Setup Development Environment for the openMDM(R) Application

Eclipse mdmbl project

Document history:

Author	**Date**	**Affects Version**	**Description**
Angelika Wittek	9.6.2017	0.6	initial version
Angelika Wittek	12.6.2017	0.6	User Preference Service added
Alexander Nehmer	22.6.2017	0.6	Review and additions
Angelika Wittek	30.6.2017	0.6	Mailing Lists and ECA infos added
Angelika Wittek	18.7.2017	0.7	Comments from Ganesh inserted, Glassfish Bugs added
Angelika Wittek	31.07.2017	0.7	Elasticsearch version added, exported to pdf for mdmbl page
Alexander Nehmer	4.8.2017	0.7	Added Eclipse Hot Deploy section
Ganesh Deshvini	14.8.2017	0.7	Parameter to skip npm install
Angelika Wittek	13.9.2017	0.8	additions for new version exported to pdf for download pages
Alexander Nehmer	19.9.2017	0.8	Added SonarQube setup in Eclipse
Angelika Wittek	07.11.2017 21.11.2017	0.9	Extending the introduction chapter
Angelika Wittek	23.11.2017	0.9	changes in service.xml, restructuring chapters, exported for V0.9
Angelika Wittek	26.02.2018	0.10	changes for version V0.10
Angelika Wittek	20.03.2018	0.11	IP Management Chapter added
Angelika Wittek	02.05.2018	0.11	Chapter 6 extended
Angelika Wittek	14.05.2018	5.0.0M1	Updates for new version
Angelika Wittek	10.9.2018	5.0.0M2 / 5.0.0M3	Updates for new version
Angelika Wittek	25.10.2018	5.0.0M4	Realm configuration changes added.

Table of contents

Note: this document is written in Google Docs, location :
https://docs.google.com/document/d/1fs5p-UQceDV4dJeSAUWughCt-c1Cc82uhSxg8BRKsQs/edit?usp=sharing

1. Introduction

1.1. General

This document serves as a guide for developers of the Eclipse mdmbl project. It describes how to setup your environment, where the source code of existing projects can be found and retrieved, how the application can be built, deployed and how to start the application.

Mailinglists:

- For development communication we use the mdmbl mailing list: https://dev.eclipse.org/mailman/listinfo/mdmbl-dev
- The openMDM Working group mailing list: https://dev.eclipse.org/mailman/listinfo/open-measured-data-wg

For contributing to the mdmbl project you need to sign the ECA
(Eclipse Contributor Agreement) : https://wiki.eclipse.org/ECA

Helpful Links:

- Eclipse Wiki - openMDM EWG: https://wiki.eclipse.org/Open-Measured-Data-Management-WG
- Eclipse Project openMDM@BL: https://projects.eclipse.org/projects/technology.mdmbl
- Eclipse Bugzilla mdmbl issues: https://bugs.eclipse.org/bugs/buglist.cgi?quicksearch=openmdm&list_id=16180271
- JIRA REQU Issues: https://openmdm.atlassian.net/secure/RapidBoard.jspa?rapidView=57&projectKey=REQU&view=planning&selectedIssue=REQU-48
- openMDM git repos: http://git.eclipse.org/c/?q=mdm

1.2. Requirements and Bugs (JIRA and Bugzilla)

Requirements:

- Requirements are created in the JIRA REQU project [1]
- Requirements are ordered by priority and maturity by the Steering Committee
- for every REQU issue one or more Eclipse Bugzilla issues [2] are created
  - a comment with the link to the Bugzilla issue is added the the REQU issue
  - the Bugzilla issue summaries follow the naming convention:
    [REQU-xx] <summary>
- mdmbl dev team works on the Bugzilla issues

[1]https://openmdm.atlassian.net/secure/RapidBoard.jspa?projectKey=REQU&rapidView=57&view=planning

[2] https://bugs.eclipse.org/bugs/buglist.cgi?quicksearch=mdmbl

Bugzilla Bugs:

- created from JIRA REQU issues: the Bugzilla issue summaries follow the naming convention: [REQU-xx] <summary>, a link back to JIRA is also provided in a comment
- “real” bugs from internal or external
- technical requirements / enhancements from the team

1.3. Use of new Frameworks

To introduce a new framework to the code the QA guidelines of the EWG and the Legal Process of Eclipse (https://www.eclipse.org/legal/EclipseLegalProcessPoster.pdf) has to be followed:

- Get the approval for use from the the Architecture Committee. The members of the AC are listed here: https://wiki.eclipse.org/Open-Measured-Data-Management-WG.
- Get the IP approval from Eclipse, at least IP Check Type A, see Section 12 IP Management and Licensing Software

Only if you have both approvals, you can use the libraries in the code.

1.4. Branching and versioning

Version number system changes after version 0.10. Version 0.11 was renamed in 5.0.0M1 (Version 5.0.0, Milestone 1).

This affects only the following projects:

1. org.eclipse.mdm.api.base
2. org.eclipse.mdm.api.default
3. org.eclipse.mdm.api.odsadapter
4. org.eclipse.mdm.nucleus

Stable versions are on the **master** branches.
The current development happens on the **dev** branches.

The latest stable version is the current master. Older stable versions are tagged with the version number, e.g. 0.6, 0.7, 0.8. These numbers refer to the releases in the openMDM JIRA:
https://openmdm.atlassian.net/projects/REQU?selectedItem=com.atlassian.jira.jira-projects-plugin:release-page&status=released-unreleased

Also see the mdmbl download page:

https://projects.eclipse.org/projects/technology.mdmbl/downloads

For each version since 0.7 find documentation and release notes in Git:
http://git.eclipse.org/c/gerrit/mdmbl/org.eclipse.mdmbl.git/tree/Releases

For version since 5.0.0M1 find also the build artefacts in Git:
http://git.eclipse.org/c/gerrit/mdmbl/org.eclipse.mdmbl.git/tree/Releases

Note I:
Only use code from the same version between the projects! There are breaking changes between the versions!

Note II:
All jar files that are build have the suffix *-1.0.0.jar - from existing versions up to V0.10. Newer jar files reflect the version numbers.

1.5. Eclipse Infrastructure

1.5.1. Gerrit

The mdmbl project is using Gerrit for code reviews. To configure, refer to chapter 4.4 Configure Gerrit.
Gerrit has a two stage reviewing system:

1. when code is checked into Gerrit, a build is automatically started. Ther Gerrit flag “verified” is set to +1, if the build succeeded
2. a committer has to review the code and if ok, the code review flag is set to +2

1.5.2. Jenkins

https://ci.eclipse.org/mdmbl/

There a nightly builds for the dev and the master branches of the following projects:

5. org.eclipse.mdm.api.base
6. org.eclipse.mdm.api.default
7. org.eclipse.mdm.api.odsadapter
8. org.eclipse.mdm.nucleus

There are also builds for Gerrit and Sonar.

