Software Dokumentation

eoda GmbH

Version 2.6.0, 2022-08-16

Inhalt

Einleitung

Diese Dokumentation dient als Nachschlagewerk und enthält Anweisungen zur Installation, Konfiguration und Bedienung der Software YUNA für die Zielgruppen:

- Fachanwender
- Data Scientist
- Dashboard Developer
- Systemadministrator

YUNA ist eine kollaborative Data-Science-Plattform und bietet die passenden Werkzeuge für den produktiven Einsatz von Datenanalysen. Sie deckt alle relevanten integrativen, Daten verarbeitenden und kollaborativen Funktionen zur Abbildung solcher Einsatzszenarien ab. Im Ergebnis reduziert YUNA so die Total-Cost-of-Ownership und sorgt für kurze time-to-market.

So ermöglicht die Plattform die Entwicklung von „Data Products“ durch:

- Komfort-Funktionen zur unternehmensweiten Zusammenarbeit
- Governance-Funktionen zur Steuerung und Überwachung
- Struktur-Funktionen zur Anpassung, Optimierung und Bewertung von Analysen im produktiven Einsatz
- Kern-Funktionen zur Steigerung der Datenqualität
- Grundlagen-Funktionen zur Entwicklung von selbstlernenden Algorithmen
- Eine skalierbare Architektur zur Unterstützung vom Prototyp zum Mission Critical Service
- Eine offene API zur Integration in die Systemlandschaft

Wer sind Data Scientists?

Sie sind diejenigen, die alles erst ermöglichen. Aus der konkreten Anforderung heraus, entwickeln Sie die Operationen oder Algorithmen um diese zu erfüllen und zu beantworten. Oder aber sie betreiben wertvolles Data Cleaning für bessere Datenqualität.

YUNA liefert eine passende Umgebung um Skripte und Algorithmen Ihrem eigentlichen Ziel zuzuführen. Unterstützt werden sie von nachvollziehbaren Prozessen und der Anpassungsmöglichkeiten.

Wer sind Fachanwender?

Sie wollen Antworten auf wichtige Fragen finden, bilden Hypothesen oder reagieren auf die Ergebnisse – und am besten in EINER Umgebung um Zeit zu sparen, statt dutzender Insel-Lösungen.

Von der Fragestellung über zielgerichtete Workflows und bis hin zur Ergebnispräsentation und der Möglichkeit – YUNA liefert alles aus einem Guss und lässt bestehende Operationen auf neue Fälle anwenden.

Wer sind Report-Empfänger?

Die Priorität liegt in der wirtschaftlichen Gesundheit des Unternehmens. Sie wägen stets Kosten

und Nutzen gegeneinander ab. Analytisch aufbereitete Ergebnisse sind Handwerkszeug – ohne geht es nicht. Jetzt müssen Sie nur noch visuell aufbereitet werden.

YUNA paart belastbare und nachvollziehbare Ergebnisse bis zur einzelnen Datenquelle mit logischer Visualisierung.

Wer sind Dashboard Developer?

Sie erstellen Dashboards und arrangieren Widgets und Daten für die anderen Anwender und sind daher in stetem Austausch mit Fachanwender, Report-Empfängern und Data Scientists. Ein tiefes Verständnis für den Aufbau von YUNA und die eigenen Datenstrukturen ermöglicht es dem Dashboard Developer, den Nutzen von YUNA kontinuierlich zu steigern .

Wer sind Systemadministratoren für YUNA?

Eine komplexe Software wie YUNA erfordert immer mal wieder, insbesondere nach Updates, das Konfigurationen auf der Systemebene angepasst werden müssen. Systemadministratoren haben Zugänge zu Datenbanken und Systemschnittstellen und benötigen keine tiefgehenden Kenntnisse im Dashboarding oder auf der Fachebene der Dateninhalte.

Symbolerklärung

Innerhalb dieser Dokumentation werden Symbole für konkrete Handlungsschritte, Funktionen, Informationen und Warnungen verwendet.

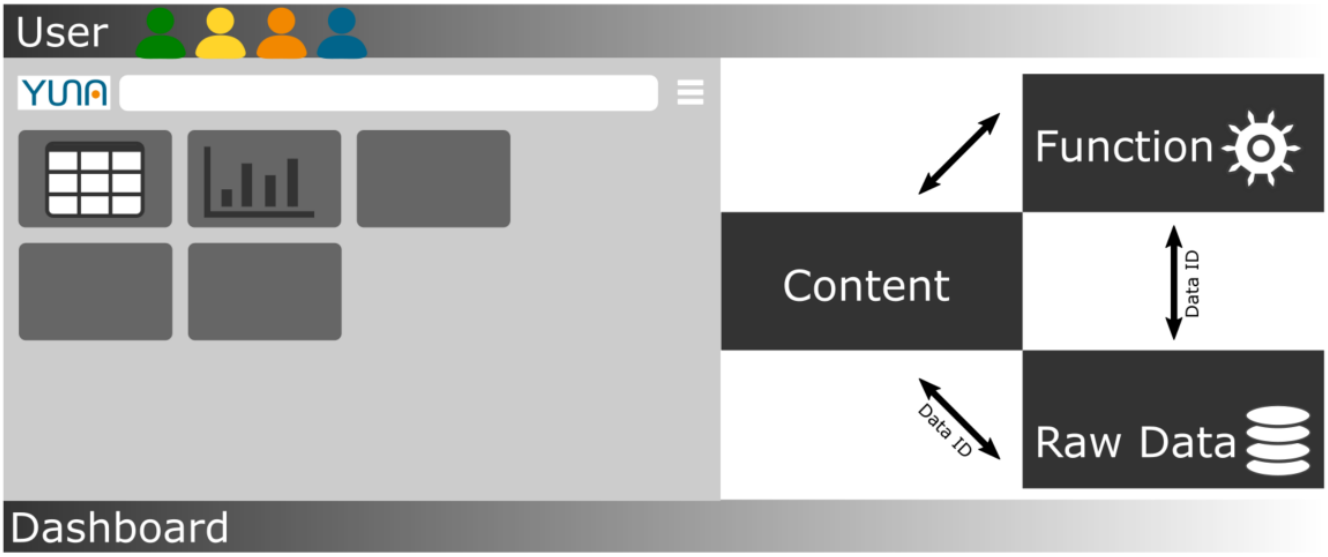
Überblick der verwendeten Symbole

Funktion	**Interagieren**	**Information**
Hier wird eine Funktion oder Konfiguration beschrieben	Hier muss ein Benutzer klicken bzw. tippen	Dieses Symbol weist auf zusätzliche Informationen hin
Warnung	**Ergebnis**	**Yuna ML**
Dieses Symbol weist auf eine Gefahr hin. Nichtbeachten kann zu Fehlern bis hin zu Systemausfall führen	An dieser Stelle finden sie das Ergebnis einer Handlungsanweisung	Hier steht YUNAML Code für die Implementierung oder Konfiguration von Dashboard Inhalten

1. Konzepte

1.1. Dashboards und Daten

Das YUNA Portal stellt für seine Benutzer Inhalte dar und bietet Interaktionsmöglichkeiten. Die dazu notwendigen und ohne Programmierung konfigurierbaren Elemente werden auf dem **Dashboard** abgebildet. Neben diesen Inhalten nutzt das Portal vor allem (Roh-) **Daten** und **Funktionen** als Konzepte, um in der Summe eine flexible Benutzerschicht darzustellen.



