

User Guide for the openMDM® Application

Eclipse mdmbl project

Version 5.3.0, 2025-11-12: {build-revision}

Table of Contents

1. Introduction	2
1.1. General	2
1.2. Features	2
2. The openMDM Web Client	4
2.1. Structure	4
2.2. Roles	5
3. General Functions and Common Controls	6
3.1. Login	6
3.2. Logout	6
3.3. Timezone	6
3.4. Datasources	6
3.5. Language	7
3.6. About	7
3.7. Control elements	7
3.8. Buttons	7
4. Navigation	10
4.1. Overview Data Structure	10
4.2. Navigator	11
5. Search	13
5.1. Defining the scope of a search via Criteria selection	13
5.2. Full Text Search	14
5.3. Advanced Search (Attribute Based Search)	14
5.3.1. Define an attribute based search	14
5.3.2. Use, create, update or delete an existing filter	16
5.3.3. Do the search	16
5.4. Search Result	17
6. Shopping Basket	19
6.1. Collect data in the Shopping Basket	19
6.2. Exporting data from the Shopping Basket (ATFX)	19
7. View editor	20
8. View data (Detail Viewer)	22
8.1. Parametersets	23
8.1.1. Create parameter set	23
8.1.2. Create parameters	23
9. Visualize data	25
9.1. Quick Viewer	25
9.2. X/Y Chart Viewer	25
9.2.1. Component bar	26

Chart	26
Table	26
9.2.2. Selection panel	28
Channel Group	28
Y-Channel	28
X-Channel	28
9.2.3. Data display	28
9.2.4. Option panel	28
10. Files	29
10.1. Attach external files	29
10.2. Shopping basket	29
11. User Preferences	30
11.1. Manage Unit Lists	30
11.1.1. Handling of Unit List sets	30
11.1.2. Channel Name	30
11.1.3. Writing a Channel Name Regular Expression pattern	31
Simple Patterns	31
Special Characters	31
Anchors	31
Quantifiers	32
Character Classes	33
External references	35
11.2. Manage Channel Lists	35
11.2.1. Handling of Channel List sets	36
11.2.2. Channel Name	36
11.3. Manage Formulas	36
11.3.1. Channel Name	37
11.3.2. Formula	37
12. Administrator Functions	38
12.1. Configure preferences	38
12.2. Add a Node Provider	38
12.3. Configure external systems	39
12.4. Validation of context attributes	40
12.5. Visibility of context attributes ordered values	40
12.6. Notification messages	41
12.7. Functional Rights	41
12.7.1. General Information	41
12.7.2. SpecialistAdmin	41
12.7.3. The Preference	41
Condition Syntax	42
12.7.4. Administration component	44

12.8. Announcements	44
12.8.1. Administration	44
12.8.2. User view	45
12.9. Context Bulk Editor	45
12.9.1. General Information	45
12.9.2. Configuration	46
13. Catalogs and Templates	47
13.1. Catalog Components	47
13.2. Templates	48
13.3. Template Versioning	49
13.3.1. REST API	49
13.3.2. Use cases	49
New version	49
Integrate new catalog attributes	49
Integrate new versions of related templates	50
14. Special HTTP API methods	51
14.1. Change TestStep Template	51
14.1.1. HTTP API	51
14.1.2. Migration Rules	51
14.1.3. Example	51
15. Audit Logging	53
15.1. What is logged with the 'audit' marker?	53
15.2. Mapped Diagnostic Context	53
15.3. Elastic Common Schema	53
15.4. Utilization in MDM5 (for contributors)	53
15.4.1. Basic Concept	53
15.4.2. Setting up the MDC	54
15.4.3. Marking the log	54
15.4.4. AuditLogger	54
15.4.5. Configuration (for administrators)	55
Logback	55

Document History

Author	Date	Version	Affects SW Version	Description
Peak Solution GmbH	21.04.2021	0.52	5.2.0Mx	Initial version
Angelika Wittek, Matthias Koller	17.10.2025	0.6	5.3.0Mx	General updates

This document is published under the Eclipse Public License 2.0:
<https://www.eclipse.org/legal/epl-2.0/>
Copyright(c) 2020-2025, Contributors to the Eclipse Foundation.

1. Introduction

1.1. General

The openMDM Application is an open source software platform for measured data management systems. In compliance with the ASAM ODS standard, this platform enables an ecosystem currently consisting of automotive companies to build in-house application as well as application vendors building commercial and open source solutions tools and systems.

The openMDM Application can serve many ASAM ODS compliant servers in parallel for data ingest, data exploration, test management, user and access management and more. Security mechanisms at data and functional level protect the stored information against unauthorized access.

In summary, it implements an industry standard platform for tools and products for ODS compliant products and tool chains.



The development of the platform and concepts is driven by the [openMDM Eclipse Working Group](#). The source code, documentaton and concepts are part of the open source [Eclipse mdmbl project](#) published under the business friendly Eclipse Public License 2.0 ([EPL-2.0](#)).

1.2. Features

The openMDM Application can be used to manage test and measurement data for a broad variety of domains. The openMDM web client offers a web based graphical user interface, that offers the following features:

- [navigate through data structures](#)
- [search and retrieve of test and measurement data](#)
- [inspect data \(*Detail Viewer*\)](#)
- [collect data for further processing \(*Shopping Basket*\)](#)
- [visualize data \(*Chart Viewer*\)](#)
- [upload / download files, update, preview or delete existing files](#)

With extended administration rights, a user can:

- [configure preferences](#)
- [configure external systems](#)

Refer to [this chapter](#).

NOTE

This document includes screenshots of the [openMDM Demo Application](#) which is provided by the Eclipse openMDM Working Group. Please be aware that different domain specific openMDM applications have different data models and so different meta and measurement data.

NOTE

The test and measurement data are usually imported by server processes and stored in an standardized ASAM ODS data source. The way of storing the data is not part of this user manual.

2. The openMDM Web Client

2.1. Structure

The openMDM web client is the graphical user interface for the openMDM application. If you start the client, you get the following view:

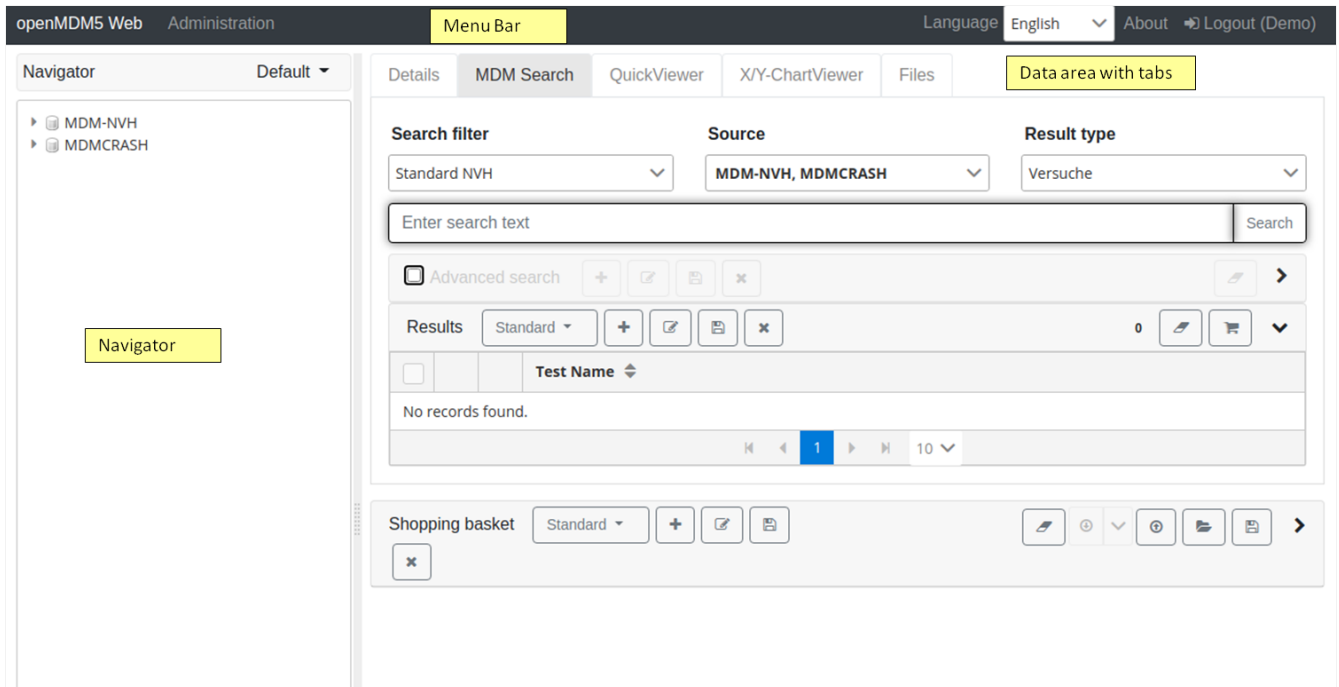


Figure 1. openMDM web client

Elements:

- **Menu bar:** Menus for language selection, About and Login / Logout, see [this section](#). There is also the Administration menu, see [this section](#).
- **Navigator:** Browse the openMDM data structures (e.g. project, pool, test, teststep, etc.), see [this section](#).
- **Data Area:**
In this area all data related functions can be found. These functions are divided into five tabs:
 - a. **Details:** The Metadata for a selected node in the Navigator are shown, see [this section](#).
 - b. **MDM Search:** Different kinds of search are available like full text search, advanced search (attribute based), defining search filters and a result view, see [this section](#).
 - c. **QuickViewer:** A simple X/Y ChartViewer to visualize the data of a measurement, see [this section](#).
 - d. **X/Y ChartViewer:** A more sophisticated viewer that can represent the data of a measurement in different views, see [this section](#).
 - e. **Files:** Upload new files and/or download, update, preview or delete existing files, see [this section](#).

NOTE The openMDM5 main elements are written in *italic font* in this document.

2.2. Roles

For the web client three different roles for users are available:

- *DescriptiveDataAuthor*: users in this role can view, edit and update data. We denote users in this role just as *user*.
- *Admin*: additional rights for configuring / customize the web client for all users. See [here](#).
- *Guest*: restricted to read only.

NOTE

When we talk about admin / administration in this document, we mean users in the *Admin* role. There are further configuration options, that can be only done on server side, we refer to this as *system administration*.

3. General Functions and Common Controls

In this section general functions from the *Menu Bar* are described, also common control elements and buttons that are used.

3.1. Login

A user name and password are required to log in to the openMDM Application. The user name and password is the same as for the related ASAM ODS user. Please ask your administrator when you aren't able to connect to the system.

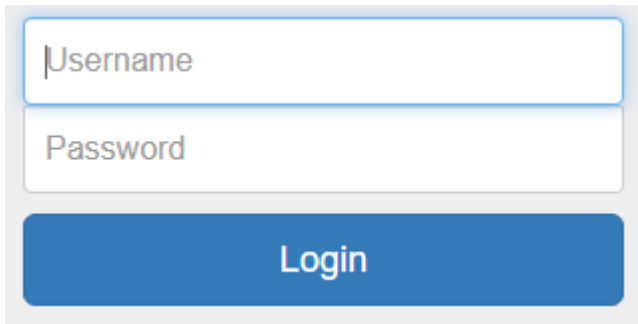


Figure 2. openMDM5 login

After a successful login, the start page of the openMDM5 Web Client appears.

NOTE

It is possible to include single-sign-on functionality to the openMDM5 application. Please ask your administrator for this feature.

3.2. Logout

Logging out will end all of your ASAM ODS user sessions / connections.

3.3. Timezone

Select a timezone in which the data will be shown. Works in detail view and X/Y Chart Viewer. There are all timezones from the whole world available. When searching, the client timezone is automatically converted to the respective server timezone.

Additional options are: - A Date Type defines if dates are printed absolute, relative or both. * Absolute: e.g. 05.07.2023, 17:21:23 * Relative: e.g. 3 years ago * Both: e.g. 3 years ago at 07/05/2023, 05:21:23 PM - If ISO Date is selected, the dates are formatted according to ISO 8601. Otherwise dates are formatted depending on the selected locale. This has no effect if **Relative** is selected.

3.4. Datasources

Select one or more datasources to work with. By clicking the **Default datasources** button. you can specify to which datasources the client should connect to after login.

3.5. Language

Select a language for text and labels. Currently available languages are English and German (Deutsch). New languages can be defined and added by the system administration.

3.6. About

All necessary legal information for the open source software is listed, this includes: Trademarks, Copyright, Declared Project Licenses, Source Code location, Third-party Content and Cryptography.

3.7. Control elements



Hide element

To hide the associated element.



Open element

To open / show the associated element.

3.8. Buttons



Refresh

To refresh the selected element.



Copy link

To copy the link of the selected element.



Shopping Basket

To add the selection to the *Shopping Basket*.



Edit

To edit the associated attributes.



Save

To save the associated changed/edited component. It can be a view or a filter or the *Shopping Basket*.



Delete

To delete the selected or associated component.



Clear

To reset changed/edited components (search conditions), or clear components (search results, *Shopping Basket*).



Download and download with options

To download the selected component. Some components have many options for download.



Upload

To upload the selected component (file, *Shopping Basket*).



Open

To open the associated component.



Link and unlink

To link or unlink the selected component to the associated component.

4. Navigation

4.1. Overview Data Structure

The test and measurement data are stored in a hierarchical structure and visualized in the *Navigator*. The top level element(s) in the tree show the "environments", the data belong to. Starting from the environment you can drill down the storage hierarchy to the top level element "Project" down to a "Test" element and its "Measurement" elements.

In this section the different levels of the test data storage and navigation will be described. The standard navigation panel uses and provides the structure of the data as stored in the ASAM ODS data source. Alternatively you can choose different navigation structures by using the *NodeProvider*.



Environment

The top level element(s) in the tree show the data source(s), the data belong to.



Project

The top level of the data is the "Project" level. It's the highest grouping criteria for test data.



Pool

The second level is the "Pool" level. It is used to categorized and group the tests in a project. It is sometimes referred as "subproject" level.



Test

The third level is the "Test" level. Beside meta data it also refers to business objects defining "Unit under test", "Test sequence", "Test equipment" and "Sensors".



Teststep

The fourth level is the "Teststep" level. Usually a teststep is defined for a limited number of measurements. A teststep belongs to a test and it includes beside teststep data the measurement data.



Measurements

The lowest level of the test data is the "Measurement" level. It includes the measurement data to be used in QuickViewer and X/Y Chart Viewer. A measurement belongs to a teststep.

