

DG EMPL

**New EURES Regulation**

NCO Default Implementation Modules V0.81



Table of Contents

[1 Purpose of the Document 6](#_Toc499899132)

[2 Obligations Covered by the Modules 7](#_Toc499899133)

[3 Default Implementation Description 9](#_Toc499899134)

[3.1 Intermediate Repository 9](#_Toc499899135)

[3.2 Modules 9](#_Toc499899136)

[3.2.1 DB Converter Module 9](#_Toc499899137)

[3.2.2 NCO Input API Module 10](#_Toc499899138)

[4 Implementation Possibilities 11](#_Toc499899139)

[4.1 Deploying Both Modules 11](#_Toc499899140)

[4.2 Deploying the NCO Input API Module Only 11](#_Toc499899141)

[4.3 Deploying the DB Converter Module Only 12](#_Toc499899142)

[4.4 Deploying Multiple DB Converter Modules 12](#_Toc499899143)

[4.5 Deploying the REST Push API Module 12](#_Toc499899144)

[4.6 Allowing Direct Writing in the Intermediate Repository 13](#_Toc499899145)

[4.7 Adapting the Code 13](#_Toc499899146)

[5 Prerequisites 14](#_Toc499899147)

[5.1 Intermediate Repository 14](#_Toc499899148)

[5.2 DB Converter Module 14](#_Toc499899149)

[5.3 NCO Input API Module 14](#_Toc499899150)

[6 Installation and Configuration 15](#_Toc499899151)

[6.1 Intermediate Repository 15](#_Toc499899152)

[6.2 DB Converter Module 15](#_Toc499899153)

[6.2.1 Global configuration 15](#_Toc499899154)

[6.2.2 Configuration of the Source Database 17](#_Toc499899155)

[6.2.3 Configuration of the Intermediate Repository 17](#_Toc499899156)

[6.2.4 Configuration of the queries 19](#_Toc499899157)

[6.2.5 Deployment of the module 19](#_Toc499899158)

[6.3 NCO Input API Module 21](#_Toc499899159)

[6.3.1 Global Configuration 21](#_Toc499899160)

[6.3.2 Configuration of the Intermediate Repository 22](#_Toc499899161)

[6.3.3 Deployment of the module 22](#_Toc499899162)

[7 Operating Instruction 26](#_Toc499899163)

[7.1 Intermediate Repository 26](#_Toc499899164)

[7.1.1 Clearing the intermediate repository 26](#_Toc499899165)

[7.2 DB Converter Module 27](#_Toc499899166)

[7.3 NCO Input API Module 27](#_Toc499899167)

[8 Release Note 29](#_Toc499899168)

[8.1 Functionalities Added in Default Implementation v0.3.0 30](#_Toc499899169)

[8.2 Functionalities Added in Default Implementation v0.4.0 30](#_Toc499899170)

[8.3 Functionalities Added in Default Implementation v0.5.0 31](#_Toc499899171)

[8.4 Functionalities Added in Default Implementation v0.6.0 31](#_Toc499899172)

[8.5 Functionalities Added in Default Implementation v0.7.0 32](#_Toc499899173)

Abbreviations and Acronyms

|  |  |
| --- | --- |
| Abbreviation | Meaning |
| API | Application Programming Interface |
| CV | Curriculum Vitae |
| DB | Database |
| DDL | Data Definition Language |
| HTTPS | Hypertext Transfer Protocol Secure |
| JAR | Java Archive |
| JNDI | Java Naming and Directory Interface |
| JV | Job Vacancy |
| NCO | National Coordination Office |
| PES | Public Employment Service |
| PrES | Private Employment Service |
| REST | Representational State Transfer |
| SQL | Structured Query Language |
| XML | Extensible Markup Language |
| WAR | Web application Archive |

*Table 1 - Abbreviations and acronyms*

Reference Documents

This section contains the list of all referenced documents. When referring to any of them, the bracketed reference will be used in the text, such as [[R01](#R01)].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ref. | Title | Reference | Version | Date |
| R01 | EURES New regulation | http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:107:TOC | N/A | N/A |
| R02 | EURES formats and standards specification - part 1  Job vacancy data standard description | EFSS\_JV | 1.03 | 24/11/2017 |
| R03 | EURES formats and standards specification - part 2  Job seeker profile data standard description | EFSS\_JSP | 1.03 | 24/11/2017 |
| R04 | EURES Functional Message Exchange Specifications | EURES-2018-INPUT-API-FMES | 1.3.1 | 11/07/2017 |
| R05 | Query mapping document | N/A | 5.10 | N/A |

*Table 2: Reference Documents*

# Purpose of the Document

This document serves these purposes:

* + - * Provide a high-level description of which IT obligations of the Member States could be covered by the “NCO default implementation modules” offered by DG EMPL;
      * Describe the intermediate repository, which is a database required for the use of each module;
      * Describe the different modules and some possible implementation possibilities;
      * List the technical requirements for the intermediate repository and for each module;
      * Provide installation and configuration instructions;
      * Provide instructions on the operational usage of the intermediate repository and of each module in case it provides a user interface, admin features or logs to consult;
      * Provide an overview of the compatible and tested environments.

The first part of this document is targeted at IT decision makers to determine whether the provided modules should/could be used for their organisation.

The requirements allow determining the feasibility of each module within your infrastructure.

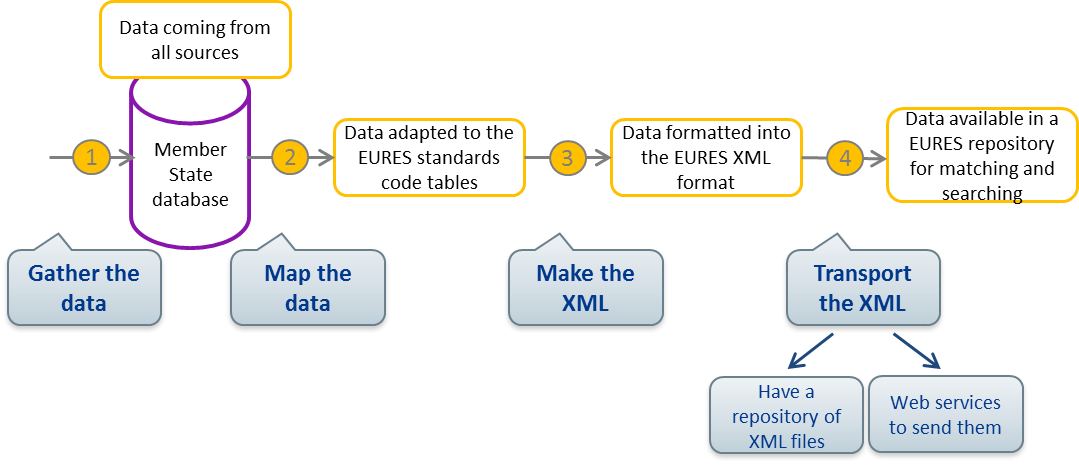
The last parts are targeted at IT operators that need to install and operate the modules.

# Obligations Covered by the Modules

Member states have the obligation to transfer job vacancies and jobseeker profiles (CVs) to DG EMPL [R01]. They must implement the “Uniform exchange system” which is described by several specification documents [R02][R03][R04]. The modules offered by DG EMPL provide support to implement parts of those documents.