1.1.1. Dashboard Inhalte

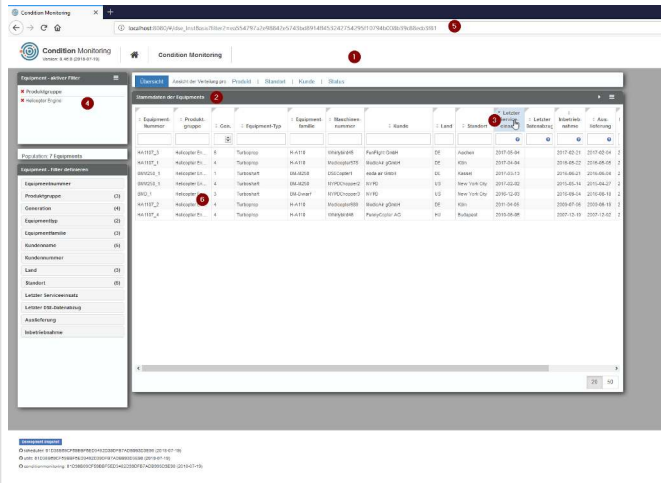
Der Inhalt wird per YUNAML, einer Auszeichnungssprache in Anlehnung an XML bereit gestellt. Nutzer mit der Rolle "Dashboard Developer" haben verschiedene Möglichkeiten, Inhalte zu erstellen:

1. Erstellen von Sichten für einzelne Benutzer-Rollen,
2. Verwalten verschiedener Versionen,
3. Verbinden von Informationen über Rollen, Sichten und Funktionalitäten.

Je Sicht ("Portal View") kann der Inhalt beispielsweise über Widgets repräsentiert werden, die ihrerseits Daten anzeigen (z.B. Standortinformationen, Kundeninformationen, Sensordaten oder Analyseergebnisse). Sowohl die Widgets selbst als auch die darin dargestellten Daten und weitere Funktionen können rollen- oder nutzerspezifisch beschränkt werden. So kann z.B. eine komplette Sicht, nur eine Tabelle dieser Sicht oder nur die Filterbarkeit dieser Tabelle lediglich bestimmten Usern zur Verfügung gestellt werden. Die Darstellung und Steuerung von Inhalt erfolgt über Widgets, Filter und DataIDs.

Beispiele für Inhalte des Dashboards sind u.a.:

1. Sichten (Portal Views)
2. Widgets
3. Widget-Eigenschaften wie "sortierbar" oder "filterbar"



1. Eine View mit 4 Widgets
2. Ein Tabellen-Widget
3. Das Tabellen-Widget hat Eigenschaften, so ist die Tabelle beispielsweise sortierbar.
4. Filter, der die angezeigten Rohdaten bestimmt
5. Die View lässt sich mit dem DeepLink anderen Benutzen mit derselben Berechtigung zur Verfügung stellen
6. Die angezeigten Rohdaten

1.1.2. Wichtige Kernfunktionen

Wesentliche Funktionen und Konzepte des Portals neben Sachverhalts-Workflows und dem Benutzer- und Rollenkonzept sind:

Filter

Mit dem Filter kann ein Benutzer eine Teilmenge einer Datenquelle selektieren, diese speichern und anderen Benutzern über die Weitergabe einer URL zur Nutzung zur Verfügung stellen. Es stehen verschiedene Filtertypen zur Auswahl, die folgender Hierarchie unterliegen: Primärfilter, Sekundärfilter, Subfilter, (lokale) Tabellenfilter. (Siehe dort)

DeepLinks

DeepLinks ermöglichen den Austausch von individuellen Views per Direktlink. Dabei enthält die URL des Links alle notwendigen Informationen wie z.B. Filtereinstellungen, um den Inhalt einer View immer bei jedem Benutzer gleich darstellen zu können, die entsprechenden Rechte vorausgesetzt. (Siehe dort)

DataID

Eine DataID referenziert eindeutig auf Daten aus der Datenbank, aber auch Datenbank-Abfragen, Bilder oder Skripte.

1.1.3. (Roh-)daten

Unter Rohdaten werden Stamm- oder Bewegungsdaten aller Art verstanden, wie zum Beispiel Daten über Maschinen oder Sensordaten. Diese können direkt in Widgets angezeigt werden. Auf die Rohdaten kann auch per Analyse-Skript zugegriffen werden.

1.2. Sharing und Deeplinks

Jede Seite des YUNA Portals lässt sich durch die Deeplink-Funktionalität über einen Link erreichen und teilen - mit Einschränkungen, die durch die Rollen des versendenden und empfangenden Benutzers definiert sind.

Filter (bis auf Spaltenfilter innerhalb einer Tabelle) und Einstellungen einer Portal View, werden direkt in der URL abgebildet und können so entsprechend gespeichert und mit anderen Nutzern

der DSP-Instanz geteilt werden. Die Basis hierfür ist die konsequente Verwendung von Parametern, die an die URL einer Portal View angehängt werden.

Die URL folgt der Syntax: `http://Server/portal/#/viewName?localParam1=abc?localParam2=xyz`

Filterparameter, die sich in einer Page-View-URL niederschlagen, können zum Beispiel Subfilter wie die MaschinenListe (*?Maschine=M12345*) oder ein ausgewählter Sensor (*?sensor=Helligkeitssensor*) sein.

> *Links sind an Rollenberechtigungen gebunden*
>
> Für alle Links gilt: Verfügt der Anwender des Links nicht über die gleichen Rechte wie der Ersteller des Links, wird die View nicht bzw. nur teilweise angezeigt.

1.3. Filtern und Begrenzen

Mit einem Filter lässt sich aus der gesamten Datenbasis ein Subset der für die jeweilige Aufgabenstellung relevanten Daten anzeigen, (teilweise) speichern und über Deeplinks teilen. Es stehen vier Arten von Filtern zur Verfügung, die hierarchisch aufeinander aufbauen:

1. Primärfilter,
2. Sekundärfilter,
3. Subfilter und
4. Tabellenfilter (wirkt nur auf einer Tabelle).

Die Datengrundlage, auf die die Filter wirken, ist vom Kunden definierbar. Dies können beispielsweise die Stammdaten der Anlagen bzw. Equipments, die in der jeweiligen Instanz des YUNA Portals zum Monitoring und zu Analysezwecken eingebunden sind, sein. Die Filterung wirkt sich direkt auf die Anzeige von Daten in abhängigen Widgets (z.B. Charts oder Tabellen) aus.

> Weitere Informationen zu den einzelnen Filterfunktionen, den Filter-Typen und zusätzlichen Eigenschaften von Filtern finden Sie in den Dokumentationsbereichen der Filterfunktionen und der Widget-Typen in den Dokumentationen für User und Dashboard Developer.