4.2. Navigator

The *Navigator* offers the possibility to browse the openMDM data sets. It is a tree, which shows the openMDM data structure. The view of the data structure in the *Navigator* is configurable and determined by the chosen Node Provider configuration defined by the system administration. The openMDM Client provides a default configuration for the standard openMDM view. This shows the metadata from the connected ODS database in the following hierarchy:

- Project (e.g PMV 2PV or PMV Model P)
- Pool/Subproject (e.g Engine Noise Measurements or PBN Measurements)
- Test (e.g PBN_UNECE_R51_20190703160224)
- Teststep (e.g PBN_UNECE_R51_Left_Acc_70)
- Measurement (e.g Channel)

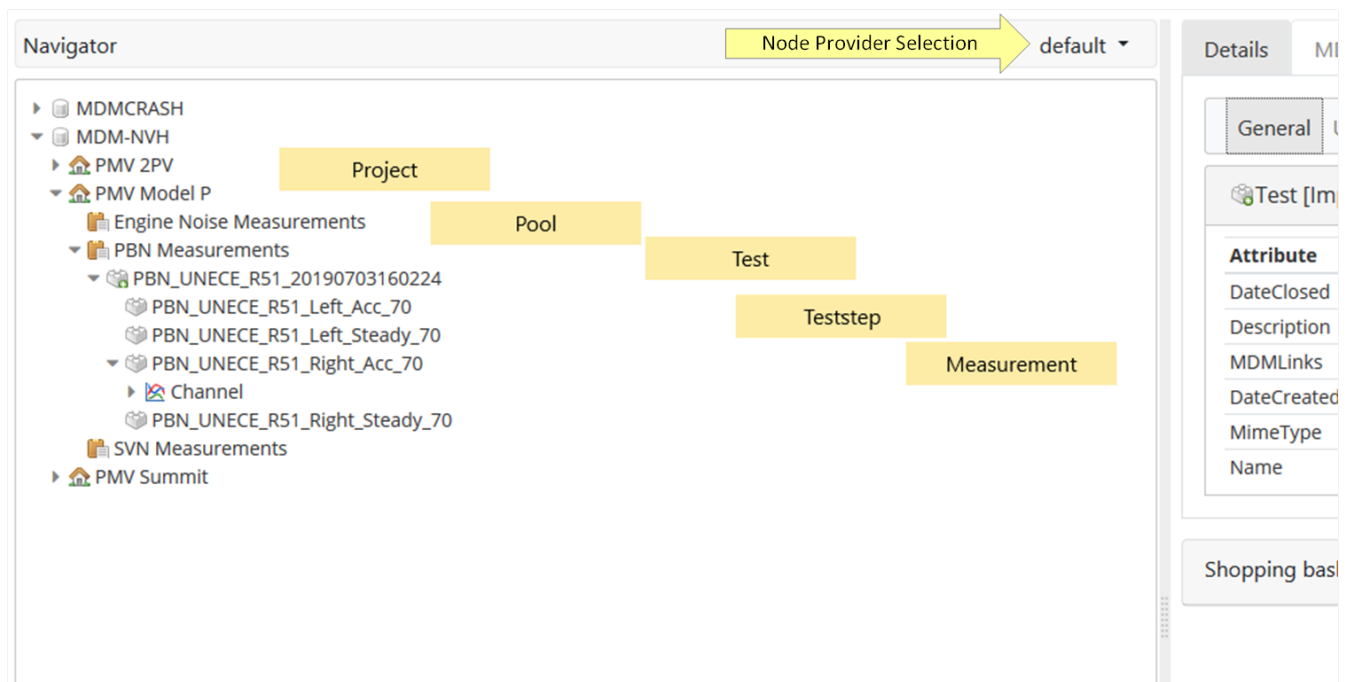


Figure 3. Hierarchical levels in the Navigator

In this screenshot the Node Provider "default" is selected.

With the right mouse button, you can invoke a context menu that offers you a choice of different functions depending on the current context.

- Refresh: Newly arrived information for the currently selected data object is shown in the navigation tree and in the selected view in the right pane of the openMDM web client.
- Refresh all: Regardless of the selected data object, all the newly arrived information is shown in the tree.
- Copy link: Copy the REST URL for the currently selected data object.
- Add to *Shopping Basket*: Storage of the currently selected data object (multi-select possible) in

the *Shopping Basket*.

- Create Project/Pool/Test/TestStep/Measurement.

NOTE

Currently there is no possibility to enter values for mandatory context attributes. Thus you cannot create TestSteps or Measurements with mandatory attributes.

- Delete Project/Pool/Test/TestStep/Measurement.

NOTE

Deleting is only available, if you have one of the roles: Admin, Specialist_Admin*

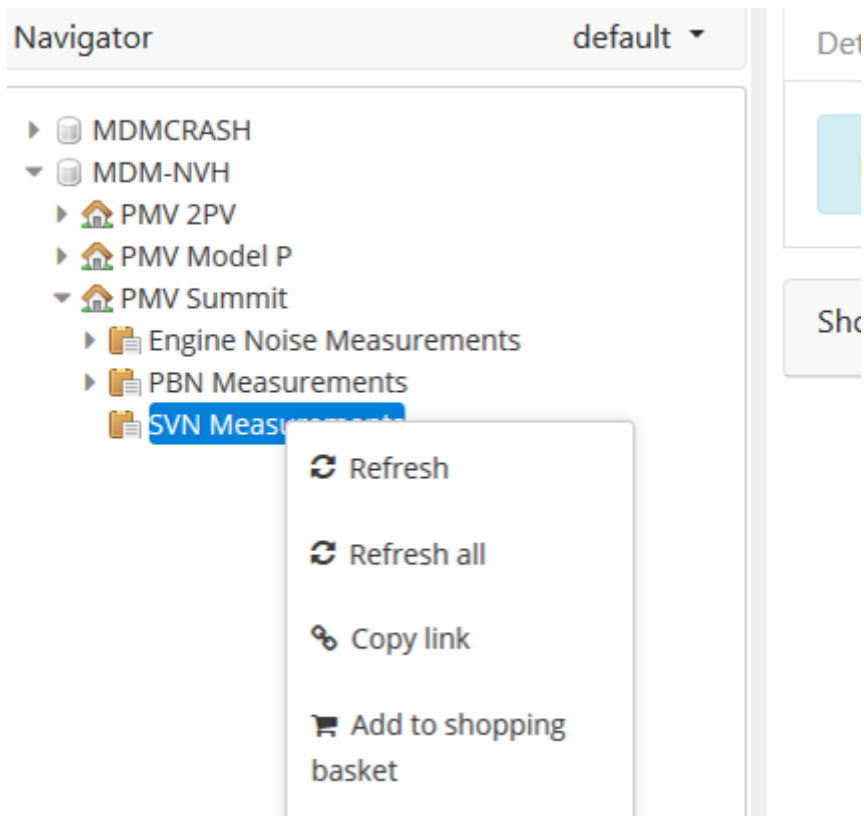


Figure 4. Context menu in the Navigator

5. Search

In the second tab in the right pane of the openMDM web client is the *MDM Search*. The *MDM Search* consists of three areas:

- Full Text Search: search for a (sub) string in the metadata. Can be combined with the advanced search.
- Advanced search: Search using multiple attributes. Named search filters can be created and updated for reuse or be deleted.
- Results: show search results in a result view.

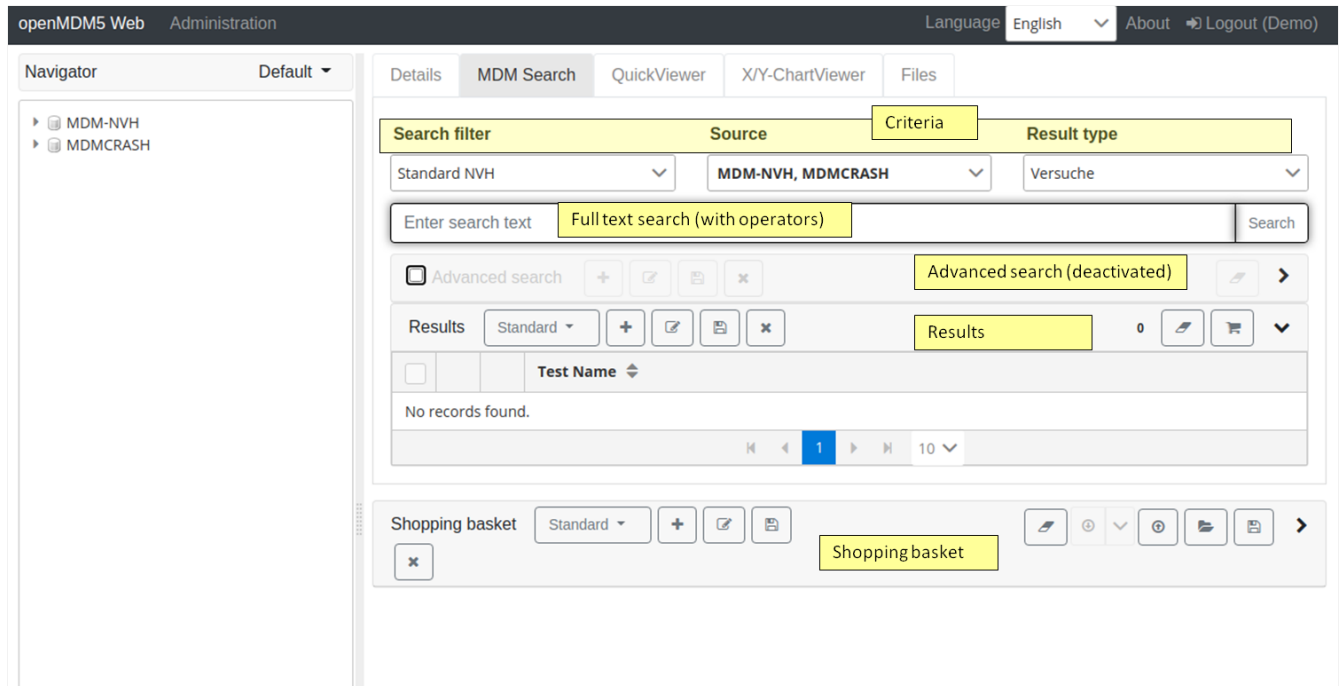


Figure 5. MDM Search

5.1. Defining the scope of a search via Criteria selection

With the criteria selection you can define the scope of your search by different options:

Search filter (for Advanced Search only): Select a named search filter. A search filter definition contains search attributes and can be reused. If a filter is selected then its search attributes are available in the Advanced Search area.

Source: You can select one or more data sources the search will be executed on. A multi-select of data sources is possible.

Result type: You can select the result type. The following result types are supported:

- Test
- Teststep
- Measurement result

- Submatrix
- Measurement quantity

NOTE

When choosing a Teststep or Measurement result as result type you should configure attributes of the Teststep or Measurement result for a better visualization and identification.

5.2. Full Text Search

The full text search (like Google text search) is a single text field. You can search for a (sub) string or text in the metadata. The criteria "Source" and "Result type" are defining the scope of the search.

You can define a more sophisticated search using operators. The available operators are described as tool tip text in the search text field.

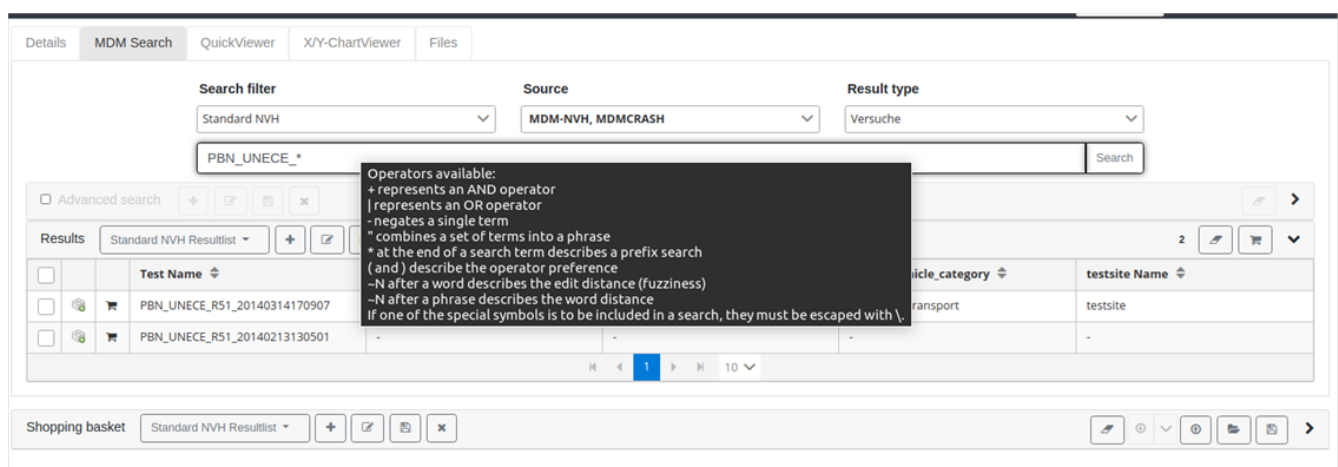


Figure 6. Search with * operator

The full text search can be combined with the advanced search, by activating the check box *Advanced Search*.

5.3. Advanced Search (Attribute Based Search)

For the Advanced Search activate the *Advanced Search* check box and expand the Advanced Search area.

IMPORTANT

The button "Search" on the right hand side of the Full Text Search starts all types to searches, even if the full text search is left empty.

5.3.1. Define an attribute based search



Create a new filter and select your attributes by opening the View Editor via this button.

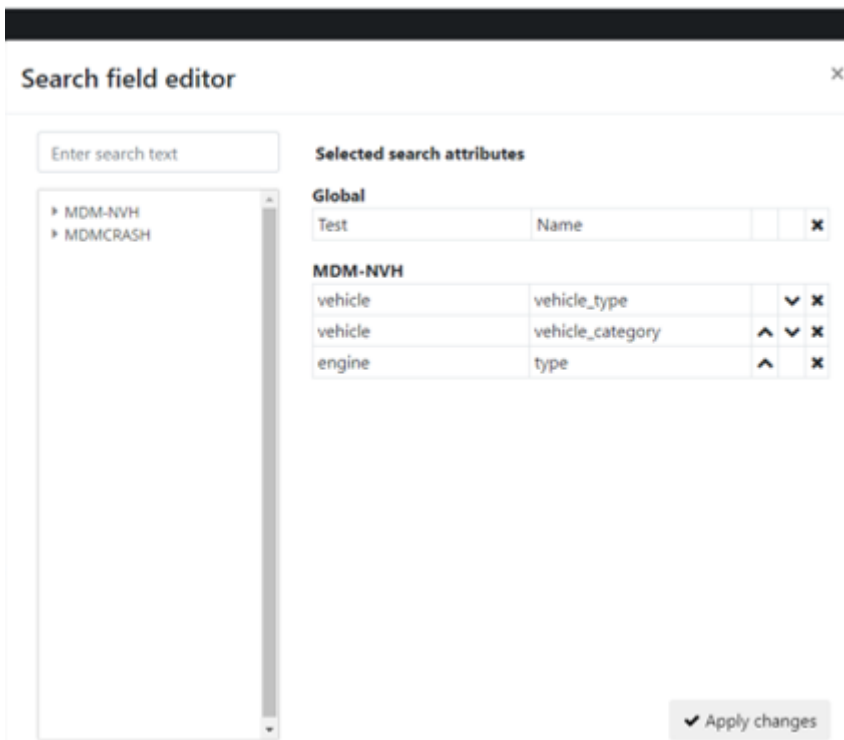


Figure 7. Select attributes in View Editor

In the Advanced Search you can see the selected attributes:

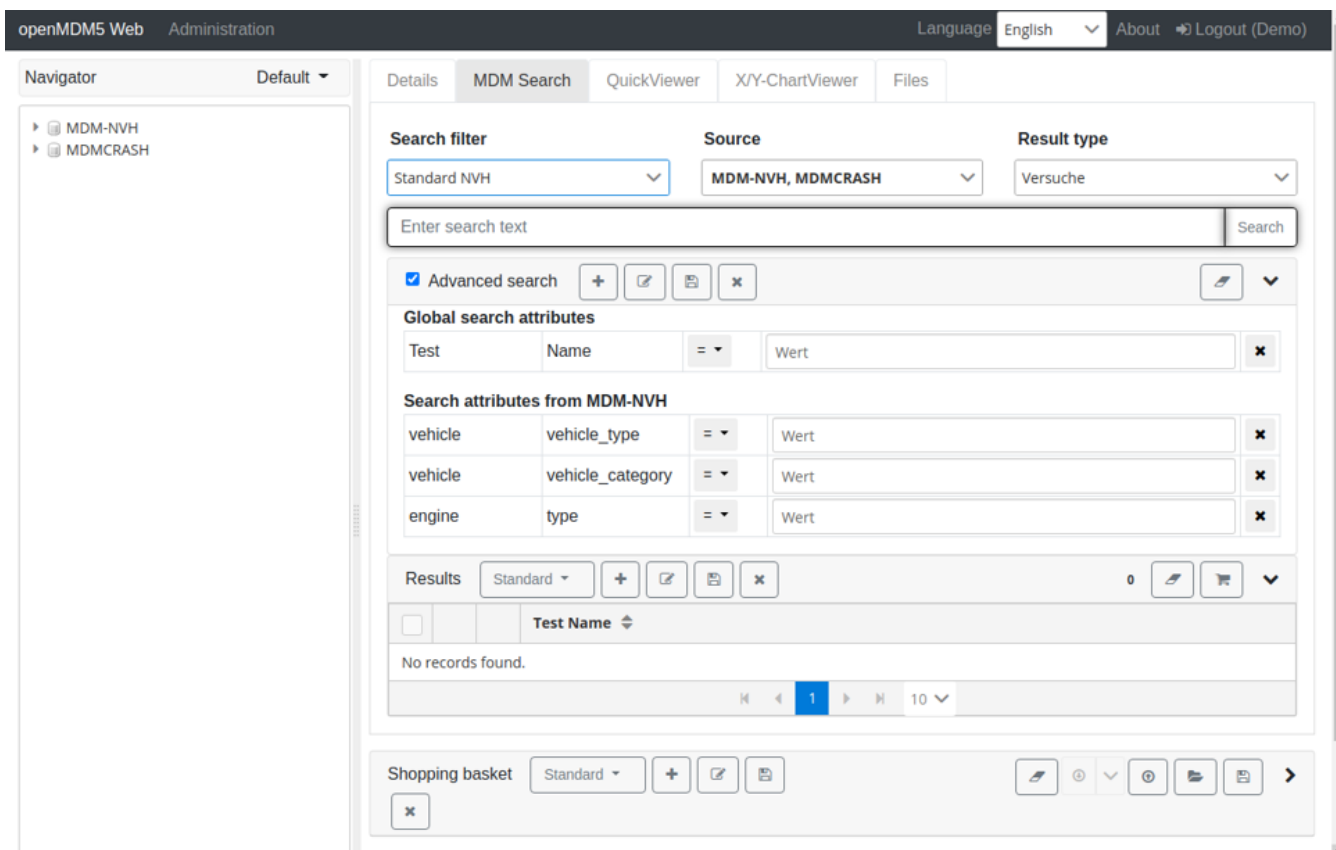


Figure 8. Attributes in the Advanced Search

If you select an attribute which is defined in all selected environments (see [Source](#)) then this attribute is a so called global search attribute. The search with a global search attribute is executed in all environments.

If you select an attribute which isn't defined in all selected environments then this attribute is an environment specific attribute and the search is executed only in the related environments.

The Advanced Search area contains the following elements (in each row):

- **Attributes:** These first two columns contain the names of the object type and the object's attribute.
- **Operators:** Operators for the attribute and their search criteria can be set in this column.
- **Value:** Search values are entered here.
- **Remove Button:** This button removes the selected row from the advanced search.

5.3.2. Use, create, update or delete an existing filter

- **Create a new filter:**
[Define attributes](#). For reuse you can define a name for the search filter and save it using the *Save search filter* button. You can assign a new name or override an existing filter by choosing the dedicated filter name.
- **Use an existing filter:**
Select a Search Filter in the *Search Filter* drop down box. Activate the Advanced Search check box.
- **Update an existing filter:**
Select a Search Filter in the *Search Filter* drop down box. Make your changes by opening the View Editor with the *Edit search filter* button. You can assign a new name or override an existing filter by choosing the dedicated filter name. Then use the *Save view* button.
- **Delete an existing filter:**
Select a Search Filter in the *Search Filter* drop down box. Use the *Delete search filter* button.

5.3.3. Do the search

For the advanced search you have to:

- define the attribute (object type and attribute name)
- define the operator (=, <, >, like)
- (optionally) fill the search text field
- define your data source(s)
- choose the result type

Use the button "Search" on the right hand side of the Full Text Search.

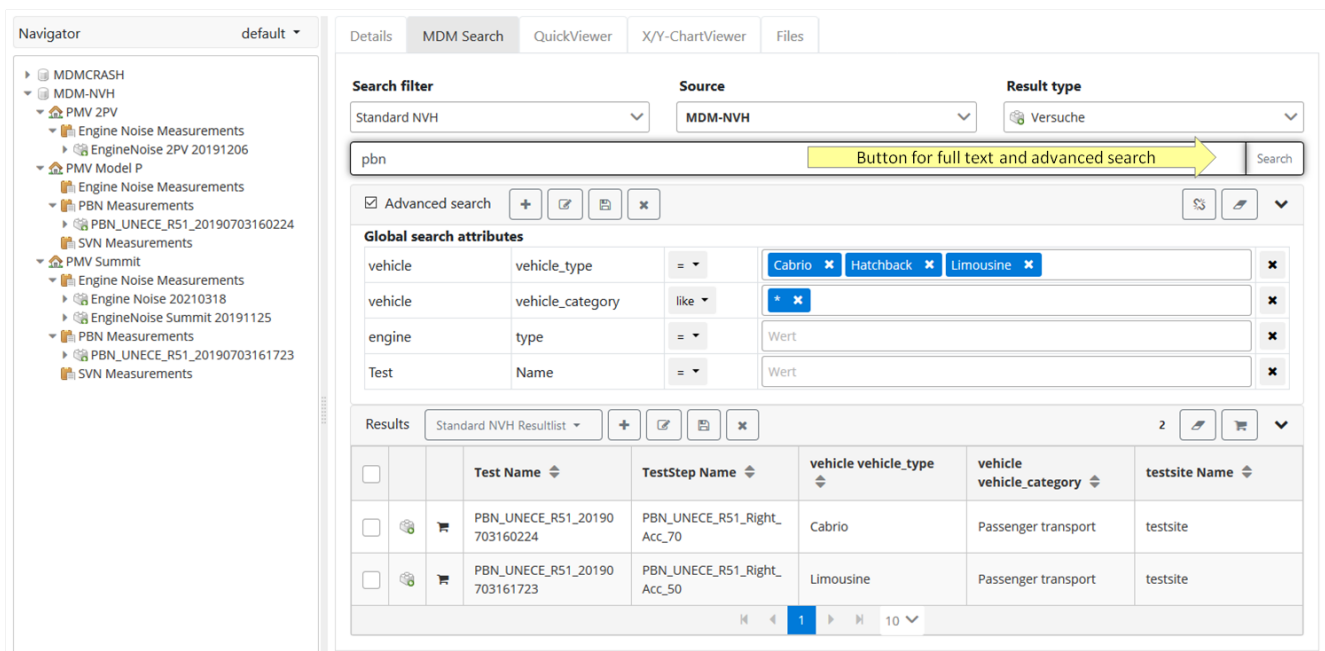


Figure 9. MDM Search

IMPORTANT

Don't forget to reset the filter criteria, if you want to execute another search.

5.4. Search Result

Using the buttons in the upper left area of the result area, you can select an existing result-view, create or edit, save or delete result-views. You create and edit results-views with the [View Editor](#).

Using the buttons in the upper right result area, you can clear the search results or add selected search results to the *Shopping Basket*.

openMDM5 Web
Administration

Language
English
About
Logout (Demo)

Navigator
Default

MDM-NVH
MDMCRASH

Details
MDM Search
QuickViewer
X/Y-ChartViewer
Files

Search filter
Source
Result type

Standard NVH
MDM-NVH
Versuche

pbn*
Search

Advanced search
+
-
x

Global search attributes

vehicle	vehicle_type	=	Cabrio x Limousine x hatchback x	x
vehicle	vehicle_category	like	* x	x
engine	type	=	Wert	x
Test	Name	=	Wert	x

Results
Standard NVH Resultlist
+
-
x

1

		Test Name	TestStep Name	vehicle vehicle_type	vehicle vehicle_category	testsite Name
☒		PBN_UNECE_R51_20140213130501	PBN_UNECE_R51_Left_Acc_50	Limousine	Passenger transport	testsite

1
10

Shopping basket
Standard
+
-
x

☐
☒

Test Name

PBN_UNECE_R51_20140213130501

1
10

Figure 10. Add result to Shopping Basket

Copyright(c) 2020-2025, Contributors to the Eclipse Foundation. Made available under the Eclipse Public License 2.0

6. Shopping Basket

Store data objects such as projects, pools, tests, teststeps, measurements etc. for further processing.

6.1. Collect data in the Shopping Basket

The *Shopping Basket* is located in the lower left area of the openMDM web client. It is a container for data objects. Any data objects can be selected in the *Navigator* (multiple selection is possible) and placed per context menu in the *Shopping Basket*, to use them as selection for further processing with other openMDM components.

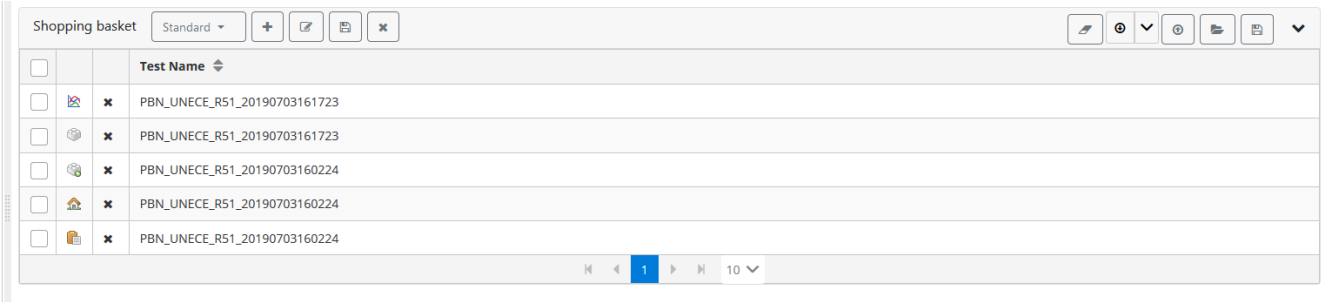


Figure 11. Storage of data in the Shopping Basket

Using the buttons in the upper left area of the area, you can create new views, edit existing views with [the View Editor](#), save edited views or delete existing views. The procedure is similar to that described [here](#).

6.2. Exporting data from the Shopping Basket (ATFX)

Using the buttons in the upper right area of the view, you can clear, download, upload, open or save data of the *Shopping Basket*.

Icons	Description
	You can download content of the selected objects in the shopping basket in form of an XML file. The download contains the link to the objects.
	You can download the data of the selected objects in the shopping basket using in the selected data format (e.g. ATFX).

7. View editor

Notiz BMW: Diese Funktionalität muss evtl. im Client vereinfacht werden (verständlich für einen Anfänger).

With this editor you can create new search filters, edit existing search filters, create new views for search results or *Shopping Basket* and edit existing views for search results or *Shopping Basket*.



Click on one of this buttons to start the View Editor dialog.

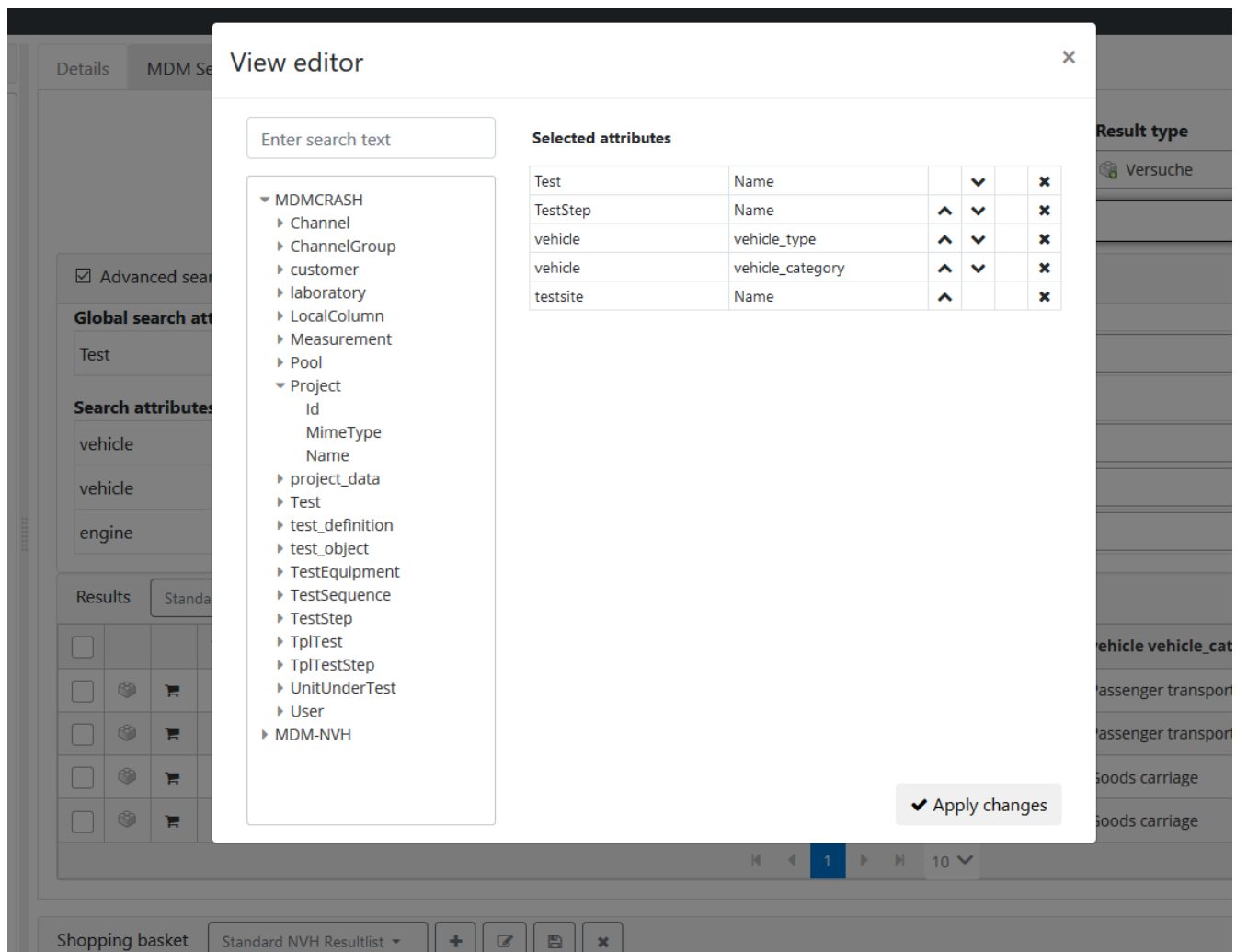


Figure 12. View Editor

You can search for elements via the search field:

Selected search attributes

vehicle

vehicle.reference_point

vehicle.maximum_authorized_mass

vehicle.vehicle_category

vehicle.mass_in_running_order

vehicle.length

vehicle.vehicle_type

vehicle.model

vehicle.id

vehicle.MimeType

vehicle.manufacturer

vehicle.Name

IVH

vehicle_type

✕

✓ Apply changes

Figure 13. View Editor search

The procedure is always similar, see as e.g. [here](#).

