Open Source Knowledge Enrichment System Administration Guide

Version 1.0

1. May 2019

Table of Contents

[1 Introduction 6](#_Toc18406984)

[2 User Authentication 6](#_Toc18406985)

[2.1 Client Certificates 6](#_Toc18406986)

[2.2 HTTP Header 7](#_Toc18406987)

[2.3 LDAP 7](#_Toc18406988)

[2.4 Local 8](#_Toc18406989)

[2.5 OAUTH (Google) 8](#_Toc18406990)

[2.6 Single User 8](#_Toc18406991)

[3 Data Directory Layout 8](#_Toc18406992)

[4 Configuration 9](#_Toc18406993)

[4.1 File-based Configuration 10](#_Toc18406994)

[4.2 Database Stored / Online Configuration 12](#_Toc18406995)

[4.2.1 Accumulo 12](#_Toc18406996)

[4.2.2 Annotations 13](#_Toc18406997)

[4.2.3 APIs 13](#_Toc18406998)

[4.2.4 Authentication 14](#_Toc18406999)

[4.2.5 AWS 16](#_Toc18407000)

[4.2.6 Azure 16](#_Toc18407001)

[4.2.7 Elasticsearch 17](#_Toc18407002)

[4.2.8 Export 17](#_Toc18407003)

[4.2.9 Faroo 18](#_Toc18407004)

[4.2.10 File 18](#_Toc18407005)

[4.2.11 HDFS 18](#_Toc18407006)

[4.2.12 Instrumentation 19](#_Toc18407007)

[4.2.13 Kafka 19](#_Toc18407008)

[4.2.14 Kibana 20](#_Toc18407009)

[4.2.15 LDAP 20](#_Toc18407010)

[4.2.16 Links 20](#_Toc18407011)

[4.2.17 Miscellaneous 21](#_Toc18407012)

[4.2.18 News Feed 21](#_Toc18407013)

[4.2.19 Pastebin 22](#_Toc18407014)

[4.2.20 Source Handlers 22](#_Toc18407015)

[4.2.21 System Component 23](#_Toc18407016)

[4.2.22 Web Crawler 23](#_Toc18407017)

[4.2.23 Video 23](#_Toc18407018)

[5 Databases 24](#_Toc18407019)

[5.1 PostgreSQL 24](#_Toc18407020)

[5.2 Elasticsearch 24](#_Toc18407021)

[5.3 Other Databases 25](#_Toc18407022)

[6 External Services 25](#_Toc18407023)

[6.1 GeoCoding 25](#_Toc18407024)

[6.2 Foreign Language Translation 26](#_Toc18407025)

[6.3 Microsoft Academic API 26](#_Toc18407026)

[6.4 Google Oauth 26](#_Toc18407027)

[7 Hortonworks Data Platform 26](#_Toc18407028)

[8 Log Files 26](#_Toc18407029)

[8.1 Daemon and Web Application Log Configuration 26](#_Toc18407030)

[8.2 Collector Daemon 27](#_Toc18407031)

[8.3 Collector Microservices 27](#_Toc18407032)

[8.4 Collector Web Application 27](#_Toc18407033)

[8.5 Elasticsearch 27](#_Toc18407034)

[8.6 PostgreSQL 27](#_Toc18407035)

[9 Running OSKE 27](#_Toc18407036)

[9.1 Managing Domains 27](#_Toc18407037)

[9.2 Managing Users 29](#_Toc18407038)

[10 System Components and Updates 29](#_Toc18407039)

[10.1 Repository Layout 29](#_Toc18407040)

[10.2 System Updates 30](#_Toc18407041)

[10.3 OSKE Software (Daemon and Web Application) 31](#_Toc18407042)

[10.4 Microservices 32](#_Toc18407043)

[11 Docker 33](#_Toc18407044)

[12 Security Notes 34](#_Toc18407045)

[13 Usage Notes 34](#_Toc18407046)

Table of Tables

[Table 1 Example HTTP Header Values 7](#_Toc18407047)

[Table 2 Data Directory Layout 9](#_Toc18407048)

[Table 3 File-based Parameters 11](#_Toc18407049)

[Table 4 PubMed Specific Configuration 12](#_Toc18407050)

[Table 5 Accumulo Parameters 13](#_Toc18407051)

[Table 6 API Configuration 14](#_Toc18407052)

[Table 7 Authentication Configuration 16](#_Toc18407053)

[Table 8 Sample AWS Configuration 16](#_Toc18407054)

[Table 9 Elasticsearch Configuration 17](#_Toc18407055)

[Table 10 Export Configuration 18](#_Toc18407056)

[Table 11 File Configuration 18](#_Toc18407057)

[Table 12 HDFS Configuration 19](#_Toc18407058)

[Table 13 Instrumentation Configuration 19](#_Toc18407059)

[Table 14 Kafka Configuration 20](#_Toc18407060)

[Table 15 Kibana Configuration 20](#_Toc18407061)

[Table 16 LDAP Configuration 20](#_Toc18407062)

[Table 17 Link Configuration 21](#_Toc18407063)

[Table 18 Miscellaneous Configuration 21](#_Toc18407064)

[Table 19 RSS (News) Configuration 22](#_Toc18407065)

[Table 20 System Component Configuration 23](#_Toc18407066)

[Table 21 Video Configuration 24](#_Toc18407067)

[Table 22 Log Levels 27](#_Toc18407068)

[Table 23 OSKE Repository Layout 30](#_Toc18407069)

[Table 24 Internal Microservices used in OSKE 32](#_Toc18407070)

[Table 25 External Microservices used in OSKE 33](#_Toc18407071)

Table of Figures

[Figure 1 Sample LDAP Configuration 16](#_Toc18407072)

[Figure 2 Sample Active Directory Configuration 16](#_Toc18407073)

[Figure 3 Sample Azure Configuration 16](#_Toc18407074)

[Figure 4 Accessing Advanced Collector View 22](#_Toc18407075)

[Figure 5 Access Handlers Page 23](#_Toc18407076)

[Figure 6 Sample Web Crawler Configuration 23](#_Toc18407077)

[Figure 7 Sample Video Configuration 24](#_Toc18407078)

[Figure 8 SQL Command to Manually Create an Administrator Record 28](#_Toc18407079)

[Figure 9 SQL Commands to Delete a Domain 28](#_Toc18407080)

[Figure 10 REST Commands to Delete Elastic Indexes for a Domain 29](#_Toc18407081)

[Figure 11 Common Docker Commands 34](#_Toc18407082)

[Figure 12 Commands to Scale a Docker Service 34](#_Toc18407083)

# Introduction

This document covers how to administer an Open Source Knowledge Enrichment (OSKE/OpenKE[[1]](#footnote-1)) environment. The guide covers the various authentication methods, the numerous configuration options and various log files, and common situations that may be encountered. Refer to the OSKE Server Installation Guide for additional information.

If further assistance is required, open an issue on the project’s GitHub repository.

# User Authentication

OSKE currently supports six methods for authentication: client certificates (x509), HTTP Header, LDAP, Local, OAuth2 (via Google), and Single User. OSKE assumes that the user ID is a legitimate and functioning e-mail address. Once the user has been authenticated, a session is established for that user in the web application. Authentication is only required for the web application. The microservices are assumed to be running within a protected environment (i.e., they only receive legitimate traffic.)

Except for the development and OSKE-Lite[[2]](#footnote-2) deployments, administrators will need to create an initial user record as an *administrator* in the “system” domain; the install guide covers this process. The development and OSKE-Lite install scripts create a default user record.

Once the user is authenticated by the system, a *session* object is created for the user within the application server (i.e., Apache Tomcat). These sessions utilize the default expiration of time of the servlet container (typically 30 minutes on Apache Tomcat). Other than the user’s email, name, and roles, very little detail is stored within the session; therefore, sessions are recreated with any request to validate the user’s authorization (which occurs on every server request from the client). For the LDAP and Local methods, the user will be re-directed to the OSKE Login Page. Sessions are destroyed when the user clicks the logout button in the upper right-hand corner of the application.

## Client Certificates

For environments that support mutual authentication for SSL / TLS, OSKE supports authentication through client certificates. The system assumes that the common name (CN) value within the certificate is the user’s email address. If this assumption does not work for a particular deployment environment, the class edu.ncsu.las.webapp.Authentication can be modified to process the x509 certificates as required. Neither this guide or the install manuals cover the necessary steps to support mutual authentication for Tomcat or other Java based web containers.

