SOCIB Balearic Islands Coastal Observing and Forecasting System

Processing application environment, execution and implementation

SOCIB-Data Center Facility

Document type:	Product user manual
Date:	2016-01-01

Description:	Processing application
Authors:	ksebastian
Supervision:	ctroupin
Involved Personnel:	ksebastian, ctroupin, jpbeltran, cmuñoz, mrujula

DOCUMENT VERIFICATION LIST

Date:	Checked by (name)	SOCIB division	Ref.
2016-01-01	K. Sebastian	SOS	

DOCUMENT DISTRIBUTION LIST

Date:	Distribution to:
2016-01-01	DCF

CHANGE RECORD

#	Date	Description	Author	Checked by
1.0.0	16/03/2016	Added missing dependencies on the “Dependencies and Plugins” chapter	mrujula	jfernandez

Índice de contenido

Introduction

The aim of this document is describe and configure the environment to run the processing application in development mode, in order to test new platforms and implementations, and create the runnable **.jar** to to run in the production environment.

Since the processing application consists of 6 projects, a description and some implementations are included of each project.

Environment configuration

The Eclipse for Java EE IDE is recommended, because all configuration, development and runnable jar are done in Eclipse.

Install Java

sudo apt-add-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java6-installer

Source code

For each project the related repository is:

Socib data processing	Socib processing library	Socib datanucleus library
processing	processing-library	datanucleus-library

Socib definitions	Socib Meteo Ocean toolbox	Socib NetCDF to Kml
socib-definitions	meteo-ocean-toolbox	netcdfToKml

Notifications System		
notifications_system		

Clone each project inside your workspace directory and import it to Eclipse ‘Eclipse Projects into Workspace.

Import the projects

Add the Socib Data Discovery sources as an Eclipse project. Go to “File” -> “New” -> “Java Project”. The project name must match the cloned one and the “Project Location” must point to the parent folder of the cloned one. Finally click finish and it should be created.

Dependencies and plugins

Library and project dependencies

The libraries are located in /home/gituser/libraries and the user libraries configuration to be imported to Eclipse as well.

/home/gituser/libraries/socib_java_libraries.userlibraries

Firstly, import to Eclipse that file in the ‘User libraries’ preferences to begin with the dependency configuration. Next, go to the ‘Java Build path’ of each project and add the following system and ‘User Libraries’:

Socib data processing	Socib processing library	Socib datanucleus library
COMMONS-COLLECTIONS COMMONS-LANG EXP4J GSON KML LOG4J NETCDF SUNMARCHALLER JAVA6	CLONING COMMONS-COLLECTIONS COMMONS-IO COMMONS-LANG COMMONS-MATH CONCURRENTUNIT-V0.4.2 CPSUITE EXP4J GOOGLE-GUAVA GSON JODA-TIME JUNIT4 LOG4J MATH ??? NETCDF OBJENESIS XOM JAVA6	COMMONS-CODEC COMMONS-COLLECTIONS COMMONS-LANG CONNECTION-POOLING DATANUCLEUS-V4.1.1 EHCACHE ~~GOOGLE-GUAVA ???~~ JODA-TIME ~~JDO-API-3.1 ???~~ LOG4J MEMCACHED POSTGRES JAVA6

Socib definitions	Socib Meteo Ocean toolbox	Socib NetCDF to Kml (the lib folder contains all)

JAVA6	LOG4J EXP4J JAVA6	LOG4J KML NETCDF COMMONS-LANG SUNMARCHALLER JAVA6

Notifications System		
ASANA DATANUCLEUS-V4.1.1 GOOGLE-GUAVA GOOGLE-CLIENT-HTTP GOOGLE-HTTP-CLIENT-GSON GOOGLE-OAUTH-CLIENT GSON JavaMail JUnit 4 LOG4J JAVA6		

Finally, add the project dependency in the 'Java Build Path':

Socib data processing	**Socib processing library**	**Socib datanucleus library**
SocibDatanucleusLibrary SocibProcessingLibrary SocibDefinitions SocibNetcdfToKml SocibNotificationsSystem	SocibDatanucleusLibrary SocibMeteoOceanToolbox SocibDefinitions SocibNotificationsSystem	SocibNotificationsSystem SocibDefinitions

Socib notifications system		
SocibDatanucleusLibrary SocibProcessingLibrary SocibDefinitions		

Furthermore, the project facet java in the "Project facets" (only if it is active) property must match with the Java version added to the build path and all projects must have the same Java version.

