OpenSextant User’s Guide

© The MITRE Corporation 2012

Table of Contents

[1 Introduction 1](#_Toc326138882)

[2 The GATE Home Directory 1](#_Toc326138883)

[3 Gazetteers 1](#_Toc326138884)

[4 GATE Application Files 2](#_Toc326138885)

[5 Installing the Standalone Application 2](#_Toc326138886)

[6 Running the Standalone Application 2](#_Toc326138887)

[6.1 The OpenSextant GUI 2](#_Toc326138888)

[6.1.1 Set GATE Home 3](#_Toc326138889)

[6.1.2 Select GAPP File 3](#_Toc326138890)

[6.1.3 Select Input 3](#_Toc326138891)

[6.1.4 Output Formats 3](#_Toc326138892)

[6.1.5 Set Output File 3](#_Toc326138893)

[6.1.6 Set Temp Directory 3](#_Toc326138894)

[6.1.7 Description Field Source 3](#_Toc326138895)

[6.1.8 Run OpenSextant 3](#_Toc326138896)

[6.2 Running OpenSextant from the Command Line 3](#_Toc326138897)

[7 The OpenSextant Web Service 4](#_Toc326138898)

[7.1 Deploying the OpenSextant Web Service 5](#_Toc326138899)

[7.2 Configuring the OpenSextant Web Service 5](#_Toc326138900)

[7.2.1 OpenSextant Home Page 5](#_Toc326138901)

[7.2.2 The config.properties file 6](#_Toc326138902)

[7.3 Text Chunk Upload 7](#_Toc326138903)

[7.4 File Upload 7](#_Toc326138904)

[7.5 Output Files 7](#_Toc326138905)

[7.6 Error Handling 7](#_Toc326138906)

[Appendix A: Text Upload Client 8](#_Toc326138907)

[Appendix B: File Upload Client 10](#_Toc326138908)

# Introduction

This document explains how to install and use OpenSextant, open source software for geotagging unstructured text. OpenSextant is implemented in Java and based on the open source text analytic software GATE (<http://gate.ac.uk/>).

OpenSextant can geotag documents in any file format supported by GATE, which includes plain text, Microsoft Word, PDF, and HTML. Multiple files can submitted in a compressed archive. OpenSextant can unpack both .zip and .tar archives, as well as tarball archives with .gz or .tgz extensions. The newer .zipx format is not currently supported.

Geospatial information is written out in commonly used geospatial data formats, such as KML, ESRI Shapefile, CSV, JSON or WKT. Details on how OpenSextant works and output formats it supports can be found in the document *Introduction to OpenSextant*.

OpenSextant can be run either as a standalone application or as a web service. Instructions for installing and configuring both are included in this document.

# The GATE Home Directory

GATE requires that a system property be set to point to the top-level GATE install directory. It expects to find certain files relative to this directory. Both the standalone and web service installations are provided with a GateHome directory that serves this purpose.

# Gazetteers

Within the context of text analytics, a gazetteer is a software module that matches words in the text to the names of people, places, and things. For OpenSextant, gazetteers match words to the names of geospatial features, such as countries, cities, rivers, and so on. OpenSextant offers three types of gazetteer: text ,database and Solr. The text gazetteer reads in data from text files and stores it in memory. It works best for relatively small numbers of places. Database gazetteers offer better scalability. OpenSextant’s database gazetteer is based on MySQL. The Solr based implementation scales to very large gazetteers and is also very fast.

The Gazetteer directory included in the installations contains data and instructions (in the README file) for creating a MySQL database containing around 11.5 million global geospatial features. It also contains instructions for the creation of the Solr based version containing the same data.

# GATE Application Files

OpenSextant’s document processing pipeline is specified at run time by a GATE Application (GAPP) file. The GAPP files currently shipped with OpenSextant are:

* OpenSextant\_Micro.gapp – Uses a gazetteer loaded from the text files.
* OpenSextant\_MySQL.gapp – Uses a gazetteer loaded into a MySQL database.
* OpenSextant\_Solr.gapp – Uses a gazetteer loaded into a Solr instance.

