
$ curl -X PUT -H "Content-Disposition: attachment; filename=foo.csv" --upload-file foo.csv http://localhost:9998
/detect/stream
```

Language Resource

```
/language/stream
```

HTTP PUTs or POSTs a UTF-8 text file to the LanguageIdentifier to identify its language.

NOTE: This endpoint does not parse files. It runs detection on a UTF-8 string.

Default return is a string of the 2 character identified language.

Some Example calls with cURL:

PUT a TXT file with English This is English! and get back en

```
$ curl -X PUT --data-binary @foo.txt http://localhost:9998/language/stream
en
```

PUT a TXT file with French comme çi comme ça and get back fr

```
curl -X PUT --data-binary @foo.txt http://localhost:9998/language/stream
fr
```

```
/language/string
```

HTTP PUTs or POSTs a text string to the LanguageIdentifier to identify its language.

Default return is a string of the 2 character identified language.

Some Example calls with cURL:

PUT a string with English This is English! and get back en

```
$ curl -X PUT --data "This is English!" http://localhost:9998/language/string
en
```

PUT a string with French comme çi comme ça and get back fr

```
curl -X PUT --data "comme çi comme ça" http://localhost:9998/language/string
fr
```

Translate Resource

```
/translate/all/translator/src/dest
```

HTTP PUTs or POSTs a document to the identified *translator* and translates from *src* language to *dest*

Default return is the translated string if successful, else the original string back.

Note that: * *translator* should be a fully qualified Tika class name (with package) e.g., org.apache.tika.language.translate.Lingo24Translator * *src* should be the 2 character short code for the source language, e.g., 'en' for English * *dest* should be the 2 character short code for the dest language, e.g., 'es' for Spanish.

Some Example calls with cURL:

PUT a TXT file named sentences with Spanish me falta práctica en Español and get back the English translation using Lingo24

```
$ curl -X PUT --data-binary @sentences http://localhost:9998/translate/all/org.apache.tika.language.translate.
Lingo24Translator/es/en
lack of practice in Spanish
```

PUT a TXT file named sentences with Spanish me falta práctica en Español and get back the English translation using Microsoft

```
$ curl -X PUT --data-binary @sentences http://localhost:9998/translate/all/org.apache.tika.language.translate.
MicrosoftTranslator/es/en
I need practice in Spanish
```

PUT a TXT file named sentences with Spanish me falta práctica en Español and get back the English translation using Google

```
$ curl -X PUT --data-binary @sentences http://localhost:9998/translate/all/org.apache.tika.language.translate.
GoogleTranslator/es/en
I need practice in Spanish
```

```
/translate/all/src/dest
```

HTTP PUTs or POSTs a document to the identified *translator* and auto-detects the *src* language using LanguageIdentifiers, and then translates *src* to *dest*

Default return is the translated string if successful, else the original string back.

Note that: * *translator* should be a fully qualified Tika class name (with package) e.g., org.apache.tika.language.translate.Lingo24Translator * *dest* should be the 2 character short code for the dest language, e.g., 'es' for Spanish.

PUT a TXT file named sentences2 with French comme çi comme ça and get back the English translation using Google auto-detecting the language

```
$ curl -X PUT --data-binary @sentences2 http://localhost:9998/translate/all/org.apache.tika.language.translate.
GoogleTranslator/en
so so
```

Recursive Metadata and Content

```
/rmeta
```

Returns a JSONified list of Metadata objects for the container document and all embedded documents. The text that is extracted from each document is stored in the metadata object under "X-TIKA:content".

```
$ curl -T test_recursive_embedded.docx http://localhost:9998/rmeta
```

Returns:

```
[
 {"Application-Name":"Microsoft Office Word",
  "Application-Version":"15.0000",
  "X-Parsed-By":["org.apache.tika.parser.DefaultParser","org.apache.tika.parser.microsoft.ooxml.OOXMLParser"],
  "X-TIKA:content":"embed_0 "
  ...
 },
 {"Content-Encoding":"ISO-8859-1",
  "Content-Length":"8",
  "Content-Type":"text/plain; charset=ISO-8859-1"
  "X-TIKA:content":"embed_1b",
  ...
 }
 ...
]
```

The default format for "X-TIKA:content" is XML. However, you can select "text only" with