The Member States have an obligation to produce a valid XML file of each job vacancy and jobseeker profile according to the EURES Standards and Formats specifications [R02][R03] and to transport it. This can be detailed in these steps:

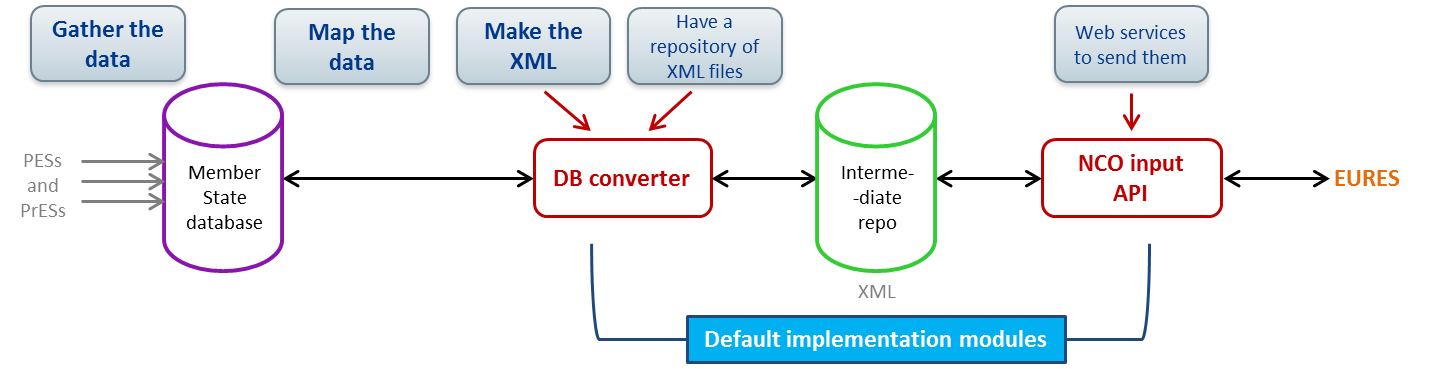
1. You need to aggregate the records and the metadata of your sources (i.e. of your PESs and PrESs) into the Member State database;
2. You need to map the data from the codifications used in the Member State database to the code tables that are required by the EURES standards;
3. You need to build valid XML files that comply with the structure required by the EURES standards, including cardinality and business rules;
4. You need to maintain an up to date inventory of these XML files to be transferred to DG EMPL on request. This includes keeping timestamps of creation and modification of these XML files, as well as a short-term inventory of the unique ID’s of closed job vacancies and/or jobseeker profiles;
5. Finally, you need to implement the web services described in the new EURES Functional Message exchange specifications [R04] that will transfer the inventory of (changed) records ID’s and/or their XML files to DG EMPL.



The modules offered by DG EMPL can cover a large part of these obligations if they are installed and configured in a proper way.

The **gathering** of the data of the sources and the **mapping** of this data are **not** covered by the modules. This will remain an obligation for which each Member State must implement its own solution.

The modules however are able to form a valid XML file with the mapped data, and maintain the repository of these files in an inventory, called the intermediate repository, with the required metadata. Finally, they implement all the needed web services to transfer the ID’s and XML files from the intermediate repository to DG EMPL.



**Important note: the modules are offered by DG EMPL to support the Member States. The usage of these modules does not remove the obligations on the Member States. In case one of the modules cannot be deployed and/or does not function as expected in the specific IT environment of your organisation, you will have to develop your own solution for the module that fails.**

The default implementation is designed for maximum modularity to allow a Member State to benefit the most of each solution offered, but also to provide the mitigation path in case one module fails.

**This version of default implementation is dealing with the transfer of JVs and Jobseeker Profiles (CVs).**

# Default Implementation Description

## Intermediate Repository

The NCO default implementation requires the configuration of an **intermediate repository** between the Member State database and the EURES database. This repository is deployed in the environment of the Member State. Its purpose is to store the records of the Member State in the XML format together with the metadata.

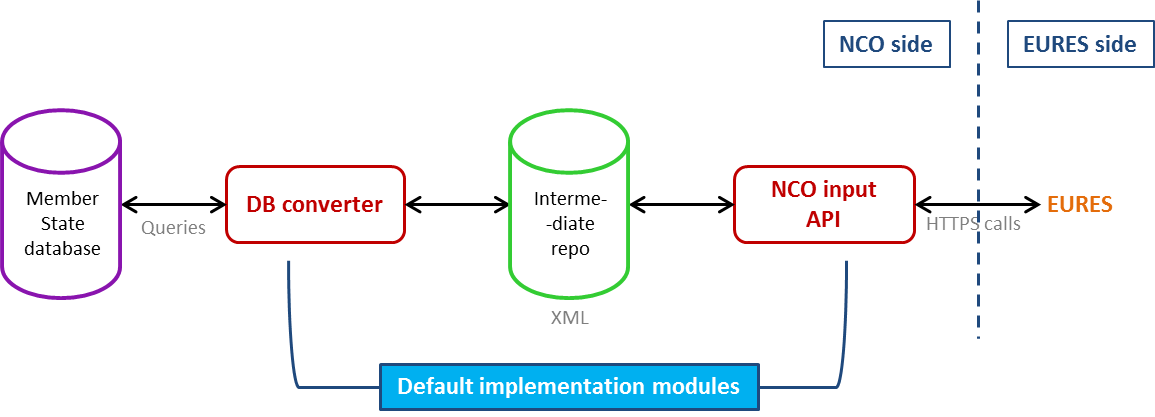
Note that the requirements for the structure of the intermediate repository can evolve over time. A version will therefore be associated to this repository, or, more precisely, to its structure. The use of the latest version can be mandatory or not, depending on the changes made. Also, the version of the intermediate repository will probably affect the versions needed for the default implementation modules.

For each intermediate repository version, the scripts enabling to configure this database will be provided to the Member States.

## Modules

The default implementation is made of two modules: the **DB converter module** and the **NCO input API module**.

The role of these modules is depicted on the figure below and their functionalities are described in the next sections.



Note that the modules of the default implementation will evolve over time and new functionalities will be added. A version will thus be associated to each module, the use of the latest one being mandatory or not, depending on the modifications made. New modules could also be implemented in the future.

### DB Converter Module

The DB converter module allows extracting the records data from the Member State database, converting it into the EURES XML format and storing the result into the intermediate repository.

To use it, the Member State will therefore need to configure some queries to extract the data from its database [R05]. In these queries, the codes of the codifications defined in the EURES standards must be provided and not the ones of the national codifications of the Member State. An easy way to achieve this objective is to create mapping tables between the national codifications and the EURES standards ones. In so doing, these mapping tables can be directly used in the queries to return the desired code in the EURES standards codification.

Note that, in order to improve the performances of the DB converter module, the conversion of data into the XML format is multi-threaded.

This module also implements the synchronization of data between the Member State database and the intermediate repository. It will in fact regularly extract the records that have been created, modified or closed since the last synchronization, and update the intermediate repository to reflect the data of the Member State database. For this purpose, the Member State will have to provide in their database the date and time corresponding to the creation, last modification and closing of each record inside this database. Note that the period of synchronization between the Member State database and the intermediate repository is configurable.

The DB converter module also handles the metadata associated to the records. First of all, it replicates the one contained in the Member State database (as the record reference, the timestamps or the status for example), which should have the imposed format. Moreover, to implement correctly the NCO input API, the intermediate repository must contain the creation timestamp corresponding to the creation of the record in this repository. It must also contain the last modification and closing timestamps corresponding to the last modification and to a potential closing in this repository. The DB converter module will therefore add this information in the intermediate repository.

The first versions of this module will only produce XML files containing EURES technical minimum fields and EURES conformant mandatory fields for job vacancies, in one language. The management of the optional fields and the possibility to give the translated content of some fields will be available later on.

Furthermore, in the future versions of the DB converter module, the user will have to possibility to enable an option allowing triggering a call to the EURES input API to request a replication each time a change has been made in the intermediate repository. Doing so, he can avoid waiting for the next call to the NCO input API in order to synchronize its data. This option will be configurable via a property.

Finally, the errors related to the XML generated in the intermediate repository and to the corresponding metadata will be reported in the log files of future versions of the module.