These GAPP files reside in the GateHome directory.

Each of these GAPPs represents essentially the same process, differing only in the implementation of the gazetteer (the component known as the NaiveTagger).

Note that the Micro version uses a very simple persistence and matching implementation (read from files and all stored in memory) it is only really practical for small ( x 100,000s) of gazetteer entries. The set of gazetteer entries to be used can be changed by editing the MicroGazetteerContents.def file in the GateHome/resources/GeoGazetteer dir. This file contains a list of gazetteer files which will be used by the Micro gazetteer. By selecting among the files created by the gazetteer cleaning process, a user can select what subset of gazetteer data to be used.

# Installing the Standalone Application

There’s no real installation required – just unpack OpenSextantRunner.zip. The resulting OpenSextantRunner directory will contain the following:

* doc directory, containing this document and its companion, *Introduction to OpenSextant*.
* GateHome directory
* Gazetteer directory
* lib directory, containing executable JARs comprising OpenSextant and its dependencies
* resources directory, containing various files required by OpenSextant
* Two shell scripts, one for Windows and the other for Linux, which start up the OpenSextant GUI.

# Running the Standalone Application

OpenSextant can be run either through a GUI or from the command line. The underlying machinery for processing documents and generating output is exactly the same for both invocation methods.

## The OpenSextant GUI

On a Windows machine, double clicking on OpenSextantGUI.bat will bring up the GUI. On a Linux machine, the GUI is started with the command ./OpenSextantGUI.sh (it may be necessary to first make the script executable with the command chmod +x OpenSextantGUI.sh). The OpenSextant GUI is shown in Figure 1. Each button is described below.

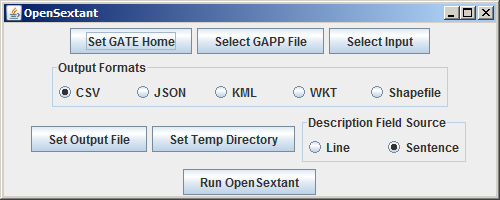


Figure . The OpenSextant GUI

### Set GATE Home

Begin by clicking on this button and selecting the GateHome directory.

### Select GAPP File

The GAPP files reside in the GateHome directory.

### Select Input

Input can be either a file or a directory. If the input is a directory, all the documents in that directory will be processed. Within a directory, GATE will ignore subdirectories and files that it does not recognize as documents. To process all the documents in a directory and its subdirectories recursively, zip up the directory and submit the zip file for processing.

### Output Formats

There is a radio button for each supported output format.

### Set Output File

Specify the destination for the output.

### Set Temp Directory

If the input is a zip or tar archive, a temporary storage directory is required for unpacking the archive. A temporary storage directory is also required if shapefile output is selected.

### Description Field Source

KML and shapefile output both contain text description fields for each geolocation. Two choices are offered for forming the text string to insert in the description field. For some types of documents (e.g., news articles) the sentence containing the geolocation is a good choice, but for other types (e.g., spreadsheets) sentence splitting may not be successful and the line of text containing the geolocation is a better choice.

### Run OpenSextant

Clicking on this button will result in either a “Document processing completed” message or an error message containing sufficient information for debugging.

## Running OpenSextant from the Command Line

From a console window open to the OpenSextantRunner directory, the Windows command for running OpenSextant is:



On a Linux machine, the command is the same except for the argument of the –cp option, which adds all the JARs in the lib directory to the classpath:



The VM option –Xmx1500m sets the heap size to the recommended minimum value. The option –Dgate.home=GateHome sets the GATE home system property to point to the GateHome directory. Invoking the org.mitre.opensextant.OpenSextantRunner main class with either no options or an invalid option will generate the message shown in Figure 2.