1.4. Lokalisierung und Internationalisierung

1.4.1. Lokalisierung

Im Rahmen von YUNA werden **drei Bereiche** unterschieden, denen sich übersetzbare Inhalte zuordnen lassen:

1. **Portal**: Die Standardoberfläche und die allgemeinen Bedienelemente
2. **Dashboard-Inhalte**: Die über YUNAML definierten Inhalte
3. **Datenbankinhalte**: Die dargestellten Datenbankinhalte

1.4.2. Internationalisierung

YUNA kann mit allen Zeichensätzen umgehen, die in UTF-8 dargestellbas sind. Bisher ist es nicht möglich, Zeichensätze zu verwenden, die von rechts nach links geschrieben werden (z.B. Arabisch).

2. Systemvoraussetzungen

2.1. Hardwarevoraussetzungen

2.1.1. Hardwarevoraussetzungen für YUNA Server

- CPU: Mind. 2-4 Prozessoren bzw. Kerne
- RAM: Mind. 12GB
- HDD: Mind. 20GB freier Speicher

2.1.2. Hardwarevoraussetzungen für YUNA Agenten:

- CPU: Mind. 2-4 Prozessoren bzw. Kerne(empfohlen Intel® Xeon® CPU E5-2680 2.70GHz oder vergleichbare)
- RAM: Dies ist stark abhängig von den Analysen, empfohlen >= 64GB
- HDD: Mind. 20GB freier Speicher

2.1.3. Systemvoraussetzungen für YUNA Clienten:

- CPU: Mind. 2-4 Prozessoren bzw. Kerne (empfohlen Intel® Core™ i5-6500 oder neuer)
- RAM: 8 GB (empfohlen 16GB)
- Browser: Google Chrome

2.2. Softwarevoraussetzungen

2.2.1. Red Hat Enterprise / CentOS

Systemvoraussetzungen für YUNA Server:

- Red Hat Enterprise Linux 7 oder CentOS 7 mit systemd
- Apache Webserver (httpd 2.4), alternativ Nginx oder Ähnliche
- Java Runtime Environment (java 11) [OpenJDK JRE oder Oracle JRE]

Systemvoraussetzungen für YUNA Agenten (R-Ausführung):

- Red Hat Enterprise Linux 7 oder CentOS 7 mit systemd
- Java Runtime Environment (java 11) [OpenJDK JRE oder Oracle JRE]
- R ab Version 3.4.1
- R-Java ab Version 1.2
- R Paket *getPass*
- dseconnect (Version mitgeliefert bei DSP)
- optional Python 3 (für Python Agent)

2.2.2. Weitere Softwarevoraussetzungen:

- Datenbank: Microsoft SQL Server ab Version 2016
- Authentifizierung: Active Directory via LDAP Schnittstelle

3. Installation

3.1. YUNA auf Linux-Systemen installieren

Die folgenden Kapitel dieser Anleitung zeigen die notwendigen Schritte zur Installation von YUNA auf.

3.1.1. Einrichtung Red Hat Enterprise / CentOS-System

Dieser Abschnitt beschreibt die Schritte zur Installation des Portals auf einem Red Hat Enterprise / CentOS-System.

Voraussetzungen installieren

Tätigen Sie die Installation als angemeldeter *root* Benutzer oder verwenden Sie das **sudo** Kommando.

```
sudo <Kommando>
```

Beispiel für den sudo-Befehl zur Installation von Python

```
sudo yum install python
```

Python 2.7

Python 2.7 sollte standardmäßig installiert sein. Wenn dies nicht der fall ist, dann muss Python installiert werden.

```
yum install python
```

Python Paketmanager pip

Der Paketmanager *pip* steht in den standard Paketrepositories von CentOS und Red Hat Enterprise Linux nicht zur Verfügung. Dafür ist die Einbindung weiterer Paketrepositories (z.B. EPEL) notwendig, oder eine manuelle Installation.

Für CentOS

CentOS: pip Installation

1. Schritt 1: Installation des EPEL Repository für den Python Paketmanager *pip*

   ```
   yum install epel-release
   ```

2. Schritt 2: Installation des pip Paketmanager

```
yum install python2-pip
```

Für Red Hat Enterprise Linux

Red Hat Enterprise Linux: pip Installation

zur Installation des pip Paketmanagers

Das EPEL Repository steht im Red Hat Enterprise Linux Paketrepository nicht zur Verfügung.

Installieren Sie das *EPEL Repository* und den *pip Paketmanager* manuell. Weitere Informationen finden Sie hier: https://fedoraproject.org/wiki/EPEL https://packaging.python.org/guides/installing-using-linux-tools/#centos-rhel

Zur Installation des *EPEL Repository* kann dieser Befehl verwendet werden.

```
rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

Für die Installation des *EPEL Repository* sind folgende Repositories vorher zu aktivieren:

1. rhel-7-server-optional-rpms
2. rhel-7-server-extras-rpms

Sofern das *EPEL Repository* verfügbar ist kann *pip* installiert werden:

Installation des pip Paketmanager

```
yum install python2-pip
```

Python Module

Installation des untangle Python-Modules

```
pip install untangle PyYAML
```

Apache httpd (optional)

Diese Abhängigkeit wird bei der Installation von YUNA über den Paketmanager *yum* automatisch installiert.

Installation des Apache httpd

```
yum install httpd
```

Java Runtime Environment (optional)

Diese Abhängigkeit wird bei der Installation von YUNA über den Paketmanager *yum* automatisch installiert.

- Es kann das OpenJDK JRE oder das Oracle JRE verwendet werden.
- Für die Installation des Oracle JRE müssen zusätzliche Repository eingebunden werden. Weitere Informationen finden Sie hier: https://access.redhat.com/solutions/732883

Installation des Java Runtime Environment

```
yum install java-11-openjdk
```

RPM Instalation

1. Schritt 1: Download eoda-dsp-<version>.x86_64.rpm und eoda-dsp-frontend-<version>.x86_64.rpm

```
wget http://<Pfad einfügen>/eoda-dse-portal-<version>.x86_64.rpm
wget http://<Pfad einfügen>/eoda-dse-frontend-<version>.noarch.rpm
```

2. Schritt 2: Installation der YUNA Pakete

```
yum install eoda-dse-portal-<version>.x86_64.rpm
yum install eoda-dse-portal-frontend <version>.noarch.rpm
```

3.1.2. Einrichtung der Datenbank

Diese Befehle können im Microsoft SQL Server Management Studio oder über Microsoft SQLCMD ausgeführt werden.

1. **Erstellen einer Datenbank auf einem Microsoft SQLServer für YUNA**

```
CREATE DATABASE [DSE-PortalDB];
GO
```

2. **Benutzeraccount für Datenbank anlegen**

```
CREATE LOGIN dseuser WITH PASSWORD = 'example';
GO
```


Ersetzen Sie im obigen Befehl das Passwort **example** durch ein sicheres Passwort Ihrer Wahl!

3. **Benutzerrechte für Benutzeraccount auf der YUNA Datenbank vergeben**