### NCO Input API Module

The purpose of the NCO input API module is to implement the API services that must be provided by the Member States and that will be called by the EURES system to replicate its job vacancies and jobseeker profiles. This module will query the intermediate repository to retrieve the needed data. Note that no conversion will be made since this repository will already contain the data in the format required for the NCO input API.

# Implementation Possibilities

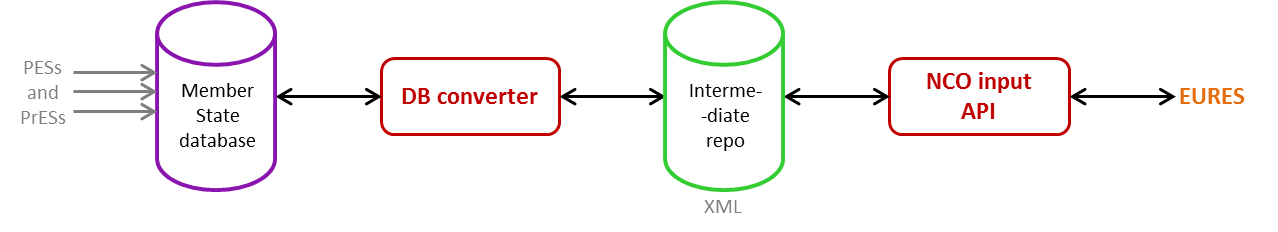
The modules can be used and combined in several ways. This chapter describes some possibilities of implementation of the Member State solution that make use of the NCO default implementation. Those possibilities can be combined.

For each solution, the versions of the modules used do not need to be the same. They must however support the same version of the intermediate repository structure.

In the case where the Member State decides to not use a module and to implement its own solution for this module instead, it is its own responsibility to implement a solution in line with the version used for the intermediate repository. Adjustments to the Member State’s solution can be necessary if the structure of the intermediate repository changes.

## Deploying Both Modules

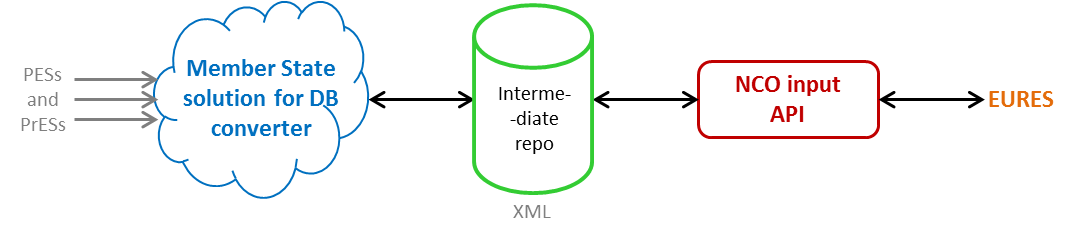
The default solution consists in using the DB converter module together with the NCO input API module. In this case, the Member State gathers the records of its sources in its database and deploys the intermediate repository. The modules are then used to connect the two databases together and to transmit records to DG EMPL.



## Deploying the NCO Input API Module Only

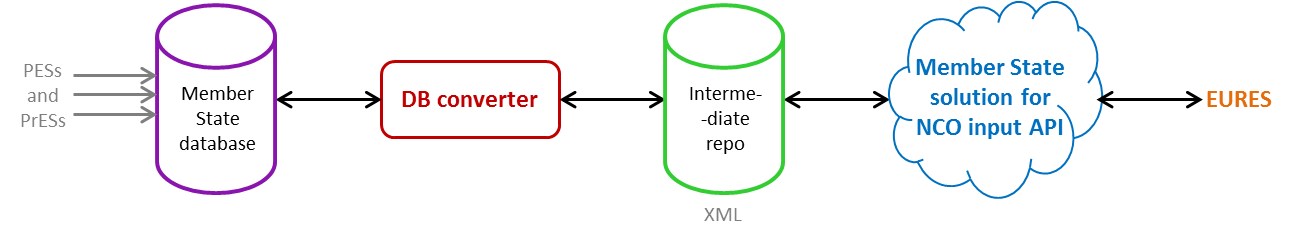
The Member State can choose to use only the NCO input API module. In this case, it deploys the intermediate repository and stores the XML records and the corresponding metadata in this repository using its own solution. The NCO input API is then deployed to implement the services needed for the transmission of data.

When developing its solution for the DB converter module, the Member State must make sure that all the metadata needed in the responses to the calls to the NCO input API is contained in the intermediate repository in the appropriate format.



## Deploying the DB Converter Module Only

The Member State can choose to use only the DB converter module. In this solution, it deploys the DB converter module and the intermediate repository, and implements the NCO input API services that will allow retrieving the data from this database.

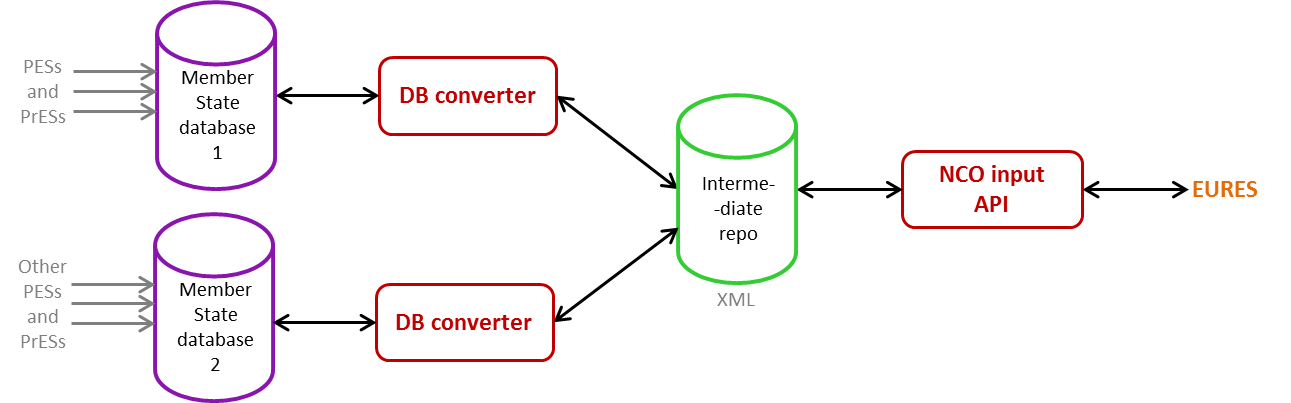


## Deploying Multiple DB Converter Modules

The Member State has the possibility to have multiple databases and deploy one DB converter module for each of them to fill the intermediate repository.

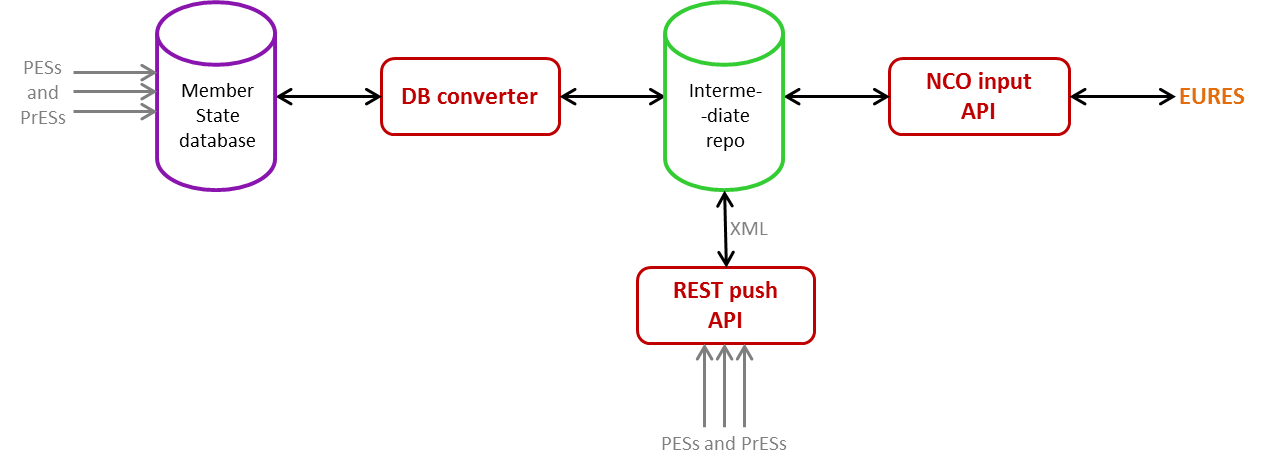
In this case, those databases have to contain distinct records and the records references have to be unique among the references of all the Member State databases.