8. View data (Detail Viewer)

The first tab of the right pane of the openMDM web client is the *Details* which contains the metadata of the selected object. It responds to the selection within the *Navigator*. The component *Details* has 5 tabs.

- General
- Unit Under Test
- Test Sequence
- Test Equipment
- Sensors
- Parametersets

Notiz (BMW): bzgl. Bulletpoints:

Sollten wir in der GUI evtl. intuitiver gestalten (weil ein Anwender vielleicht nichts von UUT, TS, TE weiß)

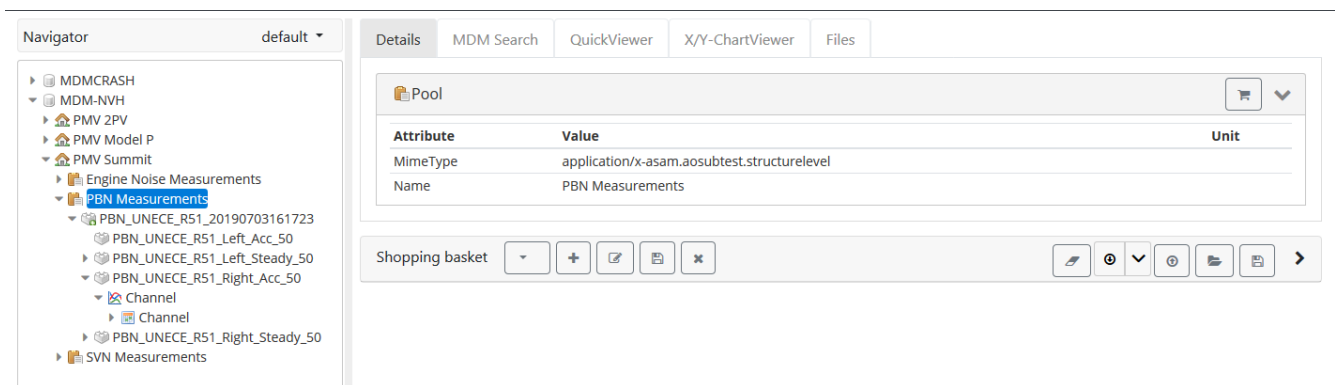


Figure 14. Metadata in the Details

The first tab is called General. It shows the attributes (= metadata) of the currently selected data object in the *Navigator*. If the selected data object is a Test or a Teststep or a Measurement then the additional tabs (see above) are visible.

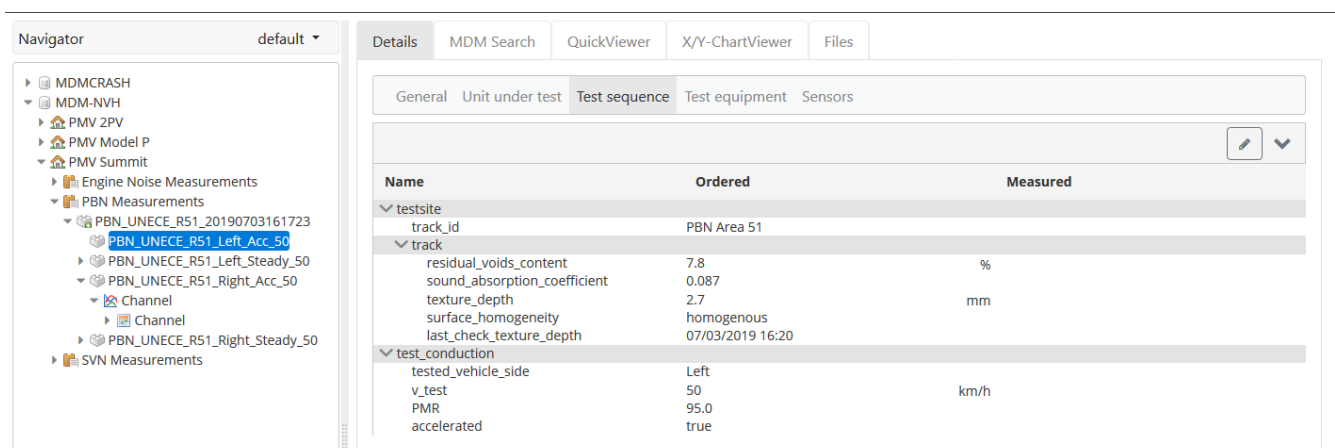


Figure 15. Test sequence tab in the Details

In these tabs you can edit the defined attribute values. Editing of attributes values follows the same concept as described for ParameterSets below.

8.1. Parametersets

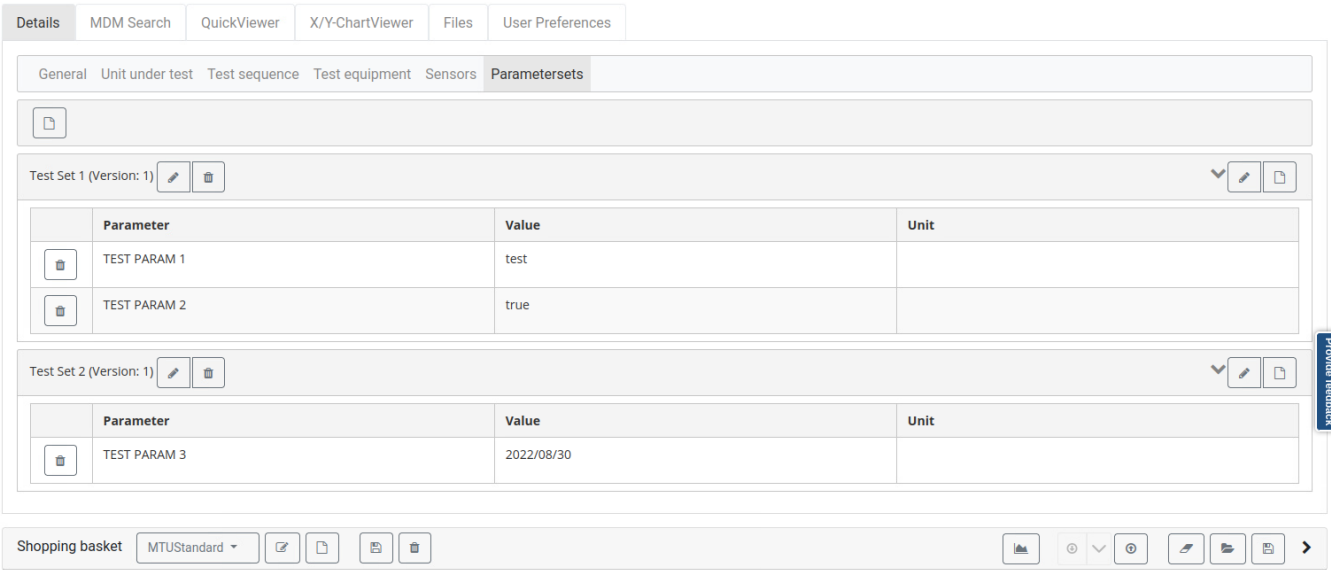


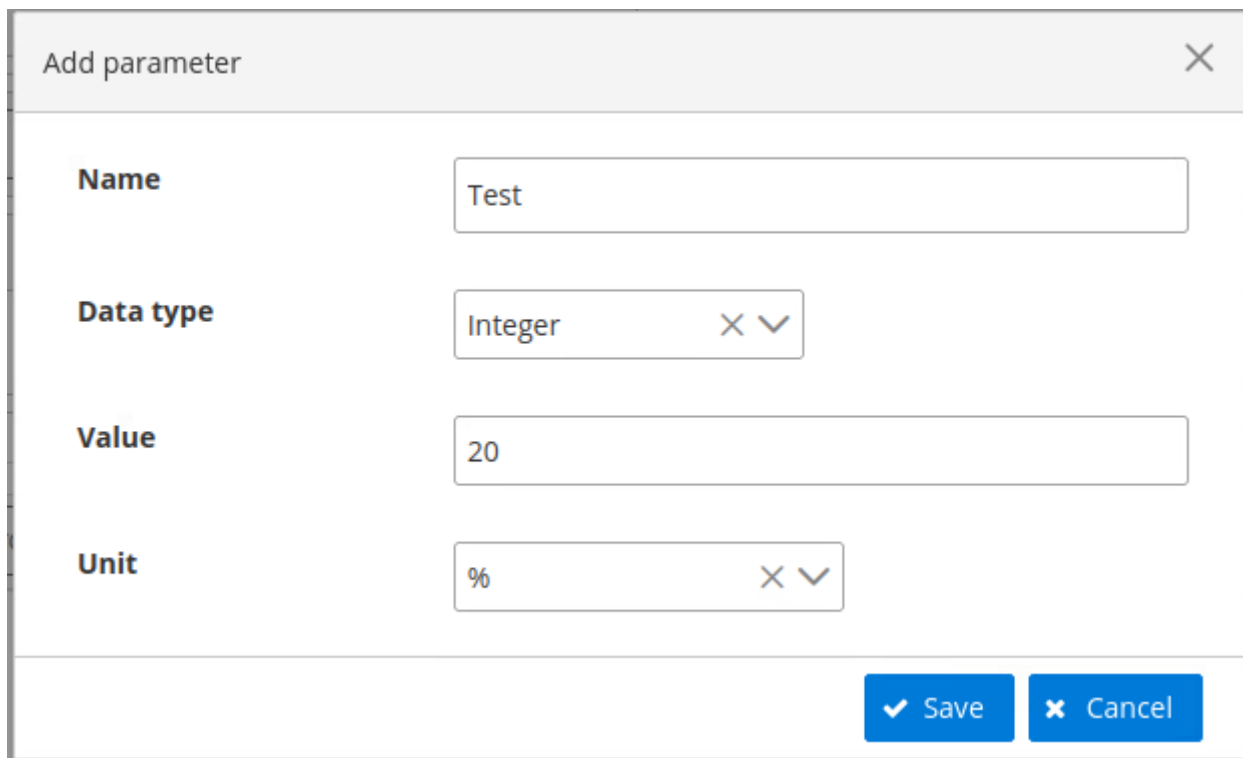
Figure 16. Parametersets overview

This tab gives the user possibility to create, remove and edit parameter sets and add parameters to them.

8.1.1. Create parameter set

Click the "Add paramater set" button to create a new parameter set and give it name. You can delete or rename a paramater set once it has been created.

8.1.2. Create parameters

A dialog box titled "Add parameter" with a close button (X) in the top right corner. It contains four labeled input fields: "Name" with the text "Test", "Data type" with a dropdown menu showing "Integer", "Value" with the text "20", and "Unit" with a dropdown menu showing "%". Each dropdown menu has a small "X" and a checkmark icon. At the bottom right, there are two blue buttons: "Save" with a checkmark icon and "Cancel" with an "X" icon.

Name	Test
Data type	Integer X ✓
Value	20
Unit	% X ✓

✓ Save X Cancel

Figure 17. Create parameter in set

Click the "Edit parameterset" button on the right side to create a parameter in this set. Specify the name, data type and the value. In case of an integer or double data type you can also specify an engineering unit.

You can also edit a parameter's value, change its engineering unit or delete it once it has been created.

9. Visualize data

9.1. Quick Viewer

The component *Quick Viewer* is a simpler version of the *X/Y Chart Viewer* with the possibility to switch between graphically or tabular view of channel values.

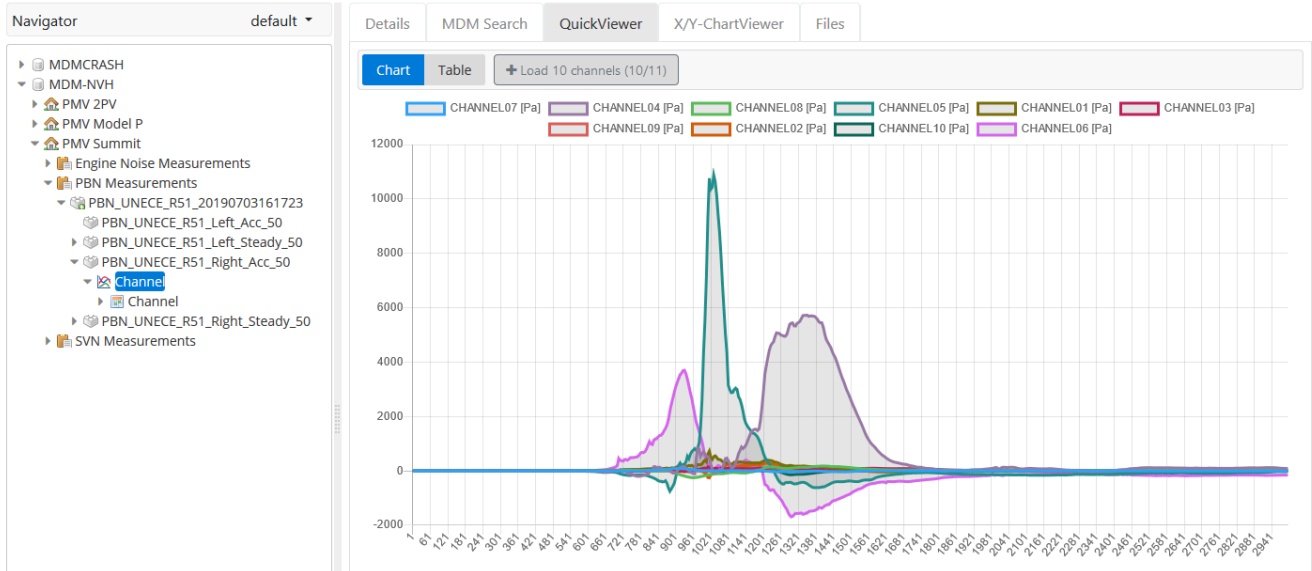


Figure 18. Quick Viewer

9.2. X/Y Chart Viewer

The *X/Y Chart Viewer* represents the data of measurements in different views. This includes the graphical and tabular view of channel values.

You can view one or more ChannelGroups, these are shown in the ChannelGroup selection panel. Each channel of a channel group has an associated axis type. If the axis type of a channel is y-axis, this channel is listed in the Y-Channel selection panel. If the type is x-axis, it is listed in the X-Channel selection panel. If the type is x/y-axis, it is listed in both, the Y-Channel and the X-Channel selection panel.

The following figure shows the general structure of the *X/Y Chart Viewer* on the basis of example data (= Channel).