```
USE [DSE-PortalDB];
GO
CREATE USER dseuser FOR LOGIN dseuser
GO
EXEC sp_addrolemember 'db_owner', 'dseuser'
GO
```

4. **Datenbank initialisieren**

Die Initialisierung wird durch *liquibase* ausgeführt.Die notwendigen Dateien finden Sie unter **/opt/eoda/dse/portal/db-versions/**

```
/opt/eoda/dse/portal/db-versions/liquibase \
  --driver=com.microsoft.sqlserver.jdbc.SQLServerDriver \
  --classpath='/opt/eoda/dse/portal/db-versions/sqljdbc4-4.0.jar' \
  --url='jdbc:sqlserver://<db server>;databaseName=DSE-PortalDB' \
  --username='<username>' --password='<password>' \
  --changeLogFile=xml/db.master.xml \
  update
```

Liquibase kann auch Flags aus einer Properties Datei einlesen.

liquibase.properties

```
driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
classpath=sqljdbc4-4.0.jar
url=jdbc:sqlserver://<db server>;databaseName=DSE-PortalDB
changeLogFile=xml/db.master.xml
defaultSchemaName=dbo
```

Der Liquibase Befehl muss nun nurnoch mit den fehlenden Flags gestartet werden.

Liquibase mit Auslagerung von Flags in liquibase.properties

```
/opt/eoda/dsp/db-versions/liquibase \
  --username='<username>' --password='<password>' \
  update
```

Die Flags username und password können auch wie die anderen Einträge ausgelagert werden.

Bei der Initialisierung wird das Datenbankschema für den Betrieb von YUNA angelegt. Zusätzlich werden für den Betrieb notwendige Informationen in der Datenbank abgelegt.

Zum Anzeigen der SQL Befehle kann das obige Kommando **update** durch

updateSQL ersetzt werden.

5. **Portal-Admin Benutzer anlegen**

   Initial muss ein Administrator Benutzer in der Datenbank angelegt werden. Zum erstellen der Passworthashes siehe Abschnitt Benutzerauthentifizierung

```
INSERT INTO [core].[user] (username, firstname, lastname, email, status, [type], password)
VALUES ('admin', 'firstname', 'lastname', 'first.last@example.com', 'ACTIVE', 'internal', PASSWORDHASH);

INSERT INTO [core].[userroles] (user_id, role_id)
  SELECT b.[id] user_id, a.[id] role_id
  FROM [core].[role] a, [core].[user] b
  WHERE a.[name] = 'System_Admin' AND b.[username] = 'admin';
```

6. **Benutzer anlegen**

   Zusätzlich müssen in der Datenbank Benutzer angelegt werden, die auf YUNA zugreifen dürfen. Diese werden über das ActiveDirectory authorisiert, freigeschaltet werden sie über die Datenbank. Dazu müssen die Benutzernamen in die Tabelle **user** der YUNA Datenbank eingetragen werden. Desweiteren benötigt der Benutzer eine Rolle, dazu muss in der Tabelle ***userroles*** die BenutzerID ***user_id*** und die zugehörige RollenID ***role_id*** eingetragen werden. Zur Unterstützung wird folgende Stored Procedure bereitgestellt:

   Stored Procedure zum Anlegen neuer Portal-User

```
--============================================================
-- DECLARATION
--============================================================
CREATE PROCEDURE [sp].[sp_createDseUser] @userName varchar(200), @email varchar(200), @role bigint, @firstname
varchar(200) = null, @lastname varchar(200) = null
AS
BEGIN
    BEGIN
      DECLARE @uid bigint = -1;
      BEGIN TRY
          IF NOT EXISTS (SELECT ID FROM [core].[user] WHERE ISNULL(username, '') = @userName)
          BEGIN
              INSERT INTO [core].[user] (username, firstname, lastname, email, status, [type], password)
              VALUES (@userName, @firstname, @lastname, @email, 'ACTIVE', 'ldap', null);
              SELECT @uid = ID FROM [core].[user] WHERE ISNULL(Username, '') = @userName;
              INSERT INTO [core].[userroles] ([user_id], [role_id]) VALUES (@uid, @role);
          END
          ELSE
          BEGIN
              print 'User already exists for login: ' + @userName;
          END
      END TRY
      BEGIN CATCH
          PRINT 'Unable to insert user for login: ' + @userName;
          SELECT ERROR_MESSAGE() AS ErrorMessage;
          THROW;
      END CATCH
    END
END
```

```
--=========================================================
--Stored procedure ausführen
--=========================================================
EXEC [sp].[sp_createDseUser] @userName = <USERNAME>, @email = <EMAIL>, @role = <ROLEID>, @firstname = <FIRSTNAME>,
@lastname = <LASTNAME>
```


Rollen werden seperat definiert und können den Nutzern zugeordnet werden.

3.1.3. Notwendige Konfiguration des Portals nach der Installation

Bevor der YUNA Backend Service ordnungsgemäß starten kann muss eine Konfiguration für den Service erstellt werden. Zusätzlich muss ein Apache proxy konfiguriert werden.

Die Konfigurationsdateien für YUNA werden unter */opt/eoda/dse/portal/configuration/* abgelegt. In diesem Verzeichnis liegen Beispielkonfigurationen mit der Dateiendung *.example*.

Zum Betrieb erforderlich sind die Dateien *dse.conf.xml* und *config.yaml*.

Datenbank

Die Konfiguration der Datenbankverbindungen sind essentiell für den Betrieb von YUNA. Um diese zu Konfigurieren, müssen die unter Verbindung zur Datenbank, Datenbankverbindung - Core und Datenbank Objekte aufgeführten Einstellungen vorgenommen werden.

Verbindung zum Datenbankserver definieren. (dse.conf.xml)

```
<dbservice id="default">
    <url>jdbc:sqlserver://mssql.example.com;applicationName=DSP-Default</url>
    <driver>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver>
    <name>DSE-PortalDB</name>
    <user>{dbuser}</user>
    <password>{dbpassword}</password>
    <initialSize>20</initialSize>
    <maxTotal>100</maxTotal>
    <type>mssql</type>
</dbservice>
```

Alternativ zur Hinterlegung des Passworts im Klartext kann es Base64-kodiert definiert werden. Damit das Passwort anschließend korrekt aufgelöst wird, muss es zusätzlich mit dem Prefix '$dse1$' versehen werden.

Das Passwort ‘12345’ würde dann als ‘$dse1$MTIzNDUK’ in die Konfiguration eingetragen werden.

Dieses Vorgehen dient ausschließlich dazu das Passwort zu verschleiern. Es ist nicht mit einer Verschlüsselung zu verwechseln und stellt keinen Zugewinn an Sicherheit dar.

Verbindung zum Datenbankserver definieren. (config.yaml)

```
org.ops4j.datasource: [
  # WARNING: The database configuration (server, database name, credentials) must always be the same as in dse.conf.xml
  {
   # Currently fully supported database is MS SQL-Server
   osgi.jdbc.driver.class: "com.microsoft.sqlserver.jdbc.SQLServerDriver",
   # server connection configuration
   serverName: "mssql.hostname.example",
   portNumber: "1433",
   databaseName: "DSE-PortalDB",
   dataSourceName: "systemdb",
   # database credentials
   user: "DatabaseUserName",
   password: "SuperSecretDatabasePassword",
   # database connection pool settings, defaults should be fine for most use cases
   pool: "dbcp2",
   jdbc.pool.maxTotal: "10",
   jdbc.pool.maxWaitMillis: "1000"
 }
]
```

Name der Datenbank im Tag portaldb ersetzen.