## HTTP Header

HTTP Header looks for the user ID contained within a series of header fields. Each header field is processed in order until a valid user is found. This mechanism is commonly used in many single sign-on environments. From a security standpoint, the application server should not accept connections from untrusted locations (i.e., they should only receive traffic from a reverse web proxy such as nginx[[3]](#footnote-3)). Without this restriction, clients or other systems (i.e., man in the middle attacker) can alter the HTTP header fields and assume another identity. Table 1 displays example header values that are sent to an application. The OSKE system reads the *remote\_user* header field to identify the user through the email address.

|  |  |
| --- | --- |
| Name | Description |
| host | openke.domain.com |
| x-real-ip | 192.168.1.1 |
| x-forwarded-for | 192.168.1.1 |
| x-forwarded-proto | https |
| remote\_user | *userEmailAddress* |
| connection | Upgrade |
| user-agent | Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... |
| accept | text/html,application/xhtml+xml,application/xml; |
| referer | https://openke.domain.com/collector/domainHome |
| accept-language | en-US,en;q=0.9 |
| cookie | ... |

Table 1 Example HTTP Header Values

## LDAP

With LDAP, the system presents the user with a login form. Once the user enters their credentials (email address and password), the system will bind to an LDAP server. If that bind is successful, then the user has been successfully authenticated. Currently, OSKE assumes that the portion of the email address prior to the “@” symbol corresponds to the user’s UID value in the LDAP server. For Windows Active Directory, the name is assumed to be in the format of “UID@domain”.

## Local

OSKE can also manage user credentials within the system. Passwords are hashed with PBKDF2-HMAC-SHA256 for 10,000 iterations and a random 16-byte salt[[4]](#footnote-4). Users can enter spaces within their password; however, any whitespace at the start or end of a password is removed.

If a user forgets their password, then they can enter their email address and click on the “Forgot Password” link. The system will send them a one-time use token to authenticate and change their password. As the one-time token could be intercepted from an email message, the security of this approach largely depends upon the user’s email system and corresponding infrastructure.

If the user receives the message “Unable to authenticate”, then an exception has occurred; the web application log should be checked for the relevant exception.

## OAUTH (Google)

The application can also directly integrate with Google for OAuth authentication. The *OSKE Install Guide* covers using nginx as reverse proxy and receiving the authentication via an HTTP Header field.

## Single User

In the Single User mode (primarily used for development and OSKE-Lite), the system assumes that only one user can access the application and, hence, is automatically authenticated. The installation process creates the necessary user record.

# Data Directory Layout

This next session covers how the system stores data (primarily, the original crawled documents) on the application server. OSKE also stores data within Elasticsearch[[5]](#footnote-5) and PostgreSQL[[6]](#footnote-6). OSKE can also export data to the local filesystem. Other systems can then read data directly from the filesystem (especially, if a shared mounting system such as the Network File System (NFS)[[7]](#footnote-7) or the Amazon Elastic File System (EFS)[[8]](#footnote-8) is utilized. Another possibility would be to install iNotify[[9]](#footnote-9) on the export directory to act (such as copying the data to another server) on the data files when they are saved.

As part of the system’s configuration, the *system.properties* file defines an attribute, “fileRoot”, which is the base location for the data directories used by OSKE. Under the directory, the layout in Table 2 exists. Both the daemon and the web application write to this location and, hence, why they execute as the same user.

|  |  |
| --- | --- |
| Name | Description |
| {domain}/files\_archive | Not currently in use. The original intention was to place collected data that was no longer necessary into this area. |
| {domain}/files\_normal | Files (original content) downloaded by the OSKE daemon are stored here. Data collected through the discovery process/area is also stored in this location. The filename is the ID used for the matching record in the JSON record. |
| {domain}/files\_sandbox | Files downloaded through the domain discovery area.  *Note: This area is no longer in use. All downloaded content is currently placed in the “normal” area.* |
| {domain}/import | Directory where users may place content to be uploaded into OSKE. The web application also writes directly to this location for uploaded files. |
| externalExport | Directory where exported content is saved from OSKE. Files will remain here for 24 hours until deleted by the daemon. |
| logs | Log files used by the OSKE daemon. By default, these log files are rotated daily. No automated process exists to delete or compress older log files. Logs produced by the microservices are managed by Docker. The web application logs are maintained separately as well. |

Table 2 Data Directory Layout

# Configuration

OSKE requires a substantial number of configuration parameters to function correctly. These parameters are stored in two separate locations: *file* and *database*. The parameters stored in the *file* location cannot be altered at runtime. File-based parameters includes the file root as mentioned in the previous section as well as the PostgreSQL connection parameters. Parameters that have been defined in the Java class edu.ncsu.las.model.collector.type.ConfigurationType as file parameters must be present in the properties file. If file-based parameters are missing in that file (system\_properties.json), the system will log the missing parameters and then exist. Database parameters are defined for the different domains [[10]](#footnote-10)that users have established within OSKE. A special domain, “system”, is defined as the base configuration for all domains. Each domain can override the system configuration as necessary. Parameters stored in the *database* location are alterable during runtime.

The ConfigurationType class also defines which parameters should be stored encrypted (e.g., the PostgreSQL database password). OSKE ensures those parameters are encrypted while stored. For file-based parameters, those parameters are checked for encryption (the value will start with “{aes}” if encrypted) when the application starts. If a field needs to be encrypted, the system will encrypt the value and then save the configuration file (system\_properties.json) into the same location. For the database parameters, these parameters are checked both when the domains are being initialized during system startup as well as when the domain is updated with new configuration.

The following sections document the various properties. As the properties are stored using javascript object notation (JSON)[[11]](#footnote-11), the properties are presented in tables, but using indention to define child objects. If there is no indention or the property does not start with a period, “.”, then the property name is at the top level of the JSON object.

## File-based Configuration

With the OSKE daemon’s root directory[[12]](#footnote-12) (“Collector”), the system\_properties.json file contains a json object with the properties defined from Table 2. As the application minifies (removes whitespace and comments) the file during the load process, you can place comments within the file. However, those comments may disappear if they system needs to update the configuration file (such as when encrypting the database password).

*Note: Please ensure file permissions are limited to OSKE System account. While the database password is encrypted, a portion of that encryption key is contained within the file itself. The other part of the key is static string with the Collector java class file.*

|  |  |  |
| --- | --- | --- |
| Name | Description | Sample Value |
| collector |  |  |
| .id | Identifier for the collector daemon and associated web application. This needs to be unique across any "cluster" / distributed set of collectors. | collect1 |
| .jobPoolSize | The number of concurrent threads that can run within a collector daemon to process jobs. | 5 |
| .taskPoolSize | Maximum number of threads that are used for miscellaneous tasks such as sending emails or exporting search results. | 5 |
| .searchPoolSize | Maximum number of threads used to search URLs for user-based requests in the Domain Discovery area. | 10 |
| .idleKillTimeSec | How long a job be inactive in seconds before it is killed. | 300 |
| .sleepTimeSec | Time the collector daemon sleeps in between performing the primary processing loop (e.g., such as searching for new jobs to run, checking for jobs to stop, etc.) | 60 |
| .workingDirectory | The OSKE system uses the user's (collector) temporary working directory and creates a directory starting with this prefix. This directory contains crawl data (e.g., such as which URLs have been processed) that supports a particular job. | openKEwork |
| .allowServices | Does the current collector daemon instance utilize services such as watching a directory for file uploads? | true |
| .requireUserAgreement | Does the system require a "signed" user agreement before a user can access the system? | false |
| database | This section contains the information needed to configure access to the system's database through a JDBC driver. |  |
| .datasourcename | Name used to identify the database connection. This value can be anything.  *Note: This is not currently used by the application.* |  |
| .driver | Full class name of the driver used to connect to the database |  |
| .url | JDBC URL to connect to the database. The format of this URL is specific to the underlying database system. Refer to the relevant JDBC document for specific details. |  |
| .portnumber | Port that the database server listens for connections. This number will vary by database system and you may change from the system default. |  |
| .user | Database user for the OSKE system. This user requires insert, update, delete, and select permissions on all OSKE database tables. The user doesn't require any further special permissions. |  |
| .password | What password does the user need to authenticate to the database system? You may enter this value as plaintext. The system will encrypt the value when it starts and prefix the encrypted value with “{aes}” when stored. |  |
| .maxconnections | What is the maximum number of connections the database can have open to the database? OSKE utilizes Apache DBCP 2 to provide connection pooling. |  |
| fileroot | The file root is the file system directory used to store original crawled content, logs, and exported files. Users may also place content into a directory to be uploaded into the system. |  |
| secretPhrase | The "secretPhrase" is a text string that is used along with another string hard-coded within the application to form a key used to encrypt configuration parameters for the database and job-level configuration. The actual key is formed by concatenating the two values and using the SHA-256 hash as the key for AES. |  |

Table 3 File-based Parameters

The pool size parameters in Table 3 are designed to find an appropriate balance between performance and responsiveness. For instance, while it may be beneficial to allow many jobs to run simultaneously, those jobs may cause thrashing (and, hence slow processing) as the system has to switch among threads for the processing.