```
/rmeta/text
```

HTML with

```
/rmeta/html
```

and no content (metadata only) with

```
/rmeta/ignore
```

Multipart Support

Metadata Resource also accepts the files as multipart/form-data attachments with POST. Posting files as multipart attachments may be beneficial in cases when the files are too big for them to be PUT directly in the request body. Note that Tika JAX-RS server makes the best effort at storing some of the multipart content to the disk while still supporting the streaming:

```
curl -F upload=@test_recursive_embedded.docx URL http://localhost:9998/rmeta/form
```

Note that the address has an extra "/form" path segment.

Skip Embedded Files/Attachments

```
$ curl -T test_recursive_embedded.docx http://localhost:9998/rmeta --header "X-Tika-Skip-Embedded: true"
```

Specifying Limits

As of Tika 1.25, you can limit the maximum number of embedded resources and the write limit per handler.

To specify the maximum number of embedded documents, set the `maxEmbeddedResources` in the header. Note that the container document does not count towards this number. The following will return the results for the container document only.

```
curl -T test_recursive_embedded.docx --header "maxEmbeddedResources: 0" http://localhost:9998/rmeta
```

To specify a write limit per handler, set the `writeLimit` parameter in a header. This limit applies to each handler (each embedded document). If this is triggered, the parse will continue on to the next embedded object and store `X-TIKA:Exception:write_limit_reached = "true"` in the metadata object for the embedded file that triggered the write limit.

```
curl -T test_recursive_embedded.docx --header "writeLimit: 1000" http://localhost:9998/rmeta
```

NOTE: In Tika 2.0, the writeLimit applies to the full document including the embedded files, not to each handler.

Filtering Metadata Keys

The `/rmeta` endpoint can return far more metadata fields than a user might want to process. As of Tika 1.25, users can configure a MetadataFilter that either `includes` or `excludes` fields by name.

Note: the `MetadataFilters` only work with the `/rmeta` endpoint. Further, they do not shortcut metadata extraction within Parsers. They only delete the unwanted fields after the parse. This still can save resources in storage and network bandwidth.

A user can map Tika field names to names they prefer. If `excludeUnmapped` is set to true, only those fields that are included in the mapping are passed back to the client.

FieldNameMappingFilter

```
<properties>
  <metadataFilters>
    <metadataFilter class="org.apache.tika.metadata.filter.FieldNameMappingFilter">
      <params>
        <excludeUnmapped>true</excludeUnmapped>
        <mappings>
          <mapping from="X-TIKA:content" to="content"/>
          <mapping from="a" to="b"/>
        </mappings>
      </params>
    </metadataFilter>
  </metadataFilters>
</properties>
```

A user can set the following in a `tika-config.xml` file to have the `/rmeta` end point only return three fields:

IncludeFieldMetadataFilter

```
<properties>
  <metadataFilters>
    <metadataFilter class="org.apache.tika.metadata.filter.IncludeFieldMetadataFilter">
      <params>
        <include>
          <field>X-TIKA:content</include>
          <field>extended-properties:Application</include>
          <field>Content-Type</include>
        </param>
      </params>
    </metadataFilter>
  </metadataFilters>
</properties>
```

To exclude those three fields but include all other fields:

ExcludeFieldMetadataFilter

```
<properties>
  <metadataFilters>
    <metadataFilter class="org.apache.tika.metadata.filter.ExcludeFieldMetadataFilter">
      <params>
        <exclude>
          <field>X-TIKA:content</field>
          <field>extended-properties:Application</field>
          <field>Content-Type</field>
        </param>
      </params>
    </metadataFilter>
  </metadataFilters>
</properties>
```

Filtering Metadata Objects

A user may want to parse a file type to get at the embedded contents within it, but s/he may not want a metadata object or contents for the file type itself. For example, `image/emf` files often contain duplicative text, but they may contain an embedded PDF file. If the client had turned off the `EMFParser`, the embedded PDF file would not be parsed. When the `/rmeta` endpoint is configured with the following, it will delete the entire metadata object for files of type `image/emf`.

ClearByMimeMetadataFilter