```
<dbservice-tablereference>
    <portaldb>DSE-PortalDB</portaldb>
    <default-datadb>DSE-DataDB</default-datadb>
</dbservice-tablereference>
```

Benutzerauthentifizierung

Alle Benutzer von YUNA werden in der Datenbank geführt. Die Authentifizierung kann sowohl durch in der Datenbank hinterlegte Passworte erfolgen als auch über die Anbindung von LDAP.

Gespeicherte Passworte müssen immer durch Hashing und Salting gesichert werden.

Authentifizierung mit in der Datenbank hinterlegten Passwörtern

Erstellen der verschlüsselten Passwörter

Die verschlüsselten Passwörter können mit dem Apache Shiro-Tools-Hasher generiert werden:

1. Download `shiro-tools-hasher.jar (letzter stabiler release)` - Alternative: https://shiro.apache.org/download.html
2. Mit `"java -jar shiro-tools-hasher-{version}-cli.jar -p"` das Programm im Passwort-Modus starten (z.B. `"java -jar shiro-tools-hasher-1.3.2-cli.jar -p"`) und den Anweisungen des Programms folgen `
3. Das gewünschte Passwort eingeben (es wird keine Eingabe angezeigt) und durch erneute Eingabe bestätigen

Die Ausgabe des Programms ist ein Hash in der Form: `$shiro1$SHA-256$500000$akubPQXLSO3vCvWjSxbVsQ==$cZa3OiIv+gB4+tQ3VtY0+g+5l44Kxy2BIyW5tz6JpkQ=`.

Anschließend muss der generierte Hash in die Tabelle `core.user` in der Spalte `password` für den

Benutzer eingetragen werden und der Wert in Spalte `type` auf `'internal'` gesetzt werden.

Danach kann sich der Nutzer mit dem so vergebenen Passwort authentifizieren.

Anbindung von LDAP

Die Benutzerauthentifizierung kann auch über LDAP erfolgen. Dazu benötigt YUNA die Verbindungseinstellungen für das zu verwendende LDAP-Verzeichnis.

Beispielkonfiguration für /opt/eoda/dse/portal/configuration/config.yaml

```
de.eoda.dse.core.ldaprealm.provider.AuthorizingLdapRealmProvider: {
  ldapServer: "ldap://<URL:PORT>",
  ldapUserPostfix: "@domain.example.com",
  ldapSearchBase: "dc=domain,dc=example,dc=com",
  # ldap filter for active directory usage, please change to your ldap schema
  ldapFilter: "(&(objectClass=user)(samAccountName=$ldapUsername))"
}
```

Unter dem Tag *ldapServer* ist **<URL:PORT>** durch die Adresse des LDAP Servers zu ersetzen. Zusätzlich muss die *ldapSearchBase* an die genutzte Verzeichnisstruktur angepasst werden.

Apache und YUNA Frontend Konfiguration

Der Apache HTTP Server wird benötigt um statische Resourcen (HTML-, Bild-, Javascript-Dateien) zur Verfügung zu stellen. Zusätzlich müssen Anfragen des Frontends an das Backend über die HTTP und HTTPS Standardports (80, 443) an den Backend-Service weitergeleitet werden.

Hinweis zu DSE URL-Pfad Änderungen

Für YUNA können individuelle URL-Pfade definiert werden.

Die hier aufgeführte Anleitung stellt eine Beispielkonfiguration dar. Da die Konfiguration abhängig von der eingesetzten Systemumgebung ist (wie z.B. Apache httpd, usw.), kann eoda keine allgemeingültige Installationsanweisung liefern.

Apache HTTP Server konfigurieren

1. **Erstellen einer Apache *httpd* Konfigurationsdatei */etc/httpd/conf.d/eoda-dsp.conf***

   Eine Beispielkonfiguration ist unter _/opt/eoda/dsp/example/apache2.4_eoda-dsp_mod-proxy_example.conf zu finden_

   Weitere Informationen zu Optionen der Apache *httpd* Konfiguration finden Sie hier: https://httpd.apache.org/docs/2.4/

2. **Apache *httpd* neu starten**

   Neustart mit dem Befehl:

   ```
   systemctl restart httpd
   ```

Bei dem Einsatz von SElinux auf Ihrem System ist es notwendig die Konfiguration für den Apache *httpd*anzupassen. Apache *httpd* wird als Reverseproxy verwendet und muss zu diesem Zweck Netzwerkverbindungen aufbauen können. Nutzen Sie dafür den Befehl:

```
/usr/sbin/setsebool -P httpd_can_network_connect 1
```

Die Standard-URL für das YUNA-Frontend ist `http://<servername>/` (z.B. `http://dsp.example.com/`). Das Frontend kommuniziert mit dem Backend standardmäßig über die URL `http://<servername>/backend/ (z.B. http://dsp.example.com/backend/`). Soll YUNA unter anderen URL-Pfaden verwendet werden, muss die Apache httpd Konfiguration angepasst und das Frontend konfiguriert werden.

Frontend konfigurieren

Bei der Beispielkonfiguration wird das Frontend vom Apache *httpd* über die URL `http://<servername>/` (z.B. `http://dsp.example.com/`) bereitgestellt. Das Frontend kommuniziert mit dem Backend über die URL `http://<servername>/backend/` (z.B. `http://dsp.example.com/backend/`). Wurden diese Pfade in der Apache *httpd*Konfiguration verändert muss das Frontend dafür konfiguriert werden. Für die Änderung des Backend Pfad im Frontend muss in der Datei */opt/eoda/dsp-frontend/config/env.json* die Option **"baseUrl": "/backend"** auf den Pfad für das Backend gesetzt werden. Dieser Pfad muss dem Pfad der Apache *httpd* Konfiguration in der *ProxyPass* Direktive entsprechen.

YUNA service Starten

Starten Sie nun den YUNA Service, sofern Sie die Agenten nicht benötigen. Andernfalls starten Sie den YUNA Service nach der Agenteninstallation.

DSE Service Starten

```
systemctl start eoda-dse-portal
```

Firewall konfigurieren

Falls auf dem Serversystem für YUNA eine Firewall eingesetzt wird müssen folgende Ports für den Zugriff freigeben werden.

Offene Ports YUNA:

Protkoll	Zweck	Port	Vorausstzung
tcp	HTTP DSE Frontend/Backend	80	Ja
tcp	HTTPS DSE Frontend/Backend	443	Nein (bei https Nutzung)
tcp	HTTP DSE Backend	8080	Ja (für Datenzugriff aus R)

3.1.4. R für Agenten installieren und vorbereiten

Eine R Laufzeitumgebung wird benötigt wenn R-Analysen über Agenten ausgeführt werden soll.

Voraussetzungen installieren

Java

Es kann das OpenJDK JRE oder das Oracle JRE verwendet werden.Für die Installation des Oracle JRE müssen zusätzliche Repository eingebunden werden. Weitere Informationen finden Sie hier: https://access.redhat.com/solutions/732883

```
yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel
```

R

Installation von R auf CentOS

1. **Installation des EPEL Repository für R**

   ```
   yum install epel-release
   ```

2. **Installation von R**

   ```
   yum install R
   ```

Installation von R auf Red Hat Enterprise Linux

Das EPEL Repository steht im Red Hat Enterprise Linux Paketrepository nicht zur Verfügung.

Installieren Sie das *EPEL Repository* und den *R* manuell. Weitere Informationen finden Sie hier:https://fedoraproject.org/wiki/EPEL https://cran.r-project.org/bin/linux/redhat/README

Zur Installation des *EPEL Repository* kann dieser Befehl verwendet werden.