As part of a separate research effort, processing data from PubMed[[13]](#footnote-13) was also explored. Unless those specific classes are invoked from the command-line, these parameters may be ignored. Table 4 contains these parameters.

|  |  |
| --- | --- |
| Name | Description |
| pubmedImporter | Configuration options to import and process PubMed data |
| .ftp\_server | FTP server that hosts the PubMed data |
| .baseline\_dir | Local directory that stores the annual baseline data |
| .update\_dir | Local directory that stores daily update files from PubMed |
| .sleepTimeSec | How much time should the process sleep in between checking for data updates? |
| .baseDirectory | Working directory for the PubMed data. |

Table 4 PubMed Specific Configuration

## Database Stored / Online Configuration

The majority of the configuration for OSKE is specified through online configuration stored in the database. Most of this configuration should be defined for the "system" domain. Other domains assume these values with the ability to override individual values at the top level.

Administrators defined in the “system” domain can access the *database* database by clicking on “Manage Domains” on the system’s home page in the web application. Parameters for the entire deployment are store in the “system” domain. Parameters defined in other domains will override that configuration for the specified domain. If necessary, the configuration can be manipulated directly in the PostgreSQL table “domain”.

### Accumulo

Accumulo defines the parameters used to collect to an Accumulo database as well as what data should be stored.

|  |  |
| --- | --- |
| Name | Description |
| accumulo | Defines the parameters used to connect to an Accumulo database to store raw data collected. A small amount of metadata is stored as well. |
| .instanceName | Instance name for the Accumulo database. |
| .userName | User name to connect to the database. |
| .password | Password used to authenticate to the database. This parameter will be encrypted. |
| .zooKeepers | Comma separated list of ZooKeeper instances. ZooKeeper tracks configuration and connection details for Accumulo. Each instance should be in the format of *hostname*:*port*. |
| .storeRaw | For each of the storage areas, defines whether or not raw collected data should be stored in accumulo |
| .normal | Should the system store the raw data collected during jobs? true/false |
| .sandbox | Should the system store the raw data collected during domain discovery sessions? true / false. As the normal and sandbox areas have been combined, this value is no longer directly used. |
| .archive | Should the system store raw data that has been archived? Currently the archive area is not used. |

Table 5 Accumulo Parameters

### Annotations

Annotations are a variety of data enhancements that are performed on the collected data during processing. This parameter (“annotations”) is a comma-delimited list of the annotator codes that this domain utilizes. These codes match the values from the getCode() method in classes that descend from edu.ncsu.las.annotator.Annotator.

### APIs

APIs are a set of single-value parameters that point to microservices that the application uses to perform a specific function. With these microservices, the desired behavior can be implemented in languages other than Java (e.g., Python). Also, the number of dependencies that exist directly within the OSKE system can be minimized. Prior to separating some of these microservices from the main application, sub-dependency conflicts began to appear (e.g., two classes may have used Apache Tika, but different incompatible versions of that code).

|  |  |
| --- | --- |
| Name | Description |
| dbPediaSpotlightAPI | Web service that annotates text with links to entities found within DBPedia. <https://www.dbpedia-spotlight.org/> |
| geoCodeAPI | Points to a web-based service that implements Nominatum[[14]](#footnote-14) for returning latitude and longitude coordinates for a specified address.  *Note: Currently, only the PubMed processing utilizes this service.* |
| geoTagAPI | Points to a micro-service that calls CLAVIN to identify locations within natural language text. |
| graphAPI | Microservice that allows the system to run queries to return nodes and edges from a graph database.  *Note: This is currently not utilized.* |
| microformatAPI | Points to a microservice written in Python that extracts “microformat” data from HTML tags. <http://microformats.org/> |
| nlpAPI | Points to a microservice that wraps around Stanford’s Core Natural Language Processing (NLP) library. This library performs part-of-speech tagging, named-entity recognition, and open-information extraction. |
| siteInfoAPI | Points to a microservice that scraps information about a particular domain from <http://siteinfo.com/>. |
| spacyAPI | Points to a Python-based microservice that performs named-entity recognition on natural language text using the spaCy library. <https://spacy.io/> |
| textRankAPI | Points to a Python-based microservice that performs keyword extraction, keyphrase extraction, summarization, and discovery index creation. |
| topicModelAPI | Points to a microservice that implements LDA topic modeling. |
| whoisAPI | Points to a Java-based microservice that contacts a WHOIS[[15]](#footnote-15) service to retrieve information about a particular domain. |

Table 6 API Configuration

### Authentication

The authentication parameters define the set of parameters used to authenticate individuals into the OSKE web-based application.

|  |  |
| --- | --- |
| Name | Description |
| webapp | Collection of parameters for the web application. |
| .auth | Collection of parameters that specify how users should be authenticated into the system. |
| .method | Specific method on how users are authenticated.  Possible values: http\_header, ldap, local, oauth2, singleuser, x509 |
| .http\_header | If the authentication method is set to “http\_header”, which header fields should be checked for the user ID value? The user ID is expected to be an email address. |
| .oauth\_clientid | If oauth2 is utilized, what is the client ID? See install guide for information on how to register a web application with Google and to define this field. |
| .signout\_text | Text to place on the logout button. The logout button will destroy the user’s current session in the web application – thus, it is useful to have the button available if the user needs to re-establish themselves for the web application when their security roles have been changed (e.g., the user has been granted access to a new domain). |
| .signout\_url | URL where the user’s browser will be directed to when the logout button is clicked. |
| .ldap | Collection of parameters used to configure the system to use an LDAP server for authentication. The system attempts to bind the user to the LDAP sever. If successful, then the user is authenticated. |
| .dnFormat | Format used for distinguishing name. The USERID will be replaced with the user’s ID they enter.  Active directory DN Format: USERID@domainName.  Other LDAP servers: uid=USERID,ou=accounts,dc=ncsu,dc=edu  (the ou & dc are specific to your ldap installation) |
| .emailDomain | What is the domain used by the user for their email address? The userID’s name will be prepended to this value to form the complete email address. Note: this restriction means that only one email domain can be used. To overcome this restriction, the system would need to query the LDAP server for the user’s email address. |
| .server | LDAP URL used by JNDI. Typically, this will be in the format of ldaps://hostname:636 or ldap://hostname:389 |
| .activeDirectoryDomain | If this value is set, then attempt to discover the active directory domain controllers through DNS searches by prepending \_ldap.\_tcp.dc.\_msdcs and then searching for SRV records. Discovered controllers must support LDAPS (SSL) on port 636. See https://technet.microsoft.com/pt-pt/library/cc759550(v=ws.10).aspx for more details |
| .local | Collection of parameters used to configure the local/internal authentication. |
| .minPasswordLength | Minimum password length. |
| .maxPasswordLength | Maximum password length. As the system hashes passwords, this value has no effect on storage requirements. |
| .lowerCaseRequired | Are lower-case letters required to be part of the password? |
| .upperCaseRequired | Are upper-case letters required to be part of the password? |
| .digitRequired | Are digits (numbers) required to be part of the password? |
| .specialCharRequired | Are other characters (than letters and numbers) required to be part of the password? |
| .maxPasswordAgeDays | How long (in days) can a password be used until it must be changed? |
| .maxFailures | How many failures can be made before an account is locked? To unlock an account, the user will need to reset their password. |

Table 7 Authentication Configuration

Below are two LDAP configuration examples.