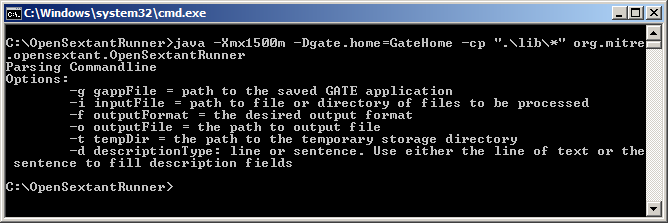


Figure . Running OpenSextant from the command line.

The command line options correspond to the GUI buttons:

* -g The path to the GAPP file
* -i The path to the input file or directory
* -f The choices are CSV, JSON, KML, WKT and Shapefile, and are not sensitive to case
* -o The destination for the output
* -t The path to the temporary storage directory
* -d Description type, either line or sentence. This is also not case sensitive.

The output to the console window will either be a success message or an error message sufficient for debugging purposes.

# The OpenSextant Web Service

Clients can access the OpenSextant web service through two RESTful interfaces. One allows for uploading a file to be geotagged, and the other geotags a chunk of text. Each interface corresponds to a servlet that responds to HTTP POST requests. The underlying machinery is identical for both servlets.

## Deploying the OpenSextant Web Service

Deploying the web service is accomplished by dropping OpenSextant.war into the webapps directory for Tomcat or the standalone/deployments directory for JBOSS. However, some system-specific configuration is required.

## Configuring the OpenSextant Web Service

The files that need to be edited to configure the web service are described below.

### OpenSextant Home Page

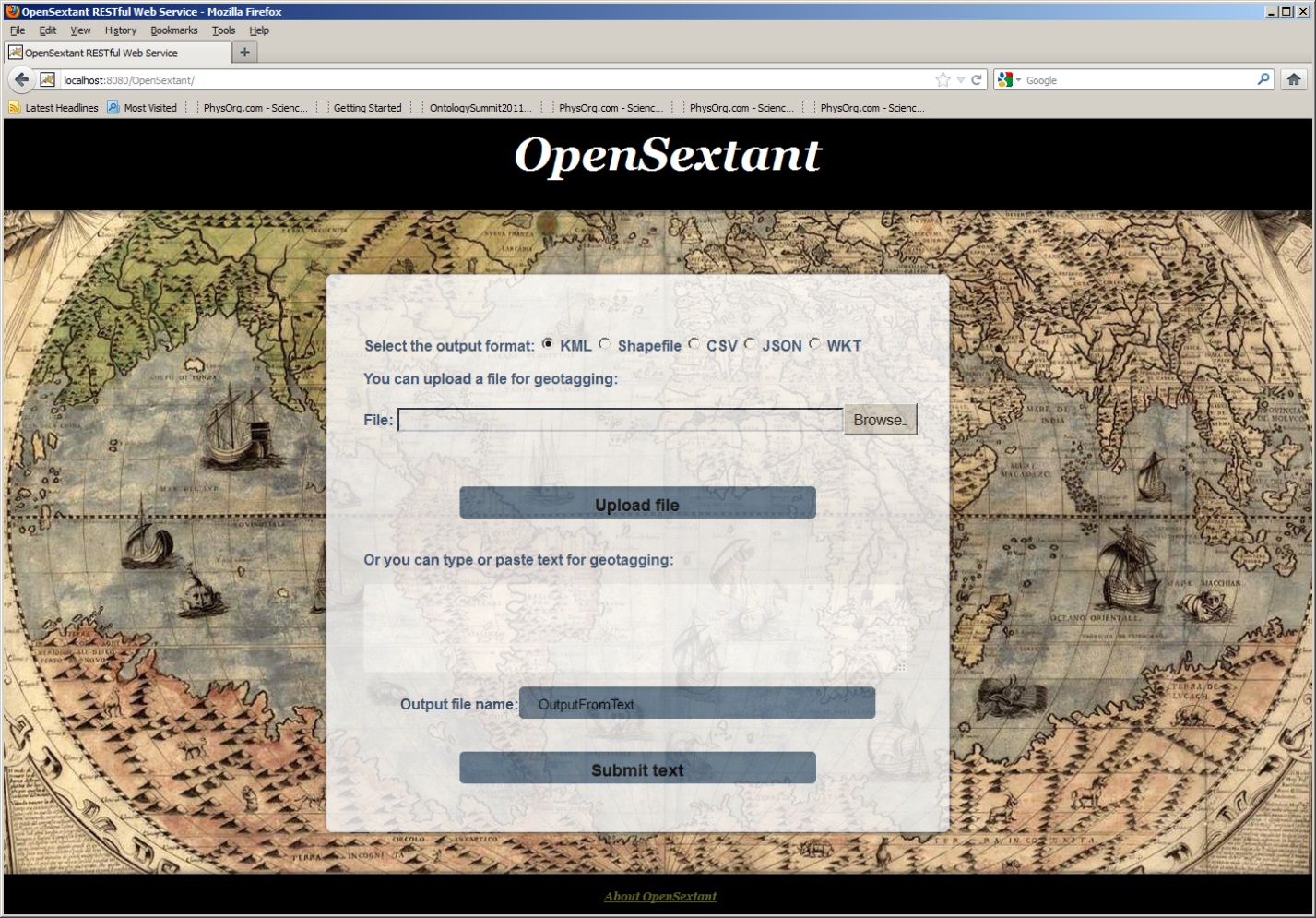
The OpenSextant home page can be used to manually submit files or text for geotagging. A screenshot is shown in Figure 3. The file for the home page is index.html. 