1.5.3. Sonar

The Eclipse SonarQube server is available via https://sonar.eclipse.org.
Rules: Qualitygate - sonar way with findbugs

Static code analysis is configured for the dev branches of the following projects:

1. org.eclipse.mdm.api.base
2. org.eclipse.mdm.api.default
3. org.eclipse.mdm.api.odsadapter
4. org.eclipse.mdm.nucleus

Generated and test sources are excluded.

1.6. ODS Server used for Developer Tests

All released versions are tested with the following ODS Server:

- HiQSoft GmbH, Version 4.6: https://www.highqsoft.com/de/avalon-asam-ods-server/
- Peak Solution GmbH, Version 2.5.8 http://www.peak-solution.de/de/produkte-leistungen/versuchs-messdatenmanagement/softwareloesungen/peak-ods-server/

NOTE: There is an issue with the an older Avalon Server version 4.3b (from 2013): if multiple catalog sensors are deleted, the server application throws an error and and leaves the application model in a broken state.

2. Prerequisites

Already installed on your machine or install it:

- Java 8: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Maven: https://maven.apache.org/
- Git: https://git-scm.com

Test, if programs are available from a console:

- > java -version
- > mvn --version
- > git --version

If not, add the programs to your PATH variable or set corresponding *_HOME variables.

3. Installations

3.1. Gradle

See https://gradle.org/install/ for installation on you system.
OR:

- Download Gradle from http://gradle.org/downloads
- Unzip the file into a directory without spaces (recommended).
- Create a GRADLE_HOME environment variable that points to this directory.
- Adjust your PATH environment variable to include $GRADLE_HOME/bin (%GRADLE_HOME%\bin on Windows).
- Test your setup by invoking `gradle --version`.

3.2. Eclipse IDE

Get the current JEE Distribution from https://www.eclipse.org/downloads
It is recommended to use the Eclipse Installer.
Note: installation is tested with Eclipse Neon and Eclipse Oxygen

Install Plugins:

- Gradle: follow this tutorial http://www.vogella.com/tutorials/EclipseGradle/article.html. Already included in the Eclipse Oxygen JEE Distribution
- SonarLint via Eclipse Marketplace
- EclEmma Code Coverage Tool (optional)

Create a new workspace for the mdmbl projects and configure it:

- set encoding to UTF-8

- set formatting rules to the Eclipse default rules

3.3. Database for the User Preference Service

The Preference service stores its data to a relational database. The database connection is looked up by JNDI and the JNDI name and other database relevant parameters are specified in src/main/resources/META-INF/persistence.xml. The default JNDI name for the JDBC resource is set to jdbc/openMDM.

For the User Preference Service you need a database with a schema “openMDM”.
Note: “openMDM” is the default schema name, it can be changed in the file:
`/org.eclipse.mdm.nucleus/org.eclipse.mdm.preferences/src/main/resources/META-INF/persistence.xml`

3.3.1. Apache Derby Database

The Glassfish Application Server has included the Derby libraries, so first get the Glassfish, as described this chapter.

Create database and tables:

- Change to your <glassfish_root> directory
- Start the database:.

```
<glassfish_root>$ ./glassfish/bin/asadmin start-database
```

- Start the ij Tool (http://db.apache.org/derby/papers/DerbyTut/ij_intro.html):

```
<glassfish_root>$ java -jar ./javadb/lib/derbyrun.jar ij
```

- Create database with default name “openMDM”

```
ij> CONNECT
'jdbc:derby://localhost:1527/openMDM;create=true';
```

- Create the table “preference”

```
ij> CREATE TABLE OPENMDM.PREFERENCE (ID BIGINT GENERATED BY
DEFAULT AS IDENTITY NOT NULL, keyCol VARCHAR(255), SOURCE
VARCHAR(255), username VARCHAR(255), valueCol
CLOB(2147483647) NOT NULL, PRIMARY KEY (ID));
```

- Add constraints

```
ij> CREATE TABLE OPENMDM.PREFERENCE (ID BIGINT GENERATED BY
DEFAULT AS IDENTITY NOT NULL, keyCol VARCHAR(255), SOURCE
VARCHAR(255), username VARCHAR(255), valueCol
CLOB(2147483647) NOT NULL, PRIMARY KEY (ID));
```

```
ij> ALTER TABLE OPENMDM.PREFERENCE ADD CONSTRAINT
UNQ_PREFERENCE_0 UNIQUE (source, username, keyCol);
```

- Check table:

```
ij> SELECT * FROM OPENMDM.PREFERENCE;
```

- close ij

```
ij> exit;
```

- Stop database.

```
<glassfish_root>$ ./bin/asadmin stop-database
```

3.3.2. Other Database Products

There is also the possibility to use other database products. For Postgres DB you will find the according sql scripts after the nucleus build in the following directory:
`$workspace/org.eclipse.mdm.nucleus/build/distributions/schema/org.eclipse.mdm.preferences`

Other database products supported by EclipseLink may also work, but are neither tested nor supported by the mdmbl project.

3.4. Glassfish

- Version 4.1.2 required
- Download Glassfish from https://javaee.github.io/glassfish/download
- Unzip the file into a directory without spaces (recommended).
- Test it:
  - change to the glassfish-root directory
  - invoke `./glassfish/bin/asadmin start-domain`
  - change to your browser
  - URL localhost:8080 should show you a welcome page
  - URL localhost:4848 should show you the admin page
  - Note: if you need to change the default ports 8080 or 4848 please see the glassfish documentation

Refer to this chapter for patching glassfish, patches are necessary for 4.2.1.

Not evaluated now for Glassfish 5.0.x. (2018-01-17). It seems to work without the patches, has to be tested some more. (TODO)

3.4.1. Configure JDBC for the User Preference Service DB

For the **User Preference Sevice DB** configure JDBC resource and its dependent JDBC Connection Pool. It has to be created and configured within the glassfish web administration console or through asadmin command line tool.

Description for the Derby Database with defaultname “openMDM”:

- Start Glassfish: `./glassfish/bin/asadmin start-domain`
- Start Derby DB: `./glassfish/bin/asadmin start-database`
- start: http://localhost:4848/
- Menu Item: JDBC-> JDBC Connection Pools -> new
    - poolname: <mypool_name>
    - Resource Type: javax.sql.DataSource
    - Database Driver Vendor: Derby
    - -> next
    - set properties: User, Password, DatabaseName to openMDM
    - -> finish
    - check it: open Connection Pool, try the ping button
- Menu item JDBC -> JDBC Resources -> new
    - JNDI NAME: jdbc/openMDM
    - Pool Name: <mypool_name>
- stop glassfish `./glassfish/bin/asadmin stop-domain domain1`
- stop Derby DB: `./glassfish/bin/asadmin stop-database`

3.5. Database for ODS-Server

The database product is dependent of the ODS Server you use. To start an ODS Server you need a loaded ASAM ODS Application Model in the DB.