|  |
| --- |
| "ldap": {  "dnFormat" : "uid=USERID,ou=accounts,dc=ncsu,dc=edu",  "emailDomain" : "ncsu.edu",  "server": "ldap://ldap.ncsu.edu:389"  } |

Figure 1 Sample LDAP Configuration

|  |
| --- |
| "ldap": {  "dnFormat" : "USERID@domain",  "emailDomain" : "ncsu.edu",  "server": "ldap://ldap.ncsu.edu:389"  } |

Figure 2 Sample Active Directory Configuration

### AWS

AWS specifies credentials and other settings to utilize services provide by Amazon Web Services (AWS). Primarily, this configuration is used by the Discovery Sessions to provide foreign language translation capabilities. If this configuration is not established, then that functionality will be hidden from end users.

|  |  |
| --- | --- |
| Name | Description |
| AWS |  |
| .key | Access key ID used to identified your deployed toAWS. |
| .region | In which region are you using AWS services. Typically, “us-east-1”. |
| .secretKey | Additional value used by AWS to authenticate incoming service requests. |
| .translateMaxTPS | Maximum number of requests that can be sent to AWS for translation per second. You may need to open a support request with AWS if the default limits are too low for your use cases. |

Table 8 Sample AWS Configuration

### Azure

Azure specifies keys used to access specific services provided by Microsoft’s Azure service. Currently, only the academic API key is utilized. To register for this key, see Microsoft Academic API under External Services.

Sample configuration:

|  |
| --- |
| "azure": {  "academicAPI\_key": "123567890abcdefghi"  } |

Figure 3 Sample Azure Configuration

### Elasticsearch

The Elasticsearch parameters define a variety of parameters used to connect to and configure Elasticsearch for use within the application.

|  |  |
| --- | --- |
| Name | Description |
| elastic |  |
| .default.settings | Defines the settings used by any ElasticSearch indexes used to support OSKE. Generally, the mapping.total\_fields.limit is set to 3000. This value may need to be increased.The index.number\_of\_shards is set to 5, and the number\_of\_replicas is set to 1. If using larger domains, adjust these values higher. |
| .default.mappings | Default mappings (field types) for the “web” type. Additional mappings are defined within annotators and applied when the daemon and/or web application starts. Once a mapping is applied, it may not be directly changed. If the “enabled” field is set to “false”, then that value (and, possibly, entire object) is not indexed. If a value is not indexed, it is not searchable. If a field is an array of values, then it should be defined as “nested”.  [How to Update Elasticsearch Mappings through Re-indexing](https://www.elastic.co/guide/en/elasticsearch/reference/6.4/mapping.html#_updating_existing_field_mappings) |
| .port | Port (9300) used to contact Elasticsearch using the native interface. |
| .clusterName | If using the native interface, what is the Elasticsearch cluster name? |
| .host | If using the native interface, what is the initial Elasticsearch host to connect to? The port, clusterName, and host values are not currently utilized[[16]](#footnote-16). |
| .restEndPoint | What is the REST end point used to access Elasticsearch? |
| .storeJSON | Should the json documents (which are the primary OSKE document records be stored in ElasticSearch? |
| .normal | Should json documents be stored in ElasticSearch for jobs? |
| .sandbox | Should json documents be stored in ElasticSearch for discovery sessions?  Not current utilized. |
| .archive | Should json documents be stored in ElasticSearch for the archive?  Not currently utilized. |

Table 9 Elasticsearch Configuration

### Export

The Export parameters define a series of options used to export data from the application.

|  |  |
| --- | --- |
| Name | Description |
| export |  |
| .maxRecordCount | Maximum number of records that can be exported. |
| .timeToLiveHours | Time in hours that exports exist before they are deleted. Downloading an export does not delete the export. |
| .voyant | For exports that are sent to Voyant () |
| .accessURL | URL that end-users access Voyant. |
| .postURL | URL that OSKE posts exported data so that Voyant can process and display the corresponding record set. |
| .maxRecordCount | Maximum number of records that can be sent to Voyant. Typically, this should not be more than 1,000. |

Table 10 Export Configuration

### Faroo

Faroo was an openly available search API that did require registration to get the API key. The company appears to be re-directing themselves as “SeekStorm”. <https://seekstorm.com/>

Currently, it does not appear possible to register for a new API key. The “faroo” handler should be excluded from the list of handlers in “sources” if you do not have a valid key.

### File

The File parameters define what data should be stored on the filesystem when collecting/processing data from the internet. Refer to Data Directory Layout section of this document for more information on the storage areas.

|  |  |
| --- | --- |
| Name | Description |
| File |  |
| .storeRaw | For each of the following storage areas, should raw data be stored? “Raw data” is the file received from the server without any processing of the data. Each value is true/false. |
| .normal |  |
| .sandbox |  |
| .archive |  |

Table 11 File Configuration

### HDFS

The HDFS defines what data should be stored in HDFS when collecting/processing data from the internet. Refer to the Data Directory Layout section of this document for more information on the storage areas. JSON files are the completed record of the processed data with the same content as the data stored within ElasticSearch.

*Note: The current implementation does not support authentication for the HDFS user.*

|  |  |
| --- | --- |
| Name | Description |
| hdfs |  |
| .hdfsUser | User account for HDFS. |
| .hdfsBaseURI | URI root / base that will be used to store collected data from OSKE. |
| .storeJSON | For each of the following storage areas, should the json record be stored? Each value is true/false. |
| .normal |  |
| .sandbox |  |
| .archive |  |
| .storeRaw | For each of the following storage areas, should raw data be stored? “Raw data” is the file received from the server without any processing of the data. Each value is true/false. |

Table 12 HDFS Configuration

### Instrumentation

Instrumentation tracks detailed usage statistics for the web application client, the web application APIs, and the collector daemon.

|  |  |
| --- | --- |
| Name | Description |
| instrumentation |  |
| .sendEvents | Should events be sent to Elasticsearch and/or Skylr. true / false |
| .projectID | Project identifier used within Skylr for OSKE. |
| .useElastic | Events are sent to Elasticsearch when set to true. |
| .useSkylr | Events are set to “Skylr” when set to true. “Skylr” is an instrumentation service developed for LAS. Most organizations will set this value to false. |
| .api | If Skylr is used, what is the end-point to send events? |
| .token | If Skylr is used, what is the token value to authenticate OSKE to the Skylr service? The HTTP Header “AuthToken” is set with this value. |
| .elastic | Parameters used to configure an Elasticsearch instance to handle events from OSKE. The Elasticsearch index will be named “skylr”. |
| .settings | Settings used for the Elasticsearch index. |
| .mappings | Mappings used for the Elasticsearch index. Primarily set to ensure date/time fields are appropriately mapped. |

Table 13 Instrumentation Configuration

### Kafka

Kafka parameters define the attributes to send JSON records to a Kafka-broker. This mechanism provides a way to send data to one or more downstream systems with a minimal amount of coupling. (i.e., from the OSKE perspective, this queue is “publish and forget”.)

Visit <https://kafka.apache.org/> for more details on Kafka.

|  |  |
| --- | --- |
| Name | Description |
| kafka |  |
| .zookeeper | Comma-separated list of zookeeper instances in hostname:port format. |
| .replicationFactor | Number of brokers that must acknowledge a write. |
| .numPartitions | Number of partitions that are created for the Kafka queue. |
| .properties |  |
| .compression.type | Type of compression is used when sending data to the queue. Typically “snappy”. |
| .key.serializer | Class used to serialize keys to the Kafka brokers. Typically, this value is org.apache.kafka.common.serialization.StringSerializer |
| .value.serializer | Class used to serialize values to the Kafka brokers. Typically, this value is org.apache.kafka.common.serialization.StringSerializer |
| .bootstrap.servers | List of comma-separated brokers in hostname:port format |
| .send | For each of the following storage areas, should the JSON record be sent to the Kafka broker(s)? |
| .normal |  |
| .sandbox |  |
| .archive |  |

Table 14 Kafka Configuration

### Kibana

Kibana is a visualization system used in conjunction with Elasticsearch. Our primary usage of the application is to show a small dashboard on the application home page.

|  |  |
| --- | --- |
| Name | Description |
| kibana |  |
| .utilizeDashboard | Defines whether or not the application will show a Kibana dashboard on the application home page. |
| .hometitle | Title for the dashboard on the OSKE Home Page. |
| .homeDashboard | Link to a dashboard object within Kibana that will be displayed on the OSKE Home Page. |