```
<properties>
  <metadataFilters>
    <metadataFilter class="org.apache.tika.metadata.filter.ClearByMimeMetadataFilter">
      <params>
        <mimes>
          <mime>image/emf</mime>
        </mimes>
      </params>
    </metadataFilter>
  </metadataFilters>
</properties>
```

Integration with tika-eval

As of Tika 1.25, if a user adds the `tika-eval` jar to the server jar's classpath, the `/rmeta` endpoint will add key "profiling" statistics from the `tika-eval` module, including: language identified, number of tokens, number of alphabetic tokens and the "out of vocabulary" percentage. These statistics can be used to decide to reprocess a file with OCR or to reprocess an HTML file with a different encoding detector.

To accomplish this, one may put both the tika-eval jar and the server jar in a `bin/` directory and then run:

```
java -cp bin/* org.apache.tika.server.TikaServerCli
```

See the TikaEval page for more details. Please open issues on our JIRA if you would like other statistics included or if you'd like to make the calculated statistics configurable.

Unpack Resource

```
/unpack
```

HTTP PUTs an embedded document type to the /unpack service and you get back a zip or tar of the raw bytes of the embedded files. Note that this does not operate recursively; it extracts only the child documents of the original file.

You can also use /unpack/all to get back the text and metadata from the container file. If you want the text and metadata from all embedded files, consider using the /rmeta end point.

Default return type is ZIP (without internal compression). Use "Accept" header for TAR return type.

Some example calls with cURL:

PUT zip file and get back met file zip

```
$ curl -X PUT --data-binary @foo.zip http://localhost:9998/unpack --header "Content-type: application/zip"
```

PUT doc file and get back met file tar

```
$ curl -T Doc1_ole.doc -H "Accept: application/x-tar" http://localhost:9998/unpack > /var/tmp/x.tar
```

PUT doc file and get back the content and metadata

```
$ curl -T Doc1_ole.doc http://localhost:9998/unpack/all > /var/tmp/x.zip
```

Text is stored in `TEXT` file, metadata cvs in `METADATA`. Use "accept" header if you want TAR output.

PUT zip file and get back met file zip and bump max attachment size from default 100MB to custom 1GB

This is available in tika-server versions greater than 2.8.0.

```
$ curl -X PUT --data-binary @foo.zip http://localhost:9998/unpack --header "Content-type: application/zip" --
header "unpackMaxBytes:  1073741824"
```

Information Services

Available Endpoints

```
/
```

Hitting the route of the server in your web browser will give a basic report of all the endpoints defined in the server, what URL they have etc

Defined Mime Types

```
/mime-types
```

Mime types, their aliases, their supertype, and the parser. Available as plain text, json or human readable HTML

Available Detectors

```
/detectors
```

The top level Detector to be used, and any child detectors within it. Available as plain text, json or human readable HTML

Available Parsers

```
/parsers
```

Lists all of the parsers currently available

```
/parsers/details
```

List all the available parsers, along with what mimetypes they support

Specifying a URL Instead of Putting Bytes in Tika >= 2.x

In Tika 2.x, use a FileSystemFetcher, a UrlFetcher or or an HttpFetcher. See: tika-pipes (FetchersInClassicServerEndpoints)

We have entirely removed the `-enableFileUrl` capability that we had in 1.x because it posed a security threat.

Transfer-Layer Compression

As of Tika 1.24.1, users can turn on `gzip` compression for either files on their way to `tika-server` or the output from `tika-server`.

If you want to `gzip` your files before sending to `tika-server` , add

```
curl -T test_my_doc.pdf -H "Content-Encoding: gzip" http://localhost:9998/rmeta
```

If you want `tika-server` to compress the output of the parse:

```
curl -T test_my_doc.pdf -H "Accept-Encoding: gzip" http://localhost:9998/rmeta
```

Making Legacy Tika Server Endpoints Robust to OOMs, Infinite Loops and Memory Leaks in Tika >= 2.x

As of Tika 2.x, the default behavior is that the main code forks the server process to handle parsing. If there's an OOM or a timeout or other crash during the parse, the forked process will shutdown and restart.

If the child process is in the process of shutting down, and it gets a new request it will return `503 -- Service Unavailable`. If the server times out on a file, the client will receive an IOException from the closed socket. Note that all other files that are being processed will end with an IOException from a closed socket when the child process shuts down; e.g. if you send three files to tika-server concurrently, and one of them causes a catastrophic problem requiring the child to shut down, you won't be able to tell which file caused the problems. In the future, we may implement a gentler shutdown than we currently have.

To turn off this behavior and to go back to the more dangerous Tika 1.x legacy behavior, configure tika-server with the `<noFork>true</noFork>` or add `--noFork` as the commandline argument.

NOTE: As mentioned above, clients will need to be aware that the server could be unavailable temporarily while it is restarting. Clients will need to have a retry logic.

Making Tika Server Robust to OOMs, Infinite Loops etc. with the tika-pipes handlers

There are two handlers (`/pipes` and `/async`) that use the tika-pipes framework that was added in Tika 2.x. These run the parses, a single file at a time, in forked processes so that the tika-server is always "up" even when a parser strikes catastrophe. See tika-pipes.

Logging

In Tika 1.x, you can customize logging via the usual `log4j` commandline argument, e.g. `-Dlog4j.configuration=file:log4j_server.xml`. If using `-spawnChild`, specify the configuration for the child process with the `-J` prepended as in `java -jar tika-server-X.Y-jar -spawnChild -JDlog4j.configuration=file:log4j_server.xml`. Some important notes for logging in the child process in versions <= 1.19.1: 1) make sure that the `debug` option is off, and 2) do not log to stdout (this is used for interprocess communication between the parent and child!).

The default level of logging is `debug`, but you can also set logging to `info` via the commandline: `-log info`.

In Tika 2.x, you can set log4j2.xml configuration for the forked process in the <jvmArgs/> element. See below.

Monitoring

ServerStatus

Tika Server uses ServerStatus object to maintain and track current status of server. The data can be exported to REST resource and JMX MBean (from 1.26).

To enable REST endpoint and JMX MBean:

```
java -jar tika-server-x.x.jar -status
```

REST resource to access server status:

```
/status
```

MBean:

Object name: org.apache.tika.server.mbean:type=basic,name=ServerStatusExporter

NOTE: In Tika 2.x, this endpoint is enabled by either `enablingUnsecureFeatures` or by specifying it as an endpoint.

```
<properties>
  <server>
    <params>
      <enableUnsecureFeatures>true</enableUnsecureFeatures>
    </params>
  </server>
</properties>
```

```
<properties>
  <server>
    <params>
      <port>9999</port>
      <forkedJvmArgs>
        <arg>-Xmx2g</arg>
        <arg>-Dlog4j.configurationFile=my-forked-log4j2.xml</arg>
      </forkedJvmArgs>
      <endpoints>
        <endpoint>status</endpoint>
        <endpoint>rmeta</endpoint>
      </endpoints>
    </params>
  </server>
</properties>
```

SSL (Beta)

Tika Server now has the ability to be spawned with SSL enabled by providing a keystore/Truststore as part of the configuration, this is likely to change but is available as part of Tika 2.4.0.

Example:

```
<properties>
  <server>
    <params>
      <port>9999</port>
      <forkedJvmArgs>
        <arg>-Xmx2g</arg>
      </forkedJvmArgs>
      <endpoints>
        <endpoint>rmeta</endpoint>
      </endpoints>
    </params>
    <tlsConfig>
      <params>
        <active>true</active>
        <keyStoreType>myType</keyStoreType>
        <keyStorePassword>pass</keyStorePassword>
        <keyStoreFile>/something/or/other</keyStoreFile>
        <trustStoreType>myType2</trustStoreType>
        <trustStorePassword>pass2</trustStorePassword>
        <trustStoreFile>/something/or/other2</trustStoreFile>
      </params>
    </tlsConfig>
  </server>
</properties>
```

If you are new to TLS, see our README.txt for how we generated client and server keystores and truststores for our unit tests.

Configuring Parsers at Parse time/per file

See Configuring Parsers At Parse Time in tika-server.

Architecture

Tika Server is based on JSR 311 for a network serve. The server package uses the Apache CXF framework that provides an implementation of JAX-RS for Java.