Figure . The OpenSextant web service home page.

The file index.html begins with the Javascript function shown in Figure 4. The URLs and port numbers (“platecaree.mitre.org:8080” in Figure 4) should be edited to conform to the system on which the service is deployed.

Figure . JavaScript within index.html

### The config.properties file

The config.properties file, shown in Figure 5, resides in the WEB-INF directory. It is used to set the following properties at deployment time:

* outputDir – The directory where the output files appear.
* tempDir – A temporary storage directory is required for uploading files. This directory is also used for unpacking archives and creating shapefiles.
* serviceUrl – This is the URL for the servlet that allows downloading of the output files. The URL and port number need to be set to match the system on which the service is deployed.
* gappFileUrl – The URL of the GAPP file
* appName – The name of the application as displayed to the user. Can be set to any value.
* descriptionType - Description type, either line or sentence. Note that this is case sensitive.

Figure . The config.properties file

## Text Chunk Upload

The relative URL for text chunk upload is /OpenSextant/TextUpload. The POST request has three string parameters:

* outputType – The recognized values are KML, Shapefile, CSV, JSON and WKT.
* textChunk – The text to be geotagged.
* outFile – The name of the output file.

The MIME type is application/x-www-form-urlencoded.

Appendix A shows an example of a Java client that uploads text for geotagging. The Apache HttpClient software (<http://hc.apache.org/httpcomponents-client-ga/>) is used to provide the HTTP functionality.

## File Upload

The relative URL for file upload is /OpenSextant/FileUpload. The POST request has a string parameter and a file parameter:

* outputType – The recognized values are KML, Shapefile, CSV, JSON and WKT.
* The file to be uploaded. The name of the parameter is unimportant.

The MIME type is multipart/form-data.

Appendix B shows an example of a Java client that uploads a file for geotagging.

## Output Files

The output for both text chunk and file upload is a file in one of the supported formats as specified in the POST request. Output files are written to the directory specified in the config.properties file. For text chunk upload, the file name is provided in the POST request. The extension appropriate to the format (e.g., .kml, .csv) is appended to this file name. For file upload, the output file has the same name as the input file with the extension changed to match the output format. For example, the KML output from a file SomeDocument.doc would be SomeDocument.kml.

Shapefile output generates multiple files. These are written out in a zip archive, so that the Shapefile output from SomeDocument.doc would be SomeDocument.zip.

## Error Handling

If the POST request is processed successfully, the response from the server contains the OK status code (integer value 200). If there are any errors during processing, the Internal Server Error status code (integer value 500) is returned. All details about the error required for diagnosing the problem, including stack traces, is written to the server log file.

# Appendix A: Text Upload Client





# Appendix B: File Upload Client