Table 15 Kibana Configuration

### LDAP

LDAP is a set of optional parameters to allow name searches when managing users and add collaborators to document buckets. This requires anonymous access to the LDAP server.

|  |  |
| --- | --- |
| Name | Description |
| ldap |  |
| .utilize | For each of the following storage areas, should raw data be stored? “Raw data” is the file received from the server without any processing of the data. Each value is true/false. |
| .searchFields | Fields to search on the user’s input to find their corresponding records. Format: JSON array of string values. |
| .baseDN | Base DN used for people within the LDAP server. |
| .url | URL to access |

Table 16 LDAP Configuration

### Links

OSKE has three different areas from which a configurable list of hyperlinks can be displayed to users. These hyperlinks can reference a wide number of different resources such as mind-maps, project documentation, and other information sources. The first area is *headerlinks*, which presents a list of links to the users in the top right section of the page. Most likely, no more than two or three links for this section will be defined. The second area is the *resourceSection*, which is a “box” that appears on the domain home page. Finally, *hyperlinks* define a list presented on the advanced collector view.

|  |  |
| --- | --- |
| Name | Description |
| headerLinks | Defines a set of links that can appear in the header of all pages. |
| .utilize | Defines whether or not links will appear on the header |
| .hyperlinks | JSON array of objects. Each object should have two string fields: displayText and link. |
| resourceSection | Box on the domain home page that can be used to display static links to users within that domain. |
| .utilize | Defines whether or not the resource box will be displayed to users. |
| .title | Title to be used for the resource section. |
| .hyperlinks | JSON array of objects. Each object should have two string fields: displayText and link. |
| hyperlinks | List of links to be displayed on the Collector View main menu. JSON array of objects. Each object should have two string fields: displayText and link. |

Table 17 Link Configuration

### Miscellaneous

This section documents a number of single-value parameters.

|  |  |  |
| --- | --- | --- |
| Name | Description | Sample Value |
| accessAuthenticatedSites | Defines whether or not users can create and execute jobs that must authenticate to sites. | true |
| allowDuplicateText | If a record is already found with the same text, should a new record be created? | true |
| allowOnlineDomainPurge | Should domain administrators be allowed to purge this domain? Purging will delete all collected data, discovery sessions, and job history. Any configuration and jobs will remain. | false |
| applicationLogo | Logo that appears in the upper left-hand corner of all pages. The image should be sized to 250 pixels wide by 75 pixels tall. | "resources/images/LAS\_Logo.png" |
| blockedTLDs | JSON array of top level domains (prefixed with a period) that OSKE should not access. | [".de",".es",".fr"] |
| conceptCacheSec | How long should concepts be cached in the concept annotator before they are refreshed? | 1800 |
| secretPhrase | Similar to the value in file configuration, this value may be used to produce a domain-specific encryption key. |  |

Table 18 Miscellaneous Configuration

### News Feed

News feed allows results from one or more RSS feeds to be displayed on the domain home page.

|  |  |
| --- | --- |
| Name | Description |
| newsFeed |  |
| .utilize | Defines whether or not the news feed section is visible to users |
| .cacheTimeMinutes | Defines how long results from the RSS feed(s) should be cached before being checked for changes. |
| .keywords | JSON array of string values. If one or more values exist, then at least one of those values must be present in an RSS entry’s title or description for the result to be shown to the user. |
| .title | Title of the news feed section on the domain home page. |
| .urls | JSON array of string values where each value is an RSS feed. |

Table 19 RSS (News) Configuration

### Pastebin

Pastebin-type websites allow users to store plain text files that are then retrievable by others who know the specific URL. Many of these sites have made available search engines on their sites to bring back posted content. As an example of nefarious use, hackers used pastebin.com to communicate how to download the Sony Hack files through Torrent.

Several websites have created custom Google Search engines (<https://cse.google.com/cse/>) to search a number of Pastebin-type sites.

*Note: This same pattern could be duplicated as needed for other searches.*

There is a single top-level parameter, “pastebinSearchURL”, that needs to be defined to use the Pastebin Handler. To generate this value:

1. Start the developer tools for your web browser.
2. Access one of the known search aggregators.
3. Search for the text “REPLACEME”.
4. Within the network traffic of the developer tools, look for a URL that contains “vElement”. Copy that link address and use it as the value for this parameter.

Known Pastebin search aggregators:

* <https://netbootcamp.org/pastesearch.html>
* <https://inteltechniques.com/osint/menu.pastebins.html>

### Source Handlers

The *sourceHandler* is a JSON array of source handlers that can be used within a particular domain. Each entry is a string.

To view the available handlers, first select the “Advanced Collector View” as shown in Figure 4. Then click on the “Handlers” button as shown in Figure 5. The Handlers page lists the available source handlers, document handlers, and annotators within OSKE.

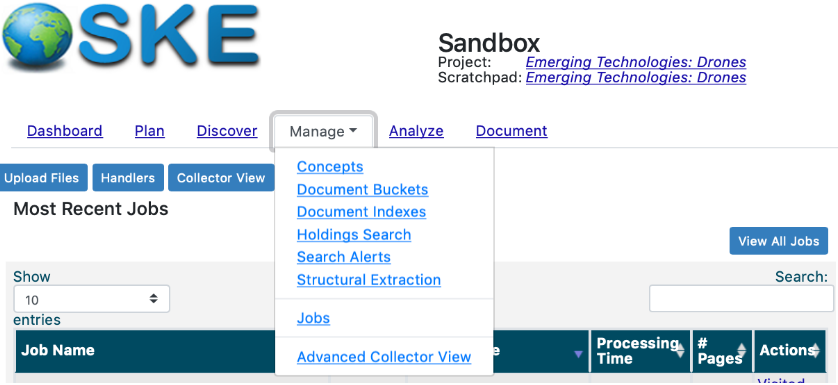


Figure 4 Accessing Advanced Collector View

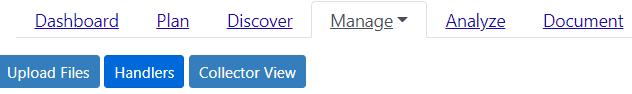


Figure 5 Access Handlers Page

### System Component

System Component has a set of parameters which define (at a very high-level) if that particular component is usable within a particular OSKE instance. If the value is set to true, then each parameter section will further define whether data is stored for a particular document area.

|  |  |
| --- | --- |
| Name | Description |
| systemComponent |  |
| .accumulo | Defines if Accumulo can be used to store raw collected data. |
| .kafka | Defines if JSON records of the collected data can be sent to Kafka brokers. |
| .hdfs | Defines whether or not HDFS is used to store RAW or JSOn data. |

Table 20 System Component Configuration

### Web Crawler

Web crawler defines the standard configuration that can be used for any jobs that utilize the web crawler (or one of the other source handlers that depend upon it such as feed). This allows parameters such as “respectRobots.txt” to be defined globally for the system or a specific domain. Refer to the web-source handler configuration online for the complete set of fields allowed. These parameters may be overridden by specific jobs.

Sample configuration:

|  |
| --- |
| "webCrawler": {         "maxPagesToFetch": -1,         "politenessDelay": 200,         "includeBinaryContentInCrawling": true,         "maxDepthOfCrawling": -1,         "userAgentString": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36",         "respectRobotsTxt": true,         "maxDownloadSize": 20000000     } |

Figure 6 Sample Web Crawler Configuration

### Video

The video configuration defines if the system will download video files.

|  |  |
| --- | --- |
| Name | Description |
| video |  |
| .utilize | Defines whether or not the domain allows videos to be downloaded |
| .url\_regex | JSON array of string values, which are regular expressions, that define whether or not a particular URL links to a video site such as YouTube. |

Table 21 Video Configuration

|  |
| --- |
| "video": {  "utilize": true,  "url\_regex": [  "vimeo.com/[0-9]+",  "youtu.be/\\w+",  "www.youtube.com/watch\\?v=",  "www.dailymotion.com/video/\\w+"  ]  } |

Figure 7 Sample Video Configuration

# Databases

OSKE must use both PostgreSQL and Elasticsearch database systems to function properly.

## PostgreSQL

The PostgreSQL database stores the configuration and meta-data for the OSKE instance. The file *database\_setup.sql* contains the scripts used to create the necessary tables. This script creates comments on most of the table and columns; refer to that file for specific details. Once these tables are created, the database user needs the standard “CRUD” permissions on all of the tables (select, update, insert, and delete).