3.5.1. Embedded Apache Derby Database and Peak ODS Server

If you use a Peak ODS Server from Peak Solutions GmbH:
they provide an embedded demo Derby Database with an Application Model and Data.
Follow their instructions.

3.5.2. Oracle 11g XE Release 2

- Download and install it: http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html
- Note for Ubuntu Users: see this article http://tuhrig.de/3-ways-of-installing-oracle-xe-11g-on-ubuntu/
- Note for Linux Users: Installing the Docker Container works, see: https://hub.docker.com/r/alexeiled/docker-oracle-xe-11g/

To load an empty Application Model to the DB: please refer to Jira-Ticket ORGA-178

3.6. ODS Server

ASAM ODS-Server e.g. from

- Peak Solution:
  http://www.peak-solution.de/de/produkte-leistungen/versuchs-messdatenmanagement/softwareloesungen/peak-ods-server/
  Note: you also need the **notification service plugin** for Peak ODS Server
- HiQSoft:
  https://www.highqsoft.com/de/avalon-asam-ods-server/
- or another compliant data source (e.g. PAK adapter)

- Get a (test) license from the vendors and follow the installation instructions.
- If you already imported an Application Model to your database and database is running, then the ODS Server should start.

Notes for running a Peak ODS Server:

- You need the notification service. Check if there is the
  `notification-plugin-1.1.0.jar`
  in the `$odsserver_root/lib directory` or copy it there.
- Configure parameters in `$odsserver_root/cfg/server.properties:`
  - Add the following line to enable the notification service:
    `JMS_FORWARDER.PORT=8089`
  - The Peak ODS Server can run its own ORB daemon. Just leave the following parameter blank: `NAMESERVICE=`
- Use the Peak demo Derby DB (MDMNVH DB) in embedded mode, the database starts automatically with the Peak ODS Server startup.
  Username and password are the name must be the name of the schema.
  - `DB_DRIVER = DERBY_EMBEDDED`
  - `DB_URL =`
    `jdbc:derby:/`**`<path_to_MDMNVH_DerbyDB>`**`/MDMNVH;create=false`
  - `DB_USER = MDMNVH`
  - `DB_PASSWORD = MDMNV`

3.7. ElasticSearch

ElasticSearch can be downloaded at https://www.elastic.co/products/elasticsearch
Use version 2.x., e.g.
https://www.elastic.co/de/downloads/past-releases/elasticsearch-2-4-2

Upgrade to version 5.x is planned, but it is not working yet, see:
https://bugs.eclipse.org/bugs/show_bug.cgi?id=520297

For testing purpose, it can be simply started by executing bin/run.bat

4. Get and build the code

4.1. Source Code Repositories

The application is split into several modules, each one dedicated to a certain scope of functions. All source code has been made available in git repositories hosted by the Eclipse Foundation https://eclipse.org/org/foundation/
The version control system used to track all changes is Git: https://git-scm.com/

We strongly recommend to use a separate workspace for the openMDM projects.

4.1.1. Manually checkout

- Change to the workspace directory
- Run the following command to checkout the source code. You have to adapt the URL for each project: `git clone <repository URL>`

Checkout the following projects

```
git clone http://git.eclipse.org/gitroot/mdmbl/org.eclipse.mdm.api.base.git
git clone http://git.eclipse.org/gitroot/mdmbl/org.eclipse.mdm.api.default.git
git clone http://git.eclipse.org/gitroot/mdmbl/org.eclipse.mdm.api.odsadapter.git
git clone http://git.eclipse.org/gitroot/mdmbl/org.eclipse.mdm.nucleus.git
```

Additional tools:

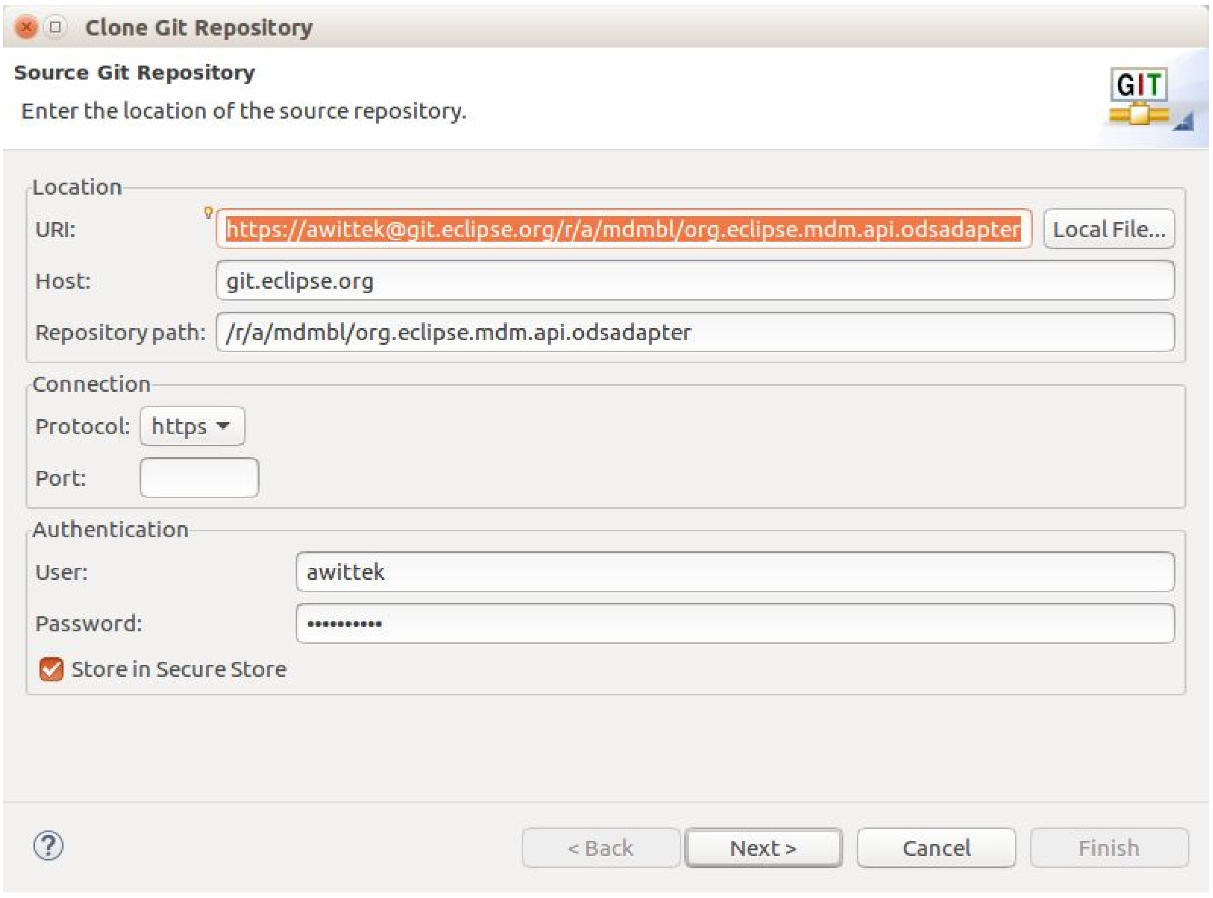
```
git clone http://git.eclipse.org/gitroot/mdmbl/org.eclipse.mdm.openatfx.mdf.git
git clone http://git.eclipse.org/gitroot/mdmbl/org.eclipse.mdm.mdfsorter.git
```

4.1.2. Checkout in Eclipse IDE

Alternatively configure eGit in your Eclipse IDE and checkout there.