```
rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

Für die Installation des *EPEL Repository* sind folgende Repositories vorher zu aktivieren:

1. rhel-7-server-optional-rpms
2. rhel-7-server-extras-rpms

Installation von R

```
yum install R
```

R Pakete

Für den Betrieb von YUNA mit *R* sind einige R-Pakete zwingend notwendig. Diese Zusatzpakete stellen die Kommunikation zwischen *R* und YUNA sicher.

R starten Sie einfach mit dem Befehl *R* auf der Kommandozeile.

```
R
```

Um R zu verlassen nutzen Sie den Befehl

```
quit(save='no')
```

Starten Sie dafür nach der Installation *R* und installieren Sie folgenden Pakete.

getPass

```
install.packages('getPass')
```

rJava

Installation von rJava

```
install.packages('rJava')
```

Führen sie folgende Befehle als **root** Benutzer auf der Linux Shell aus um rJava zu konfigurieren:

```
R CMD javareconf
```

dseconnect Paket installieren

Die Installation von dseconnect wird ausserhalb von *R* durchgeführt. Das dseconnect R-Paket wird mit Yuna ausgeliefert. Sie finden es nach der Installation des *eoda-dse-portal-<version>.x86_64.rpm* Paket auf dem System mit der YUNA Installation unter dem Pfad **/opt/eoda/dse/portal/dseconnect/** zu finden.

Installation von dseonnect

```
R CMD INSTALL /<pfad zu dseconnect>/dseconnect<version>.tar.gz
```

3.1.5. Verbindung zwischen YUNA & Agenten prüfen

Die Verbindung zwischen YUNA und *Agenten* kann zur Fehlerdiagnose über die folgende URL im Browser überprüft werden.

```
http://<dse-servername>/backend/de.eoda.dse.core.agent.rest/
```

Wenn die Verbindung erfolgreich ist erhalten Sie eine dem folgenden Beispiel ähnliche folgende Ausgabe in Ihrem Browser.


```
[
  {
    "id": "a374caea-8a67-4db6-9b27-11cb39f5920a",
    "capabilities": {
      "name": "tlsragent",
      "packages": "cluster;version='2.0.7-1'"
    },
    "remoteUri": "<dse-agent>",
    "scriptLanguage": "r",
    "scriptLanguageVersion": "3.4.4"
  },
  {
    "id": "b6cb9b29-f1cb-42b7-b9de-47634d7bb93a",
    "capabilities": {},
    "remoteUri": "<dse-agent>",
    "scriptLanguage": "echo",
    "scriptLanguageVersion": "0.1.0"
  }
]
```


3.2. Auslieferung und Installation von Updates

3.2.1. Auslieferung

Die Auslieferung der Portalaktualisierungen erfolgt über RPM-Pakete, die in der DMZ (demilitarisierte Zone) von eoda bereitgestellt werden. Dort wird für jeden Kunden ein eigenes Repository gepflegt, in dem sich die kundenspezifischen Installationspakete sowie die Dokumentation befinden.

URL des Repositories: https://git.eoda.de/

Zugangsdaten: Der Benutzername und das zugehörige Passwort sollten Ihnen in einer separaten Mail zugekommen sein.

Sofern Ihnen diese Informationen fehlen oder Probleme vorliegen, wenden Sie sich bitte an den Support.

3.2.2. Installation von Updates

1. **Services stoppen**

```
service eoda-dse-agent stop
```

```
service eoda-dse-portal stop
```

2. **Rpm Pakete installieren**

```
yum install eoda-dse-portal-1.6.1.20190529T1317Z-1.x86_64.rpm
```

```
yum install eoda-dse-portal-frontend-1.6.1.20190529T1316Z-1.noarch.rpm
```

3. **Prüfen ob ein Liquibase update notwendig ist**

```
service eoda-dse-portal status -l
```

Falls ein update notwendig ist

```
/opt/eoda/dse/portal/db-versions/liquibase --classpath="/opt/eoda/dse/portal/db-versions/sqljdbc4-
4.0.jar:/opt/eoda/dse/portal/db-versions" --driver="com.microsoft.sqlserver.jdbc.SQLServerDriver" --url
="jdbc:sqlserver://localhost;applicationName=DSP-Default;databaseName=CM-PortalDB" --username="IHR BENUTZERNAME"
--password='IHR PASSWORT' --changeLogFile='xml/db.master.xml' update
```

4. **dseconnect updaten**

```
R CMD INSTALL /opt/eoda/dse/agent/spconnect/dseconnect_<release-version>.tar.gz
```

5. **connectivity Pakete updaten**

Aktivieren sie – sofern nicht bereits geschehen – die forcierte Installation der connectivity Pakete (Agentenkonfiguration) beim Start des Agenten.

6. **Portal Service starten**

```
service eoda-dse-portal start
```

7. **Agent updaten**

```
yum install eoda-dse-agent-1.6.1.20190529T1318Z-1.x86_64.rpm
```

8. **Agent starten**

```
service eoda-dse-agent start
```


Optional: Installation des R-Connectivity Pakets in anderen globalen Bibliotheken

Nach dem erfolgreichen Start des Agenten finden sie das aktuelle R-Connectivity Paket als Source im Verzeichnis /opt/eoda/dse/agent/connectivity/R. Für Entwicklungszwecke können sie dieses gegebenenfalls manuell in anderen R-Bibliotheken installieren.

9. **Prüfen ob Portal und Agent laufen**

```
service-eoda-dse-portal status
```

10. **Log Files überprüfen**

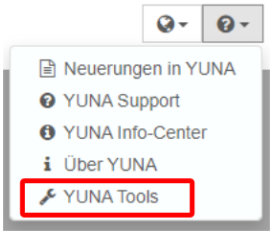
```
tail -f /opt/eoda/dse/portal/dselog.log
```


Aktuelle dsedep Version

Die aktuelle Version des Dashboard-Deployment-Tools (dsedep) finden Sie nach Installation des Frontend RPM-Pakets (z.B eoda-dse-portal-frontend-1.6.1.20190529T1316Z-1.noarch.rpm) im Verzeichnis :

/opt/eoda/dse/portal/tools/

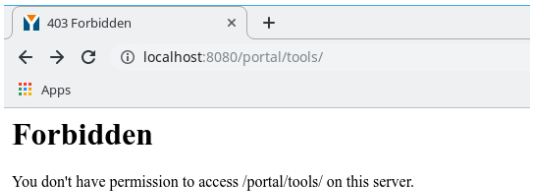
3.2.3. Manueller Download



Klicken Sie auf den Infobutton im Portal und dann auf YUNA Tools, um dseconnect und dsedep herunterzuladen.