IMPORTANT: The NetCDF library in processing project must be the first one in the "Order

and Export" tab (Project "Properties" -> "Java Build Path" Menu). If not, then the exception "Exception in thread "main" java.lang.NoSuchMethodError: ucar.nc2.Attribute.getShortName()Ljava/lang/String;" will be throw.

Plugins

Datanucleus

The Datanucleus Eclipse Plugin. From the web page: Development of a DataNucleus-enabled project using Eclipse would benefit from the DataNucleus Eclipse plugin providing enhancement of classes and the opportunity to create the database schema. You can download this plugin by adding the DataNucleus "Eclipse Update" site of http://www.datanucleus.org/downloads/eclipse-update to your Eclipse configuration.

You don't have to configure anything, because the datanuclues configuration is performed via the datanucleus.properties file, located in the SocibDatanucleus library. You only have to add DataNucleus support to the SocibDatanucleusLibrary (right click on the project -> DataNucleus -> Add Datanucleus support).

EclEmma Java Code Coverage

EclEmma is a free Java code coverage tool for Eclipse, available under the Eclipse Public License. It brings code coverage analysis directly into the Eclipse workbench.

http://eclemma.org/

Eclipse-pmd

PMD is a source code analyzer. It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth. It supports Java, JavaScript, PLSQL, Apache Velocity, XML, XSL.

http://sourceforge.net/projects/pmd/files/pmd-eclipse/update-site/

Sphinx environment

See the javasphinx section in the master document.

Execution

Local environment (develop)

The main class is the **ProcessingManager.java** located in the es.socib package of the Socib Data Processing project. This class implements the main method that process the active platforms and generate the related products (NetCDF files, kml files, ...).

1. First of all check the properties the **processing.properties** file located in the Socib Processing library project. It must be in developing mode (**developing = true**) and the **developing paths set to your local paths** (see the ProcPropertiesConf.java in the Socib Processing library project for an extended explanation of each property). In develop mode, the processing doesn't archive the input data and set the level of the processing logger to INFO.

2. Secondly, select the instruments and deployments that you want to process. The Socib Data Processing should process all active platforms with input data. You may want to process a subset of instruments and deployments. The **DeploymentService.java** located in the package **es.socib.database.service.DeploymentService** of the **Socib Datanucleus Library** implements the function ***getActiveDeployments()***. You can modify the behaviour carefully to retrieve the desired instruments and deployments. See also the ***constructCondition()*** function comment to know the filtering options.

3. Finally, you can run via Eclipse "Run As" the **ProcessingManager.java** or the main method as a "Java Application". The Socib Data Processing will process the input data and generate the products, according to the current configuration.

Run the unit tests

The tests in the Socib Processing Library project are implemented in JUnit4 and there are defined some suites using cpsuite to run the unitests. The cpsuite requires Java 7+, so the suites must be run with Java 7+. Eclipse has the features to run and debug the tests adding to the project the junit library as a dependency.

All the tests are implemented in the test folder. By default the tests run with the properties:

- ProcPropertiesConf.developing = true;
- ProcPropertiesConf.developingRtdataPath = "/home/dataprocuser/test_data";
- ProcPropertiesConf.developingOpendapPath = "/home/dataprocuser/test_data/opendap";

It contains some tests that could be used as an example to process instrument and product data. See the es.socib.ProcessingTestCase class for further information.

Production

The properties in the processing.properties file must have these values:

- outputBasePath = /data/current/opendap
- currentDataPath = /data/current/
- activePlatformsCacheEnable = true
- filterDataOutOfRange = true

- developing = false

Furthermore, the getActiveDeployments() function of the DeploymentService.java file must retrieve all active deployments, checking that line “sqlQuery = constructCondition(instrument, instrumentNames, depCodes);” is commented.

Deploy

To create a new jar you must do right click over the Socib Data Processing Project -> Export -> Select the “Runnable JAR file” -> then a window with some options is prompted. Select the SocibDataProcessing in “Launch Configuration”, the path to save the jar (empty folder!!) and other files in “Export destination” and select the “Copy required libraries….”. Finally click “Finish”.