- Open the Git Perspective
- Open "Clone a Git Repository …."
- Checkout with Username / Password:
  If you follow this way, the Git / Gerrit Configuration described in this chapter will be obsolete.
  https://<eclipse_username>@git.eclipse.org/r/a/mdmbl/<my_repo>,
  the password is the Gerrit Password, look there:
  https://git.eclipse.org/r/#/settings/http-password
  e.g. https://awittek@git.eclipse.org/r/a/mdmbl/org.eclipse.mdm.api.base
- Or anonymous with the URIs above.

- Screenshot:



Enter the location of the source repository.

Location
URI: https://awittek@git.eclipse.org/r/a/mdmbl/org.eclipse.mdm.api.odsadapter Local File...
Host: git.eclipse.org
Repository path: /r/a/mdmbl/org.eclipse.mdm.api.odsadapter

Connection
Protocol: https
Port:

Authentication
User: awittek
Password: ••••••••••
[✓ Store in Secure Store]

4.2. Importing projects to the Eclipse IDE

Import the project to your Eclipse IDE via File -> Import -> Import as existing gradle project. If importing all projects at once via the parent directory fails, import each project separately. Build the projects in the order they are listed above, because of the dependencies between them.
Refer to the documentation and follow it:

- http://git.eclipse.org/c/gerrit/mdmbl/org.eclipse.mdm.nucleus.git/tree/README.md

If there are still java errors after importing and building, right click on the project -> Gradle -> Refresh Gradle Project
For other build problems. see the section 4.4 Known setup bugs. If your problem is not listed there, create a Bugzilla Bug for it:
https://bugs.eclipse.org/bugs/enter_bug.cgi?product=MDMBL

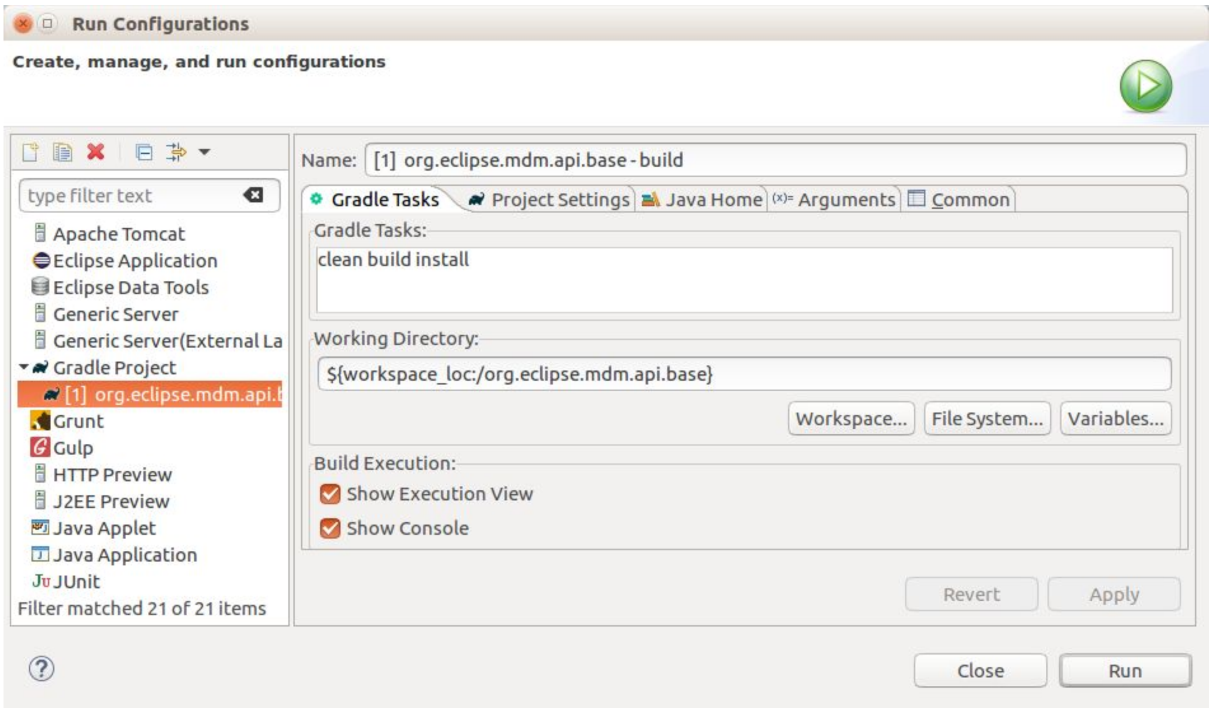
After importing the projects to the IDE, you have all repository connections in the Git perspective.

4.3. Building the projects

4.3.1. Building projects separately

Run `./gradlew clean build` in the root directory of each project to build it.
Execute `./gradlew clean build install` also generates a JAR to be placed in your local cache. From there it can be retrieved when subsequent projects are build.

Or you build the projects in the Eclipse IDE via a Run Configuration:



Name: [1] org.eclipse.mdm.api.base - build

Gradle Tasks: clean build install

Working Directory: ${workspace_loc:/org.eclipse.mdm.api.base}

Build Execution:
- [✓] Show Execution View
- [✓] Show Console

Please see the README.md files in the root directory of the projects for exact information about how the projects are build.

Note: some projects depend on each other, so you have to build them in a specific order:

1. org.eclipse.mdm.api.base
2. org.eclipse.mdm.api.default
3. org.eclipse.mdm.api.odsadapter
4. org.eclipse.mdm.nucleus

The other projects can be built independently.

4.3.2. Building all projects for the core application

Running the gradlew script from org.eclipse.mdm.nucleus project will build the following projects:

1. org.eclipse.mdm.api.base.git
2. org.eclipse.mdm.api.default.git

3. org.eclipse.mdm.api.odsadapter.git
4. org.eclipse.mdm.nucleus.git

4.3.3. Building the projects without web frontend

To build the application without the web frontend, pass the argument **skipNode** to the build script of the org.eclipse.mdm.nucleus project.

```
> ./gradlew clean build -PskipNode
```

4.4. Configure Git and Gerrit

For checking in code to a repository you need to configure Gerrit in the Git perspective.