Apache Konfiguration für den Tools Verzeichnioszugriff

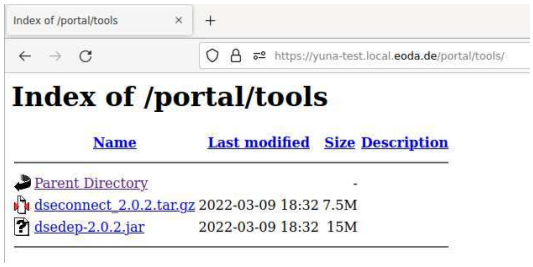
Aufruf der URL des Tool-Ordners ohne aktiviertes Directory-Listing:



Beispiel für Einrichtung des Directory-Listings in der Apache-Konfigurationsdatei im Ordner `/etc/httpd/conf.d/` zur Anzeige des Toolverzeichnisses im Browser (Diese Konfiguration ermöglicht den Zugriff Aller auf das Verzeichnis):

```
<Directory /opt/eoda/dse/portal-frontend/tools/>
    Options Indexes
    Require all granted
</Directory>
```

Beispiel für die Anzeige im Browser bei erfolgreicher Konfiguration:



Aussehen und Formatierung des Listings werden ebenfalls im Apache konfiguriert!

Weitere Infos unter: https://httpd.apache.org/docs/2.4/en/mod/core.html#directory

4. Konfiguration

4.1. Konfiguration von YUNA

4.1.1. Mit Version 1.0 entfallen folgende Einträge in der *dse.conf.xml*:

Alle Einstellungen unter dem Element *user*:

```
<!-- User database connection -->
<user>
    <database>default</database>
</user>
```

Alle Einträge unter dem Element *ldapsecurityservice* (Siehe LDAP-Konfiguration):

```
<ldapsecurityservice>
    <devLocal>i_know_its_a_test_flag_and_I_dont_use_it_in_production</devLocal>
    <ldapServer>ldap://172.20.193.10:389</ldapServer>
    <ldapSearchBase>CN=Users,dc=local,dc=eoda,dc=de</ldapSearchBase>
    <ldapUserPostfix>@local.eoda.de</ldapUserPostfix>
</ldapsecurityservice>
```

Alle Einträge unter dem Element *localdata*:

```
<localdata>
    <ld.errorpatterns>/var/errorpatterns/</ld.errorpatterns>
</localdata>
```

Alle Einträge unter dem Element *rservice*:

```
<rserveservice>
    <host>localhost</host>
    <port>6311</port>
    <user></user>
    <password></password>
</rserveservice>
```

Alle Einträge unter dem Element *token*:

```
<token>
    <usercreation>eodaIT</usercreation>
    <sqlrequest>Administrators</sqlrequest>
</token>
```

Alle Einträge unter dem Element *scheduler*:

```
<scheduler>
    <disabledays>6, 7</disabledays>
</scheduler>
```

Alle Einträge unter dem Element *local*:

```
<local>
    <default>en_US</default>
<local>
```

4.1.2. Mit 1.2.0 ändern sich einige Einstellungen in der RAAS Konfiguration

Vorher:

```
<raas>
    <version>1.0</version>
    <vendor>eoda GmbH</vendor>
    <vendor_url>https://www.eoda.de</vendor_url>
    <localdata>/var/raasimages/</localdata>
    <spconnectversion>45</spconnectversion>
    <spconnect>/opt/virgo/rpackges/dseonnect_0.4.tar.gz</spconnect>
    <!-- die nachfolgenden Konfigurationselemente entfallen-->
    <debuguser>false</debuguser>
    <debuguserlogin>test@test</debuguserlogin>
    <debuguserpassword>YourPassword</debuguserpassword>
    <cron>true</cron>
</raas>
```

Es entfalle n:

- debuguser
- debuguserlogin
- debuguserpassword
- cron

Stattdessen kommt hinzu:

```
<raas>
    ...
    <connectpackage>dseconnect</connectpackage>
</raas>
```

4.1.3. Services (config.yaml)

Es besteht die Möglichkeit die einzelnen Services, welche von Core und Portal genutzt werden zu konfigurieren. Dabei können nicht nur die von eoda definierten Services eingestellt werden, sondern auch die im Core und Portal importierten Services (z.B. 'org.ops4j.datasource'). Die Konfigurationsmöglichkeiten der einzelnen Services werden in den folgenden Unterkapiteln beschrieben.

Datenbankverbindung - Core

Beispiel

```
# General database connection configuration section
# ====
org.ops4j.datasource: [
  # WARNING: The database configuration (server, database name, credentials) must always be the same as in dse.conf.xml
  {
   # Currently fully supported database is MS SQL-Server
   osgi.jdbc.driver.class: "com.microsoft.sqlserver.jdbc.SQLServerDriver",
   # server connection configuration
   serverName: "mssql.hostname.example",
   portNumber: "1433",
   databaseName: "DSE-PortalDB",
   dataSourceName: "systemdb",
   # database credentials
   user: "DatabaseUserName",
   password: "SuperSecretDatabasePassword",
   # database connection pool settings, defaults should be fine for most use cases
   pool: "dbcp2",
   jdbc.pool.maxTotal: "10",
   jdbc.pool.maxWaitMillis: "1000"
 }
]

# Default database setup configuration
#
# Do not change without instructions from support!
# ====
de.eoda.dse.core.db.api.DaoConfiguration: {
  createTables: false
}
```

Dieser Service wird dazu genutzt die Verbindung zur Datenbank, welche vom Core genutzt wird, zu konfigurieren. Die Dokumentation für die einzelnen Optionen ist hier zu finden.

Ist die Option

```
de.eoda.dse.core.db.api.DaoConfiguration: {
  createTables: false
}
```

auf 'false' gesetzt, wird beim Start des Servers keine Datenbank-Tabelle erstellt.

Verschlüsselte Verbindung

Zum Aufbau einer verschlüsselten Verbindung muss in der config.yaml die Verschlüsselung eingeschaltet werden. Dafür müssen zusätzlich die Parameter "encrypt" und "trustServerCertificate" gesetzt werden.

Beispiel:

```
....
encrypt: true,
trustServerCertificate: true,
....
```

```
```

Konfiguration von Wartungsfenstern

Die Konfiguration eines Wartungsfensters bewirkt, dass in dem definierten regelmäßigen Zeiträumen keine Jobs ausgeführt werden.Damit kann beispielsweise erreicht werden, dass jeden ersten Montag im Monat von 3:00 bis 6:00 Uhr keine Jobs ausgeführt werden und währenddessen Wartungsarbeiten problemlos durchgeführt werden können.

Ausführung wird nicht nachgeholt

Jobs die durch das Wartungsfenster nicht ausgeführt werden, werden nicht wiederholt.

