

# Configuration Property Guide

## Contents

- [SWEET](#)
- [Central Search](#)
- [Search \(Central and Node\)](#)
- [SWEET & Search \(node and central\)](#)
- [SWEET, Search \(Node and Central\) and Glossary](#)
- [SPARQLER](#)

This guide describes the configuration properties for the eagle-i sweet and search applications, and the public sparql endpoint that may be installed alongside the repository. This document is broken up into several sections. Properties are grouped by application; shared properties form a separate group. Each group lists the required properties and optional properties separately. Required properties must be set in order for the application to function.

The default name for the property file to use is `eagle-i-apps.properties` which lives in the eagle-i configuration directory, e.g. `/opt/eaglei/conf`

There are some properties which are listed in **red**. These properties contain sensitive information, such as credentials to the repositories. We recommend that you place these properties in a separate file `eagle-i-apps-credentials.properties` and place the file in a directory that is only accessible to `ROOT` to further protect the credentials, e.g. `/opt/eaglei/.config`

## SWEET

In addition to the properties below, which are specific to the SWEET application, there are additional properties, both required and optional, that are shared with other applications listed [here](#) and [here](#).

### Required Properties

- **`eaglei.datatools.repository.url`**  
The base-url of the eagle-i repository against which the SWEET runs. This property needs to be provided even if the SWEET runs in the same server as the repository  
*Default:* None  
*Requirements:* **MUST** be an HTTPS end-point, since the SWEET back-end uses HTTP basic authentication. `localhost` **cannot** be used  
*Example:* `https://somenode.eaglei.net`
- **`eaglei.ui.centralSearch.url`**  
The full url for the central search application. This property is used to set the links in the menu of the SWEET application  
*Default:* `http://search.eagle-i.net/central`  
*Requirements:* None  
*Example:* `http://search.eagle-i.net/central`
- **`eaglei.catalyst.user`**  
A repository user that has anonymous access. This user is used for the SWEET webservices. If you would like to expose contact information to the webservices user, add this user to the `Contact Properties ACL` via the repository's administrative panel.  
*Default:* None  
*Requirements:* Valid repository username  
*Example:* `specialRepoUser`
- **`eaglei.catalyst.password`**  
The corresponding password for the above mentioned user.  
*Default:* None  
*Requirements:* Valid password  
*Example:* `specialRepoPW`

### Optional Properties

- Global Data Repository  
If you would like to use a repository to store instances that are global (in programming parlance), you will need to set the following properties in all SWEET applications that will be using the global repository to supply the global information.  
**Please note** that if your global repository will be hosted in the same Tomcat that is also serving a central search application, you will **also** need to follow the optional properties instructions for a central search application to co-exist with a SWEET application.
  - **`eaglei.datatools.uses.globals`**  
True if the SWEET application is supposed to also be gathering information from a global repository.  
*Default:* False  
*Requirements:* Boolean value
  - **`eaglei.datatools.globalRepository.url`**  
The full url for where the global data repository resides.  
*Default:* None  
*Requirements:* None  
*Example:* `https://global.eagle-i.net`
  - **`eaglei.datatools.globalPolling.frequency`**  
The number component of the frequency at which SWEET should poll the global data repository for updated information.  
*Default:* 24  
*Requirements:* Integer value

- **eaglei.datatools.globalPolling.unit**  
The time unit component of the frequency at which SWEET should poll the global repository for updated information.  
*Default:* HOURS  
*Requirements:* **MUST** be a [java.util.concurrent.TimeUnit](#), i.e. DAYS, HOURS, MINUTES, SECONDS, MILLISECONDS, MICROSECONDS, NANOSECONDS
- **eaglei.datatools.globals.user**  
The global repository username for the automated harvesting of data from the global repository to the local proxy graph for storing the global information, which is then used by the sweet application. This user needs to have **add**, **remove** and **read** access to the NG\_GlobalProxy graph on the client repository, where the client repository is where the local sweet application uses as it's home repository. This user needs to have **read** access to the Default Workspace on the global repository.  
*Default:* None  
*Requirements:* Valid repository user on both the client and global repository. The username **must** match in both repositories.  
*Example:* [globalUser](#)
- **eaglei.datatools.globals.password**  
The password for the user specified by `eaglei.datatools.globals.user`.  
*Default:* None  
*Requirements:* Valid repository password for the user on both the client and global repository. The password **must** match in both repositories.  
*Example:* [globalPassword](#)

## Central Search

The following properties **only** to a central installation of the search application. Be sure to also look at the additional properties, both required and optional, that are common to both installations of search [here](#) and also those that are shared with other applications listed [here](#) and [here](#).