Additional informations:

- Gerrit can be found at https://git.eclipse.org/r/
- There is a tutorial, it also explains how to use Gerrit with Eclipse: http://www.vogella.com/tutorials/Gerrit/article.html
- See documentation from Eclipse: https://wiki.eclipse.org/Gerrit

Prerequisites:

- Get your Gerrit Password via: https://git.eclipse.org/r/#/settings/http-password
- Configure your Gerrit settings: https://git.eclipse.org/r/#/settings/ -> Watch the mdmbl projects.

4.4.1. Configure repositories

Go to your Eclipse IDE -> Git Perspective:
For all projects do:

- Expand the project
- Remotes -> origin -> Select "Configure Push" from context menu -> OK
- URI -> click change button
- Configure URI Window:
    - URI: https://<eclipse_username>@git.eclipse.org/r/a/mdmbl/<projectname> e.g. https://<eclipse_username>@git.eclipse.org/r/a/mdmbl/org.eclipse.mdm.api.base
    - Protocol: choose HTTPS
    - Fill in username and password (the Gerrit one from the settings above)
    - Check the option "Store in secure store"
- Back to the "Configure Push" Window, section "Ref mapping" -> add
    - Local Branch: HEAD
    - Choose remote branch:
        - for the dev branch e.g. refs/for/dev
        - for the master branch: refs/heads/master
    - OK
- Try the "Dry-Run" Button to test your configuration

- Save
- “Gerrit Configuration” (right click on the remote)
    - URI: same as above
    - Username: same as above
    - Destination branch: dev (or another)
- Finish

4.4.2. Configure Push

To push code to Gerrit you can

1) As a committer
    a) Push code directly to Gerrit without any review (in Git repositories view: Push to upstream)
    b) Push code to Gerrit for reviewing (in Git repositories view: Push to Gerrit)
2) As a non-committer
    a) Push code to Gerrit for reviewing (in Git repositories view: Push to Gerrit)

4.4.3. Commits to Gerrit

- Add Signed-Off-By when you commit to sign off the commit
- In Eclipse: open the git staging perspective, there is a window called "Commit message" and next to the label "Commit Message" there are 3 buttons. the middle button is the "Add signed-off-by" button. Click it.

5. Deploy and configure application

before deploying the application:

- start ORB (`$JAVA_HOME/bin/orbd -ORBInitialPort 2809`) (skip this if you are using Peak ODS Server with no NAMESERVICE specified in the server.properties)
- start ODS Server with corresponding database
- start Elasticsearch
- start Glassfish
    - `> asadmin start-database`
    - `> asadmin start-domain`

The build of the nucleus module creates:

```
/org.eclipse.mdm.nucleus/build/distributions/mdm_web-VERSION.zip
```

The ZIP archive contains:

- the backend org.eclipse.mdm.nucleus-VERSION.war
- the configurations in /configuration
- sql scripts for the User Preference Service in /schema.

Configuration:

- copy the content of the extracted /configuration folder to `$GLASSFISH_ROOT/domains/domain1/config`
- Configuration file: `org.eclipse.mdm.property/`**`global.properties`**
  To enable globally the freetext search set the parameter: `freetext.active=true`
- if you enable the freetext.active parameter (=true), make sure
  - that ElasticSearch is started and that the port in the property `elasticsearch.url` is set correct (check it in the ElasticSearch log and via your browser with the url and port, the result should be a json response)
  - the freetext parameters are set in the service.xml
  - freetext search is active for at least one service
- Configuration file: **`org.eclipse.mdm.connector/service.xml`**
  - configure the data sources, for ODS Servers look into your ODS Server log file to determine the corba URL
  - To use the freetext search configure for each datasource separately:
    - specific parameters for the NotificationService and the freetext search
    - set the freetext.active parameter to true (Example2)
  - Disable the freetext search for a datasource:
    - set the freetext.active parameter to false (Example3)

    or
    - leave away the freetext.* parameters, as they are optional (Example1)

```
<services>
<!-- Example1: ODS Server without freetext.* parameters -> freetext search is not active -->
<service entityManagerFactoryClass="org.eclipse.mdm.api.odsadapter.ODSContextFactory">
		<param name="nameservice">corbaloc::1.2@YOUR_HOST1:2809/NameService</param>
		<param name="servicename">YOUR_SERVICE1.ASAM-ODS</param>
</service>
<!-- Example2: Peak ODS-Sever with active freetext search -->
<service entityManagerFactoryClass="org.eclipse.mdm.api.odsadapter.ODSContextFactory">
		<param name="nameservice">corbaloc::1.2@YOUR_HOST2:2809/NameService</param>
		<param name="servicename">YOUR_SERVICE2.ASAM-ODS</param>
		<!--The indexing requires a user to get the DataItems from the ODS Server. Those are the
credentials for the user -->
		<param name="freetext.active">true</param> <!--to configure the Avalon, set to false (*) -->
		<param name="freetext.user">sa</param>
		<param name="freetext.password">sa</param>
		<param name="freetext.notificationType">peak</param>
		<param name="freetext.notificationUrl">http://YOUR_HOST2:8089/api</param>
</service>

<!-- Example3:  Avalon ODS-Sever -> freetext search is not active-->
<service entityManagerFactoryClass="org.eclipse.mdm.api.odsadapter.ODSContextFactory">
		<param name="nameservice">corbaloc::1.2@YOUR_HOST3:2809/NameService</param>
		<param name="servicename">YOUR_SERVICE3.ASAM-ODS</param>
		<!--The indexing requires a user to get the DataItems from the ODS Server. Those are the
credentials for the user -->
		<param name="freetext.active">false</param> <!--to configure the Avalon, set to true (*) -->
		<param name="freetext.user">sa</param>
		<param name="freetext.password">sa</param>
```

```
        <param name="freetext.notificationType">avalon</param>
        <param name="freetext.pollingInterval">5000</param>
</service>
</services>
```

- configure the LoginModule (see org.eclipse.mdm.realms - README.md)
- restart the application server

Deploy the backend (org.eclipse.mdm.nucleus-VERSION.war) on your Glassfish server => do it via the admin console at http://localhost:4848/ and adapt the Context Root and the Application Name to “org.eclipse.mdm.nucleus”.