Note that applying this strategy should not significantly improve the performances of the XML conversion since the DB converter module is already multi-threaded.



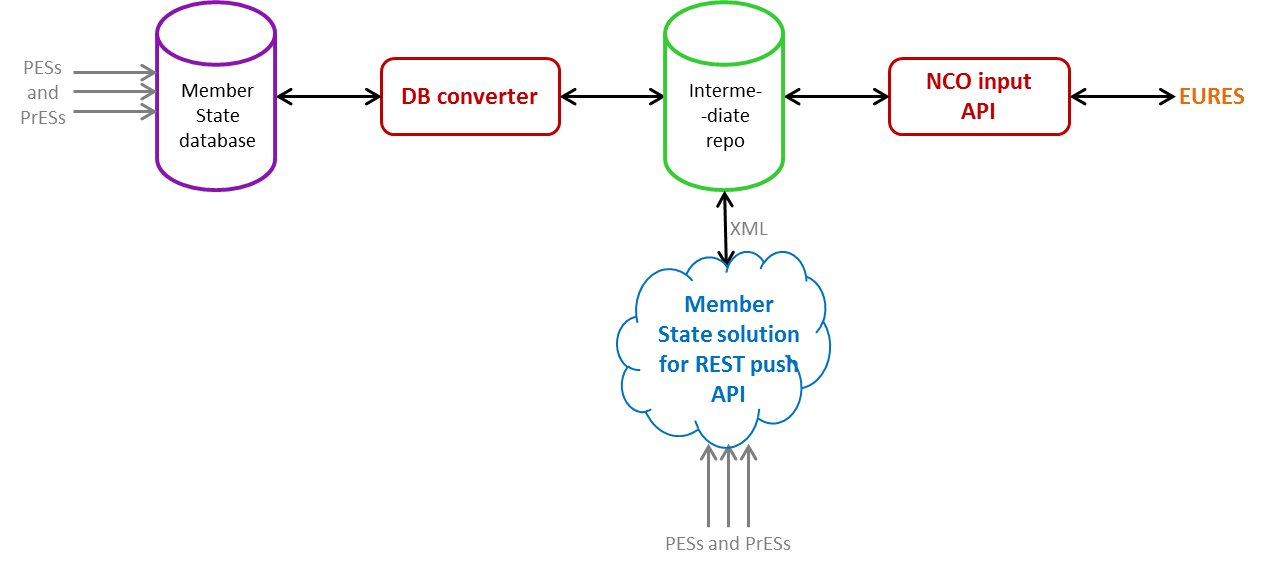
## Deploying the REST Push API Module

Depending on the needs and requirements of the Member States, a REST Push API module **could** be created in the future to permit to directly push into the intermediate repository the records of one or more sources in the EURES XML format. This module could be used without the other ones.



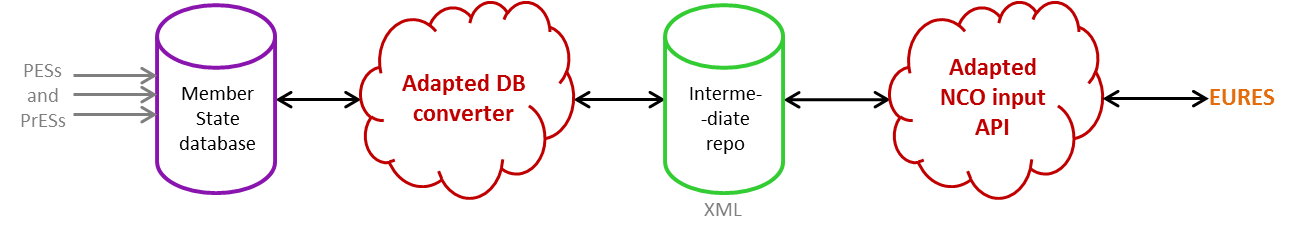
## Allowing Direct Writing in the Intermediate Repository

The Member State could allow some sources to insert their data directly into the intermediate repository by developing its own solution for the REST push API module.



## Adapting the Code

The sharing of the source code of the modules with the Member States is currently being investigated and is therefore **not** guaranteed. It would permit to adapt the modules of the default implementation to be more adjusted to the strategy and the system of the Member State.



# Prerequisites

## Intermediate Repository

The intermediate repository can either be a MongoDB or an SQL database (Oracle, PostgreSQL, or MySQL[[1]](#footnote-2)). For the MongoDB, no setup is required. However, for using an SQL database, some provided scripts must be executed before the default implementation modules can be used.

## DB Converter Module

The use of the DB converter module is only possible if the following conditions are met:

* The Member State database must have an environment supporting Java 8. Indeed, the source code of the NCO default implementation is written using version 8 of Java. Some features implemented in Java 8 are used in this code. The NCO default implementation is therefore not backward compatible with previous Java versions;
* The Member State database must be an SQL database. It must of course contain the records mandatory fields and the needed metadata (such as the creation, last modification and closing timestamps for example);
* As mentioned in the previous section, the intermediate repository must be deployed as an SQL or a MongoDB database. It must be properly configured to contain the records mandatory fields and the needed metadata.

## NCO Input API Module

The use of the NCO input API module is only possible if the following conditions are met:

* The Member State database must have an environment supporting Java 8. Indeed, the source code of the NCO default implementation is written using version 8 of Java. Some features implemented in Java 8 are used in this code and the NCO default implementation is therefore not backward compatible with previous Java versions;
* The intermediate repository must be deployed as an SQL or a MongoDB database. It must of course be configured to contain the records mandatory fields and the needed metadata.

# Installation and Configuration

## Intermediate Repository

The installation of the intermediate repository is mandatory to be able to use any of the modules provided by the NCO default implementation. The installation depends on the database technology chosen by the Member State to implement the intermediate repository. The Member State can choose between a MongoDB (noSQL) and one of the two following SQL database systems: Oracle, PostgreSQL or MySQL.

**Using MongoDB**

Installation instructions for MongoDB can be found at [the official MongoDB installation manual](https://docs.mongodb.com/manual/installation/). No scripts must be executed as the collections in the MongoDB will be created automatically by the modules on start up.

With the release 0.6.0, the collections jvDetail and jvMetadata can be deleted from the intermediate repository since these collections are now named recordDetail and recordMetadata. A reset should be performed to be resynchronized with the source.

**Using an SQL database**

To use an SQL database, the following scripts of the specific database system, located in eures-reg2018-nco-int-repo-0.6.2-sql, must be executed in the given order in the database:

1. <dbsystem>\_IREP\_TABLES\_changeset-0.0.0.sql
2. <dbsystem>\_IREP\_TABLES\_changeset-0.1.0.sql
3. <dbsystem>\_IREP\_TABLES\_changeset-0.4.0.sql
4. <dbsystem>\_IREP\_TABLES\_changeset-0.6.0.sql

where <dbsystem> is ORACLE, POSTGRESQL, or MYSQL. Make sure that the used database system is case insensitive (by default or by configuring it).