### Required Properties

- **eaglei.search.is.central**  
Indicates the type of search application installation. A value of true indicates the search application is a central search, false a node search application.  
*Default:* False  
*Requirements:* Boolean value
- **eaglei.noderegistry.url**  
The fully qualified URL to where the node registry service exists. This is used to inform the central search application as to what nodes it needs to harvest and index for searching.  
*Default:* None  
*Requirements:* A resolvable URL  
*Example:* <http://search.eagle-i.net/node-registry>

### Optional Properties

- **eaglei.search.harvester.polling**  
The time period, in milliseconds, for the central search to perform a harvesting operation.  
*Default:* 30000  
*Requirements:* Integer value
- Global Repository  
If you would like to use a global repository to store instances that are global (in programming parlance), **and** your global repository will be hosted in the same Tomcat that is also serving a central search application, you will **also** need to define the following properties for the sweet application to co-exist with central search.
  - **eaglei.datatools.central.coexist**  
Indicates whether or not a sweet application will be co-existing in the same tomcat as a central application.  
*Default:* FALSE  
*Requirements:* Boolean value
  - **eaglei.datatools.central.coexist.filename**  
The name of the secondary non-credentialed property file that contains the properties for the sweet application.  
*Default:* None  
*Requirements:* A fully qualified filename
- **eaglei.identity.url**  
If login is required for the application, the location of where the identity service exists **must** be specified here.  
*Default:* None  
*Requirements:* A fully qualified URL  
*Example:* <https://search.eagle-i.net/eagle-i-webapp-identity-service/identity>

## Search (Central and Node)

The following properties apply to both a central and a node installation of the search application. Be sure to also look at the additional properties that are shared with other applications listed [here](#) and [here](#).

### Required Properties

None

## Optional Properties

- **eaglei.search.requires.login**  
Indicates whether or not the search application requires a login. In the case of a node search application, login is performed by querying the underlying repository's whoami graph. Currently, in the case of a central application will require a functional instance of the identity-service. Login for the central application will be done using the identity-service authentication.  
*Default:* True  
*Requirements:* Boolean value
- **eaglei.search.logout.url**  
If login is required for the search application, this property can be used to set where the user should be redirected to upon logging out of the application.  
*Default:* None  
*Requirements:* A resolvable URL  
*Example:* <http://www.eagle-i.net>
- **eaglei.dev.mode**  
For developers to experiment with certain features that are not ready for production. To be used in conjunction with code that references `SearchApplicationContext.getInstance().isDevMode()`.  
*Default:* False  
*Requirements:* Boolean value
- Search Usage Database Module  
If you would like to make use of the asynchronous logger to have more specific search logs.
  - **eaglei.logger.jdbc**  
The prefix to the connection url for the underlying database to specify the type of connector and database. Be sure to **include** the trailing double slashes. For example: `jdbc:mysql://`.  
*Default:* None  
*Requirements:* Valid prefix in the form of: `jdbc:database type://`  
*Example:* `jdbc:mysql://`
  - **eaglei.logger.host**  
The hostname for the database where the asynchronous logger should send log messages to, including any port number.  
*Default:* None  
*Requirements:* Valid hostname and a port number, if applicable, in the form of: `hostname` or `hostname:1234`, where 1234 is the port number for accessing the database.  
*Example:* `somehostname:1234` or `hostname`
  - **eaglei.logger.database**  
The name of the database to connect to, which will be where the asynchronous logger writes the log messages to.  
*Default:* None  
*Requirements:* A valid database that has been configured per the instructions for the asynchronous logger module.  
*Example:* `searchLogDB`
  - **eaglei.logger.database.user**  
Username to use for connecting to the database used by the asynchronous logger. This user must have the requisite permissions for writing to the database.  
*Default:* None  
*Requirements:* A valid user with the correct permissions  
*Example:* `dbUser`
  - **eaglei.logger.database.password**  
Password for the above mentioned username.  
*Default:* None  
*Requirements:* A valid password for the user.  
*Example:* `dbpassword`

## SWEET & Search (node and central)

The following properties apply to SWEET as well as both the node and central installation of the search application.