PostgreSQL could be replaced by another relational database system. However, some of the queries related to retrieving current date and time would need to be updated as well one “upsert” statement.

The *Visited Pages* table will hold the largest amount of records as a record is created for each page visited.

## Elasticsearch

Elasticsearch is used to store the JSON records created from the collected data. These records contain the extracted text from the visited page, metadata associated with the page, provenance data, and extracted/enriched information such as named entities (people, organizations, etc.) or geographic locations.

For each domain in OSKE, three indexes are created within Elasticsearch. The naming convention is *domainCodeName*\_s*torageArea*. While *\_archive* indexes are created, they are not currently used. Additionally, depending the configuration, the *\_normal* and \_sandbox domains may just be combined into *\_normal*.

For smaller use cases (e.g., where a relatively small number of pages may be collected), the default configuration for Elasticsearch can be used. However, for larger instances, multiple servers may be used for Elasticsearch.

When configuring the memory used by Elasticsearch, no more than half of the available memory on the server for Elasticsearch should be used. Elasticsearch will then use the default system caching to improve performance when reading and writing data to disk. Additionally, maximum virtual heap sizes larger than 31GB should not be used. At this point, Java will begin to use a larger internal memory point scheme that increase memory demands – performance will scale better by using a server cluster at that point.

By default, Elasticsearch only allows a maximum of 1000 distinct fields to be indexed. As many of the metadata fields that can be collected (e.g., http headers, meta tags on an HTML page, etc.) are defined by the various sources, it is not difficult to exceed the maximum limit of fields. The default configuration for OSKE sets this maximum field limit to to 3,000. Additionally, the default OSKE configuration sets some specific fields as “not enabled” in the mapping. For fields that are not enabled, no corresponding indices in the underlying Lucene Search Engine exist, and, hence, those fields cannot be searched. Some other program to analyze those fields will need to be used.

Another common problem with Elasticsearch is that a field may be mapped to the wrong type. Unless a mapping has been specifically defined for a field, Elasticsearch infers the field type based upon the first value seen for a field. For example, if the system first sees a number in a field that generally holds text values, the text values will not be stored – one of the settings has Elasticsearch ignore fields where the types do not match correctly. Changing mappings when values exist is a non-trivial task as the index must be completely rebuilt. See [How to Update Elasticsearch Mappings through Re-indexing](https://www.elastic.co/guide/en/elasticsearch/reference/6.4/mapping.html#_updating_existing_field_mappings) for more details.

## Other Databases

OSKE can make use of other databases system. First, it can use ldap-based systems to perform user authentication as well as user searches. The system has also been developed to work with Apache Accumulo to store meta-data on crawled records as well as the underlying raw crawl data.

# External Services

OSKE makes use of external services to provide additional data and/or services for the platform.

## GeoCoding

To assign latitude and longitude coordinates for an address (can be full or partial), OSKE uses a couple of existing services based upon Nominatim[[17]](#footnote-17): OpenStreetMap[[18]](#footnote-18) and LocationIQ[[19]](#footnote-19). If necessary, an organization can establish their own Nominatim environment. For OpenStreetMap, an administrator should include some sort of organizational identifier in the user agent field. For LocationIQ, an administrator will need to register for an account at <https://locationiq.com/register>.

## Foreign Language Translation

To support foreign language translation capabilities for Discovery Sessions, OSKE utilizes Amazon’s Translate Service[[20]](#footnote-20). <https://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#access-keys-and-secret-access-keys> provides more details on AWS access keys.

Utilizing AWS’s translation API does cost money. Pricing details are available at <https://aws.amazon.com/translate/pricing/>.

## Microsoft Academic API

To search academic papers through Microsoft’s services, OSKE contacts their services within Azure’s Academic Knowledge API. To enable this capability, the identifying key must be provided in the configuration. Documentation and subscription details are available at <https://labs.cognitive.microsoft.com/en-us/project-academic-knowledge>. Currently, this service is free of charge.

## Google Oauth

The installation page covers configuring an account to support oauth authentication

# Hortonworks Data Platform

While the current documentation does not describe how to install the Hortonworks Data Platform[[21]](#footnote-21), the system was initially created to work with the platform. OSKE can stored crawled records in HDFS as well as send them to downstream services by using Kafka. Additionally, we explored using Zeppelin as interactive data analysis platform to analyze records stored in HDFS.

# Log Files

OSKE and the various components utilized produce a number of different log files.

## Daemon and Web Application Log Configuration

For both the daemon and the web application, administrators can configure the setting files in the “config” directory. As some of the components use different logging frameworks, both logging.properties and logging.xml need to be updated. The primary level settings are in Table 22.

|  |  |
| --- | --- |
| Level | Description |
| SEVERE | Significant error that needs to be examined. Will directly impact OSKE’s use. |
| WARNING | Processing issue. Often because of temporary issue or specific data. May need to be examined if re-occuring. |
| INFO | General information about the state of the applications to assist developers |
| FINE  FINER  FINEST | Detailed log messages. Generally, these are not utilized unless researching a specific problem |

Table 22 Log Levels

## Collector Daemon

The daemon log files are in the “log” directory for the collector. These are rotated daily through the use of the “rotatelogs[[22]](#footnote-22)” program provided by the Apache HTTP Server Project. This rotation is established in the “bin/start.sh” file.

## Collector Microservices

The Microservices use Docker’s logging mechanism. Generally, you will be able to use “docker logs *containerName”* to view the log files. <https://docs.docker.com/config/containers/logging/> provides more details for Docker.

## Collector Web Application

As the web application typically runs within Tomcat or some other servlet container, the log files for the web application are placed in that container’s established location. For Tomcat and using OSKE’s installation instructions, the log files are in /opt/tomcat/current/logs. Primarily, you will be log messages will be put into catalina.out, but you may need to look at the other files.

## Elasticsearch

For Elasticserch, log files are located at /var/log/elasticsearch on the relevant server unless otherwise configured differently in ElasticSearch.

## PostgreSQL

PostgreSQL log files are placed into /var/lib/pgsql/*VERSION*/data/log where “*VERSION*” should be replaced by the primary version (e.g., “10”) of the installation.

# Running OSKE

## Managing Domains

To add a new domain to OSKE, click on the “Manage Domains” button from the application home page. Users will only see this button if they are setup as “administrators in the system domain. To manually add a record in the PostrgreSQL database, you can perform the following:

|  |
| --- |
| INSERT INTO system\_user  (email\_id,name,domain\_instance\_name, role,status, status\_dt,  changed\_by\_email\_id)  VALUES ('*emailAddress*','u*serName*','system','administrator',  'active',now(),'*yourEmailAddress*'); |

Figure 8 SQL Command to Manually Create an Administrator Record

Within the application, additional administrators can be added in the “Manage -> Users” Screen for the “system” domain. Only administrators can be added to the “system” domain.

When a domain is created, the current user is automatically added as an administrator for the new domain. That person will then need to setup additional users (see section 9.2).

Currently, OSKE does not provide a mechanism to fully delete a domain. One alternative is to set “allowOnlineDomainPurge” to true for the domain and then perform “purge” on that domain within the “Manage Domains” area of OSKE. Further, the domain can be set to “Offline” to prevent further use.