The execution of the 0.6.0 changeset removes the contents of the tables of the intermediate repository. A reset should be performed to be resynchronized with the source.

## DB Converter Module

In order for the DB converter module to work properly, a configuration file “*application.properties*” must be created first (included in the deliverable). The location of the file depends on the deployment strategy, covered in section 6.2.5.

All the configuration properties described in the next subsections must be added/updated in this “*application.properties*” file.

### Global configuration

***Table 3*** summarizes the properties to globally configure the DB converter module. Optional properties (i.e. a default value exists) are suffixed with *(optional)*.

|  |  |  |
| --- | --- | --- |
| Property | Description | Example |
| logging.file | The location of the file used for logging. If the file doesn’t exist, it will be created. | /logs/db-converter.log |
| jv.sync.batch.frequency.cron  (optional) | Cron expression altering the frequency with which the JVs in the intermediate repository are synced with the source database. The expression denotes the times at which a new sync is started. If the property is not specified, the batch for the JVs will not run. The example specifies a sync is started every thirty minutes on the first second of that minute.  To disable the synchronisation of JVs, this property must be removed from the properties file (or commented out by using the “#” prefix). | 0 0/30 \* \* \* \* |
| cv.sync.batch.frequency.cron  (optional) | Cron expression altering the frequency with which the CVs in the intermediate repository are synced with the source database. The expression denotes the times at which a new sync is started. If the property is not specified, the batch for the CVs will not run. The example specifies a sync is started every thirty minutes on the first second of that minute, starting at the fifth minute of the hour.  To disable the synchronisation of CVs, this property must be removed from the properties file (or commented out by using the “#” prefix). | 0 5/30 \* \* \* \* |
| sync.thread.pool.size | The number of threads in the thread pool used for the synchronization operations between the Member State database and the intermediate repository.  Default if not provided: 10 | 10 |
| nco.source.db.id | In case of multiple DB converters deployed, it represents the ID of the DB converter. Not mandatory in case of :   * a single DB converter for both JVs and CVs; * a single DB converter for the JVs and a single one for the CVs;   Mandatory in case of several DB converters containing the same type of records (JVs or CVs). | db-conv-1 |
| management.port  (optional) (needed if endpoints.enabled is set to true) | The management port used for reaching the monitoring endpoints. | 8082 |
| endpoints.enabled  (optional) | Property that allow enabling or disabling the monitoring endpoints. | false |
| endpoints.\*.enabled  (where \* is an endpoint)[[2]](#footnote-3)  (optional) | Property that allow enabling or disabling a specific monitoring endpoint (/health for example). | true |

*Table 3: Global Configuration of the DB converter module*

### Configuration of the Source Database

The database connection can be configured either via a JNDI name or using the direct configuration. ***Table 4*** and ***Table 5*** describe the properties required respectively for a configuration using JNDI datasource and using a direct datasource.

Only one configuration is required. If both configurations are set, the JNDI datasource will be used.

|  |  |  |
| --- | --- | --- |
| Property | Description | Example |
| nco.source.db.jndi-name | The JNDI name for the source database (if used). | java:comp/env/jdbc/ncodb |

*Table 4: Configuration of the database connection using a JNDI datasource*

|  |  |  |
| --- | --- | --- |
| Property | Description | Example |
| nco.source.db.url | The Member State database URL (if not configured through JNDI). | jdbc:oracle:thin:@member-state.lan:1341:ORCL |
| nco.source.db.username | The Member State database username (if not configured through JNDI). | user |
| nco.source.db.password | The Member State database password (if not configured through JNDI). | password |
| nco.source.db.driverClassName | The database driver for the Member State database. | oracle.jdbc.OracleDriver |

*Table 5: Configuration of the source database connections using a direct datasource*

### Configuration of the Intermediate Repository

The following property must be set for the module to be able to determine which type of database system will be used as intermediate repository (the one installed on the environment of the Member State in the previous section):

|  |  |  |
| --- | --- | --- |
| Property | Description | Example |
| spring.profiles.active | This property is used by the module to determine the technology used for accessing the intermediate repository. If an SQL database is used as intermediate repository, this property must be set to “jpa”. If using MongoDB, this property must be set to “mongo”. | jpa |

*Table 6: Configuration of the type of database used for the intermediate repository*

Depending on the chosen database system for the intermediate repository, some specific properties must be set:

**When using MongoDB**

Only use this configuration when the property the following property is set: “*spring.profiles.active=mongo*”.

|  |  |  |
| --- | --- | --- |
| Property | Description | Example |
| spring.data.mongodb.host | The hostname of the MongoDB the module will connect to. | localhost |
| spring.data.mongodb.port | The server port for the MongoDB. | 27017 |
| spring.data.mongodb.database | The name of the database that will be used in the MongoDB. | jvdb |
| spring.data.mongodb.username | The login user of the MongoDB server. | user |
| spring.data.mongodb.password | The login password required to login to the MongoDB. | password |

*Table 7: Configuration of the intermediate repository using mongo*

**When using an SQL database**

Only use this configuration when the following property is set: “*spring.profiles.active=jpa*”.

The connection to the intermediate repository can be configured either using a JNDI datasource or a direct datasource. ***Table 8*** and ***Table 9*** describe respectively how to configure the properties for a JNDI datasource and a direct datasource.

Only one configuration is required. If both configurations are set, the JNDI datasource will be used.

|  |  |  |
| --- | --- | --- |
| Property | Description | Example |
| nco.irep.db.jndi-name | The JNDI name for the intermediate repository. | java:comp/env/jdbc/intrepo |

*Table 8: Configuration of the intermediate repository using JPA with a JNDI datasource*

|  |  |  |
| --- | --- | --- |
| Property | Description | Example |
| nco.irep.db.url | The intermediate repository database URL. | jdbc:postgresql://member-state-host:5432/jvdb |
| nco.irep.db.username | The intermediate repository database username. | jvdb |
| nco.irep.db.password | The intermediate repository database password. | password |
| nco.irep.db.driverClassName | The database driver for the intermediate repository. | org.postgresql.Driver |

*Table 9: Configuration of the Intermediate repository using JPA with a direct datasource*

### Configuration of the queries

Finally, to be able to get all necessary data from the Member State database, the module uses some queries that must be configured correctly in the “*application.properties*” file. These queries are described in the query mapping document [R05].

### Deployment of the module

After the “*application.properties*” file is filled in correctly, the module can be started. This can be done in several ways:

#### Executing the executable JAR

On Unix/Linux systems a specially crafted WAR file can be directly executed: eures-reg2018-nco-db-converter-executable .jar. This will start an embedded application container using Java. Therefore, only Java is required.

The “*application.properties*” file must be located in the same folder as the executable war file.

##### Installing the module as a service on Unix

To install the module as a service on Unix, refer to the following guide: <http://docs.spring.io/spring-boot/docs/current/reference/html/deployment-install.html>.

The “*application.properties*” file must be located in the same folder as the executable war file.

##### Using the java command

On non-Unix systems the jar can be started using the command java –jar eures-reg2018-nco-db-converter-executable.jar.

The “*application.properties*” file must be located in the same folder as the executable jar file.[[3]](#footnote-4)

##### Supplying database drivers

To be able to upgrade the JDBC drivers for the intermediate repository and not to limit the supported[[4]](#footnote-5) database systems, the database drivers are not included in the war but need to be provided – as is standard practice when using an application server. To provide the database driver, place the necessary jar files in a folder. Provide this folder to the executable war by setting the environment variable “*LOADER\_PATH*”.

##### Example service setup on modern Linux

This example contains instructions on how to create a unit file for a systemd based Linux distribution[[5]](#footnote-6). This will create a service on the operating system for the database converter.