### Required Properties

- **eaglei.model.url**  
The full url for the model service that provides ontology term suggestion lists and access to the model class information.  
*Default:* None  
*Requirements:* None  
*Example:* <https://search.eagle-i.net/model>

### Optional Properties

- **eaglei.connection.acceptAllCerts**  
This allows self signed certificates in development and test environments, for http communications that are programatically handled (e.g. between sweet/search and repo).  
**DO NOT** set this to true in a production environment: it presents a security risk.  
*Default:* false  
*Requirements:* None
- Google analytics  
If you are using google analytics to track page views, set the following property.
  - **eaglei.ui.analyticsId**  
The google analytics ID used for tracking page views.  
*Default:* None

Requirements: Valid google analytics ID  
Example: [asdf1234](#)

# SWEET, Search (Node and Central) and Glossary

## Required Properties

- Feedback Module  
Feedback mechanism that allows users of the SWEET, Search and Glossary applications to provide feedback about their use to you and your developers. We have parameterized these properties to allow to you to use either JIRA for tracking the issues or have an email sent.
  - **eaglei.feedback.method**  
Specify which method to use for submitting feedback. Be sure to set the corresponding required properties depending on which method you are using.  
Default: jira  
Requirements: Must be either 'jira' or 'email'
  - **eaglei.feedback.jira.url** **TODO - rename to non Jira-specific**  
If you are using JIRA, this will be the full url for connecting to JIRA.  
If you would like to have emails sent, this will be the full hostname for sending the email.  
Default: None  
Requirements: A valid url, including any port information needed to connect to the issue client. OR A valid hostname needed to send the email.  
Example: <http://jira.eagle-i.net:8080/rpc/soap/jirasoapervice-v2>
- Feedback Module (Jira)
  - **eaglei.feedback.jira.user**  
If the method for submitting feedback is **JIRA**, this is the corresponding JIRA user that will be used for the submission. If the method selected is **EMAIL**, this is the user for sending the email.  
Default: None  
Requirements: A valid JIRA User or EMAIL User.
  - **eaglei.feedback.jira.password**  
The password for the above mentioned user.  
Default: None  
Requirements: A valid password for the JIRA User or EMAIL User
- Feedback Module (EMAIL)  
The following properties are required for a functioning feedback mechanism using email. In order to use this method, be sure to set the optional property specifying the method to use. If you choose to use the default JIRA method for submitting feedback, you do not need to use any of the following properties.
  - **eaglei.feedback.email.from**  
The email to use as the from address of the email. This email will also be used in the even the email message is bounced.  
Default: None  
Requirements: A valid email address  
Example: [email@from.me](mailto:email@from.me)
  - **eaglei.feedback.email.to**  
The email where the feedback should be sent to.  
Default: None  
Requirements: A valid email address  
Example: [email@to.you](mailto:email@to.you)

## Optional Properties

- Feedback Module (Jira)
  - **eaglei.feedback.jira.projectKey**  
The key for the JIRA project to use for the issues submitted by the feedback module.  
Default: FBK  
Requirements: None
- Feedback Module (EMAIL)  
The following properties are required for a functioning feedback mechanism using email. In order to use this method, be sure to set the optional property specifying the method to use. If you choose to use the default JIRA method for submitting feedback, you do not need to use any of the following properties.
  - **eaglei.feedback.email.port**  
The port to use for sending the email.  
Default: 25  
Requirements: Valid port for sending the email.
  - **eaglei.feedback.email.ssl**  
Indicates whether or not SSL should be used for sending the email.  
Default: false  
Requirements: boolean value

# SPARQLER

A Public SPARQL Endpoint (aka "sparqler") may be installed, but is not required. If it is to be installed, certain properties must be set, and others may be set. The sparqler is a repository containing the public data (and meta-data) copied from a (non-public) repository (its source-repository). The sparqler is kept in sync with its source-repository by copying all updates (among the public data) from the source-repository to the target sparqler-repository. The properties (if set) are found in three different files.