Once the processing.jar file and processing_lib folder you can copy them to the /home/dataprocuser/processing_application/vX.X.X. Before copy them, is highly recommended to backup the previous file and lib folder, check that the processing.jar is not running (ex: with top) and comment the crontab line that executes the processing.jar . Note that there is a latest_version symbolic pointing to the latest folder version, because the crontab line that executes the processing.jar points to the latest_version symbolic link.

Raw input, archive and archive daily folders

The SOCIB Processing Application will search inside this platform “Path”, for each installed instrument, a folder with the instrument name. Inside this folder, depending on the parsing function implementation, should be the:

- raw input: The folder where files to be processed must be stored
- raw archive daily: The folder where processed files will be move from the input directory for some platform (usually fixed stations). Files could be removed at the end of the each day.
- raw archive: This folder is currently used in two ways. It can be the folder where files will be move from the input directory for some platforms (usually none fixed platform). Or, it can be the folder where for example daily files from some fixed station will be store. This directory can be organized, see the “Organize directory” python project for for details (gituser@portal.socib.es:repositories/organize_directory)

If the instrument name folder does not exist, then will search these folders in the “Path” folder.

Check folder permissions

The folder where input files are located and where input files will be archived, must have permissions “rwx” for the “dataprocuser” user, usually the rawInput and rawArchiveDaily directories.

Caches

VERY IMPORTANT: See the cache document

Datanucleus notes

The database has been modified over the time and will be modified in the future and therefore the Socib Datanucleus Library. These modifications can concern to JDO mapping or services, and if you run the Socib Processing and error related with the Datanucleus arises and looks like this:

```
javax.jdo.JDOFatalUserException: Clase "es.socib.database.instrumentation.Calibration" campo "es.socib.database.instrumentation.Calibration.sensorVariable" : definido en el MetaData, pero este campo no existe en la clase!
	at org.datanucleus.api.jdo.NucleusJDOHelper.getJDOExceptionForNucleusException(NucleusJDOHelper.java:528)
	at org.datanucleus.api.jdo.JDOPersistenceManagerFactory.freezeConfiguration(JDOPersistenceManagerFactory.java:781)
	at org.datanucleus.api.jdo.JDOPersistenceManagerFactory.createPersistenceManagerFactory(JDOPersistenceManagerFactory.java:326)
	at org.datanucleus.api.jdo.JDOPersistenceManagerFactory.getPersistenceManagerFactory(JDOPersistenceManagerFactory.java:195)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

It is an expected error that is easy to fix. Right click on the Socib Datanucleus Library -> Datanuclues -> Run enhancer tool.

Log files

The Socib Data Processing project generates three main log files: the processing.log, parsing_input_files.log and datanucleus.log. In production, the log files are located in /home/dataprocuser/processing_application/latest_version. In develop, the logs files are located inside the Socib Data Processing project.

The log level can be defined at the log4j.properties file, located in the src folder.

Socib data processing

See the README.md and sphinx documentation.

Socib processing library

See the README.md and sphinx documentation.

Implement a new parsing function

See the NewDataManager and ModelParser documentation (source code is a plus). Furthermore study the ModelParser implementations that are mentioned. The files are located in the es.socib.processing.parsinginputdata.

Implement a new deriving function

See the DerivingFunctionContainer documentation and source file.

Implement a new qc test

See the QCTestContainer documentation and source file.

Related documents

Sea level issue

https://docs.google.com/document/d/11dd32a8mcSKkIGuuvbDBIpb_02j65nOThGCt6dbS91s

Quality control

https://docs.google.com/document/d/1uRQZU1F68fV2GgjNH5h6M5LYToYo3meQR6ceSqhNtpw/edit

Rename variable names

https://docs.google.com/document/d/14zRxaGova1Qs0OzoeGKnBp5ejfoZgi1EyM-t9IVLamQ

Socib datanucleus library

See the README.md and sphinx documentation.

Socib definitions

See the README.md and sphinx documentation.

Implement a new service definition

Socib Meteo ocean toolbox

See the README.md and sphinx documentation.

Socib NetCDF to kml

See the README.md and sphinx documentation.