* Create the folder */opt/eures/db-converter*
* Copy the file eures-reg2018-nco-db-converter-executable.war to that folder.
* Copy the application.properties to that folder.
* Create the folder */opt/eures/db-drivers* and put the chosen database driver in that folder.
* In /*etc/system/system* create a file named eures-db-converter.service with the content below.

[Unit]

Description=Eures database converter

After=syslog.target

[Service]

User=eures

Environment=”LOADER\_PATH=/opt/eures/db-drivers”

ExecStart=/opt/eures/db-converter/eures-reg2018-nco-db-converter-executable.jar

SuccessExitStatus=143

[Install]

WantedBy=multi-user.target

* Start the service by using *systemctl start eures-db-converter.service*
* To run the service at system startup execute the command  
  *systemctl enable eures-db-converter.service*

##### Running from the command line example

For testing it may be simpler to run the database converter in userspace from a normal, non-root account, before moving to a more robust setup as described in the previous section. Instructions to that end can be found below.

* Create a folder named *eures* in the home directory of the user.
* In that folder create subfolders named *db-drivers* and *db-converter.*
* In *db-drivers* put the jars for your database system.
* In *db-converter* put eures-reg2018-nco-db-converter-executable.war and the application.properties file
* On the command line cd into *db-converter*.
* Execute the command *export LOADER\_PATH=$HOME/eures/db-drivers*
* Execute the *eures-reg2018-nco-db-converter-executable.jar* file.

#### Using Tomcat

The artefact eures-reg2018-nco-db-converter.war can be deployed on a Java Servlet Specification v3 compliant application server like Apache Tomcat.

The Tomcat Webapp will need to know the location of the “*application.properties*” configuration file. The simplest way to achieve this is to configure it as an environment property.

In the Tomcat configuration directory, create a file with the same name as the war to be deployed in the directory conf/Catalina/localhost. For instance, when deploying *eures-reg2018-nco-db-converter.war*, create *eures-reg2018-nco-db-converter.xml*. The file needs to have the following content:

<Context><Environment name="spring.config.location" type="java.lang.String" value="<path-to>/application.properties"/></Context>

The context element can also be added directly to the Host element of the conf/server.xml file with the following modification:

<Context path="eures-reg2018-nco-db-converter"><Environment name="spring.config.location" type="java.lang.String" value="<path-to>/application.properties"/></Context>

#### Using Weblogic

In Weblogic, an alternative is required as there is no simple way to specify environment properties. Therefore, start the Weblogic server with the following argument

* Dspring.config.location=<some-path>

where *“<some-path>*” must be replaced by the absolute path to the folder containing the “*application.properties*” file. Then deploy the application on the Weblogic application server.

## NCO Input API Module

In order for the NCO input API module to work properly, a configuration file “*application.properties”* must be created first (included in the deliverable). The location of the file depends on the deployment strategy, covered in section 6.3.3.

All the configuration properties described in the next subsections must be added/updated in this “*application.properties*” file.

### Global Configuration

***Table 10*** summarizes the properties to globally configure the Input API module.

|  |  |  |
| --- | --- | --- |
| Property | Description | Example |
| input.api.ping.message | The message returned in the response to a call to the Ping service (see [R04]). | Hello from Input API |
| logging.file | The location of the file used for logging. If the file doesn’t exist, it will be created. | /logs/input-api.log |
| active.cv.controller  (optional) | Property that allows activating or deactivating the CV endpoint. If the property is not set, it will be deactivated. | true |
| active.jv.controller  (optional) | Property that allows activating or deactivating the JV endpoint. If the property is not set, it will be deactivated. | true |
| management.port  (optional) (needed if endpoints.enabled is set to true) | The management port used for reaching the monitoring endpoints. | 8081 |
| endpoints.enabled  (optional) | Property that allow enabling or disabling the monitoring endpoints. | false |
| endpoints.\*.enabled  (where \* is an endpoint)[[6]](#footnote-7)  (optional) | Property that allow enabling or disabling a specific monitoring endpoint (/health for example). | true |

*Table 10: Global Configuration of the Input API module*

### Configuration of the Intermediate Repository

The same configuration as the one used for the intermediate repository in the DB converter module can be used. This configuration is entirely described in section 6.2.3.

### Deployment of the module

After the “*application.properties*” file is filled in correctly, the module can be started. This can be done in several ways:

#### Executing the executable JAR

On Unix/Linux systems a specially crafted WAR file can be directly executed: *eures-reg2018-nco-input-api-executable.jar*. This will start an embedded application container using Java. Therefore, only Java is required.

The “*application.properties*” file must be located in the same folder as the executable war file.

##### Installing the module as a service on Unix

To install the module as a service on Unix, refer to the following guide: <http://docs.spring.io/spring-boot/docs/current/reference/html/deployment-install.html>.

The “*application.properties*” file must be located in the same folder as the executable war file.

##### Using the java command

On non-Unix systems the jar can be started using the command calling java –jar eures-reg2018-nco-input-api-executable.jar.

The “*application.properties*” file must be located in the same folder as the executable war file.[[7]](#footnote-8)

##### Supplying database drivers

To be able to upgrade the JDBC drivers for the intermediate repository and not to limit the supported[[8]](#footnote-9) database systems, the database drivers are not included in the war but need to be provided – as is standard practice when using an application server. To provide the database driver, place the necessary jar files in a folder. Provide this folder to the executable war by setting the environment variable “*LOADER\_PATH*”.

##### Example Setup on modern Linux

This example contains instructions on how to create a unit file for a systemd based Linux distribution[[9]](#footnote-10).

* Create the folder */opt/eures/input-api*
* Copy the file eures-reg2018-nco-input-api-executable.jar to that folder.
* Copy the application.properties to that folder.
* Create the folder */opt/eures/db-drivers* and put the chosen database driver in that folder.
* In /*etc/system/system* create a file named eures-input-api.service with the content below.

[Unit]

Description=Eures input api

After=syslog.target

[Service]

User=eures

Environment=”LOADER\_PATH=/opt/eures/db-drivers”

ExecStart=/opt/eures/input-api/eures-reg2018-nco-input-api-executable.jar

SuccessExitStatus=143

[Install]

WantedBy=multi-user.target

* Start the service by using *systemctl start eures-input-api.service*
* To run the service at system startup execute the command  
  *systemctl enable eures-input-api.service*

##### Running from the command line example

For testing it may be simpler to run the database converter in userspace from a normal, non-root account, before moving to a more robust setup as described in the previous section. Instructions to that end can be found below.

* Create a folder named *eures* in the home directory of the user.
* In that folder create subfolders named *db-drivers* and *input-api.*
* In *db-drivers* put the jars for your database system.
* In *db-converter* put *eures-reg2018-nco-input-api-executable.war* and the application.properties file
* On the command line cd into *db-converter*.
* Execute the command *export LOADER\_PATH=$HOME/eures/db-drivers*
* Execute the *eures-reg2018-nco-input-api-executable.jar* file.

#### Using Tomcat

The artefact eures-reg2018-nco--input-api.war can be deployed on a Java Servlet Specification v3 compliant application server like Apache Tomcat.

The Tomcat Webapp will need to know the location of the “*application.properties*” configuration file. The simplest way to achieve this is to configure it as an environment property.

In the Tomcat configuration directory, create a file with the same name as the WAR to be deployed in the directory conf/Catalina/localhost. For instance, when deploying *eures-reg2018-nco--input-api.war*, create *eures-reg2018-nco--input-api.xml*. The file needs to have the following content:

<Context><Environment name="spring.config.location" type="java.lang.String" value="<path-to>/application.properties"/></Context>

The context element can also be added directly to the Host element of the *conf/server.xml* file with the following modification:

<Context path="eures-reg2018-nco-input-api"><Environment name="spring.config.location" type="java.lang.String" value="<path-to>/application.properties"/></Context>

In this case, the path provided can be an absolute path to the Tomcat directory or a relative path (that is relative to the Tomcat home directory).

