Installation and Setup Guide for the openMDM® Application

Eclipse mdmbl project

Version 5.2.0, 2024-05-01: {build-revision}

Table of Contents

Document history:

Author	Date	Affects Version	Description
Angelika Wittek, Alexander Nehmer, Matthias Koller	9.6.2017-14.5.2019	<5.0.0	Initial version User Preference Service added Mailing Lists and ECA infos added Comments from Ganesh inserted, Glassfish Bugs added Elasticsearch version added Added Eclipse Hot Deploy section Parameter to skip npm install exported to pdf for download pages Added SonarQube setup in Eclipse Extending the introduction chapter changes in service.xml, restructuring chapters, exported for V0.9 changes for version V0.10 IP Management Chapter added Chapter 6 extended
Angelika Wittek, Matthias Koller	14.05.2018-20.2.202	5.0.0++	Updates for new version
Angelika Wittek, Matthias Koller	09.03.2020	5.1.0	Updates for new version merged Setup Guide and Installation Guide to this document.
Angelika Wittek, Simon Skoczylas	27.04.2020	5.2.0M1	Small fixes from Simon's review Added updates for database setup and roles.
Angelika Wittek	20.05.2020	5.2.0M2	Repository restructuring
Matthias Koller	10.11.2020	5.2.0M3	Updated New and Noteworthy
Angelika Wittek, Matthias Koller, Simon Skoczylas	15.03.2021	5.2.0M4	Document transferred to AsciiDoc Nodeprovider Config added Updates for new milestone
Angelika Wittek, Matthias Koller	15.6.2023	5.2.0M5	Updates for new milestone

1. Milestones & Releases: New and Noteworthy

1.1. Milestone 5.2.0M5

A user guide is available, see the project page, section project links .

1.2. Milestone 5.2.0M4

Node Provider: The navigation structure of the client application can be adapted to the needs with one or more configurations of the Node Provider, see here.

Documentation: This documentation was transferred to AsciiDoc, the sources are now part of the mdm Git repository: `org.eclipse.mdm/doc`

1.3. Milestone 5.2.0M3

Support for importing/exporting ATFx external component files directly was added. Therefore some API methods in **org.eclipse.mdm.api.base** were changed. See release_notes.md for a detailed list of changes. The quickviewer and x-y-charviewer in the web client was extended to show multiple channels in its data table.

1.4. Milestone 5.2.0M2

The old repositories are not used any more, they will be archived soon. Do not use!

New Repository Structure, all code from following the repositories moved to **org.eclipse.mdm**

- org.eclipse.mdm.api.base
- org.eclipse.mdm.api.default
- org.eclipse.mdm.api.odsadapter
- org.eclipse.mdm.nucleus

The move includes the history, the branches master, dev, release_candidate and existing committer feature branches. The following tags were created: 5.0.0, 5.1.0, 5.2.0M1. If you need an older tag file a Bugzilla Bug.

The structure in the new repository:

- root: build scripts, documentation and legal documentation
- api/
  - base/: code from former repos org.eclipse.api.base and org.eclipse.api.default, package names are unchanged
  - odsadapter/: code from repo org.eclipse.api.odsadapter

    - atfxadapter/: code from repo org.eclipse.mdm.nucleus/org.eclipse.mdm.api.atfxadapter
- nucleus/
    - code from repo org.eclipse.mdm.nucleus except for the atfxadapter component
- gradle/: the gradle wrapper
- doc/: documentation from org.eclipse.mdm.nucleus/doc
- tools/: new folder for common tooling, e.g. formatter, repository migration file

New build parameter for headless build and deployment:
see section Headless Deployment.

1.5. Milestone 5.2.0M1

Changes to MDMRealm:

Up to version 5.1.0 all Users had to be assigned the group `MDM` to use openMDM.
Since 5.2.0M1 users need one of the following groups: `Admin`, `DescriptiveDataAuthor`, `Guest`.
See `org.eclipse.mdm/readme.md#Configure LoginModule` for more information.

Database schema change for User Preference Service:

In this release openMDM 5 database schema was extended.
If you have an existing installation make sure to apply the provided update script to your database.

2. Introduction

2.1. General

This document serves as a setup and installation guide for the Eclipse mdmbl project.
For **developers** is describes how to setup the environment, where the source code of the project can be found, retrieved,build and deployed.

For **users** interested only in deploying / installing the openMDM Application, start here.

Note: This guide describes the deployment on a local machine only, for installing the application in a company infrastructure, please contact your administrators to support you with firewalls and proxy configurations, grant permissions, etc.

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/automotive.mdmbl
- Eclipse Bugzilla mdmbl issues
- openMDM git repos:
  https://projects.eclipse.org/projects/automotive.mdmbl/developer

Downloads: Get the artefacts from the Project Download Page: openMDM_application-<version>.zip
https://projects.eclipse.org/projects/technology.mdmbl/downloads

2.2. Requirements and Bugs

For all issues the Eclipse Bugzilla is used.

Requirements:

- Requirements are created in the Eclipse Bugzilla System
- Requirements are ordered by priority and maturity by the Steering Committee

- for every REQU issue one or more Eclipse Bugzilla Tasks are created
- mdmbl dev team works on the Bugzilla issues

Bugzilla issues:

- Requirements
- “real” bugs from internal or external
- technical requirements / enhancements from the team

2.3. Use of third-party content / IP Management

To introduce a new framework to the code the QA guidelines of the EWG and the Legal Process of Eclipse 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
  Note: since 2020 there is no active Architecture Committee
- Get the IP approval for third party content from the Eclipse Foundation.

See IP Checks for 3rd party content.

2.4. Branching and versioning

There are releases and milestones for the code in org.eclipse.mdm, milestones are marked with “M”, e.g. 5.2.0M2 is the 2nd milestone on the way to the 5.2.0 release.

Stable versions are on the **master** branch.
The current development happens on the **dev** branch.
There are also several other branches, please contact us via the mailing list if you need more information.

The latest stable version is on the current master. Older stable versions are tagged with the version number, e.g. 5.1.0, 5.2.0M1.

For all stable versions the build artefacts and documentation are made available on the mdmbl download page:
https://projects.eclipse.org/projects/automotive.mdmbl/downloads

Documentation and release notes in Git:
https://git.eclipse.org/c/mdmbl/org.eclipse.mdm.git/tree/

2.5. Eclipse Infrastructure

2.5.1. Gerrit

The mdmbl project is using Gerrit for code reviews. To configure, refer to chapter Configure Gerrit.

Gerrit has a two stage reviewing system:

1. when code is checked into Gerrit, a build is automatically started. The Gerrit flag “verified” is set to +1, if the build succeeded
2. a committer/contributor has to review the code and if ok, the code review flag is set to +2
3. a commiter has to submit the changes, the Gerrit branch will be merged in the destination branch

2.5.2. Jenkins

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

There a nightly builds for the dev and the master branches of org.eclipse.mdm
There are also builds for Gerrit.

2.6. ODS Server used for Developer Tests

All released versions are tested with the following ODS Server:

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

3. Prerequisites

Already installed on your machine or install it:

- Java 8:
  - Oracle Java SE 8u201 and higher http://www.oracle.com/technetwork/java/javase/downloads/
  - AdoptJDK: openJDK 8 with HotSpot https://adoptopenjdk.net/
- Git: https://git-scm.com

Test, if programs are available from a console:

- `>java -version`
- `>git --version`

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

3.1. Gradle

See https://gradle.org/install/ for installation on you system. Version: 4.10.2 (updated 21.2.2020) This is optional, as the projects are using the Gradle Wrapper which is delivered with the source code.

3.2. Eclipse IDE

Get the current "Eclipse IDE for Enterprise Java Developers" Distribution from https://www.eclipse.org/downloads It is recommended to use the Eclipse Installer.

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

- set encoding to UTF-8
- set Java Code Style formatting rules to the Eclipse default rules [built in]

Formatter:

We highly recommend to use the Eclipse 2019/03 or higher IDE. Set the formatting rules to the Eclipse default rules [built in]. If you use a former version, then import the formatting rules from: org.eclipse.mdm/tools/eclipse_formatter.xml

This is **important** as the default rules in Eclipse have changed with the releases.