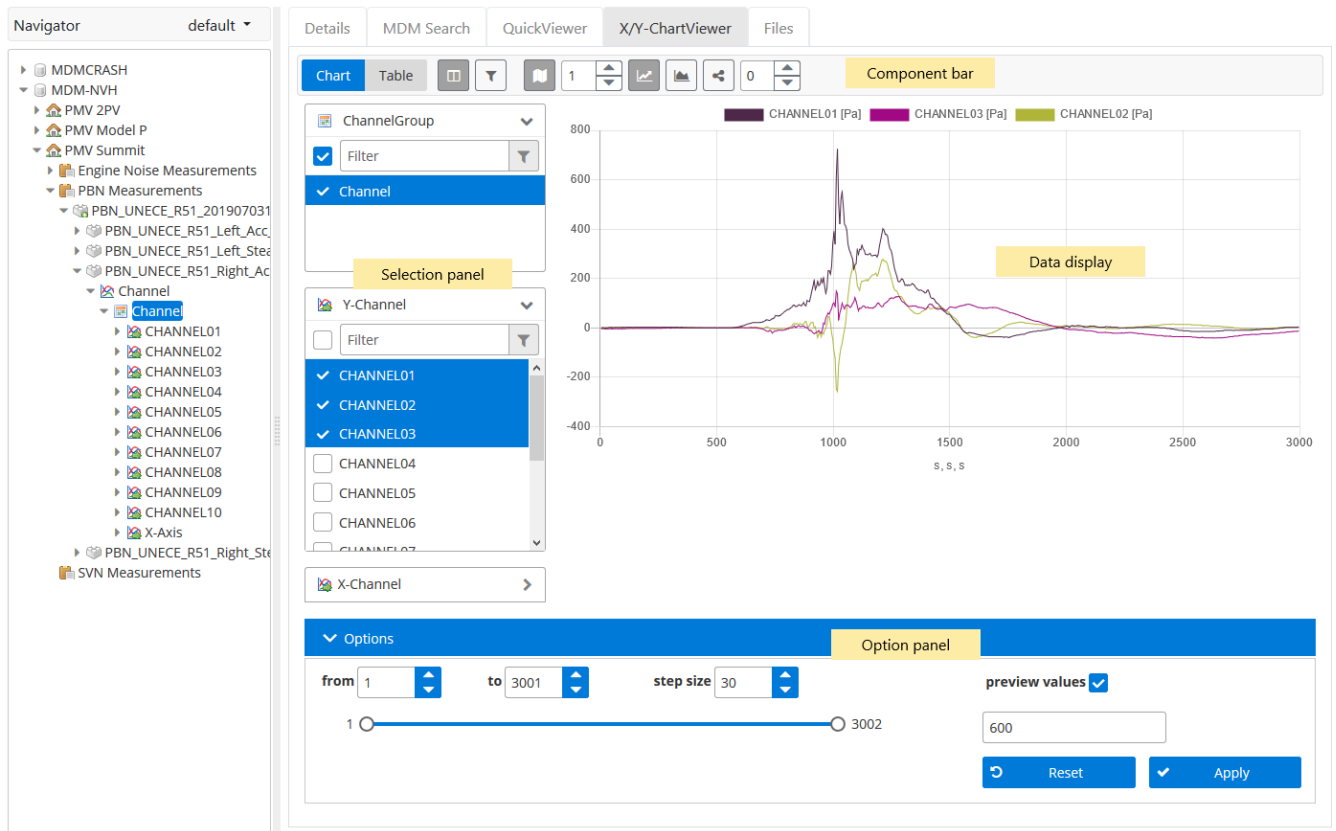


Figure 19. Structure of the X/Y Chart Viewer

Use the Selection panel define the objects to be displayed.

9.2.1. Component bar

In the Component Bar you configure the various display options such as Chart or Data Table and their specific behavior such as legend, decimal points, etc.

Chart

Use Chart for the graphically view of channel values in a 2D axis system. All other functionality of the Component bar and the Option panel are only used to configure this view.

Table

By the selection of this button starts tabular view of channel values. This view has only the page-navigatation and the selection for number of lines.

Icons	Description
	Hide the Selection panel.
	Show the Selection panel.

Icons	Description
	For filter channel options by axis-type.
	For provide all channel options.
	Hide the chart legends of graphical representation of data.
	Show the chart legends of graphical representation of data.
	Hide the data as graphical representation.
	Show the data as graphical representation.
	Fill the area underneath the graph.
	Clear the area underneath the graph.
	Hide data points marker of the curve.
	Mark data points of the curve.
	Group measurement Y channels at the graph.
	Ungroup measurement Y channels at the graph.
	Width of data line or data line tension.
	Includes 3 different options to add Items to the shopping basket: Measurement : adds only the measurement to the shopping basket. Channel Group : adds selected channel group to the shopping basket. Channels : adds selected Ychannels + related Xchannels to the shopping basket.

9.2.2. Selection panel

This panel is used to select the channels to be displayed. It consists of three areas. Filtering with regular expressions is possible for the Channel Group and Y-Channel.

Channel Group

The upper area contains all currently available measurements to the data object selected in the *Navigator*. Use right-click in the *Navigator* to add multiple measurements to the *X/Y Chart Viewer*.

Y-Channel

Defines the channels that can be used as Y-axis, as well as its display unit. Multi-select of Y-axes is possible. Additionally, configured *User Preferences* can be chosen here. For example selecting unit lists, channel filters or virtual channel calculations. The user preferences are directly assigned to the displayed graphic in *X/Y Chart Viewer*.

X-Channel

Defines the available X-axis of the measurement. As X-axis you can only select one channel.

9.2.3. Data display

In this dialog area, the selected channels are displayed in a Chart or Table View. A right-click menu provides additional functions.

9.2.4. Option panel

At this panel different options for displaying channels can be set. For example, the size of X-axis, the step size or whether and how many preview channels should be displayed.

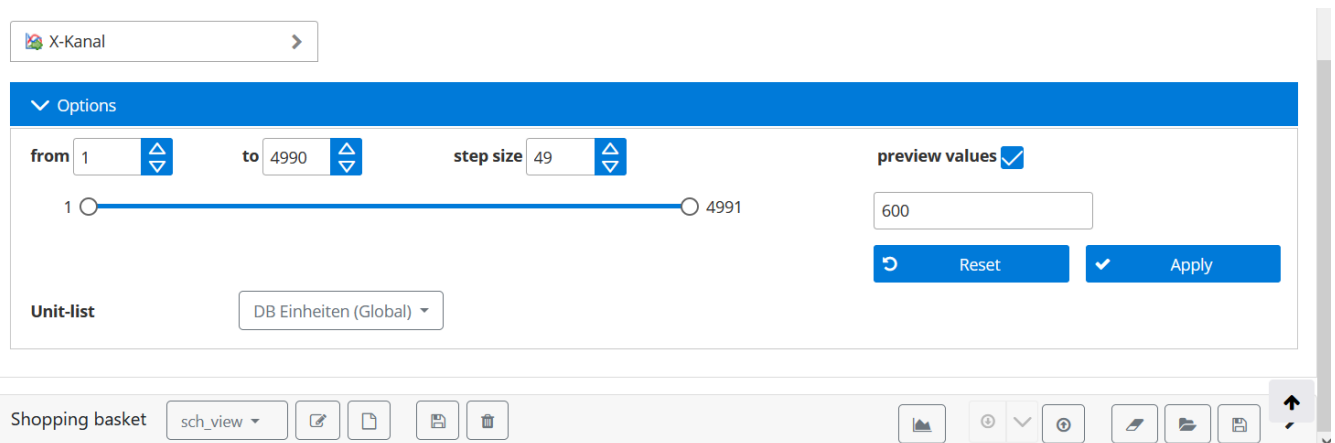


Figure 20. Options

10. Files

10.1. Attach external files

You can attach external files (such as drawings, descriptions, evaluations, etc.) to data objects of type test, test step and measurement. This is done via the tab *Files* of the openMDM Web Client. The view File attachments appears as an tab in the right pane of the openMDM web client.

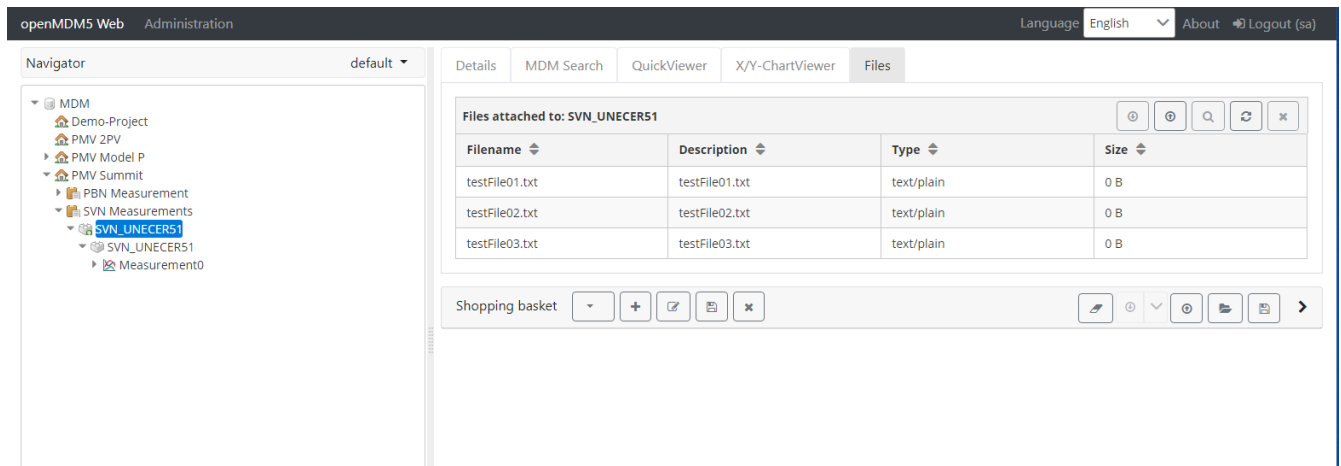


Figure 21. File attachments

Using the buttons in the upper right area of the view, you can upload new files and/or download, update, preview or delete existing files.

10.2. Shopping basket

Data results collected in the Shopping Basket can be downloaded, e.g. as ATFX. These files can be also uploaded. The data can be saved in the application with a specific named and opened again. Result entries can be cleared. See [here](#).

11. User Preferences

11.1. Manage Unit Lists

The User can manage a set of unit lists, to define an individual preferred unit for each combination of a Physical Dimension and a Channel. These definitions can be used in the *X/Y-Chart Viewer*.

The screenshot shows the 'User Preferences' page in the openMDM5 Web application. The interface includes a sidebar with a navigation tree and a main content area with three sections: 'Unit-list', 'Channel filters', and 'My Formulas'. The 'Unit-list' section has a dropdown for '01 Force MdmUser (CRASHDEMO)' and a table with columns 'Source', 'Phys. Dimension', 'Channel Name', and 'Unit'. The 'Channel filters' section has a dropdown for 'Common List 2022-03-01 (Global)' and a table with columns 'Source' and 'Channel Name'. The 'My Formulas' section has a table with columns 'Display Name' and 'Formula'. At the bottom, there is a 'Shopping basket' section with a dropdown for 'Standard_hhe_2021-06-25'.

Source	Phys. Dimension	Channel Name	Unit
MDMCRASH	energy	^16NECKUP0000MO(XYZ)P\$	J

Source	Channel Name
Global	HEAD.+XP\$
Global	CHST.+XP\$

Display Name	Formula
ABSERI U*I [W]	\$(ABSERI\d+C) * \$(ABSERI\d+V)
NECK UP+LOW Pythagoras	abs(\$(NECKUP) * \$(NECKLO)) ^ 0.5

Figure 22. User Preferences

The Administrator can copy the content of an existing preference from some user's **USER SCOPE** into the *Preference Administration* **SYSTEM SCOPE** with the key "**user-prefs.unit-prefs**". This kind of preset will then be available for all users.

11.1.1. Handling of Unit List sets

- For each set of definitions a display name can be defined.
- A definition set can be defined for all data sources, or for one specific data source.
- A definition set contains at least one Physical Dimension → Channel Name → Unit mapping.
- One of the definition sets can be marked as default.

11.1.2. Channel Name

The Channel Name can be **exactly** the Channel's name as displayed in the Navigation Tree, or a **Regular Expression Pattern** for a Channel Name. Hints on using Regular Expressions are displayed in the tooltip.

11.1.3. Writing a Channel Name Regular Expression pattern

Regular expressions are patterns used to match character combinations in strings. They have certain syntax/certain rules of its own, here an introduction:

Simple Patterns

Simple patterns are constructed of characters for which you want to find a direct match. For example, the pattern `UP0000AC` matches character combinations in strings only when the exact sequence "`UP0000AC`" occurs, meaning where this substring is contained at **any** place in the Channel Name.

```
UP0000AC
```

E.g.: This would match:

- 16SPIN`UP0000AC`XP
- 16SPIN`UP0000AC`YP
- 16SPIN`UP0000AC`ZP
- 16NECK`UP0000AC`XP
- 16NECK`UP0000AC`YP
- 16NECK`UP0000AC`ZP

Special Characters

When the search for a match requires something more than a direct match, such as finding one or more "`0`" 's, or finding white space, or find Channels "`beginning with`", or "`ending with`", you can include special characters in the pattern.

```
UP0+ACX
```

Then the use of:

- [Anchors](#)
- [Quantifiers](#)
- and [Character Classes](#)

might be necessary.

Anchors

Anchors tell if the pattern matches at the left or right boundary of a Channel Name, thus enables us to specify, if the pattern shall match for Channel Names beginning with, or ending with that text pattern.

Syntax Definition:

```
^ -- match relative to the beginning of the text (start with)
$ -- match relative to the ending of the text (ends with)
```

Examples:

Match **empty** Channel Names (no text between start and end):

```
^$
```

Match all Channel Names starting with **"16SPINUP0000AC"** :

```
^16SPINUP0000AC
```

E.g.: This would match:

- **16SPINUP0000ACXP**
- **16SPINUP0000ACYP**
- **16SPINUP0000ACZP**

Match all Channel Names ending with **"0000ACZP"** :

```
0000ACZP$
```

E.g.: This would match:

- **16CHST00000000ACZP**
- **16HEADCG0000ACZP**
- **16SPINUP0000ACZP**
- **16SPINLO0000ACZP**
- **16NECKUP0000ACZP**
- **16NECKLO0000ACZP**

Quantifiers

Syntax Definition:

x can be any Character or Character Class. The quantifier **must follow** a character, place holder, or Character Class.

IMPORTANT

*A Regular Expression Pattern **cannot start** with a quantifier.*

E.g.: ***SOME thing** or **?SOME thing** are invalid expressions.

```
x{n}    -- occurs exactly n times
x{n,}   -- occurs at least n times
x{n,m}  -- occurs at least n times, not more than m times
?       -- occurs never or exactly 1 time (same as x{0,1})
*       -- occurs never or unlimited times (same as x{0,})
+       -- occurs at least 1 time or unlimited times (same as x{1,})
```

Examples:

Match Channel Names containing **"P"** followed by anything or followed by **"0"** :

```
P0*
```

E.g.: This would match:

- 16SPINUP**P0000**ACZ**P**
- 16SPINLO0000ACZ**P**
- 16NECKUP**P0000**ACZ**P**
- 16NECKLO0000ACZ**P**

Match Channel Names containing **"P0"** followed by anything or followed by **"0"** :

```
P0+
```

E.g.: This would match:

- 16SINUP**P0000**ACZ**P**
- 16SINUP**P0**815ACZ**P**
- 16NECKUP**P0000**ACZ**P**

Character Classes

Syntax Definition:

A list of characters or Character Classes enclosed between **[** and **]** ; when enclosed between **[^** and **]** then this means all characters that are listed are **excluded** . The **-** sign (reserved character) has special meaning, this is to define a **range** of characters. Reserved characters can be **escaped** via **** backslash.