#### Using Weblogic

In Weblogic, an alternative is required as there is no simple way to specify environment properties. Therefore, start the Weblogic server with the following argument

-Dspring.config.location=<some-path>

where *“<some-path>*” must be replaced by the absolute path to the folder containing the “*application.properties*” file. Then deploy the application on the Weblogic application server.

# Operating Instruction

## Intermediate Repository

The contents of the intermediate repository can be checked using an interface tool to connect to the used database (e.g. SQL Developer, MongoDB Compass…).

### Clearing the intermediate repository

**Warning:** The execution of the following commands will drop all the data inside intermediate repository. Use them carefully.

#### MongoDB

1. Make sure the DB converter module NCO Input API modules are not running;
2. Drop the database configured in the *spring.data.mongodb.database* using one of the following methods:
   1. Connect to the MongoDB database using a mongo client like *Robomongo*[[10]](#footnote-11) and drop the database of the intermediate repository (see Figure 1 for an illustration in Robomongo);
   2. Connect to MongoDB database using mongo shell and execute the db.dropDatabase() command[[11]](#footnote-12).
3. On the next sync, the database will be automatically recreated.

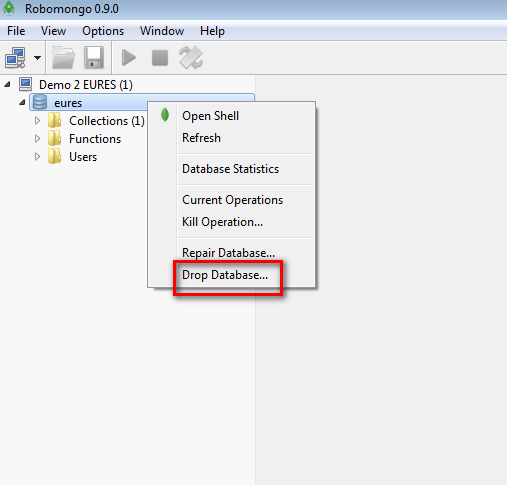


Figure 1: Drop a database in Robomongo

#### SQL Databases

1. Make sure the DB converter module NCO Input API modules are not running;
2. Connect to the intermediate repo that is configured per the instructions of section 6.2.3;
3. Empty following tables in the order specified using TRUNCATE TABLE or DELETE FROM commands:
   * IREP\_JV\_DETAIL;
   * IREP\_JV\_METADATA;
   * IREP\_SYNC\_WARNINGS;
   * IREP\_SYNC\_EXCEPTION\_DETAILS;
   * IREP\_EXCEPTION\_DETAILS;
   * IREP\_SYNC.

## DB Converter Module

Administrative services for the DB converter module will be exposed on the following path.

{host}/db-converter/{type}/{service}

The type can be *jv* or *cv*. It indicates the type of records that will be synchronized.

The service can be *ping*, *reset* or *syncDbs*:

* The *ping* service can be used to check if the DB Converted module is deployed correctly;
* The *reset* service can be used to reset the intermediate repository, purging any JVs that have disappeared from the source database;
* The *syncDbs* service can be used to manually trigger synchronization between intermediate repository and source database.

Care must be taken not to expose these services to the outside world as this would allow unknown entities to interfere with the correct operation of the DB converter module. This can be easily checked using the ping service. The ping service can be accessed by entering {host}/db-converter/{type}/ping in any web browser. The request to ping should return an error if accessed from outside the premises of the NCO.

The log files for this module can either be found in the logging facility of the application server the module is deployed on or in the file specified in configuration (see section 6.2.1). In many application server implementations, the configuration of the logging facilities of the application server itself take precedence over the configuration specified in the configuration file of the application. This can influence the location of the log files.

## NCO Input API Module

The services of the NCO input API will be available at the configured location at the following path:

{host}/input/api/{type}/v0.1/{service}

The type can be *jv* and *cv* and indicates the type of records for which information will be returned.

The *ping* service can be used to check if the NCO input API is correctly deployed, while the other services (*getAll*, *getChanges* and *getDetails*) can be used to check if it correctly transmits the data in the intermediate repository.

The log files for this module can either be found in the logging facility of the application server the module is deployed on or in the file specified in the configuration (see section 6.3.1). In many application server implementations, the configuration of the logging facilities of the application server itself take precedence over the configuration specified in the configuration file of the application. This can influence the location of the log files.

# Release Note

The following chapter presents a quick overview of the compatible and tested environments for the default implementation. The intermediate repository and the two modules are tackled separately.

The test and compatibility matrix for the 2 repositories (intermediate repository and source database) are displayed in respectively ***Table 11*** and ***Table 12***. The modules DB converter and Input API are displayed in respectively ***Table 13*** and *Table 14*.

For each feature, the following values are displayed:

* Tested: displays whether the feature was fully tested, partially tested or not tested;
* Compatible: displays whether the feature is guaranteed to be compatible, should be compatible (i.e. not fully tested) or is not compatible (tests revealed bugs related to that feature).

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Tested | Compatible | Versions |
| Use mongo DB | Full | Yes | 3.2.x,3.4.x |
| Use Oracle DB | Full | Yes | 12c |
| Use PostgreSQL | Partial | Should | >9.3.x |
| Use MySQL | Partial | Yes | >5.7.x |
| Use h2 | Full | Yes |  |
| Other SQL database[[12]](#footnote-13) | No | Should |  |

*Table 11: Compatibility table for the intermediate repository*

|  |  |  |
| --- | --- | --- |
| Feature | Tested | Compatible |
| Use Oracle DB | Full | Yes |
| Use PostgreSQL | No | Should |
| Use MySQL | No | Should |
| Use h2 | Full | Yes |
| Other SQL database | No | Should |

*Table 12: Compatibility table for the source database*

|  |  |  |
| --- | --- | --- |
| Feature | Tested | Compatible |
| Execute executable WAR | Full | Yes |
| Install as a service in UNIX | Partial | Yes |
| Start using the java command | Partial | Should |
| Deploy in Tomcat | Partial | Should |
| Deploy in WebLogic | Full | Yes |
| Other application server[[13]](#footnote-14) | No | Should |

*Table 13: Compatibility table for the DB converter module*

|  |  |  |
| --- | --- | --- |
| Feature | Tested | Compatible |
| Execute executable WAR | Full | Yes |
| Install as a service in UNIX | Partial | Yes |
| Start using the java command | Partial | Should |
| Deploy in Tomcat | Partial | Should |
| Deploy in WebLogic | Full | Yes |
| Other application server[[14]](#footnote-15) | No | Should |

*Table 14: Compatibility table for the Input API module*

## Functionalities Added in Default Implementation v0.3.0

The changes made to the default implementation modules between versions 0.2.0 and 0.3.0 can be summarized as follows:

* Adaptation of the URLs of the NCO input API according to version 1.20 of the FMES [R04];
* Adaptation of the JSON content returned by the services of the input API according to version 1.20 of the FMES [R04];
* Support of Gzip compression in the transferred data as described in the FMES [R04];
* Upgrade of Spring Boot version to 1.4.5.RELEASE.

Note that no change has been made to the DB converter module or to the intermediate repository scripts, so only the NCO input API module should be upgraded.

## Functionalities Added in Default Implementation v0.4.0

The changes made to the default implementation modules between versions 0.3.0 and 0.4.0 can be summarized as follows:

* Updates of existing queries and adding of new queries to support the extraction of the fields in the Conformant Optional layer (see also [R03]) of the HR-XML standard (see also [R05]);
* Change of the way the status of a JV sync is saved in the intermediate repository from integer to String (text);
* Adding of SQL scripts for the intermediate repository to create indexes on some tables.