- goto http://localhost:8080/org.eclipse.mdm.nucleus and Login with sa/sa

5.1. Configure Login Module (Realm Configuration)

The realm is configured in a standardized way. Configure your LDAP, AD or other systems according to the glassfish documentation:
https://javaee.github.io/glassfish/doc/4.0/security-guide.pdf

For local installations you can setup and configure a file realm, described in the readme.md file of the org.eclipse.mdm.nucleus project:
http://git.eclipse.org/c/mdmbl/org.eclipse.mdm.nucleus.git/tree/README.md

Note: The component "org.eclipse.mdm.realms" is not used any longer, since 5.0.0M4

5.2. Headless Deployment

The default configuration for the authentication method is set to “FORMULAR” . If you want to do a headless deployment change it to “BASIC”.

Change this file:
org.eclipse.mdm.nuclues/org.eclipse.mdm.application/src/main/webconfig/web.xml

from
`<auth-method>FORMULAR</auth-method>`
to
`<auth-method>BASIC</auth-method>`

6. Start application

- start ORB (`$JAVA_HOME/bin/orbd -ORBInitialPort 2809`) (skip this if you are using Peak ODS Server with no NAMESERVICE specified in the server.properties)

- start the database for the ODS Server (if necessary)
- start the ODS server
- start Elasticsearch
- start Glassfish (including database for UPS)

Change to your browser URL is http://localhost:8080/org.eclipse.mdm.nucleus.
You should see the openMDM LoginPage. Look for user/ password in the database in the userXX table, e.g. sa/sa.

7. Known (setup) problems and solutions

7.1. Build

- The build of the org.eclipse.mdm.api.odsadapter project does need a “gradle clean install” as a “gradle install” breaks with build errors that package “com.google.protobuf” does not exist

7.2. Glassfish

7.2.1. Glassfish - Inconsistent Module State

- `org.glassfish.deployment.common.DeploymentException: Error in linking security policy for org.eclipse.mdm.nucleus -- Inconsistent Module State`
    - Deployment went wrong
        - delete `$glassfish_root/glassfish/domains/domain1/applications/org.eclipsemdm.nucleus`
        - delete `$glassfish_root/glassfish/domains/domain1/generated`
        - restart Glassfish

7.2.2. Glassfish - java.lang.ClassNotFoundException

If you run into "java.lang.ClassNotFoundException: javax.xml.parsers.ParserConfigurationException not found by org.eclipse.persistence.moxy" this is a bug described in https://bugs.eclipse.org/bugs/show_bug.cgi?id=463169 and https://java.net/jira/browse/GLASSFISH-21440.

This solution is to replace GLASSFISH_HOME/glassfish/modules/org.eclipse.persistence.moxy.jar with this: http://central.maven.org/maven2/org/eclipse/persistence/org.eclipse.persistence.moxy/2.6.1/org.eclipse.persistence.moxy-2.6.1.jar

7.2.3. Glassfish - org.osgi.framework.BundleException

If you run into “org.osgi.framework.BundleException: Unresolved constraint in bundle com.fasterxml.jackson.module.jackson-module-jaxb-annotations”: this is a compatibility problem with the installed jackson libraries.
The solution is to replace
GLASSFISH_HOME/glassfish/modules/jackson-*.jar with these files:

- http://central.maven.org/maven2/com/fasterxml/jackson/core/jackson-annotations/2.8.1/jackson-annotations-2.8.1.jar
- http://central.maven.org/maven2/com/fasterxml/jackson/core/jackson-core/2.8.1/jackson-core-2.8.1.jar
- http://central.maven.org/maven2/com/fasterxml/jackson/core/jackson-databind/2.8.1/jackson-databind-2.8.1.jar
- http://central.maven.org/maven2/com/fasterxml/jackson/jaxrs/jackson-jaxrs-base/2.8.1/jackson-jaxrs-base-2.8.1.jar
- http://central.maven.org/maven2/com/fasterxml/jackson/jaxrs/jackson-jaxrs-json-provider/2.8.1/jackson-jaxrs-json-provider-2.8.1.jar

Rename the jars to the replaces (remove the versions).

7.2.4. Glassfish - java.lang.NoSuchMethodError

If you are using the REST-API to access the MDM entities (CREATE und UPDATE) and encounter the following error:

```
java.lang.NoSuchMethodError:
com.fasterxml.jackson.databind.deser.std.UntypedObjectDeserializer
.<init>
```

replace the jackson libraries as mentioned in 7.2.3.

7.2.5. Glassfish - WEB9102: Login failed: Lookup failed

If you cannot login to the webclient and the glassfish log shows the following error:

```
WEB9102: Web Login Failed:
com.sun.enterprise.security.auth.login.common.LoginException:
Login failed: Lookup failed for
'java:global/org.eclipse.mdm.nucleus/ConnectorService!org.eclipse.
mdm.connector.boundary.ConnectorService' in
SerialContext[myEnv={java.naming.factory.initial=com.sun.enterpris
e.naming.impl.SerialInitContextFactory,
java.naming.factory.state=com.sun.corba.ee.impl.presentation.rmi.J
NDIStateFactoryImpl,
java.naming.factory.url.pkgs=com.sun.enterprise.naming}|#]
```

check that the war file was deployed with the Application Name “org.eclipse.mdm.nucleus”

8. Extended configurations (run & IDE)

8.1. Hot Deployment

- On the command line run "gradlew clean install" in org.eclipse.mdm.nucleus
- Eclipse setup:
- Import all projects into Eclipse (tested with Buildship 2.2.x)
- Import -> Gradle Project
    - org.eclipse.mdm.nucleus
- In project "org.eclipse.mdm.nucleus"
    - Project Properties
        - Project Facets (if project is not faceted yet)
            - Convert to faceted form ...
            - Select "Dynamic Web Module"
    - Builders (optional)
        - Disable "Validation" (speeds up redeployment)
    - Deployment Assembly
        - Remove all entries
        - Add following sources
            - To Deploy Path WEB-INF/lib
                - Java Build Path Entries
                    - Add Project and External Dependencies
                - Project
                    - org.eclipse.mdm.api.base
                    - org.eclipse.mdm.api.default
                    - org.eclipse.mdm.api.odsadapter
                    - org.eclipse.mdm.businessobjects
                    - org.eclipse.mdm.connector
                    - org.eclipse.mdm.filerelease
                    - org.eclipse.mdm.freetextindexer
                    - org.eclipse.mdm.preferences
                    - org.eclipse.mdm.property
            - To Deploy Path WEB-INF/classes
                - Folder
                    - org.eclipse.mdm.application/bin
            - To Deploy Path /
                - Folder
                    - org.eclipse.mdm.application/build/node/dist
            - To Deploy Path /WEB-INF
                - Folder
                    - org.eclipse.mdm.application/src/main/webconfig

- In projects
    - org.eclipse.mdm.api.base
    - org.eclipse.mdm.api.default
    - org.eclipse.mdm.api.odsadapter
    - org.eclipse.mdm.businessobjects
    - org.eclipse.mdm.connector
    - org.eclipse.mdm.filerelease
    - org.eclipse.mdm.preferences
    - org.eclipse.mdm.property

    - Project Properties
        - Project Facets
            - Set Java Version to 1.8
- In project "org.eclipse.mdm.connector"
    - Gradle -> Refresh Gradle Project

- Check that in project org.eclipse.mdm.nucleus the Deployment Assembly settings were not resetted. Otherwise redo.