.	-- any character (ONE character place holder)
[XYZ]	-- one of the characters 'X', 'Y' or 'Z'
[.]	-- the character '.'
\.	-- the character '.'
[0-9]	-- same as [0123456789], match one digit
[a-z]	-- lower-case ASCII letters, match one lower-case ASCII letter
[A-Z]	-- upper-case ASCII letters, match one upper-case ASCII letter
[^0-9]	-- same as [^0123456789], match if character is NOT a digit
[^a-z]	-- no lower-case ASCII letter
[^A-Z]	-- no upper-case ASCII letter
\d	-- short form for [0-9] (a digit)
\D	-- short form for [^0-9], [^\d] (not a digit)
\w	-- word character: short form for [A-Za-z0-9_]
\W	-- not a word: short form for [^A-Za-z0-9_], [^\w]
\s	-- any blank space character
\S	-- any none blank space character
[\w\~]	-- word character including '~'

Examples:

Match Channel Names that match exactly "16SPINUP0000ACXP" or "16SPINUP0000ACYP" :

```
^16SPINUP0000AC[XY]P$
```

E.g.: This would match:

- 16SPINUP0000ACXP
- 16SPINUP0000ACYP

Match X / Y Channel Names for "LO", ending with "P" (Lower Body region sensor data):

```
L00000..[XY]P$
```

E.g.: This would match:

- 16NECKL00000FOXP
- 16NECKL00000FOYP
- 16NECKL00000MOXP
- 16NECKL00000MOYP
- 16SPINL00000ACXP
- 16SPINL00000ACYP
- 16STRNL00000ACXP

Match X / Z Channel Names for acceleration or deceleration, with 6 digits before:

\d{6}[AD][CS][XZ]

E.g.: This would match:

- 16CHST000000ACXP
- 16CHST000000ACZP
- 16CHST000000DSXP
- 16CHST020406ACZP
- 16CHST503010DSXP

External references

Everything that goes beyond our introduction you can find on-line; e.g.:

- regular-expressions.info
- regex101.com
- regexr.com

11.2. Manage Channel Lists

The User can manage a set of channel lists, to define an individual pre-selected set of channels. These definitions can be used in the *X/Y-Chart Viewer*'s. All Y-Channels inside the *X/Y-Chart Viewer* where **any Channel Name (Regular Expression patterns)** match, will be preselected:

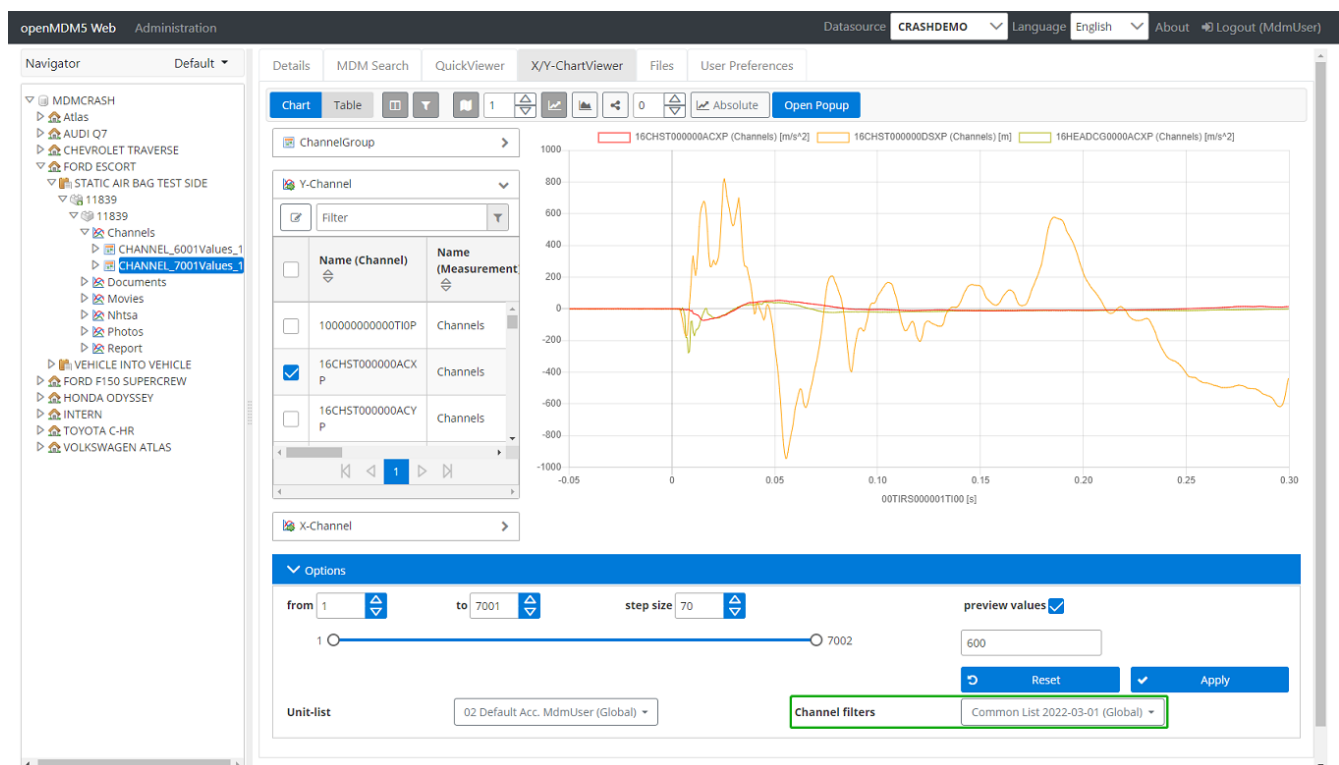


Figure 23. Channel List in X/Y-Chart Viewer

11.2.1. Handling of Channel List sets

- For each set of definitions an unique display name must be defined.
- A definition set can be defined for all data sources, or for one specific data source.
- A definition set contains at least one *Channel Name (Regular Expression pattern)*.
- One of the definition sets can be marked as default.

11.2.2. Channel Name

The Channel Name can be **exactly** the Channel's name as displayed in the Navigation Tree, or a **Regular Expression Pattern** for a Channel Name. For writing a *Channel Name Regular Expression pattern* see [here](#).

11.3. Manage Formulas

The User can manage a set of formulas, to define an expression on how certain Channel measurement values will be calculated. Each formula has an unique (display) name and a formula expression, that contains both: The calculation and the channels being used specified by their *Channel Name (Regular Expression pattern)* wrapped inside a placeholder expression $\${}$.

All defined formulas where **all Channel Name (Regular Expression patterns)** match, will be available inside the X/Y-Chart Viewer:

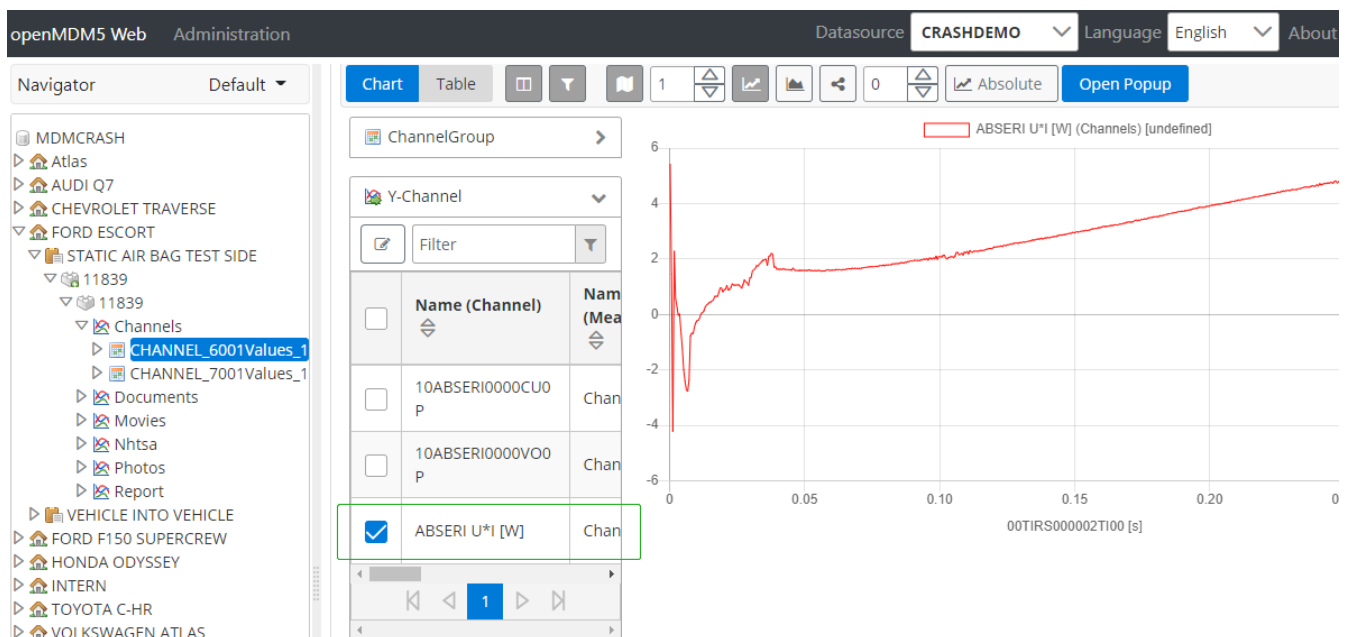


Figure 24. Formula in X/Y-Chart Viewer

e.g.: $\${Current} * \${Voltage}$

Would search for the 1st Channel-Name containing 'Current' (e.g. 'Battery Current'), and search for the 1st Channel-Name containing 'Voltage' (e.g. 'USB Voltage Min').

Then it will calculate the power 'P = I * U' [W = A * V]; of the matched measurement values.

11.3.1. Channel Name

The Channel Name can be defined by the **exact** Channel name as displayed in the Navigation Tree, or a **Regular Expression Pattern**. For writing a *Channel Name Regular Expression pattern* see [here](#).

11.3.2. Formula

When adding a new formula, or editing an existing formula:

Formula definition

Display Name

Formula

Syntax Hint:

The formula takes a Channel-Name Regular Expression Pattern inside place holders `'${Channel-Reg-Expr}'`:

e.g.: `'${Current} * ${Voltage}'`

Would search for the 1st Channel-Name containing 'Current' (e.g. 'Battery Current'), and search for the 1st Channel-Name containing 'Voltage' (e.g. 'USB Voltage Min').

Then it will calculate the power `'P = I * U' [W = A * V]`; of the matched measurement values.

✓ Ok ✗ Cancel

Figure 25. Formula Input

The formula input is **not** permanently validated, only under following conditions:

- *Display Name* is filled and valid
- **Tabulator** or **Enter** key was pressed, when inside the *Display Name* field.
- **Tabulator** or **Enter** key was pressed, when inside the *Formula* field.
- keys other than **Space**, **+**, **-**, *****, **/**, **^**, were pressed, when inside the *Formula* field.

12. Administrator Functions

An overview of advanced functions for administrators on the Administrator tab are described in this chapter.

NOTE

These functions are expert functions and very technical. For a deeper dive, we recommend the following technical documentation:

- * [Installation and Setup Guide for the openMDM Application](#)
- * [Readme file of the Eclipse mdmbl project](#)

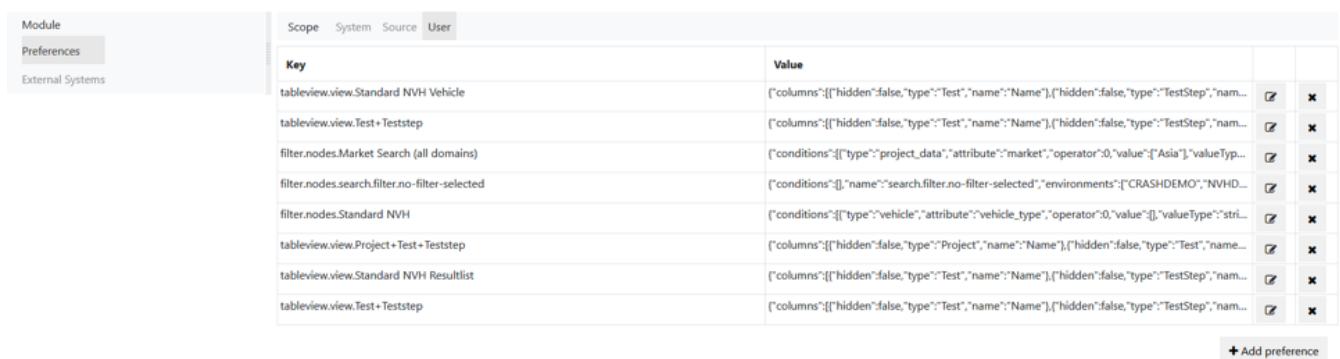
To use these functions, your user has to be assigned to the administrator role, please ask your system administrator.

12.1. Configure preferences

The component *Preferences* includes the following scopes:

- **System:** preferences that are valid system wide
- **Source:** preferences are valid for the dedicated data source (environment)
- **User:** user specific preferences. Preferences are e.g. search filters a user has defined, they are stored automatically.

A preference is a pair of a unique key and a value. The key is composed of a prefix defining the purpose of the preference followed by an arbitrary but unique identifier string. It is recommended to choose the identifier the same as the preferences "name" field, in case there is one. The value holds the preference's data in a JSON string. [More information](#).



The screenshot shows the 'Preferences' window with the 'User' tab selected. The table lists various preferences with their keys and values. The 'filter.nodes.Standard NVH' key is highlighted.

Scope	System	Source	User
Key	Value		
tableview.view.Standard NVH Vehicle	{"columns":[{"hidden":false,"type":"Test","name":"Name"}, {"hidden":false,"type":"TestStep","nam...		
tableview.view.Test+Teststep	{"columns":[{"hidden":false,"type":"Test","name":"Name"}, {"hidden":false,"type":"TestStep","nam...		
filter.nodes.Market Search (all domains)	{"conditions":[{"type":"project_data","attribute":"market","operator":"0","value":["Asia"],"valueTyp...		
filter.nodes.search.filter.no-filter-selected	{"conditions":{"name":"search.filter.no-filter-selected","environments":["CRASHDEMO","NVHD...		
filter.nodes.Standard NVH	{"conditions":[{"type":"vehicle","attribute":"vehicle_type","operator":"0","value":{"valueType":"stri...		
tableview.view.Project+Test+Teststep	{"columns":[{"hidden":false,"type":"Project","name":"Name"}, {"hidden":false,"type":"Test","name...		
tableview.view.Standard NVH Resultlist	{"columns":[{"hidden":false,"type":"Test","name":"Name"}, {"hidden":false,"type":"TestStep","nam...		
tableview.view.Test+Teststep	{"columns":[{"hidden":false,"type":"Test","name":"Name"}, {"hidden":false,"type":"TestStep","nam...		

+ Add preference

Figure 26. Preferences (User)

In the screenshot you can see the "filter.nodes.Standard NVH" key, this row describes the search filter that was defined [here](#).

12.2. Add a Node Provider

Node Providers are defined using system preferences. The key must be prefixed with "nodeprovider." followed with the name of the Node Provider, for instance: "nodeprovider.example".

The screenshot shows a 'Preferences Editor' dialog box. It has three main sections: 'Key:', 'Scope:', and 'Value:'. The 'Key:' section contains a text field with the value 'nodeprovider.example'. The 'Scope:' section contains a button labeled 'SYSTEM'. The 'Value:' section contains a large text area with the text '<Json Definition>'. At the bottom left is a button labeled 'xCancel' and at the bottom right is a button labeled 'Save' with a save icon.

Figure 27. Add a Node Provider

[More information.](#)

12.3. Configure external systems

The component *External Systems* includes environment specific mapping information between an external system and the openMDM system attributes. You can import data from other data sources (e.g. ATFX files, csv files, xml files, ODS Server,...) that are not compliant to the openMDM5 application model, by defining a mapping of the metadata included in the source to the openMDM5 model. The openMDM5 application import function has to support the specific data source format.