In the release 0.4.0, both DB converter and NCO input API modules are updated. Here is a summary of the operations to be performed to install the release 0.4.0 of the default implementation starting from release 0.3.0 if using a **MongoDB** as intermediate repository (to be executed in the provided order):

1. [Optional] Empty the intermediate repository (necessary because of the update of queries for Conformant Optional layer) as described in section 7.1.1. Alternatively, the *reset* service described in section 7.2 can be called after deployment to reset the database provided that the query *query.jv.getActiveIds* is properly configured (see step 3).;
2. Deploy the new DB converter and NCO input API modules;
3. [If step 1 not executed] Call the endpoint of the *reset* service described in section 7.2.

Here is a summary of the operations to be performed to install the release 0.4.0 of the default implementation starting from release 0.3.0 if using a **SQL database** as intermediate repository (to be executed in the provided order):

1. Empty the intermediate repository (necessary because of the update of queries for Conformant Optional layer) as described by section 7.1.1;
2. Execute the script <dbsystem>\_IREP\_TABLES\_changeset-0.4.0.sql corresponding to the type of DB;
3. Deploy the new DB converter and NCO input API modules.

## Functionalities Added in Default Implementation v0.5.0

The changes made to the default implementation modules between versions 0.4.0 and 0.5.0 can be summarized as follows:

* Database drivers are no longer supplied with the runnable jar. This allows more freedom to provide support for other database systems, as well as upgrading drivers without running into version conflicts with the supplied drivers. See sections 6.2.5 and 6.3.3 for details on how to provide the necessary drivers;
* Updates of existing queries and adding of new queries to support the extraction of all the fields for a JV (see also [R03]) of the HR-XML standard (see also [R05]);
* The intermediate repository now supports MySQL databases;
* Upgrade to the version 1.1 of the EURES standard.

In the release 0.5.0, both DB converter and NCO input API modules are updated. Here is a summary of the operations to be performed to install the release 0.5.0 of the default implementation starting from release 0.4.0 (to be executed in the provided order):

1. [Optional] Empty the intermediate repository as described in section 7.1.1. Alternatively, the *reset* service described in section 7.2 can be called after deployment to reset the database provided that the query *query.jv.getActiveIds* is properly configured (see step 3).;
2. Deploy the new DB converter and NCO input API modules;
3. [If step 1 not executed] Call the endpoint of the *reset* service described in section 7.2.

## Functionalities Added in Default Implementation v0.6.0

The changes made to the default implementation modules between versions 0.5.0 and 0.6.0 can be summarized as follows:

* The possibility to have multiple source databases synchronized with the intermediate repository has been added (see section 4.4 for more details);
* The DB converter can fetch CVs containing a few fields, detailed in [R05]. It can be configured to fetch JVs and/or CVs, which, for example, would allow deploying two DB converters, one for the JVs and one the CVs;
* The synchronisation of JVs and CVs can de disabled by removing the relevant properties (see jv.sync.batch.frequency.cron and cv.sync.batch.frequency.cron properties in ***Table 3*** for more details);
* The NCO input API can transmit information about CVs. As for the DB converter, it can activate or deactivate the JV and/or CV endpoints (see active.jv.controller and active.cv.controller properties in ***Table 10*** for more details);
* The number of threads used when resetting or synchronizing the intermediate repository with the Member State database can be configured by means of a new property. If not configured, the value is set to 10 threads. Consequently, the Member States need to explicitly set it to 1 if they want to avoid multi-threading (see property sync.thread.pool.size in ***Table 3*** for more details);
* The possibility to monitor the Input API and DB Converter modules using Spring Boot Actuator[[15]](#footnote-16) (see new properties “management.port “,“endpoints.enabled” and “endpoints.\*.enabled” added in ***Table 3*** and ***Table 10***). Note that by default, this feature is disabled;
* Adding of SQL scripts for the intermediate repository to create indexes on some tables;
* Support for MongoDb 3.4.x;
* Upgrade to Spring boot 1.5.7.

Here is a summary of the operations to be performed to install the release 0.6.0 of the default implementation starting from release 0.5.0 (to be executed in the provided order):

1. if using an **SQL database** as intermediate repository, execute the script <dbsystem>\_IREP\_TABLES\_changeset-0.6.0.sql corresponding to the type of DB (note that execution of this script will automatically remove all the JVs stored in the intermediate repository);
2. Add the new properties to the DB converter and to the NCO input API modules;
3. Deploy the new DB converter and NCO input API modules.

## Functionalities Added in Default Implementation v0.7.0

The changes made to the default implementation modules between versions 0.6.0 and 0.7.0 can be summarized as follows:

* Upgrade to the version 1.3 of the EURES standard (see [R02] and [R03]);
* Addition of privacy queries so that processing instructions can be added to the CV xml (see “Processing Instruction codes” sheet in [R05] for more information);
* Update of the queries (CV and JV) (see [R05] for more information. Two new Excel sheets have been added in the document listing the changes made in the queries).

In the release 0.7.0, the DB converter module is updated (Since the NCO input API module has not changed, the previous version can be used). Here is a summary of the operations to be performed to install the release 0.7.0 of the default implementation starting from release 0.6.0 (to be executed in the provided order):

1. [Optional] Empty the intermediate repository as described in section 7.1.1.
2. Deploy the new DB converter;
3. [If step 1 not executed] Call the endpoint of the *reset* service described in section 7.2.

1. MySQL is not currently supported as an intermediary repository but it could be in a future version of the default implementation if requested (see chapter 8 for more details) [↑](#footnote-ref-2)
2. More information here: https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-endpoints.html [↑](#footnote-ref-3)
3. When running the java –jar <jar> command, it MUST be run from the directory the jar and application properties reside in. Otherwise the application.properties file will not be detected correctly. [↑](#footnote-ref-4)
4. Given the multitude of database systems available only the compatibility with a few major database drivers will be extensively tested. [↑](#footnote-ref-5)
5. Ubuntu, Red Hat Enterprise, Debian, CentOS, and SUSE Linux Enterprise server have adopted systemd as a default since v15.04,v7.0,v8,v7.14.04, and v12 respectively. [↑](#footnote-ref-6)
6. More information here: https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-endpoints.html [↑](#footnote-ref-7)
7. When running the java –jar <jar> command, it MUST be run from the directory the jar and application properties reside in. Otherwise the application.properties file will not be detected correctly. [↑](#footnote-ref-8)
8. Given the multitude of database systems available only the compatibility with a few major database drivers will be extensively tested. [↑](#footnote-ref-9)
9. Ubuntu, Red Hat Enterprise, Debian, CentOS, and SUSE Linux Enterprise server have adopted systemd as a default since v15.04,v7.0,v8,v7.14.04, and v12 respectively. [↑](#footnote-ref-10)
10. <https://robomongo.org> [↑](#footnote-ref-11)
11. <https://docs.mongodb.com/manual/reference/method/db.dropDatabase/> [↑](#footnote-ref-12)
12. No scripts are provided to generate DDL for another SQL database. Therefore, if another database engine is to be used, the administrator might have to adapt the scripts generated for oracle DB, PostgreSQL, or MySQL for the target database engine. [↑](#footnote-ref-13)
13. In order to deploy the WAR file in another application server, a deployment descriptor file might have to be provided. [↑](#footnote-ref-14)
14. In order to deploy the WAR file in another application server, a deployment descriptor file might have to be provided. [↑](#footnote-ref-15)
15. More information about Spring Boot Actuator can be found on <https://docs.spring.io/autorepo/docs/spring-boot/1.4.5.RELEASE/reference/html/production-ready-monitoring.html> [↑](#footnote-ref-16)