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/

4.1.1. Manually checkout

Change to the workspace directory. Run the following command to checkout the source code:

```
git clone <repository URL>
```

Get the clone URLs from the Project Page

Checkout the following repositories for the openMDM Application:

- org.eclipse.mdm

There are additional openMDM tools:

- org.eclipse.mdm.openatfx.mdf.git
- org.eclipse.mdm.mdfsorter.git

4.1.2. Checkout in Eclipse IDE

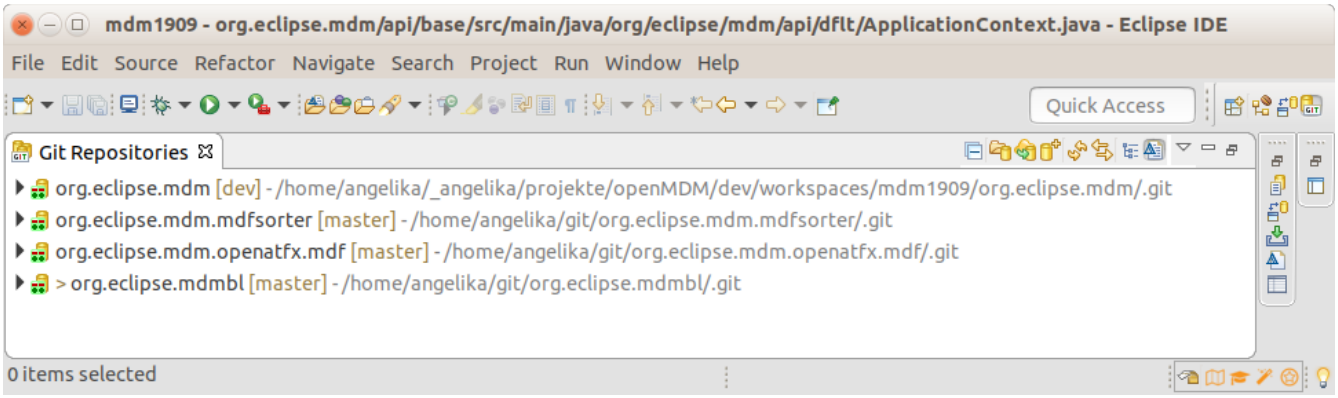
Alternatively configure eGit in your Eclipse IDE and checkout there.

- Open the Git Perspective
- Open “Clone a Git Repository ....”
- Checkout with Username / Password

The password is the Gerrit Password, look there:
https://git.eclipse.org/r/#/settings/http-password

After cloning the repositories, your Git perspective:

NOTE	If these "green symbols" are not present, it means that Gerrit has not yet been configured correctly. Follow the instructions in this chapter.



4.2. Importing projects to the Eclipse IDE

Import the project to your Eclipse IDE via File → Import → Import as existing gradle project.

If the import is not working, check if the Gradle version is compatible with the Java Version your IDE is using.
Gradle documentation to find the compatible combinations

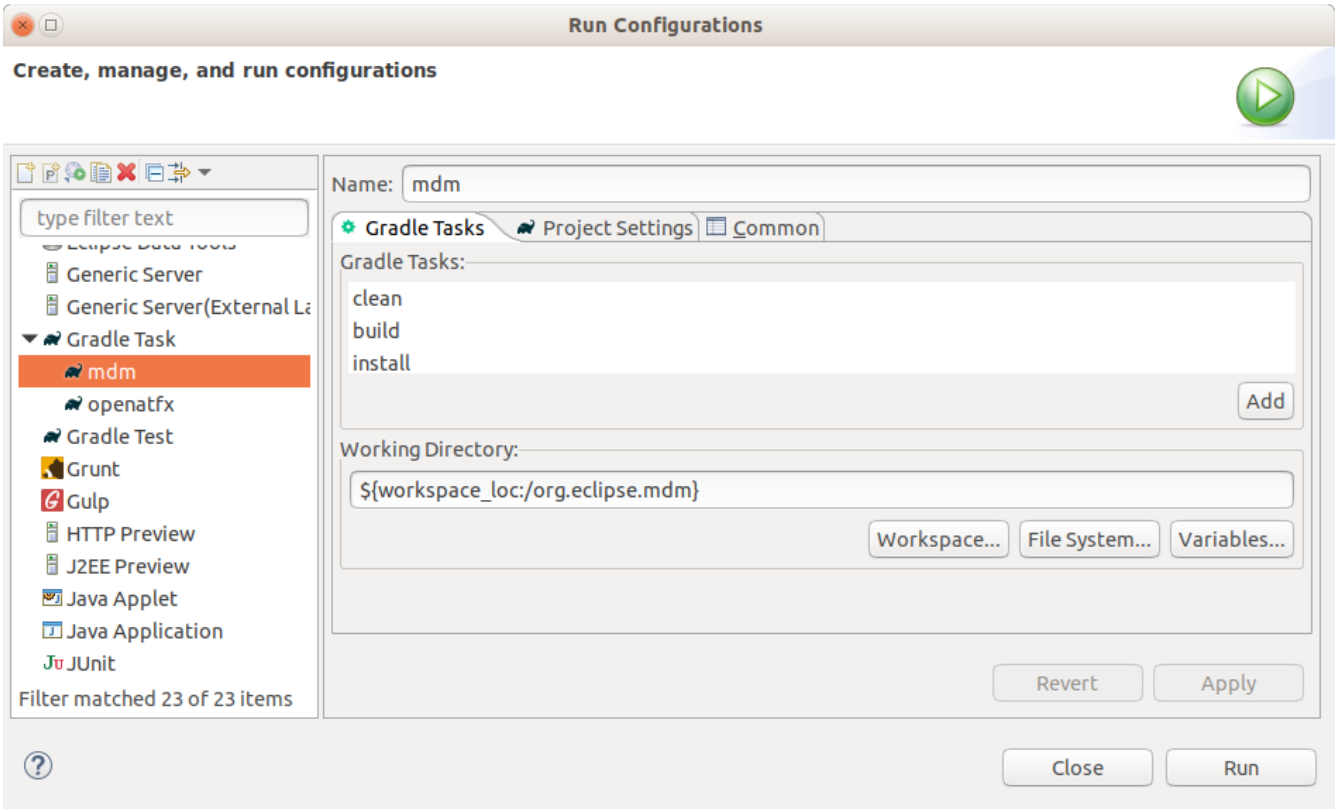
Tested for Eclipse IDE 2023-06:

- IDE runs with Java 17
- install Gradle 8.1.1
- import as existing gradle project → import options
  - select "Override worspace settings"
  - check and configure "Local installation directory"

4.3. Building the projects

Run `./gradlew clean build` in the root directory of org.eclipse.mdm. Execute `./gradlew clean build install` also generates the jars, and the war file.

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



The following projects are build:

1. apicopy
2. application
3. businessobjects
4. connector
5. filerelease
6. freetextindexer
7. preferences
8. property
9. webclient

If you do not see these projects after the build in your Eclipse IDE, just import the org.eclipse.mdm project again as “existing gradle project”.