To manually delete a domain, you will need to first delete the relevant records within the PostrgreSQL database (see Figure 8), delete the Elasticsearch indices (Figure 9), and the file storage areas for the domain (execute rm -rf on the directory named by the domain ID on the file root).

|  |
| --- |
| delete from collection where domain\_instance\_name='*domainID*';  delete from document\_bucket\_collaborator where domain\_instance\_name='*domainID*';  delete from collection\_collaborator where domain\_instance\_name='*domainID*';  delete from concept\_categories where domain\_instance\_name='*domainID*';  delete from concepts where domain\_instance\_name='*domainID*';  delete from discovery\_index where domain\_instance\_name='*domainID*';  delete from document\_bucket where domain\_instance\_name='*domainID*';  delete from domain where domain\_instance\_name='*domainID*';  delete from domain\_discovery\_session where domain\_instance\_name='*domainID*';  delete from domain\_discovery\_session\_execution  where domain\_instance\_name='*domainID*';  delete from job where domain\_instance\_name='*domainID*';  delete from job\_archive where domain\_instance\_name='*domainID*';  delete from job\_status\_history where domain\_instance\_name='*domainID*';  delete from project where domain\_instance\_name='*domainID*';  delete from job\_history where domain\_instance\_name='*domainID*';  delete from project\_document where domain\_instance\_name='*domainID*';  delete from search\_alert\_notification  where alert\_id in (select alert\_id from search\_alert where domain='*domainID*');  delete from search\_alert where domain='*domainID*';  delete from site\_crawl\_rule where domain\_instance\_name='*domainID*';  delete from structural\_extraction where domain\_instance\_name='*domainID*';  delete from system\_user where domain\_instance\_name='*domainID*';  delete from system\_user\_option where domain\_instance\_name='*domainID*';  delete from visited\_pages where domain\_instance\_name='*domainID*'; |

Figure 9 SQL Commands to Delete a Domain

|  |
| --- |
| DELETE *domainID*\_archive  DELETE *domainID*\_sandbox  DELETE *domainID*\_normal |

Figure 10 REST Commands to Delete Elastic Indexes for a Domain

## Managing Users

To manage users for a domain, an individual must either be an individual for that domain or for the “system” domain. To manage users for a specific domain, select “Users” under the “Manage” tab for the domain (or use the “system” domain to manage users for any of the domains).

Users records may not be fully deleted online – records can only be marked as “inactive” or “removed”. “inactive” is meant to be a temporary condition. If marked as “removed”, the user will no longer be visible in the “Manage User” page unless “show all users is selected”. If you need to remove a user record fully, you will need to delete the records in the system\_user and system\_user\_option tables.

# System Components and Updates

As with any software-based system, maintenance will a re-occurring task – to add functionality, correct issues, fix security issues, update components, and many other reasons.

For the Java based components, OSKE utilizes Apache Maven[[23]](#footnote-23) for its build system, which includes downloading and installing external dependencies. Each project stores the Maven configuration in a file called pom.xml. To check for component updates, execute

maven versions:display-dependency-updates

## Repository Layout

Table 23 presents the top-level directories in the OSKE repository along with their role description. Additional details are available in README.md files in various directories in the repository.

|  |  |
| --- | --- |
| Top-level Directory | Description |
| Collector | Contains the daemon as well as the business logic and persistence classes for OSKE. |
| CollectorWebApp | User interface specific code for OSKE utilizing web technologies. All external JavaScript files have been copied into this code base. |
| LAS-Common | Various java components that could be utilized in any Java based project (i.e., these packages and classes are not specific to OSKE) |
| LAS-Common-NLP | Text processing software, largely based upon Stanford’s CoreNLP[[24]](#footnote-24). |
| Micrsoservices | Series of REST-based services that extend OSKE’s functionality. These components are written in either Java or Python. In most cases, these services can execute outside of the OSKE context and provide value to other projects. |
| OSKE-Lite | Contains a docker-compose file and other configuration files that allows users to quickly establish a full OSKE environment for a single user. This environment can also be used as a development platform. |
| development | Contains a minimal docker-compose file that stands up the PostgreSQL and Elasticsearch databases, but with no other components. |
| documentation | Contains a quick start guide, user manual, install guide, and adminstrator’s guide. |
| externalProjects | Contains several external projects that have been copied into the repository for convenience.  Boilerpipe: <https://github.com/kohlschutter/boilerpipe>  crawler4j: <https://github.com/yasserg/crawler4j>  minIE: <https://github.com/gkiril/minie> |

Table 23 OSKE Repository Layout

## System Updates

For the development of OSKE, CentOS 7.x was used as the primary operating system. System updates for CentOS can be performed by executing “sudo yum update”. With the exception of Java and Elasticsearch updates, the OSKE developers did not discover any issues with system updates. For Java, the developers needed to restart the various java-based components that relied upon TLS to contact other sites (this is the Daemon and Web Application). The trust store file that contains the root certificates to establish trust had moved to a new directory location as the prior version of Java was removed from the operating system.

For Elasticsearch, the main issue for the past two major releases (6.x, 7.x) has been the planned removal of multiple types from Elasticsearch[[25]](#footnote-25). At one point, the developers did rely upon on multiple types in Elasticsearch, but types other than the storage of the primary crawl data was moved into the PostgreSQL database during the migration to Elasticsearch 6. The migration to Elasticsearch 7 required updating the various REST API calls to Elasticsearch to remove type information.

## OSKE Software (Daemon and Web Application)

OSKE utilizes Apache Maven to perform builds. The following presents the necessary commands to build each component. These builds do not have to occur on the same server as your deployment environment.

Build crawler4j:

cd install\_directory/OpenSourceKnowledgeEnrichment/externalProjects/crawler4j

mvn install

Build Boilerpipe:

cd install\_directory/OpenSourceKnowledgeEnrichment/externalProjects/boilerpipe/

mvn install

Build minIE

cd install\_directory/OpenSourceKnowledgeEnrichment/externalProjects/minIE

mvn install

Build LAS-Common:

cd install\_directory/OpenSourceKnowledgeEnrichment/LAS-Common

mvn install

Build LAS-Common-NLP

cd install\_directory/OpenSourceKnowledgeEnrichment/LAS-Common-NLP

mvn install

Build the Collector Daemon

cd install\_directory/OpenSourceKnowledgeEnrichment/Collector

mvn dependency:copy-dependencies package install

Build the Collector Web Application

cd install\_directory/OpenSourceKnowledgeEnrichment/CollectorWebApp

mvn package

Deploying the OSKE Daemon files:

Copy application jar file and associated libraries from the build area: (note the name change on the collector jar)

cp install\_directory/OpenSourceKnowledgeEnrichment/Collector/target/dependency/\* \

***OPENKE\_DIRECTORY\_SOFTWARE***/collector/lib/

cp install\_directory/OpenSourceKnowledgeEnrichment/Collector/target/Collector-0.0.2.jar \

***OPENKE\_DIRECTORY\_SOFTWARE***/collector.jar

Deploying the OSKE Web Application to Apache Tomcat:

1. Stop tomcat

systemctl stop tomcat

1. Copy the web application from the build directory

cp install\_directory/OpenSourceKnowledgeEnrichment/CollectorWebApp/target/CollectorWebApp-0.1.0-SNAPSHOT.war /opt/tomcat/current/webapps/collector.war

1. Start tomcat

systemctl start tomcat

## Microservices

OSKE uses a number of REST-based microservices which are deployed as Docker containers. Table 24 lists the images created for the system. The first column contains the image name(tag) that is created in the docker-compose files. Comments in the dockerfiles also suggest to use these names. The second column, “Images Used”, lists any base images (or intermediate images) used during the build processes. To minimize the image size, the build process will often use a full development environment, but then copy files to a smaller base image to reduce the size. The third column contains the location of the dockerfile used to build the image. The last column contains the container name that will be present if one of the system’s docker-compose files is used to start the application. If docker swarm is used for all of the components, then “openke-base” should be removed once components are built – otherwise the docker swarm will try to continually restart that image which was only present as a foundation for other components in the build process.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Images Used | Dockerfile | Docker Compose  Container Name |
| openke\_base | centos:7.5.1804 | Microservices/base\_images/Dockerfile\_base |  |
| openke\_daemon | openke\_base  centos:7.5.1804 | OpeKE-Lite/Dockerfile\_daemon | daemon |
| openke\_webapp | openke\_base  tomcat:8.5.34-jre8-alpine | OSKE-Lite/Dockerfile\_webapp | webapp |
| openke\_ms\_exif | centos:7.5.1804 | Microservices/service\_image\_exif/Dockerfile | exif-service |
| openke\_ms\_geocode | centos:7.5.1804 | Microservices/service\_text\_geo\_coding/Dockerfile | geocode-service |
| openke\_ms\_geotext | centos:7.5.1804 | Microservices/service\_text\_geo\_tagging/DockerFile | geo-service |
| openke\_ms\_microformat | python:3.7.3-stretch | Microservices/service\_html\_microformat/Dockerfile | microformat-service |
| openke\_ms\_nlp | openke\_base  openjdk:8u181-jdk | Microservices/service\_text\_nlp/Dockerfile | nlp-service |
| openke\_ms\_scrapper | node:12-slim | Microservices/service\_internet\_scraper/dockerfile |  |
| openke\_ms\_service\_structural | openke\_base | Microservices/service\_text\_contentExtraction/Dockerfile | structural-service |
| openke\_ms\_spacy | python:3.7.3-stretch | Microservices/service\_text\_ner\_spacy/Dockerfile | spacy-service |
| openke\_ms\_temporal | openke\_base  openjdk:8u181-jdk | Microservices/service\_text\_temporal/Dockerfile | temporal-servcice |
| openke\_ms\_textrank | centos:7.5.1804 | Microservices/service\_text\_analysis/Dockerfile | textrank-service |
| openke\_ms\_topic | openke\_base | Microservices/service\_text\_lda/Dockerfile | topic-service |
| openke\_ms\_whois | openke\_base  openjdk:8u181-jdk | Microservices/service\_internet\_whois/Dockerfile | whois-service |