Selected environment: NVHDEMO

Back

Name	Description
ATFX_Import_NVHDEMO	ATFX Import for NVHDEMO

Save

Ext. system attributes

Name	Description	
☐ \$[silencer]\(\\(.+))fibre_type		✕
☐ \$[silencer]\(\\(.+))manufacturer		✕
☐ \$[silencer]\(\\(.+))model		✕
☐ \$[silencer]\(\\(.+))type		✕
☐ \$[tyre]\(\\(.+))manufacturer		✕
☐ \$[tyre]\(\\(.+))size		✕
☐ \$[tyre]\(\\(.+))tread_depth		✕
☐ [PBN_test_conduction]PMR		✕
☐ [PBN_test_conduction]accelerated		✕
☐ [PBN_test_conduction]tested_vehi		✕
☐ [PBN_test_conduction]v_test		✕
☒ [vehicle]mass_in_running_order	mass_in_running_order	✕

Add new

MDM attributes

Component type	Component name	Attribute name	
Unit under test	vehicle	mass_in_running_order	✕

Add new

Figure 28. External Systems (ATFX importer mapping in NVHdemo environment)

12.4. Validation of context attributes

The values of context attributes can be validated by regular expressions. For that, an user preference must be defined to specify attributes and their according patterns. The preference must be named as "context_attribute.validators". The preferences value is a simple json string with the attribute identifier as key and the regular expression as value. It follows the rule "{context_type}.{tpl_component_name}.{attribute_name}".

For example:

```
{
  "UUT.car.vin": "^[A-Z0-9]+$",
  "TE.datalogger.serial_nr": "^[0-9]{7}$",
}
```

The expression will be validated when an attribute is created/edited. Saving would be only possible when the value is valid due to this pattern.

12.5. Visibility of context attributes ordered values

Visibility of context attributes ordered values in the details view can be managed via a preference. The preferences key is `navigator.details.context.hide_ordered`. If the preferences value is `true` the ordered values are hidden. If the preference has any other value or is not present the ordered values are displayed.

12.6. Notification messages

The duration a notification is shown in the UI before leaving into message box can be managed via a preference. The preference key is `notification.message.life_span`. The value must be a number and defines the life span in milliseconds. Default is 3000 if no preference is defined.

12.7. Functional Rights

12.7.1. General Information

Functional rights can be granted to users via a system scoped preference. The rights configuration is highly customizable. It accepts conditions based on logical expressions. The expressions may refer to dynamic context data at runtime. The conditions may also refer to structural user profile data. The user profile includes data source scoped user details and roles as well as application scoped user details and roles (from security context on web server or synchronized by identity provider).

In case no preference is defined functional rights are granted to all users.

In case preference is corrupted edit rights are granted for application scoped admin role only.

The default configuration containing all functional right keys and a description of each right can be found in the web client under `/assets/rights_config.default.json`.

12.7.2. SpecialistAdmin

The role `SpecialistAdmin` is datasource specific and provides access to administration components applied directly to an underlying datasource. This components include for example: catalog and template manager, unit and quantity administration. A user only has access to this components if he has either the application level role `SpecialistAdmin_<SOURCE1>` where `SOURCE1` is the name of the data source. Or has the Role `SpecialistAdmin` in the Datasource.

12.7.3. The Preference

The preferences key is `rights_config`.

The preferences value must be a valid json object of the form:

```
{
  "default": boolean,
  "functionalRights": {
    "KEY": {
      "condition": string,
      "description": string
    },
    ...
  }
}
```

The **KEY** is a placeholder for the key of the functional right. The key is case sensitive. The key may contain `..`. In that case the json structure expects a nested object.

The **default** field is a fallback for all functional rights that are referenced in the Mdm5-WebApplication but not explicitly specified in this preference.

The **condition** field holds the actual condition to be evaluated whether the functional right is granted. See following section for further details.

The **description** field is optional and for documentation use only.

Condition Syntax

Conditions may include data references provided as:

- string literals: `'a'` (defines the string `a`; string literals are case sensitive).
- string literal arrays: `('a', 'b', 'c')` (defines an array containing the strings `a`, `b` and `c`). However, string literal lists must not be nested.
- dynamic context data: `${test}` (evaluates to the value of the variable with the name `test`).
- application scoped user data: `username` (evaluates to the value of the property with the name `username` of the user details provided on web server). Nested properties are supported by chaining field names separated by `..`.
- data source scoped user data: `sourceName.username` (evaluates to the value of the property with the name `username` of the user details provided in data source with name `sourceName`). Nested properties are supported by chaining field names separated by `..`.

Condition expression syntax is developed to handle variables holding values of types string and string[] only. However, referencing other data will not result in an error but may cause unexpected outcome.

Conditions may include operators for strings and string arrays, resp.:

- equals: `==` (compares two strings for equality, eg `${a}=='a'`. Should a variable refer to an string array only the array references is respected.)
- not equals: `!=` (compares two strings for inequality, eg `${a}!='b'`. Should a variable refer to an string array only the array references is respected.)
- includes: `IN` (is case sensitive; checks if left side is included in right side, eg `${a} IN ('a', 'b')`. Should either side not be an array it will be wrapped in one.)

Conditions may include logical operators for strings:

- and: `&&` (connects two boolean expressions with a logical and, eg `${a}=='a' && ${b}!='c'`).
- or: `||` (connects two boolean expressions with a logical or, eg `${a}=='a' || ${b}!='c'`).
- negation: `!` (negates the following boolean expression , eg `!${a}=='b'` is equal to `${a}!='b'`).

Conditions may include parenthesis and blanks:

- parenthesis are a part of the string literal arrays syntax.

- parenthesis may be used to override operator precedence (eg. negates `!($a)=='a' && $b IN ('a', 'b'))`).
- parenthesis may be used around boolean expressions for better readability (eg. negates `!($a)=='a'` is same as `!$a=='a'`).
- parenthesis must not be used around data expressions (eg. negates `($a)=='a'` and `'a' IN 'a', 'b'` are invalid).
- blanks must not be used within operators (eg. `= =` and `I N` are invalid).
- blanks must not be used in opening tag of context data reference (eg. `$ {a}` is invalid).
- blanks may be used with in parenthesis and curly brackets (eg. `${ a }` and `( 'a'!='b' )` are valid).
- blanks may be used between any kind of expression or operator (eg: `'a' != 'b'` or `'a' IN ('a', 'b')` is valid).
- blanks in string literals are considered to be part of the string ('a is not equal to 'a ').

Valid values

```
{
  "default": false,
  "functionalRights": {
    "administration": {
      "messgroessen": {
        "show": {
          "condition": "MDMNVH.roles IN ('Admin', 'MDMSystemAdministrator') &&
MDMNVH.department==$department",
          "description": "If true, the preference menu entry is visible."
        }
      },
      "units": {
        "show": {
          "condition": "'Admin' IN roles "
          "description": "Units are visible for application role 'Admin' only."
        }
      },
    },
  },
  "modules": {
    "berechnungsauftrag": {
      "show": {
        "condition": "$department != MDMNVH.department || username == 'John Doe'"
      }
    }
  }
}
```

12.7.4. Administration component

The available functional rights are available in a tree table. For the leaf nodes a condition can be specified and a description is shown explaining the scope of the functional right.

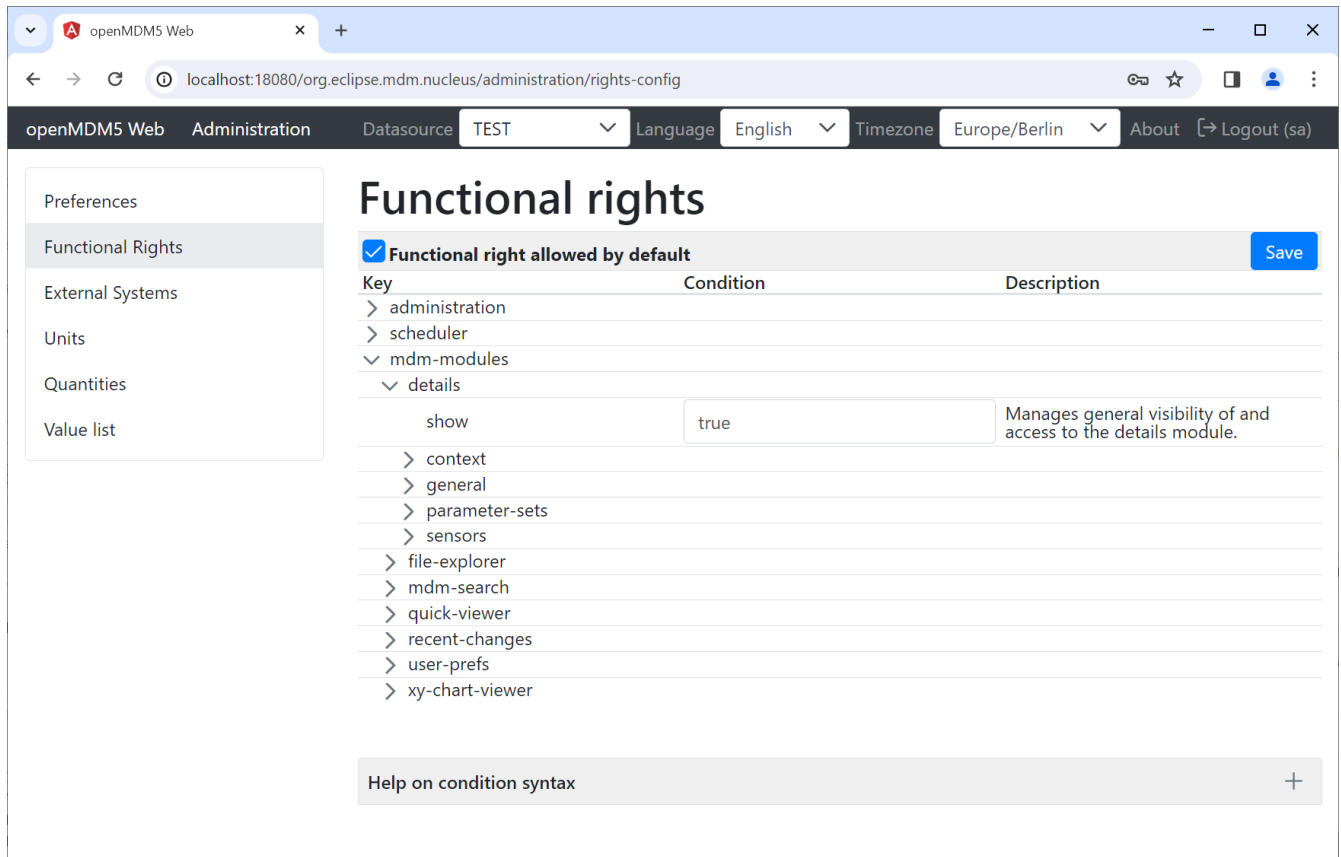


Figure 29. Functional rights

12.8. Announcements

12.8.1. Administration

The administrator is able to configure several announcements the users will be getting displayed when starting the application. The configured announcements are stored as a system preference.

An announcement consists of header, body and a validFrom timestamp that defines when the announcement should be displayed to the user.

The body of every announcement will be embedded as html. So, either normal text or valid html with formats, links etc. is possible.

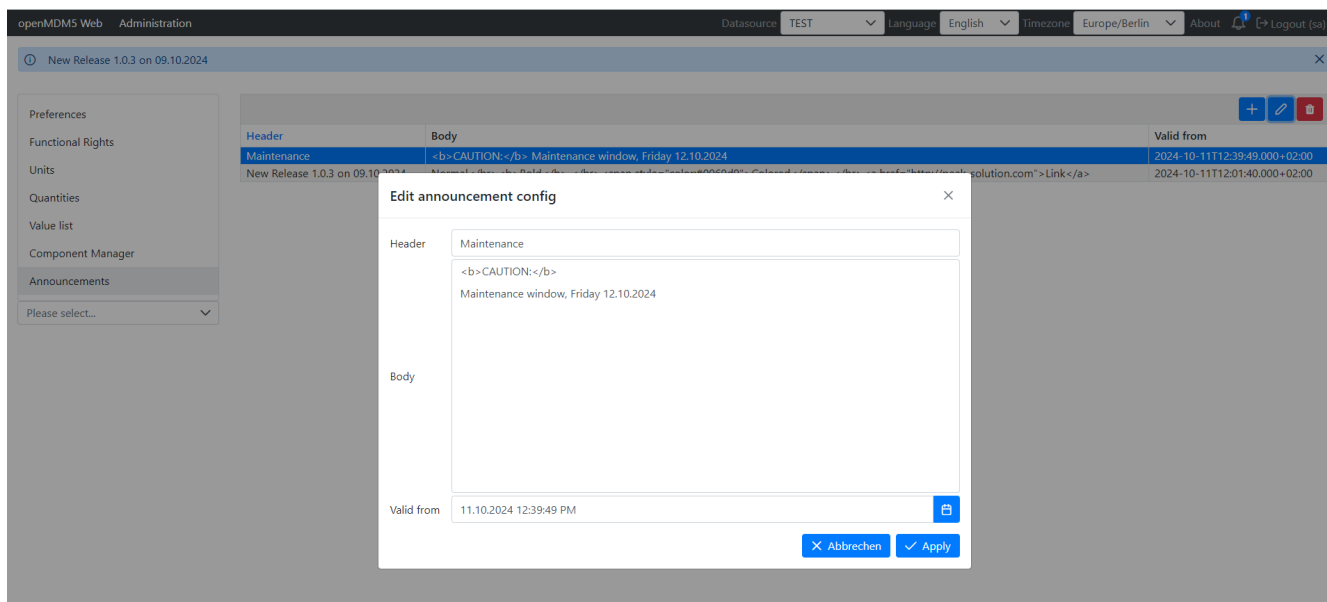


Figure 30. Announcement configuration

12.8.2. User view

Every announcement is then shown at start of the application. The user is able to click on it to see details (body).

As well the user has the possibility to dismiss these announcements by a "don't remind me"-option. If they do so this will be stored as a user preference with id and the timestamp. When loading announcements for the user the timestamps of dismissal and validFrom are compare to decide whether to show it or not.

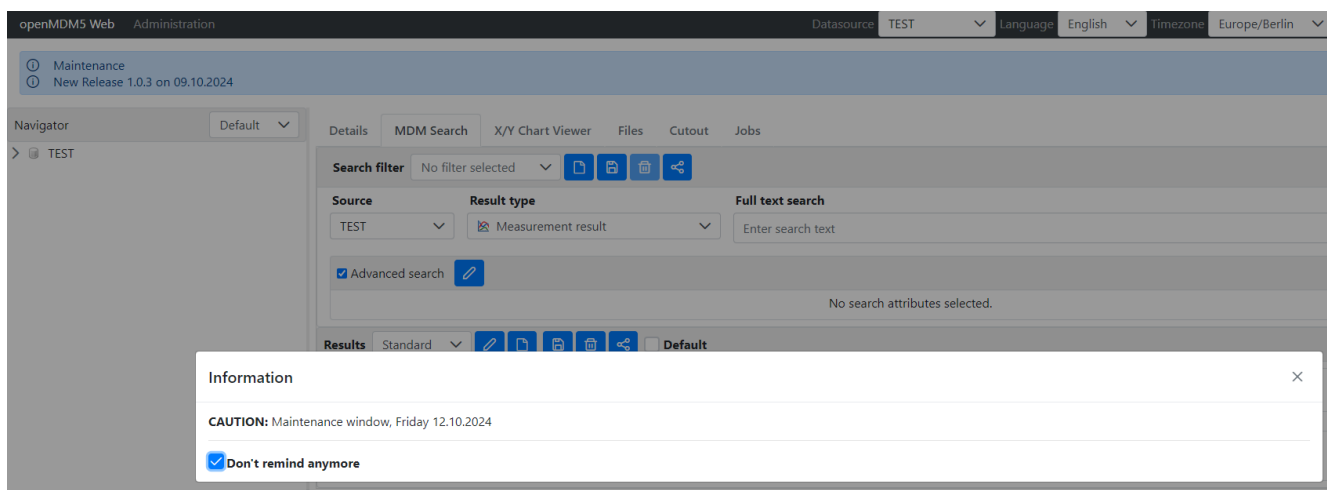


Figure 31. Announcement example

12.9. Context Bulk Editor

12.9.1. General Information

The Context Bulk Editor enables the user to edit contexts of multiple context descriptables at once. By default the selection of context descriptables is done via the shopping basket. However, it is possible to register customized context describable providers. The editor loads contexts initially (navigating into the component) and on demand (button click). The affected context types can be

managed via a preference.