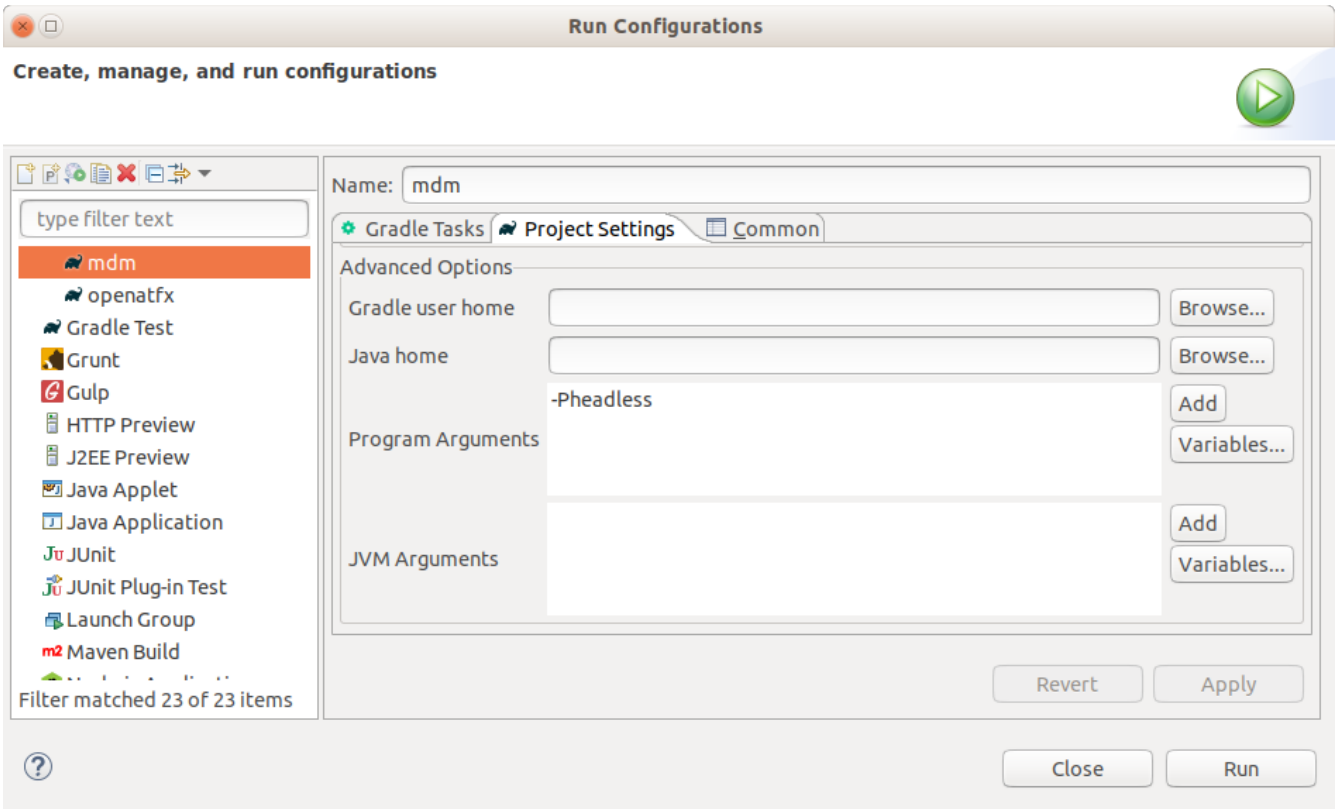
4.3.1. Building the projects without the webclient

To build the application without the webclient, pass the argument headless to the gradle script of the `org.eclipse.mdm` project. The webclient is not included to the war file.

```
./gradlew clean build -Pheadless
```

In your console you get the message:

```
Configure project :nucleus Headless build! Webclient will not be included into web
archive. To build and include the webclient remove the property headless or set
-Pheadless=false.
...
Task :nucleus:webclient:install SKIPPED
```



If there are still java errors after importing and building, right click on the project → Gradle → Refresh Gradle Project
For other build problems, see section Known (setup) problems and solutions.

If your problem is not listed there, create a Bugzilla Bug for it:

https://bugs.eclipse.org/bugs/enter_bug.cgi?product=MDMBL

4.3.2. Building the asciidoc documentation only

For generating the documentation, UserGuide and GettingStartedGuide, as pdf and html:

```
./gradlew asciidoctor asciidoctorPdf -Pheadless
```

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/
  → Watched Projects → Add the mdmbl projects.

4.4.1. Configure repositories

Go to your Eclipse IDE → Git Perspective:

- 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
    ```

  - 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
- 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. The recommenden way:
      Push code to Gerrit for reviewing (in Git repositories view: Push to Gerrit)
   b. Push code directly to Gerrit without any review (in Git repositories view: Push to upstream)
2. As a non-committer:
   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 the Eclipse IDE:
  open the git staging perspective, open the window "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. Preconditions and required installations

5.1. Glassfish

Supported versions:

Glassfish, Version 4.1.2

- Download Full Platform: https://javaee.github.io/glassfish/download
- Administration-Guide: https://javaee.github.io/glassfish/doc/4.0/administration-guide.pdf
- Security-Guide: https://javaee.github.io/glassfish/doc/4.0/security-guide.pdf
- Some patches are necessary. Refer to this chapter for patching glassfish. [TODO link to chapter]

Glassfish, Version 5.1.0

- Full Platform from https://projects.eclipse.org/projects/ee4j.glassfish/downloads
- Deployment with Glassfish Administration Console is not possible, because of a Glassfish Bug: 545170 - "Archive Path is NULL" when deploying from GUI (https://bugs.eclipse.org/bugs/show_bug.cgi?id=545170)
- Administration-Guide: https://javaee.github.io/glassfish/doc/5.0/administration-guide.pdf + Security-Guide: https://javaee.github.io/glassfish/doc/5.0/security-guide.pdf

Installation:

- 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

5.2. 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`

For activating the change, the artefacts have to be rebuilt.

5.2.1. Apache Derby Database

The Glassfish Application Server has included the Derby libraries.

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

- Check table:

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

- Create the table “system_process” (since 5.2.0M1):

```
ij> CREATE TABLE OPENMDM.SYSTEM_PROCESS (ID BIGINT GENERATED BY DEFAULT AS IDENTITY
NOT NULL, process_id VARCHAR(255), process_key VARCHAR(255), last_locked BIGINT,
PRIMARY KEY (ID));
```

- close ij

```
ij> exit;
```

- Stop database.

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

5.2.2. 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.

5.2.3. 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 through asadmin command line tool. (the glassfish web administration console is not recommended, as it is pretty buggy).

Description for the Derby Database with defaultname “openMDM”:

1. Start domain, database and admin-tool:

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

2. Create JDBC Connection Pool “openMDM” with User, Password and DatabaseName “openMDM”:

```
asadmin> create-jdbc-connection-pool --datasourceclassname
org.apache.derby.jdbc.ClientDataSource --restype javax.sql.DataSource --property
password=openMDM:user=openMDM:serverName=localhost:databaseName=openMDM:connectionA
ttributes=\;create\\=true openMDM
```

3. Check JDBC Connection Pool:

```
asadmin>ping-connection-pool
asadmin>list-jdbc-connection-pools
```

4. Create JDBC Resource:

```
asadmin> create-jdbc-resource --connectionpoolid openMDM jdbc/openMDM
```

5. Check JDBC Resource:

```
asadmin>list-jdbc-resources
```

6. Stop domain, database:

```
<glassfish_root>$ ./bin/asadmin stop-domain domain1
<glassfish_root>$ ./bin/asadmin stop-database
```

5.3. ElasticSearch

Download ElasticSearch 7 for openMDM 5.1.0++:
https://www.elastic.co/products/elasticsearch or
https://www.elastic.co/de/downloads/past-releases#elasticsearch

For openMDM 5.0.x and 5.1.Mx use version 2.x., e.g. https://www.elastic.co/de/downloads/past-releases/elasticsearch-2-4-2

For testing purpose, it can be simply started by executing:

```
<elasticsearch_root>$ ./bin/elasticsearch
```

5.4. 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.

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.

5.5. ODS Server

ASAM ODS-Server e.g. from

- Peak Solution:
  http://www.peak-solution.de/de/produkte-leistungen/versuchs-messdatenmanagement/softwareloesungen/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 two additional plugins copy them to $odsserver_root/plugins:
  - Put peakcorbafileserver-VERSION.jar into plugins
  - Put notification-plugin-VERSION.jar into plugins
- 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 same 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
```

6. Install, deploy and configure application

For deployment, you need the build artefact:
org.eclipse.mdm.nucleus-<version>.war

This file is an element of the file: openMDM_application-<version>.zip

Get this file from:

1. the Project Download Page: https://projects.eclipse.org/projects/technology.mdmbl/downloads
2. or from your build: /org.eclipse.mdm/build/distributions/

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.

6.1. Configuration Files

Copy the content of the extracted /configuration folder to:
`$GLASSFISH_ROOT/glassfish/domains/domain1/config`

Configuration file - global.properties:
`org.eclipse.mdm/nucleus/property/src/main/configuration/global.properties`

To enable globally the freetext search set the parameter:
`freetext.active=true`, if enabled, 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 - service.xml:
`org.eclipse.mdm/nucleus/connector/src/main/configuration/service.xml`

- configure the data sources, for ODS Servers look into your ODS Server log file to determine the corba URL
- the sourceName is the name of the datasource within openMDM 5. It must be unique among all services and should only contain alphanumeric characters and underscores.
- 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 sourceName="YOUR_SERVICE1"
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 sourceName="YOUR_SERVICE2"
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 sourceName="YOUR_SERVICE3"
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>
```

Restart the application server.