- Setup Glassfish (Servers View)
    - Servers -> Create new
    - Complete setup for local Glassfish instance
        - Install Glassfish tools if needed (https://marketplace.eclipse.org/content/glassfish-tools)
    - Double-click on Glassfish server
        - Publishing
            - "Automatically publish when resources change"
    - Context Menu -> Add and Remove ...
        - Add org.eclipse.mdm.nucleus to Glassfish

- Configure Glassfish
    - Open Management Console (localhost:4848)
        - Configurations
            - server-config
                - System Properties
                    - Add property org.jboss.weld.serialization.beanIdentifierIndex Optimization to false

- Start Glassfish in "Debug"-Mode to have your changed sources published

8.2. Angular Live Development Server

During development a full build of the MDM5 Webclient, which is part of the nucleus, is quite time consuming. It is more efficient to skip the client build entirely by using the parameter -PskipNode, which excludes the JavaScript-Code from being build and included in the war file. For the UI you can start the "NG Live Development Server".
It is started by invoking "npm start" in the root path of the angular application (org.eclipse.mdm.nucleus\org.eclipse.mdm.application\src\main\webapp). After successful compilation you are able to see the webclient under http://localhost:4200/.
Changes to the angular application will trigger an incremental compile of the changed sources.
Additionally the the live development server implements a proxy, which forwards all requests to http://localhost:4200/org.eclipse.mdm.nucleus/* to http://localhost:8080/. So if the Glassfish with the MDM5 backend does not run on port 8080, you have to change the proxy configuration in org.eclipse.mdm.nucleus/org.eclipse.mdm.application/src/main/webappproxies/development.config.json.
To avoid issues with the form based authentication, it is easier to switch to basic authentication by changing the web.xml in this scenario. Otherwise you may get errors for the backend requests, because you are not automatically forwarded to the login page.

8.3. SonarQube

The Eclipse SonarQube server is available via https://sonar.eclipse.org.

Note: Do not use SonarLint plugin as the current version does not support the old SonarQube version at Eclipse and the old one has less features that the SonarQube plugin

- Install SonarQube plugin version 3.5
  - Install new Software…
    - Add http://downloads.sonarsource.com/eclipse/eclipse/
    - Select all plugins to be installed
    - Oxygen might complain about Erlang Configuration Helper to not be installable
- Configure SonarQube in Eclipse: Eclipse Preferences -> SonarQube -> Servers
  - Edit the default server or create a new one if none present
    - SonarQube Server ID: Eclipse
    - SonarQube Server URL: https://sonar.eclipse.org
    - Username: your username for eclipse.org
    - Password: your password for eclipse.org
- Associate projects with Eclipse SonarQube:
  - Edit Project Properties for all projects from
    - org.eclipse.mdm.api.base
    - org.eclipse.mdm.api.default

        - org.eclipse.mdm.api.odsadapter
        - org.eclipse.mdm.nucleus
- Configure
        - Associate with SonarQube: Association should be proposed automatically
- Available Views are
    - SonarQube Issues
    - SonarQube Rule Description
- Analysis can be performed via
    - Project context menu
    - SonarQube -> Analyze
    - The results of the analysis are then written to Eclipse SonarQube

9. Development Rules

9.1. Commit comments

Prefix every commit with the Bugzilla ID if there is one, e.g. your commit message would be “518433: made some changes”.

9.2. Tasks in the code

TODO / FIXME and other tasks have to follow this naming schema:

```
// {TASK_NAME} {my_committer_id}, {yyyy-MM-dd}: {my_comment}

e.g.
// TODO mkoller, 2017-11-14: Currently only the first Adapter is
indexed.
```

You can do that manually or you can import the following snippet In the Eclipse IDE via Preferences ->Java -> Editor -> Templates

```
<!-- Import: Eclipse IDE -> Window -> Prefernces -> Java -> Editor Templates -> Import --><!-- User
Variable: add to eclipse.ini -> "-Duser.name=<my name>" →
<templates>
       <template autoinsert="true" context="java" deleted="false" description=""
enabled="true"        name="TODO">// TODO ${user}, ${d:date('yyyy-MM-dd')}:
      </template>
     <template autoinsert="true" context="java" deleted="false" description=""
enabled="true" name="FIXME">// FIXME ${user}, ${d:date('yyyy-MM-dd')}:
      </template>
</templates>
```

Usage in code: "TOD"+ [Ctrl+Space]

You have to set your Eclipse user to your committer ID:
edit the file eclipse.ini and set the parameter:
-Duser.name=your_eclipse_committer_id

9.3. Working with committer feature branches

Develop on your committer feature branch. Make code reviews as usual via Gerrit on your branch.

Committer feature branches have a special name scheme:
${my_committer_id}/my_branch.
Only for those references under refs/heads (and refs/tags) with your Committer ID you will be able to delete the branch afterwards.

- To create a branch: create the branch with the name scheme ->
  ${my_committer_id}/my_branch
- To delete a branch in Eclipse: Project → Team → Remote → Push …
    - “Next” on first wizard screen
    - Select branch to delete in “Add delete ref specification”
    - Click “Add Spec” (the branch is now listed in “Specifications for push”
    - “Finish”
- When integrating your branch into the dev branch
    - Checkout the dev branch
    - Merge your feature branch into the dev branch, but with having Git a merge commit created although it was a fast-forward merge
      ```
      git merge --no-ff your_feature_branch
      ```
    - Push the dev branch to Gerrit

Note: Without the no-ff option Gerrit refuses to accept the push due to “no new changes”.

10. Tipps+Tricks

10.1. Handling of Relations

- First, you should consider if you really need to access relations directly. For most use cases there already is a function in the OpenMDM 5 business logic available which does what is required.
- The BaseEntityManager interface allows you to traverse parent and children relations.
- If you need to read other relation types, you can implement and use a BaseEntitySearchQuery. For details look at the method changeRelation in org.eclipse.mdm.api.odsadapter.RelationTest.

- For setting relations you need to access to BaseEntityFactory.get*Store. Typically this is only possible for the adapter implementation, because the contents of the stores might not be 100% consistent over different adapters. The adapter may choose make these methods publicly accessible, as is the case for the ODSAdapter, but be aware that this is currently adapter specific. See org.eclipse.mdm.api.odsadapter.RelationTest for an example. Please note that you can only modify mutable relations (in the MutableStore), e.g. no parent child relations.