Table 24 Internal Microservices used in OSKE

Any changes within the following components require a re-build of the openke\_base image:

* Collector
* LAS-Common
* LAS-Common-NLP
* externalProjects/boilerpipe
* externalProjects/crawler4j
* externalProjects/minIE

When the openke-base changes, it may be necessary to rebuild the other microservices/docker images that depend upon the openke-base image (see the “Images Used” column in Table 23). From a convenience standpoint, it may just be easier to rebuild all of the images (requires deleting the current images). One way to perform the re-builds (rather than each image individually) is to use the docker-compose files to perform the builds with the command:

docker-compose build --no-cache

The microservices do not need to be stopped to perform the build. The distribution includes two versions of the docker-compose.yml file. The version in Microservices contains all images except the database, elasticsearch, daemon, and webapp. This version is intended to manage the microservices used by the core OSKE components. The other version in OSKE-Lite contains all of the images and is meant to provide a complete working environment of OSKE for development or evaluation purposes.

For reference, Table 24 lists external images that are used within OSKE.

|  |  |
| --- | --- |
| Image | Docker Compose  Container Name |
| postgres:11.2 | database |
| elasticsearch:7.0.0 | elasticsearch |
| kibana:7.0.0 | kibana |
| dbpedia/spotlight-english | dbpedia-service |
| redis:4.0.11 | openke\_redis |
| jacopofar/wordnet-as-a-service | wordnet-service |

Table 25 External Microservices used in OSKE

# Docker

OSKE primary utilizes Docker to containerize/execute microservices. OSKE-Lite runs under a single docker-compose[[26]](#footnote-26) file. The scripts used to establish a development environment establish containers that run Elasticsearch and PostgreSQL.

Below are common commands you may find helpful. These commands must be run as root or some other user that can execute the docker commands[[27]](#footnote-27).

|  |  |
| --- | --- |
| Situation | Command |
| Access running container | docker exec -it *container\_id* bash |
| Delete all containers | * docker rm $(docker ps -a -q) |
| Delete all images | docker rmi $(docker images -q) |
| Export an image | docker save --output name.tar *container\_id* |
| Import an image | docker load –-input name.tar |
| Remove dangling images | docker rmi $(docker images -f "dangling=true" -q) |
| Remove exited containers | docker rm -v $(docker ps -a -q -f status=exited) |
| Restart all containers | docker restart $(docker ps -a -q) |
| Start container with a different entry point | docker run -it –entrypoint=sh *imageTag* |
| Stop all containers | docker stop $(docker ps -a -q) |
| Update a container’s restart policy | docker update --restart=always *container\_id* |
| Update all containers restart policy | docker update --restart=always $(docker ps -a -q) |
| View container status | docker ps -a |
| View log files | docker logs *container\_id* |

Figure 11 Common Docker Commands

For the docker images, you should set the restart policy to “always”. <https://docs.docker.com/config/containers/start-containers-automatically/>

To scale docker instances, you can perform this utilizing Docker Swarm or Kubernetes. Docker Swarm is installed with Docker. Figure 12 demonstrates scaling the NLP service to three replicas. Note that the container to be scaled should be first stopped first. These commands must be run as the root user.

|  |
| --- |
| docker swarm init  export SERVICE\_URL=http://0.0.0.0:9001/nlp/  docker stop nlp-service  docker service create --replicas 3 -p 9006:9001 --name nlp-service las/service\_nlp |

Figure 12 Commands to Scale a Docker Service

# Security Notes

The default installation as specified by the instructions and scripts does not utilize SSL communications among the different components. Further, the communication with ElasticSearch and the Microservices are not authenticated. As such, the system should be running in its own subnet where external network traffic may not directly contact these services.

If you use the http\_header option, you must ensure that the application server only accepts connections from appropriate sources (e.g., nginx). Otherwise, users can inject their own http\_header value to spoof users.

# Usage Notes

To create RSS Feeds (for Jobs) that utilize Google Alerts, you can establish a custom RSS Feed from Google:

1. Go to [www.google.com](https://www.google.com/) and search for the topic you want to create an RSS feed for
2. On the search results page that appears, click the **News** tab.
3. Scroll to the bottom of the News results and click **Create Alert**.
4. On the Alerts page, click **Show options**.
5. From the **Deliver to** drop-down menu, choose **RSS Feed**. There are also other options you can customize in the dropdown lists in this section.
6. Click **Create Alert**.
7. Then you can select the **RSS** button on the next page to copy the address to use in OSKE.

**Note**: You can also go directly to [Google Alerts](https://www.google.com/alerts) to create an RSS feed on any topic, include news.

1. Open Source Knowledge Enrichment is referred to as both OpenKE and OSKE throughout this document and the project. [↑](#footnote-ref-1)
2. OSKE-Lite is a minimal distribution designed for evaluation or single user. [↑](#footnote-ref-2)
3. <http://nginx.org/> [↑](#footnote-ref-3)
4. A “salt” is a series of random data used to further randomize passwords when they are hashed. <https://en.wikipedia.org/wiki/Salt_(cryptography)> [↑](#footnote-ref-4)
5. <https://www.elastic.co/products/elasticsearch> [↑](#footnote-ref-5)
6. <https://www.postgresql.org/> [↑](#footnote-ref-6)
7. <https://en.wikipedia.org/wiki/Network_File_System> [↑](#footnote-ref-7)
8. <https://aws.amazon.com/efs/> [↑](#footnote-ref-8)
9. <https://en.wikipedia.org/wiki/Inotify> [↑](#footnote-ref-9)
10. A domain is an overall area of interest that is being investigated within the system. [↑](#footnote-ref-10)
11. <https://www.json.org/> [↑](#footnote-ref-11)
12. This is the directory in which the OSKE Collector Daemon has been deployed. Refer to the OSKE installation guide for a description of the Daemon an how that application has been deployed. You can also search for that file using a command such as find / | grep system\_properties.json [↑](#footnote-ref-12)
13. <https://www.ncbi.nlm.nih.gov/pubmed/> [↑](#footnote-ref-13)
14. <http://nominatim.org/> [↑](#footnote-ref-14)
15. https://en.wikipedia.org/wiki/WHOIS [↑](#footnote-ref-15)
16. When the application was first developed, the native interface was used to connect to Elasticsearch under the naïve belief that this interface was the best solution based upon prior experiences with JDBC drivers. However, the interface proved to be extremely brittle as the application’s Elasticsearch libraries had to match exactly with those used by the Elasticsearch installation. Elasticsearch (at least at some point) used object serialization to pass information among the different members of a cluster. [↑](#footnote-ref-16)
17. <http://nominatim.org/> [↑](#footnote-ref-17)
18. <https://wiki.openstreetmap.org/wiki/Nominatim> [↑](#footnote-ref-18)
19. <https://locationiq.com/> [↑](#footnote-ref-19)
20. <https://aws.amazon.com/translate/> [↑](#footnote-ref-20)
21. <https://hortonworks.com/products/data-platforms/hdp/> [↑](#footnote-ref-21)
22. <https://httpd.apache.org/docs/2.4/programs/rotatelogs.html> [↑](#footnote-ref-22)
23. <https://maven.apache.org/> [↑](#footnote-ref-23)
24. <https://stanfordnlp.github.io/CoreNLP/> [↑](#footnote-ref-24)
25. <https://www.elastic.co/guide/en/elasticsearch/reference/master/removal-of-types.html> [↑](#footnote-ref-25)
26. <https://docs.docker.com/compose/> [↑](#footnote-ref-26)
27. <https://docs.docker.com/engine/reference/commandline/docker/> [↑](#footnote-ref-27)