Note: An explanation about the the corbaloc and corbaname URLs, find here:

http://www.ciaranmchale.com/corba-explained-simply/the-corbaloc-and-corbaname-urls.html

6.2. Configure Login Module (Realm Configuration)

The realm is configured in a standardized way. Configure your LDAP, AD or other systems according to the glassfish security documentation: Glassfish 5 or Glassfish 4.

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

Note: The component "org.eclipse.mdm.realms" is not used any longer, since 5.0.0M4. And it is no longer supported.

6.3. Property lookups for parameters values in service.xml

The service.xml contains all information necessary for the Connector-Service to connect to the available data sources/adapter instances. Since this information includes secret information like passwords, it is possible to provide lookups, which gives you the possibility to specify tokens as references to properties defined elsewhere.

There are different lookups available:

- sys: Looks up variables defined as system properties
- env: Looks up variables defined as environment variables

Example:

```
<param name="password">${env:odsPassword}</param>
```

6.4. Configure system properties

- use system property "org.eclipse.mdm.api.odsadapter.filetransfer.interfaceName" to set a specific network interface name to be used
- use system property “org.eclipse.mdm.configPath” to redefine the location of the Glassfish config folder, e.g. “GLASSFISH_ROOT/glassfish/domains/domain1/config”

6.5. Deployment of org.eclipse.mdm.nucleus

Deploy the backend (org.eclipse.mdm.nucleus-VERSION.war) on your Glassfish server. Execute the following command (replacing PATH-TO-WAR and VERSION)

```
<glassfish_root>$ ./bin/asadmin
asadmin> deploy --force=true --name org.eclipse.mdm.nucleus   <PATH-TO-
WAR>/org.eclipse.mdm.nucleus-<VERSION>.war
```

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

NOTE	Deployment in GlassFish 5 Server Administration Console is not working.

6.6. Headless Deployment

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

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

from

```
<auth-method>FORM</auth-method>
to
<auth-method>BASIC</auth-method>
```

6.7. Node Provider

The navigation structure of the client application can be adapted to the needs with one or more configurations of the Node Provider. The Node Provider consists of a JSON based configuration to define the navigation structure.

The configuration of a Node Provider must be created as preference with the prefix `nodeprovider.`.

A Node Provider can be configured in three different ways.

1. Web Client (see Nodeprovider Configuration)
2. REST Interface (Preferences Endpoint)
   Take a look at `http://{SERVER}:{PORT}/{APPLICATIONROOT}/swagger.html`
3. Database access (not recommended)

6.7.1. Node Provider JSON structure

The structure of the JSON document must be as follows:

- Root node
- Nodes for structure configuration

First the root entry of the JSON document must contain an `id`, a `name` and a context mapping.

Example Node provider root

```
{
    "id" : "generic_measured",
    "name" : "Generic Measured",
    "contexts" : {
        "*" : {}
    }
}
```

The context mapping can either contain a wildcard `*` or the name of one application context available in the used openMDM configuration. For example “NVHDEMO”. The wildcard configuration will be applied to each application context available.

Each context mapping consists of nested nodes which in sum describe the navigation structure configuration.

Each of that nested nodes consist of the following attributes:

- `type` (String)
  As the type of the node from the openMDM model. For example “Environment”, “Project” or a node for the given Context where the names depend on the given model definition.
- `filterAttributes` (Array of Strings)
  The attributes of a node which should be used for filtering. For example “Id”, “Name”.
- `labelAttributes` (Array of Strings)
  The attributes of a node which should be used for displaying.
- `labelExpression`
  An EL Syntax based expression to format the display name. The attributes used as “labelAttributes” can be used here. For example “${Id)” to display the Id of a node.
- `valuePrecision` (String)
  One of: `YEAR`, `MONTH`, `DAY`, `HOUR` or `MINUTE`
  Which allows to group selected DATE attributes.
- `contextState` (String)
  One of: `MEASURED`, `ORDERED`
  Allows to set the context for Context related nodes to get either the measured or the ordered value.
- `virtual` (Boolean)
  Defines a new Node which is not part of the default navigation structure.
- `child` (Object with next structure)
  Defines the next level of the navigation structure

Example Node

NOTE	This example only shows 3 levels of nested nodes. The full configuration should at least contain configuration down to a Measurement entity.

```
"type" : "Environment",
"filterAttributes" : ["Id"],
"labelAttributes" : ["Name"],
"virtual" : false,
"child" : {
    "type" : "Project",
    "filterAttributes" : ["Id"],
    "labelAttributes" : ["Name", "Id"],
    "virtual" : false,
    "child" : {
        "type" : "vehicle",
        "filterAttributes" : ["model"],
        "labelAttributes" : ["model"],
        "labelExpression" : "Model: ${model}",
        "virtual" : true,
        "contextState": "MEASURED",
    }
}
```

7. Start application

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)
- start Glassfish
  - `asadmin start-database`
  - `asadmin start-domain`

THe browser URL is http://localhost:8080/org.eclipse.mdm.nucleus.

You should see the openMDM LoginPage. Lock in with for user/ password from the userXX table in zhe database, e.g. sa/sa.