12.9.2. Configuration

General visibility is managed by functional right with the key "context-bulk-editor".

The affected context types are managed by a preference with the key `context-bulk-editor.config`. The value must be a json setting the field `contextGroups` to a single value or an array of the following: `0`, `1`, `"Measured"`, `"Ordered"` (case-insensitive).

Example 1: Single number.

```
{
  "contextGroups": 1
}
```

Example 2: Array of strings.

```
{
  "contextGroups": ["Measured", "Ordered"]
}
```

13. Catalogs and Templates

13.1. Catalog Components

During test some component definitions are reused for lots of tests. They are split into three root areas:

catcomp context types

- □UnitUnderTest
- □TestEquipment
- □TestSequence

A more specific example is given here.

catcomp example

- □UnitUnderTest
 - vehicle
 - wheels
 - engine
 - combustion
- □TestEquipment
 - testbed
 - logger
 - sensors
- □TestSequence
 - setup step
 - execute step
 - detach step

An administrative engineer is allowed to to define those components to be used in tests. This reduces the workload when setting up tests.

An example for an UnitUnderTest Component is:

body

```
class vehicle:
    vin: str
    nr: str

class sensor:
    manufacturer: str
    serial_nr: str
```

13.2. Templates

In openMDM the descriptive Data is stored in the following structure.

descriptive data structure

- Test
 - TestStep
 - Ordered Context
 - UnitUnderTest
 - TestEquipment
 - TestSequence
 - Measurement
 - Measured Context
 - UnitUnderTest
 - TestEquipment
 - TestSequence

The Ordered and the Measured context have the same setup but might have different values.

After defining the Catalog Components we need a way to define the test structure. openMDM provides Templates to define this descriptive data.

- Test Template (tpltests)
 - TestStep Templates (tplteststeps)
 - Template UnitUnderTest (tplroots/unitundertest/)
 - Template components
 - Template TestEquipment (tplroots/testequipment/)
 - Template components
 - Template TestSequence (tplroots/testsequence/)
 - Template components

The reason to create such templates is to allow preparation of all elements (instances) of a test

before the test is executed. During test execution the Measured Context is filled with the actual values. The Ordered Context is used to define the test structure and the expected values.

This means if I create a Test based on a *Test Template* it will create all sub elements to be filled by test engineer or testbed.

To be used in an productive environment the templates need a versioning system. A template version in use (state is valid) can not be changed anymore.

13.3. Template Versioning

13.3.1. REST API

requests

```
POST /environments/{SOURCENAME}/tpltests/{ID}/version
POST /environments/{SOURCENAME}/tplteststeps/{ID}/version
POST /environments/{SOURCENAME}/tplroots/{CONTEXTTYPE}/{ID}/version
```

body

```
{
  "updateCatalog": false, ①
  "copyValidRelated": false, ②
  "useLatestRelated": false ③
}
```

- ① updateCatalog: When copying component templates, supplement with new component attributes.
- ② copyValidRelated: When copying a template, create a copy of all valid related templates. (Necessary for updating component templates or integrating new template versions on lower levels.)
- ③ useLatestRelated: If a related template is available in a new version, use it.

13.3.2. Use cases

New version

[newVersionComponentTemplate] | *newVersionComponentTemplate.gif*

Figure 32. New version of a component template.

If a new version without special settings is called all component templates are copied. The new version is set to editable.

Integrate new catalog attributes

[newVersionComponentTemplateAddAttributes] | *newVersionComponentTemplateAddAttributes.gif*

Figure 33. New version of a component template. Adding new attributes.

If `updateCatalog` is set to true, the new version of the component template is created adding the new attributes.

If `copyValidRelated` at Test or TestStep level is set to true, all valid related templates are copied. The new version of the component template is set to editable. If `updateCatalog` is also set to true, the new attributes are added to the copied templates. All related templates are set automatically to valid. This means they cannot be changed anymore. Only the top element is set to editable.

Integrate new versions of related templates

[newVersionTestStepNewRelatedVersion] | *newVersionTestStepNewRelatedVersion.gif*

Figure 34. New version of a teststep template. Using new component version.

if `useLatestRelated` is set to true a new version of related templates is used if available. To update Component Templates in a Test Template `copyValidRelated` must also be used.

14. Special HTTP API methods

This section will describe some special behavior of the openMDM REST API

14.1. Change TestStep Template

In some rare cases it might be useful to evolve the TestStep template.

It might be useful to migrate the TestStep template after creating a copy of the TestStep instance.

14.1.1. HTTP API

PUT /environments/{SOURCE_NAME}/teststeps/{ID}

body

```
{
  "TemplateTestStep": 4711
}
```

4711 is the id of the new template to be used.

14.1.2. Migration Rules

The new template must fulfill the following rules:

- The new TestStepTemplate must already be usable in TestTemplate.
- New components can be added to a ContextRoot.
- New attributes can be added to a component.
- If a new attribute is not optional it must provide a default value.
- Optional attributes can not be changed to obligatory ones.
- If more than one ComponentTemplate point to the same ComponentPart the names must match.
- All Attributes of the old Template must exist in the new one.
- ComponentPart can not be exchanged just because it has the same attributes. It must match.

14.1.3. Example

Here TestStepTemplate1 can be migrated to TestStepTemplate2. This will add the UUT component engine and the vehicle attribute model.

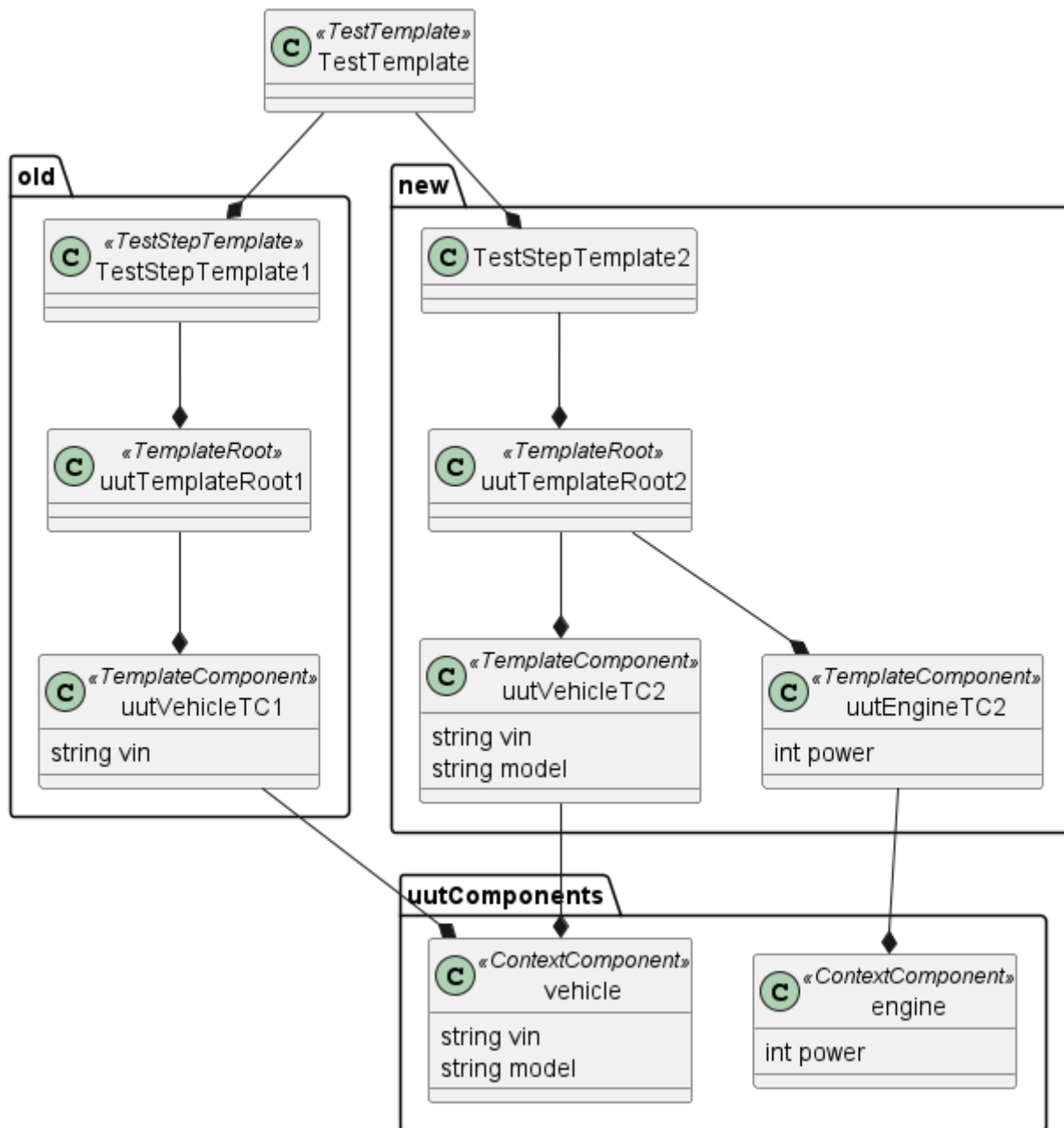


Figure 35. Migrate TestStep Template

15. Audit Logging

This documentation outlines the implementation of **audit logging** following the **Elastic Common Schema (ECS)** standard while leveraging **Mapped Diagnostic Context (MDC)** for enhanced traceability.

By integrating **MDC**, we ensure that each log entry captures relevant contextual information, such as **user actions**, **request IDs**, and **session details**, improving observability and correlation across distributed systems. The logs are structured according to ECS, enabling seamless indexing, searching, and visualization in **Elasticsearch**, **Kibana**, and **other compatible tools**.

The following sections detail the configuration, usage, and best practices for implementing ECS-compliant audit logging with MDC in *Mdm5*.

15.1. What is logged with the 'audit' marker?

- Every user interaction with the *Mdm5 RestApi* (before and after each endpoint invocation)
- User login, logout, and synchronization
- Role assignments and permission changes

15.2. Mapped Diagnostic Context

Mapped Diagnostic Context (MDC) is a mechanism that allows you to enrich log events with additional contextual information, such as **user IDs**, **session IDs**, or **request details**. This information is stored in a **thread-local map**, meaning it is unique to the current thread of execution and can be automatically added to logs without needing to manually include it in every log statement.

15.3. Elastic Common Schema

Elastic Common Schema (ECS) is an open standard developed by **Elastic** for organizing and structuring data logs, metrics, and traces in a consistent way. ECS helps unify how logs are stored and analyzed, allowing for better interoperability and easier correlation across various systems.

One of ECS's key features is **standardized field names**. ECS defines a common set of field names and data types that make it easier to process logs and metrics across different sources (e.g., web servers, applications, network devices). For example, ECS might specify a field like `@timestamp` for the event time or `user.name` for the user's name in logs.

15.4. Utilization in MDM5 (for contributors)

15.4.1. Basic Concept

1. Add all contextual information to the MDC while respecting ECS field names.
2. Mark log events with `org.slf4j.Marker` using the name `audit`.

3. Configure the logger to generate audit log output.

15.4.2. Setting up the MDC

Before each *Mdm5 RestApi* call, the following filters add request-based information to the MDC:

- The `ch.qos.logback.classic.helpers.MDCInsertingServletFilter` is used to add basic information to the MDC.
- The `@WebFilter org.eclipse.mdm.application.logging.MdmMdcInsertingServletFilter` adds more specific information (e.g., client IP, username, etc.) to the MDC on every *Mdm5 RestApi* call.

ECS requires log events to be structured using a predefined set of fields.

It is recommended to use the `org.eclipse.mdm.logging.AuditLogger` for logging. This provides an easy way to add the required information, ensures proper field naming, and keeps the code clean and simple.

However, the MDC is easily accessible, and all fields can be set or overwritten manually if needed.

15.4.3. Marking the log

It is recommended to use the `org.eclipse.mdm.logging.AuditLogger` for logging, as it automatically sets the required `org.slf4j.Marker`.

However, to include a log event in the audit log, simply adding an `org.slf4j.Marker` with the name `audit` is sufficient.

```
LOG.debug(MarkerFactory.getMarker("audit"), "some message")
```

15.4.4. AuditLogger

ECS event categorization fields tend to repeat themselves within the scope of a class or a task, with only the **type** and **outcome** fields changing.

The `AuditLoggerFactory` allows specifying all constant fields. It provides methods to create `AuditLogger` instances with **type** and **outcome** set as required.

The `AuditLogger` itself provides a method to process the actual logging call. It adds ECS-specific fields to the MDC, enriches the log call with the `audit` marker, executes the log, and cleans the MDC afterward. By taking the logging call as a lambda expression, it offers full flexibility for logging call signatures.

```

private static final AuditLoggerFactory loggerFactory = new
AuditLoggerFactory("Some action.", EcsEventCategory.Process, EcsEventType.User);

loggerFactory.start().log((m) -> LOGGER.debug(m, "Process started.)); // adds
EcsEventType.Start
loggerFactory.success().log((m) -> LOGGER.debug(m, "Process successfully
ended.)); // adds EcsEventType.End, EcsEventOutcome.Success
loggerFactory.error().log((m) -> LOGGER.error(m, e.getLocalizedMessage(), e)); //
adds EcsEventType.End, EcsEventType.Error

```

15.4.5. Configuration (for administrators)

To generate a log file, the configured logging framework must be properly set up. Since audit logging is based on **slf4j**, it should work with various logging frameworks. However, it has only been tested with **logback** so far.

Logback

Filter

The **LoggingMarkerFilter** allows you to capture relevant log events, ensuring that only the necessary events are logged based on your configuration.

Encoder

A default ECS encoder is available: **co.elastic.logging.logback.EcsEncoder**. The encoder allows setting constant fields in the MDC, defining the **application** and **environment** via **logback.xml**.

Example: `.logback.xml`

```
<appender name="DEFAULT_AUDIT_FILE" class="ch.qos.logback.core.FileAppender">
  <filter class="org.eclipse.mdm.logging.LoggingMarkerFilter">
    <markerName>audit</markerName>
  </filter>
  <file>${com.sun.aas.instanceRoot}/logs/mdm5.json</file>
  <append>true</append>
  <encoder class="co.elastic.logging.logback.EcsEncoder">
    <!-- Predefined ECS fields -->
    <serviceName>Service</serviceName>
    <serviceVersion>1.0.0</serviceVersion>
    <serviceEnvironment>development</serviceEnvironment>
    <serviceName>Node</serviceName>
    <includeMarkers>true</includeMarkers>
    <!-- Custom fields -->
    <additionalField>
      <key>some.custom.value</key>
      <value>12345</value>
    </additionalField>
  </encoder>
</appender>
```

However, it is also possible to use custom encoder implementations.