11. Troubleshooting

Look into the Logfiles:

- Glassfish: `$glassfish_root/domains/domain1/logs/server.log`
- Derby DB for User Preference Service: `$glassfish_root/databases/derby.log`
- The Logfiles of your ODS compatible datasource

12. IP Management and Software Licensing

12.1. Software Licensing under the EPL

Important points for licensing software under the EPL 1.0:

- Comply to the rules of 3rd party library licenses including compatibility to EPL 1.0
- Provide legal documentation
    - Provide EPL 1.0 copyright header
    - Provide the EPL 1.0 Licence File
    - Provide a notice file
    - Provide End User Content

Note: Licensing software under the EPL 1.0 does not mean, that the code has to be hosted at Eclipse. You can host it in any repository, e.g. www.github.com.

Explained in detail:

12.1.1. Comply to the rules of 3rd party library licenses

There are two main points:

- declare the usage of the 3rd party libraries as described in the according licenses
- check compatibility to the EPL 1.0

See: https://www.eclipse.org/projects/handbook/#ip-third-party

12.1.2. Provide EPL 1.0 copyright headers

See: https://www.eclipse.org/projects/handbook/#ip-copyright-headers

```
/*
 * Copyright © {date} {owner}[ and others]
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/epl-v10.html
 */
```

12.1.3. Provide the Licence file

- provide the license file in every repo at root level.
- provide the license file in every distribution you build and deliver (zip, war, jar, ...)

See:

- https://www.eclipse.org/projects/handbook/#legaldoc-license
- https://www.eclipse.org/projects/handbook/#legaldoc-repo

12.1.4. Provide a Notice File

- provide the notice file in every repo at root level.
- provide the notice file in every distribution you build and deliver (zip, war, jar, ...)

See:

- https://www.eclipse.org/projects/handbook/#legaldoc-notice
- https://www.eclipse.org/projects/handbook/#legaldoc-distribution

12.1.5. Provide End User Content

See: https://www.eclipse.org/projects/handbook/#legaldoc-end-user

12.2. The Eclipse Foundation IP Process

The Eclipse IP Process supports you that your Eclipse project is compliant to the EPL
That means, you have to follow the process. The IP Process is mainly an interaction between the Eclipse IP Team (and its proxies and tools) and the committers of a project. To keep things transparent, the communication in the IP Process is mainly via email and the Eclipse IP tracking tool IP ZIlla.The main points are discussed here, also refer to: https://www.eclipse.org/projects/handbook/

Main points:

- committers have to follow the committer due diligence guidelines
- regularly run through an IP Check for your provided software

- make an IP Check for 3rd party libraries you want to use BEFORE you check them into the repositories
- regularly run through an IP Check for 3rd party libraries used for test & build
- Provide legal documentation
    - Provide copyright notice (provenance)
    - provide a EPL 1.0 license file
    - provide a Notice File and keep it up to date
    - provide End User content

The Eclipse Legal Process Poster:
https://www.eclipse.org/legal/EclipseLegalProcessPoster.pdf

The Eclipse IP Process in Eight Cartoons:
https://www.eclipse.org/projects/dev_process/ip-process-in-cartoons.php

Explained in detail:

12.2.1. The committer due diligence guidelines

Every committer has signed the committer paperwork:
https://www.eclipse.org/projects/handbook/#paperwork

That includes that **a committer has to follow the due diligence guidelines** (“Sorgfaltspflicht”):
https://www.eclipse.org/legal/committerguidelines.php

12.2.2. IP Checks, CQs, Type A / Type B and reuse

12.2.2.1. Explanation Type A / Type B IP Checks:

- Type A: License Compatibility Certification
- Type B: Full IP Due Diligence (License, Provenance, Scanning)
- Exact defintion: https://www.eclipse.org/org/documents/Eclipse_IP_Policy.pdf

For every IP Check you have to create a Contribution Questionnaire (CQ) in the Eclipse IPZilla Tool.

For every CQ you can decide if a Type A or a Type B check is performed. As long as the Eclipse projects are in the incubation phase, IP Checks of Type A are sufficient.
From experience Type A checks are performed must faster at Eclipse then Type B checks.

For 3rd party libraries reuse CQs where possible. That means if you want to use a library with version x and there is already a CQ from another project for exact this library you can create an Piggyback CQ. The advantage is that these CQs are performed automatically and run trough very fast (in general about 1 hour).

See also these blog posts:
https://waynebeaton.wordpress.com/2017/01/16/eclipse-infrastructure-support-for-ip-due-due-diligence-type/
https://mmilinkov.wordpress.com/2016/06/29/overhauling-ip-management-at-the-eclipse-foundation/

12.2.2.2. Getting started with CQs (Links)

This section is for commiters of the Eclipse project, only they have acces to the CQ Pages.

1. Login to IP-Zilla: https://dev.eclipse.org/ipzilla/
   Via the search button you can define filters for your project or look for existing CQs
2. Create a new CQ:
   a. Go to yout project page and log in, e.g.
      https://projects.eclipse.org/projects/technology.openk-platform/
   b. On the right side you have the committer tools, choose "Create a Contribution Questionnai…"

12.2.3. IP Checks for Eclipse projects

Regularly run through an IP Check for your provided software.

For Eclipse Projects IP Checks have to be provided from the current dev team (committers) for every release and /or requested by the Steering Committee.

As long as the projects are in the incubation phase, IP Checks of Type A are sufficient.

Regularly accomplished IP Checks minimize the risk of unclear state of the Terms of Use.

12.2.4. IP Checks for 3rd party libraries used for productive code

- run through an IP Check for every 3rd party library you want to use BEFORE you check them into the repositories and / or reference them via your build scripts
- As long as the projects are in the incubation phase, IP Checks of Type A are sufficient.

Note: this is not only about Java libraries, it also applies for all other libraries you use, e.g. JavaScript, Python, …

The IP Team must review third party content if:

- *the Java/OSGi manifest for one of the project bundles makes a direct reference to third party content (either a bundle or package);*
- *project code includes an import statement for a package from third party content;*
- *project code uses reflection or other means to reference APIs and implementation;*
- *project code uses OSGi Services to make a reference to a specific implementation of a service; or*
- *project code invokes a "command line" tool.*

This list is not intended to be exhaustive.

From: https://www.eclipse.org/projects/handbook/#ip-third-party

https://blogs.eclipse.org/post/wayne-beaton/decoding-eclipse-ip-policy-third-party-dependencies

12.2.5. IP Checks for 3rd party libraries used for test & build only

See: https://wiki.eclipse.org/Development_Resources/IP/Test_and_Build_Dependencies

12.2.6. Provide legal documentation

See: https://www.eclipse.org/projects/handbook/#legaldoc

That includes:

- Provide Copyright Headers, see 2.1 Provide EPL 1.0 copyright headers
- Provide the EPL 1.0 Licence File, see 2.2 Provide the Licence file
- Provide a notice file, see 2.4 Provide a Notice File
- Provide End User Content, see 2.5 Provide End User Content