7.1. 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

8. Known (setup) problems and solutions

8.1. Glassfish

8.1.1. 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.bounda
ry.ConnectorService' in
SerialContext[myEnv={java.naming.factory.initial=com.sun.enterprise.naming.impl.Serial
InitContextFactory,
java.naming.factory.state=com.sun.corba.ee.impl.presentation.rmi.JNDIStateFactoryImpl,
java.naming.factory.url.pkgs=com.sun.enterprise.naming}|#]
```

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

8.1.2. Glassfish 4 only - 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.

The **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

8.1.3. Glassfish 4 only - 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 replaced ones (remove the versions).

8.1.4. Glassfish 4 only - java.lang.NoSuchMethodError

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

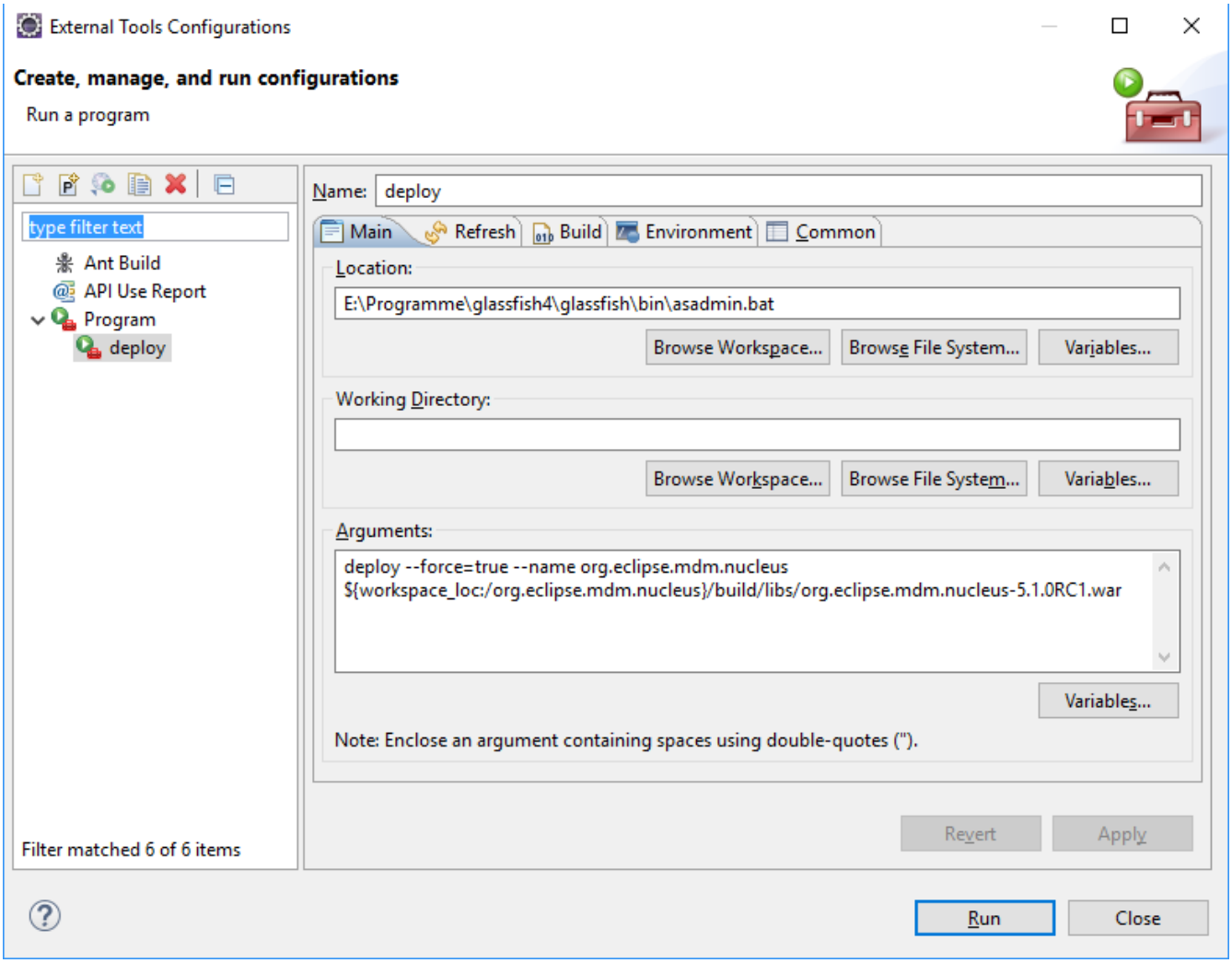
```
java.lang.NoSuchMethodError:
com.fasterxml.jackson.databind.deser.std.UntypedObjectDeserializer.<init>
```

replace the jackson libraries as mentioned here.

9. Extended configurations (run & IDE)

9.1. Deployment via Eclipse IDE

To deploy the nucleus in Glassfish asadmin can be configured as an External Tool in Eclipse:



9.2. 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 -Pheadless, 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/webclient\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/webclient/src/main/webapp/proxies/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.

10. Development Rules

10.1. Commit comments

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

10.2. 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

10.3. Working with committer feature branches

Committer Feature Branches are special branches a committer can create and delete. These branches have a dedicated naming 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

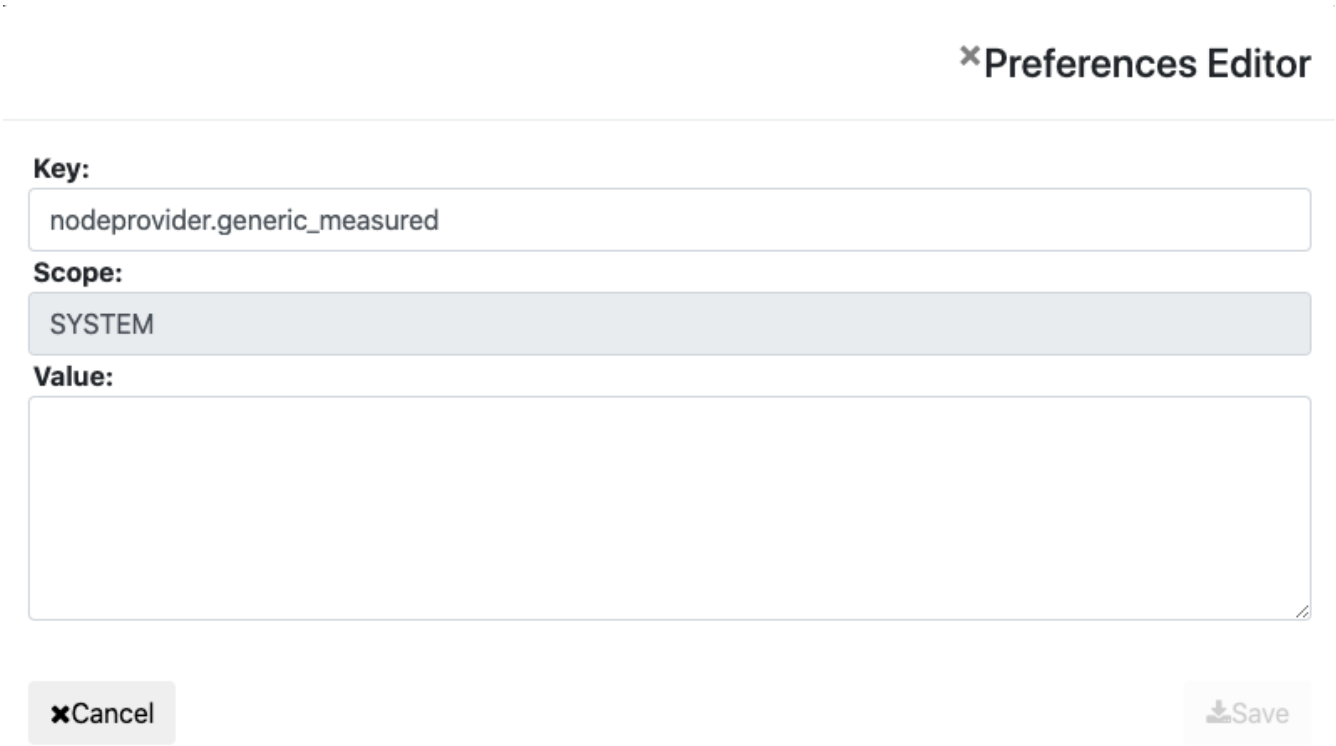
Note:
Without the no-ff option Gerrit refuses to accept the push due to “no new changes”.

11. Web Frontend Configuration

11.1. System Preferences

11.1.1. Node Provider Configuration

To create a Node Provider in the Web UI, an administrative user must login into the application. After login, it is necessary to switch to the administration page and add a system preference to the configuration. The configuration must be located in the system scope and the key must be prefixed with `nodeprovider.`.



Example Node provider key:

- nodeprovider.generic_measured

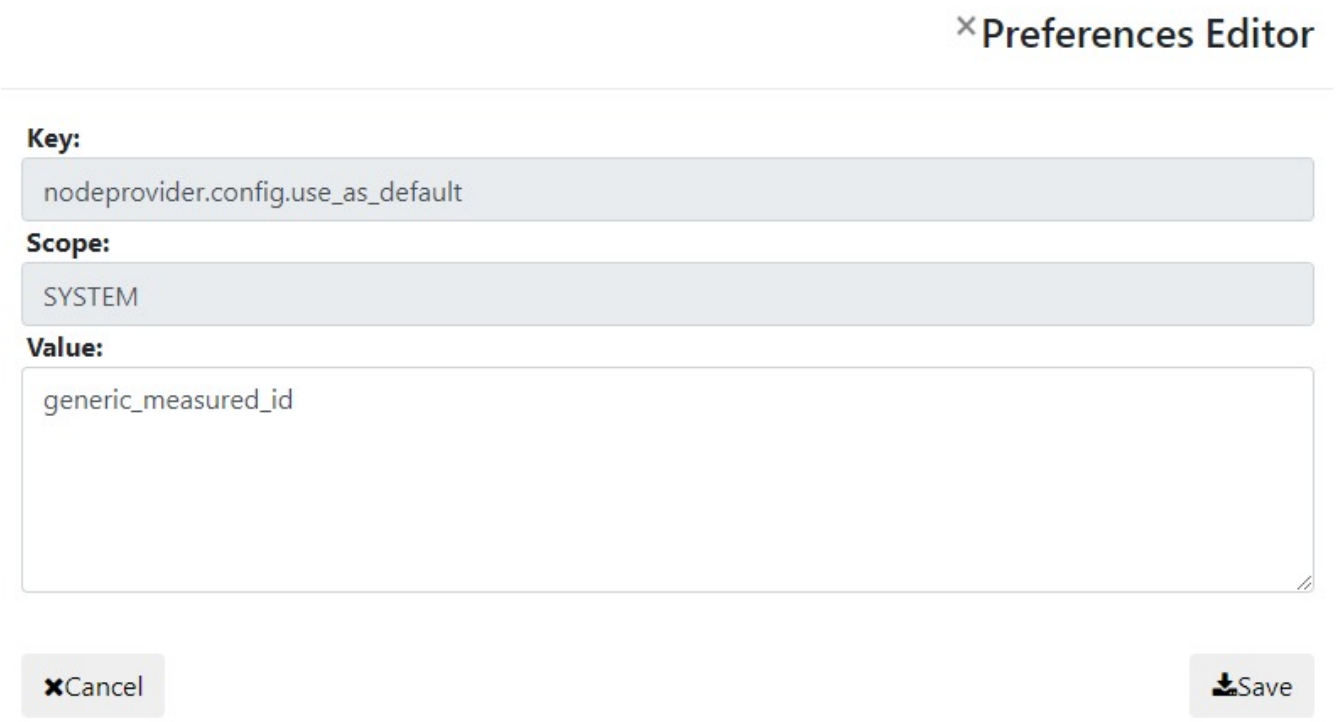
The internal default node provider can be deactivated by setting the following key to 'false':

- nodeprovider.config.default_is_active

A new default node provider can be set by providing the full key of that node provider as value for the following key:

- nodeprovider.config.use_as_default

Following the above example the value would be:



11.1.2. Search Preferences

For the search default values a System Preference can be configured.

Result types

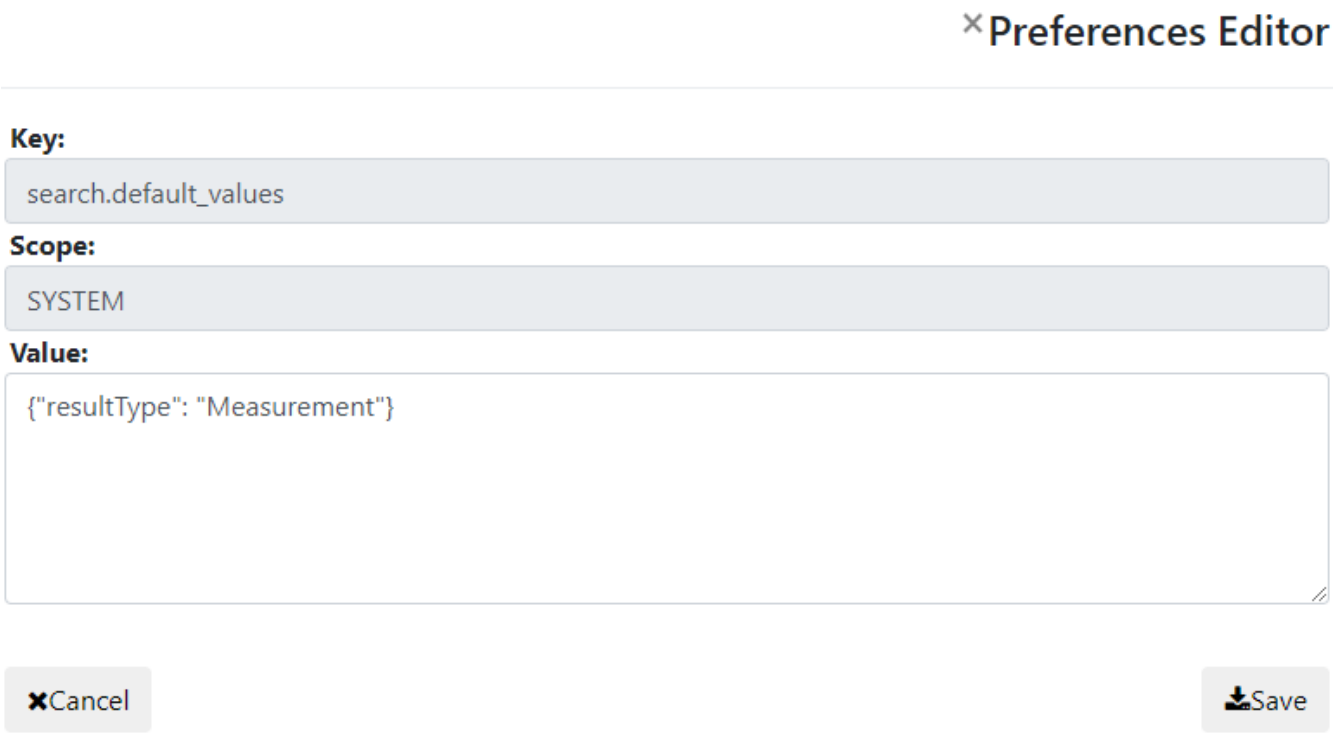
To configure the "Result type" default value:

- **Key:** search.default_values
- **Value:** a valid JSON string e.g.:

```
{"resultType": "Measurement"}
```

- Possible **resultType** values are:
  - Test
  - TestStep
  - Measurement

Following the above example the value would be:



Case Sensitivity

Search and filtering can be made case sensitive by adding the following system preference to true.

- **Key:** search.case_sensitive
- **Value:** true

JSON

To activate the button "JSON" which will provide the search request as JSON string use the following configuration:

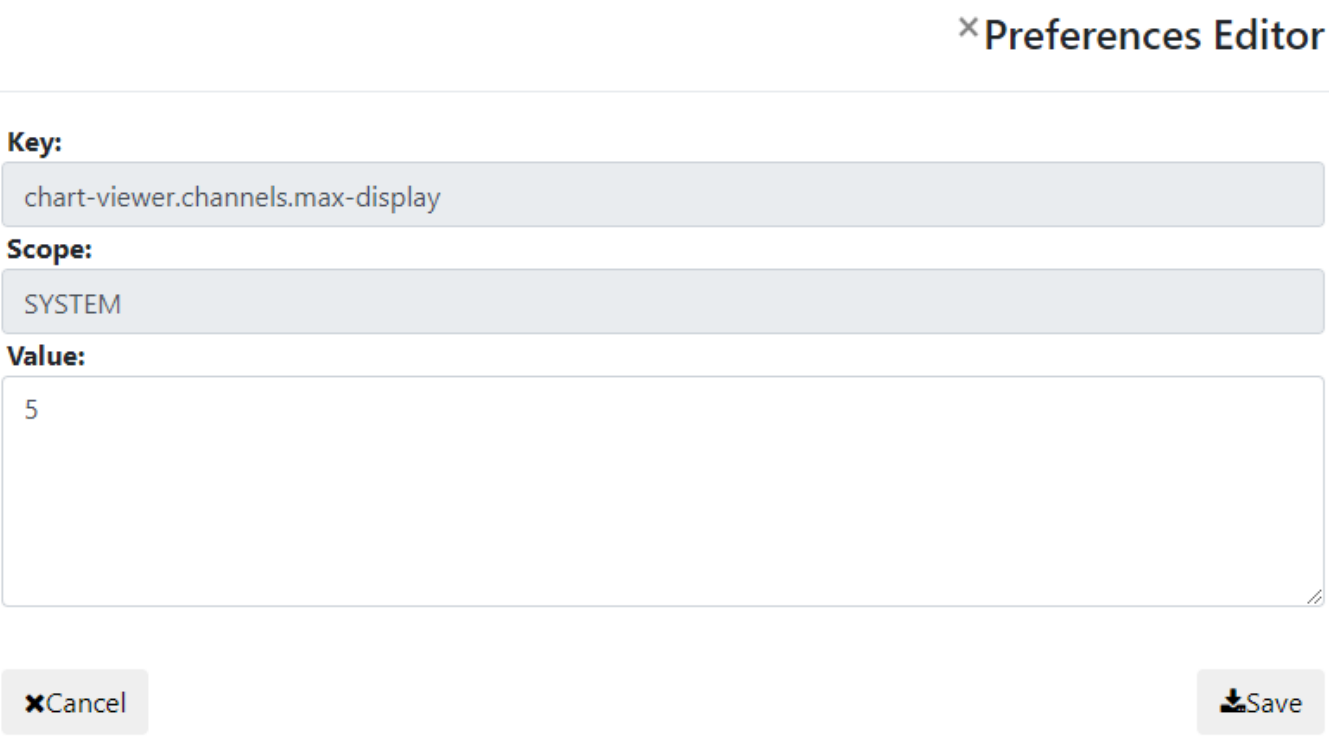
- **Key:** search.button.request-as-json
- **Value:** true

11.1.3. Quick Viewer Preferences

In the Quick Viewer the amount of loaded channels will be restricted to **10** channels, if a channel group or measurement was selected. In case this default setting does not fit your needs, you can change it with:

- **Key:** chart-viewer.channels.max-display
- **Value:** we recommend any cardinal number greater than 0 and smaller than 51, e.g.: **5**

Following the above example the value would be:



11.2. I18N Configuration

I18N support for web frontend, languages available: English, German.

For adding a new language see:
org.eclipse.mdm/nucleus/webclient/src/main/webapp/README_I18N.md

Files with translations:
org.eclipse.mdm/nucleus/webclient/src/main/webapp/src/assets/i18n/*.json

11.3. Icons Configuration

All icons for the web frontend are taken from the FAMFAMFAM Silk Icons library, version 1.3 (http://www.famfamfam.com/lab/icons/silk/). This library is licenced under the Creative Commons Attribution 3.0 License (https://creativecommons.org/licenses/by/3.0/). This library was approved by the Eclipse Foundation, see CQ 17759.

Note: A lot of users of other mdm applications are used to the icons from this document https://www.highqsoft.com/download/ao_base.htm These icons are not open source, so we do not use them in our application.

The mapping is defined in: org.eclipse.mdm/nucleus/webclient/src/main/webapp/src/styles.css

The mapping from the ao elements to FAMFAMFAM icons:

ASAM ODS 5.3.0 Base Element Definitions	Icon	FamFamFam Silk Icons, V1.3
https://www.highqsoft.com/download/ao_base.htm#AoFile		page_white_stack

ASAM ODS 5.3.0 Base Element Definitions	Icon	FamFamFam Silk Icons, V1.3
https://www.highqsoft.com/download/ao_base.htm#AoEnvironment		database
folder		folder
https://www.highqsoft.com/download/ao_base.htm#AoMeasurementQuantity		chart_curve_go
https://www.highqsoft.com/download/ao_base.htm#AoMeasurement		chart_curve
https://www.highqsoft.com/download/ao_base.htm#AoParameter		shape_square
https://www.highqsoft.com/download/ao_base.htm#AoParameterSet		shape_move_forwards
project		house
structure level		paste_plain
https://www.highqsoft.com/download/ao_base.htm#AoSubmatrix		calendar
https://www.highqsoft.com/download/ao_base.htm#AoTestEquipmentPart		monitor
https://www.highqsoft.com/download/ao_base.htm#AoTestEquipment		computer
https://www.highqsoft.com/download/ao_base.htm#AoTest		brick_add
https://www.highqsoft.com/download/ao_base.htm#AoTestSequencePart		shape_move_front
https://www.highqsoft.com/download/ao_base.htm#AoTestSequence		page_white_stack
test step		brick
https://www.highqsoft.com/download/ao_base.htm#AoUnitUnderTestPart		cog
https://www.highqsoft.com/download/ao_base.htm#AoUnitUnderTest		cog_go

12. Integration of SAML authentication

It is possible to configure a SAML authentication into openMDM.

Because Glassfish has no embedded SAML integration a SAML servlet adapter from KeyCloak is used.

You can find the whole documentation to adapter and configuration under:
https://www.keycloak.org/docs/latest/securing_apps/#java-adapters-2

All you have to do is to set up some SAML specific configuration.

12.1. Configuration Files

Configuration file - build.gradle:
`org.eclipse.mdm\nucleus\application\build.gradle`

Add the following:
compile 'org.apache.httpcomponents:httpclient:4.5.13'
compile 'org.keycloak:keycloak-saml-servlet-filter-adapter:15.0.2'

Configuration file - web.xml:
`org.eclipse.mdm\nucleus\application\src\main\webconfig\web.xml`

Add MDMSamlFilter (f.e.):