## Required Properties

These properties provide the synchronization program access to source repository and its target sparqler repository. In a normal installation, the two repositories run on the same node, indeed, within the same web-server (tomcat). For security reasons it is not recommended for the same user-credentials to be used for both source-repository and sparqler.

In file *eagle-i-apps.properties*:

- **eaglei.sparqler.source.URL**  
The base-url of the eagle-i repository against which the data tools runs. This property needs to be provided even if the data tools back-end runs in the same server as the repository  
*Default:* None  
*Requirements:* **Must** be the base-url of the source repository. Must use protocol HTTPS, since privileged access is required to the source repository. localhost **cannot** be used  
*Example:* <https://myRepository.net/>
- **eaglei.sparqler.target.URL**  
Specifies the path to the target (sparqler) repository, which is to mirror the source repository.  
*Default:* None  
*Requirements:* **Must** be the base-url of the target (sparqler) repository. Must use protocol HTTPS, since privileged access is required to the target (sparqler) repository. localhost **cannot** be used  
*Example:* <https://mySparqlerRepository.net/sparqler/>
- **eaglei.sparqler.lastSynchronizedDateFile**  
The local path to the file in which the timestamp of the most recently synchronized data is stored.  
*Example:* [/opt/eaglei/eagle-i-sparqler.last-sync-date.properties](#)  
*Requirements:* the tomcat-user **must** have permission to write to this file (and to create it if it does not yet exist).

In file *eagle-i-apps-credentials.properties*:

- **eaglei.sparqler.source.user**  
The user-name under which the synchronizer obtains data from the source repository.  
*Default:* None  
*Requirements:* the named user must have sufficient access to use the "harvest" and "sparql" requests.  
*Example:* [myUserName](#)
- **eaglei.sparqler.source.password**  
The password to be used with the user-name under which the synchronizer obtains data from the source repository.  
*Default:* None  
*Requirements:* none  
*Example:* [myPassword](#)
- **eaglei.sparqler.target.user**  
The user-name under which the synchronizer adds data to or removes data from the target sparqler repository.  
*Default:* None  
*Requirements:* the named user must have sufficient access to use the "graph" request.  
*Example:* [mySparqlerUserName](#)
- **eaglei.sparqler.target.password**  
The password to be used with the user-name under which the synchronizer adds data to or removes data from the target sparqler repository.  
*Default:* None  
*Requirements:* None.  
*Example:* [mySparqlerPassword](#)

## Optional Properties

In file *eagle-i-apps.properties*:

- **eaglei.sparqler.syncData.captureFile**  
Used for test- and debugging-purposes only. The base local path to files in which the data harvested from the source-repository, the data added to, and the data deleted from the sparqler repository.  
*Default:* None. If unset, no data is dumped to a file.  
*Requirements:* None.  
*Example:* [/opt/eaglei/data/syncCapture](#)

## Optional Properties governing scheduling of synchronization

The sparqler webapp automatically synchronizes itself with its source-repository on a regular schedule. Various properties may be set in repository's home-directory's configuration.properties file to modify the default behaviour of the scheduling.

At repository startup, the sparqler waits until both itself and its source-repository are up and running, and then sets a repeating timer to run the synchronizer at regular intervals.

- **eaglei.sparqler.sync.repeat-period**  
The time in minutes between starting runs of the synchronizer.  
*Default:* 1440 (i.e. 24 hours)  
*Requirements:* None.

- **eaglei.sparqler.sync.mgr.repositories-timeout**

The initial time in minutes the synchronizer is to wait if repositories are not yet available at startup. After the wait, if they are still not available, the wait-time is doubled, and so on, for the specified number of times. The maximum wait time is therefore  $((\text{waitTime} * 2^{\text{numTries}}) - 1)$  minutes.

*Default:* 1

*Requirements:* None.

- **eaglei.sparqler.sync.mgr.repositories-num-tries**

The maximum number of times to try to contact the source- and target-repositories (see preceding property).

*Default:* 2

*Requirements:* None.

- **eaglei.sparqler.sync.mgr.stop-wait**

At shutdown, the maximum time (in minutes) to wait for the synchronizer timer to shut down gently before resorting to killing the thread.

*Default:* 2

*Requirements:* None.