```
<filter>
    <filter-name>MDM Saml Filter</filter-name>
    <filter-class>org.eclipse.mdm.application.MDMSamlFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>MDM Saml Filter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

To have only SAML as authentication servlet, remove or comment out the sections "securityConstraints" and "loginConfig".

No-SSO To keep the form login of MDM it is neccessary to protect a No-SSO-resource. A servlet filter for this is already prepared in org.eclipse.mdm.application.NoSSOServletFilter. Furthermore the resource must be added to "passThroughPrefixes" of MDMRequestFilter.

Full example of web.xml with KeyCloakFilter and No-SSO config:

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="3.0">
<security-constraint>
    <web-resource-collection>
        <web-resource-name>MDM WEB (protected)</web-resource-name>
```

```
        <url-pattern>/nosso/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>Admin</role-name>
        <role-name>DescriptiveDataAuthor</role-name>
        <role-name>Guest</role-name>
    </auth-constraint>
</security-constraint>
<security-constraint>
    <web-resource-collection>
        <web-resource-name>MDM WEB (unprotected)</web-resource-name>
        <url-pattern>/login.css</url-pattern>
        <url-pattern>/favicon.ico</url-pattern>
    </web-resource-collection>
</security-constraint>
<filter>
       <filter-name>Keycloak Filter</filter-name>
       <filter-class>org.eclipse.mdm.application.MDMSamlFilter</filter-class>
   </filter>
   <filter-mapping>
       <filter-name>Keycloak Filter</filter-name>
       <url-pattern>/*</url-pattern>
   </filter-mapping>
<filter>
    <filter-name>MDMRequestFilter</filter-name>
    <filter-class>org.eclipse.mdm.application.MDMRequestFilter
    </filter-class>
    <init-param>
        <param-name>passThroughPrefixes</param-name>
        <param-value>/mdm,/nosso</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>MDMRequestFilter</filter-name>
    <url-pattern>/*</url-pattern>
    <dispatcher>REQUEST</dispatcher>
</filter-mapping>
<!-- If using SessionContext#isCallerInRole(), roles also have to be declared at class
level with @DeclareRoles(value = { "Admin", "DescriptiveDataAuthor", "Guest" }) -->
<security-role>
    <role-name>Admin</role-name>
</security-role>
<security-role>
    <role-name>DescriptiveDataAuthor</role-name>
</security-role>
<security-role>
    <role-name>Guest</role-name>
</security-role>
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name>MDMRealm</realm-name>
```



```
    <form-login-config>
        <form-login-page>/login.jsp</form-login-page>
        <form-error-page>/error.jsp</form-error-page>
    </form-login-config>
</login-config>
<session-config>
    <session-timeout>20</session-timeout>
</session-config>
</web-app>
```

Configuration file - keycloak-saml.xml:
`org.eclipse.mdm\nucleus\application\src\main\webconfig\keycloak-saml.xml`

Keycloak need an own xml file for configuration of service and identity provider. This file must be deployed into the WEB-INF folder (see next part).
How to configure the keycloak-saml.xml can be found under:
https://www.keycloak.org/docs/latest/securing_apps/#java-adapters-2

Configuration file - role-mappings.properties:
`org.eclipse.mdm\nucleus\application\src\main\webconfig\role-mappings.properties`

In `keycloak-saml.xml` it can be defined which attributes in SAML-Response should be used for user roles. The value of this role in the response can now be mapped to a application role. This is what the `role-mappings.properties` file is for.

Map an incoming role from IdentityProvider to an application role, f.e.:

```
CN\=mdm,CN\=Users,DC\=example,DC\=company=Admin
```

In this example the IdentityProvider delivers the distinghuished name of the AD group the user belongs to. This name will be mapped to the Admin-role in MDM.

Configuration file - build.gradle:
`org.eclipse.mdm\build.gradle`

Add keycloak-xml to WEB-INF deployment:

```
webXml = file('application/src/main/webconfig/web.xml')
webInf  {from 'application/src/main/webconfig/glassfish-web.xml'}
webInf  {from 'application/src/main/webconfig/keycloak-saml.xml'}
```

Here it is also possible to link a different web.xml file, f.e. "web-saml.xml" that should be used as deployed web.xml.

Configuration file - global.properties:
`org.eclipse.mdm/nucleus/property/src/main/configuration/global.properties`

To enable No-SSO set the parameter:

application.logoutRedirect=https://<host>:<port>/org.eclipse.mdm.nucleus/logout.jsp?nosso

13. OSS Governance

Eclipse mdmbl is an open source project hosted by the Eclipse Foundation licensed under the EPL-2.0. The legal obligations of the content must be observed in all forms of which the content is available.

The source of truth is always the Eclipse Foundation Project Handbook.

13.1. Legal Documentation

The following files must be part of your repository root folder:

- LICENSE
- NOTICE.md
- DEPENDENCIES
- SECURITY.md
- CONTRIBUTING.md
- CODE_OF_CONDUCT.md

See EF - Legal Documentation Generator

13.1.1. LICENSE FILE

Project license: EPL-2.0

- SPDX-License-Identifier: EPL-2.0
- License Text

See the Handbook#legaldoc-license.

For documentation the Creative Commons Attribution 4.0 International (CC BY 4.0) is recommended.

13.1.2. NOTICE FILE

- Add the link to your repository
- Add the link(s) to your SBOM, e.g. the DEPENDENCY file (one or more)
- Add information for third party content checks, if not covered by the Dash Tool (e.g. IP checks for icons, fonts, ...)

13.1.3. DEPENDENCY FILE

Note: Third-party dependencies need to be checked regularly to reflect your code changes. The DEPENDENCY file must be updated accordingly. This is recommended for every contribution (e.g. PR) whenever possible.

Use the [Eclipse Dash License Tool](https://www.eclipse.org/projects/handbook/#ip-license-tool)

If different technologies / package managers (e.g. npm and maven) are used you are free to have several dependency files. Use the naming convention DEPENDENCY_XYZ, e.g. DEPENDENCY_FRONTEND and DEPENDENCY_BACKEND.

13.1.4. SECURITY FILE

- The security file should at least contain information, where/how to report a vulneriability issue
- Add this template

See the Handbook#vulnerability.

13.1.5. CONTRIBUTOR GUIDE

See the Handbook#legaldoc-contributor

13.1.6. CODE OF CONDUCT

See the CODE OF CONDUCT and here in md format.

13.2. Copyright and Licence header

See the Handbook#ip-copyright-headers

Example (Java):

```
/*******************************************************************************
* Copyright (c) 20xx,20yy Contributors to the Eclipse Foundation
*
* See the NOTICE file(s) distributed with this work for additional
* information regarding copyright ownership.
*
* This program and the accompanying materials are made available under the
* terms of the Eclipse Public License v. 2.0 which is available at
* http://www.eclipse.org/legal/epl-2.0.
*
* SPDX-License-Identifier: EPL-2.0
*
*******************************************************************************/
```

Note:
Update the year in the copyright header at the start of each new year!

Example:
Copyright (c) 202x, <**new year**> Contributors to the Eclipse Foundation

13.3. IP checks for project content

Project content is roughly said, all content you created, e.g. code, scripts, documentation and configuration files. See the exact explanation for **project content** in the Handbook#ip-project-content.

Contributions by Eclipse contributors (signed ECA) must be received via an Eclipse Foundation channel (e.g. Gerrit, GitHub pull request, attachment on an issue).

A mdmbl committer **can** accept the contribution without further investigation if all of the following conditions are met:

- Was developed from scratch; written 100% by Eclipse contributors (no copied code, e.g. from StackOverflow or generated AI code e.g. from GitHub Copilot or ChatGPT)
- Was submitted under the terms of the project license (e.g. legal doc, copyright & license header)
- Contains no cryptography; and
- It contains fewer than 1,000 lines of code, configuration files, and other forms of source code.

If not, a project committer **must** engage with the IP Team to request an IP review for Code Contributions, choose the "vet-project" template.

13.4. IP checks for 3rd party content

The term third-party content refers to any content that is leveraged by the Eclipse project that is not produced and/or maintained as an asset of the project. This includes content from other Eclipse projects. See the complete explanation of third-party content.

All third-party content has to be checked and approved by the Eclipse Foundation IP Team. There are two ways:

- Creating an IP issue manually (e.g. fonts, images, ...)
- Using the Eclipse Dash License Tool to creat IP issues in an automated way (libraries)

All third party content used has to be documented in the NOTICE file or in the DEPENDENCY file.

Note: Only project committers can open IP issues, manually or via the Dash Tool!

13.4.1. Checking libraries using the Eclipse Dash License Tool

You can request the status of your used libraries via the Dash Licence Tool, see also the handbook.

Steps:

1. Download the lastest version: README ⇒ Get It ⇒ Download Link
2. For every repository / technology:
    - Create the list of transitive dependencies, see the README
    - Run the Dash Tool with the parameters "-project automotive.mdmbl" and "-summary

DEPENDENCIES" (or use the maven plugin)

3. Check for libraries with status "rejected" in the DEPENDENCIES file, if present, the dependency cannot be used and has to be removed.
4. Check for libraries with status "restricted", if so request checks by automatically creating issues via the Dash Tool:
   - include the -review option;
   - pass the API token via the -token option; and
   - pass the project id via the -project option.
5. Track the issues created by the Dash Tool
6. Provide support if an issue is labeled with "Help wanted"
7. Add the summary as DEPENDENCY file to the according repository (root level)

Important Notes:

- Get your API Token (see README of the Dash Tool), note that only committers can get an API Token
- Do **NOT** share your API Token!
- DO **NOT** integrate into automatic builds the "Automatic IP Team Review Requests" (creation of ip tickets: " -review option") via your **personal** API Token
- To integrate in your automatic builds: request a dash-bot via the EF Helpdesk.

13.4.2. Check the mdmbl backend with the Dash Tool

Generate / update the DEPENDENCIES_backend file.

1. Change into your local 'org.eclipse.mdm' repository, for each module get the dependency list via the gradle tooling and concatinate the outputs:

```
./gradlew nucleus:dependencies | grep -Poh "(?<=\s)[\w.-]+:[\w.-]+:[^:\s]+" | grep
-v "^org.eclipse" | sort | uniq > depend_backend_all.txt
./gradlew api:csvadapter:dependencies | grep -Poh "(?<=\s)[\w.-]+:[\w.-]+:[^:\s]+"
| grep -v "^org.eclipse" | sort | uniq >> depend_backend_all.txt
./gradlew api:atfxadapter:dependencies | grep -Poh "(?<=\s)[\w.-]+:[\w.-]+:[^:\s]+"
| grep -v "^org.eclipse" | sort | uniq >> depend_backend_all.txt
./gradlew api:base:dependencies | grep -Poh "(?<=\s)[\w.-]+:[\w.-]+:[^:\s]+" | grep
-v "^org.eclipse" | sort | uniq >> depend_backend_all.txt
./gradlew api:odsadapter:dependencies | grep -Poh "(?<=\s)[\w.-]+:[\w.-]+:[^:\s]+"
| grep -v "^org.eclipse" | sort | uniq >> depend_backend_all.txt
```

2. Sort the output and remove duplicates:

```
sort depend_backend_all.txt | uniq > depend_backend.txt
```

3. Run the Dash Tool:

```
java -jar dash.jar depend_backend.txt -project automotive.mdmbl -summary
DEPENDENCIES_backend.csv
```

4. Or run the Dash Tool for automated ticket creation :

```
java -jar dash.jar depend_backend.txt -review -token <your_token> -project
automotive.mdmbl
```

5. Check the summary file / tickets as described above, update the file in your repo.

13.4.3. Check the mdmbl frontend with the Dash Tool

Generate / update the DEPENDENCIES_frontend file.

```
java -jar dash.jar package-lock.json -project automotive.mdmbl -summary
DEPENDENCIES_frontend.csv
```

```
java -jar dash.jar package-lock.json -review -token <your_token> -project
automotive.mdmbl
```

13.4.4. Checking other content (fonts, images, ...)

Other 3rd party content, e.g. fonts, icons, images has also to be approved by the Eclipse Foundation IP Team. IP review, choose the "vet-third-party" template.

If the content is approved add the information to the NOTICE file. See also the NOTICE file for examples.

13.5. Legal information for distributions

The distribution form of software artifacts (often in a compiled form) generated from a project's source code repositories must also include legal information. The source of truth is always the https://www.eclipse.org/projects/handbook/

License, notice and (if existing) DEPENDENCIES files, must be included in the root of every distribution artifact (e.g. JAR file). In the most general case, these files will appear in the root of distribution unit, but the exact location varies by technology type.

For content delivered as Java archive (JAR) files, for example, the legal files should be placed in the META-INF directory.

When the distribution is an individual file (e.g. JavaScript), the file must contain a header with copyright and license information, and the best effort must be undertaken to associate the file with

the notices (e.g. a link to the source repository in the header).

Project teams should consult with their Project Management Committee (PMC) for input and advice regarding alternate names and locations for legal documentation.

(Text from the Eclipse Foundation Project Handbook)

13.6. Legal notice for end user content

Add an About Dialog (or similar) with the following informations:

- Copyright statements
- License
- Trademarks owned by the Eclipse Foundation on behalf of the project
- The source code repositories
- Used 3rd party libraries
- When applicable — a cryptography notice

The information can be provided in a static way or you can link directly to the source repository as they already contain your legal information.

Important: The links to your repostitory have to reference the tagged version (*) / commit id the component delivery has been created / build from!

Example:

```
[Your product name](URL to the YOUR repository)
* License: EPL-2.0
* Licence Path: <URL to the License in your repository (*)>
* NOTICE: <URL to the NOTICE file in your repository (*)>
* Source URL: <URL to the your repository (*)>
* Commit ID: <the commit id the component delivery has been created / build from>
```