Für die Einrichtung eines Wartungsfensters muss die Konfiguration des **JobServiceProvider** erweitert werden. Dort muss der Parameter **maintenanceDayPattern** mit einer oder mehreren CRON expressions konfiguriert werden. Diese CRON expressions müssen mit Quartz kompatibel sein.

Beim Anwendungsstart erfolgt eine Validierung der konfigurierten CRON expressions.Invalide CRON expressions werden ignoriert und es wird eine entsprechende Fehlermeldung in die Log Datei geschrieben.

Zeitzone UTC

Die konfigurierte CRON expression orientiert sich immer an der Zeitzone UTC.

Beispiel für ein Wartungsfenster an jedem Samstag und Sonntag

```
de.eoda.dse.core.job.provider.JobServiceProvider: {
    # Configure how many instances of the same job may run in parallel
    # Instances of different jobs still run in parallel and are not affected by this property
    maxParallelJobExecutionCount: 1
    # Pattern for maintenance
    maintenanceDayPattern: [
       "* * * ? * SAT",
       "* * * ? * SUN"
  ]
}
```

Falls nur eine einzelne CRON expression konfiguriert werden soll, kann dies auch in einer Kurzform erfolgen.

Beispiel für ein Wartungsfenster an jedem ersten Sonntag im Monat

```
de.eoda.dse.core.job.provider.JobServiceProvider: {
    # Configure how many instances of the same job may run in parallel
    # Instances of different jobs still run in parallel and are not affected by this property
    maxParallelJobExecutionCount: 1

    # Pattern for maintenance
    maintenanceDayPattern: "* * * ? * 1#1 *"
}
```

Ein freier Generator für Quartz kompatible CRON expressions findet sich im Web unter https://www.freeformatter.com/cron-expression-generator-quartz.html

LDAP

Der Core bietet die Möglichkeit sich auch über LDAP auf dem Portal anmelden zu können. Dazu muss der dazu passende Service konfiguriert werden.

Beispiel für eine LDAP Konfiguration in der Datei config.yaml

```
# General ldap configuration
# ====
de.eoda.dse.core.ldaprealm.provider.AuthorizingLdapRealmProvider: {
  ldapServer: "ldap://ldap.domain.example.com",
  ldapUserPostfix: "@domain.example.com",
  ldapSearchBase: "dc=domain,dc=example,dc=com",
  # ldap filter for active directory usage, please change to your ldap schema
  ldapFilter: "(&(objectClass=user)(samAccountName=$ldapUsername))",
  # On users first login with ldap credentials the user will be imported from ldap
  # to local user database. The user will be assigned to the role configured in "defaultRole".
  defaultRole: "registered",
  activateImportedUser: false
}
```

Beschreibung der Konfigurationsparameter

Key	Beschreibung	Default
ldapServer	Die Adresse des Authentifizierungsservers	
importMissingUser	Ist der Schalter auf "true" gesetzt, wird ein Benutzer nach erfolgreicher Anmeldung in die Benutzerdatenbank importiert, wenn dieser dort noch nicht existiert und bekommt die Rolle, die unter 'defaultRole' definiert wurde.	false
defaultRole	Legt die Rolle fest, welche den Nutzern zugewiesen wird, welche sich das erste mal über einen LDAP Zugang auf dem Portal anmelden.	registered

Key	Beschreibung	Default
ldapUserPostfix	Der Postfix für LDAP User	
activateImportedUser	Standardmäßig bekommt ein importierter Benutzer den Status 'REGISTERED' zugewiesen. Mit diesem Status können sich die Nutzer noch nicht anmelden. Wenn activateImportedUser auf "true" steht, wird ein importierter Benutzer nach Anmeldung automatisch aktiviert. Mit dieser Option müssen Nutzer, die sich über LDAP das erste mal auf dem Portal anmelden nicht manuell auf 'active' gesetzt werden.	false
ldapFilter	Details hierzu liefern die LDAP Spezifikationen	
ldapSearchBase	Details hierzu liefern die LDAP Spezifikationen	

Parallele Jobausführung

In der config.yaml können Parameter für die parallele Jobausführung eingestellt werden.

Beispiel für die Konfiguration der parallelen Jobausführung

```
de.eoda.dse.core.job.provider.JobServiceProvider: {
  maxParallelJobExecutionCount: 1,
  maxOverallParallelJobExecutionCount: 10,
  maxAllowedStartTimeDelayBeforeDiscard: 60000
}
```

Beschreibung der Konfigurationsparameter

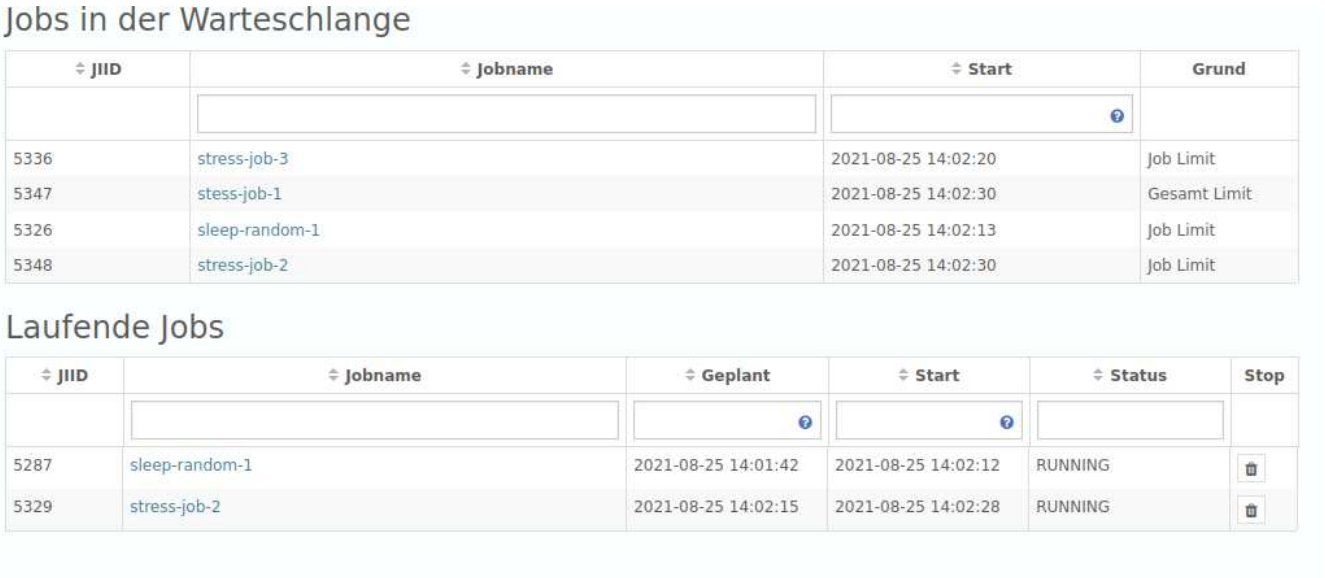
Parameter	Erlaubte Werte	Beschreibung	Default
maxParallelJobExecutionCount	Integer	Die maximale Anzahl der Ausführung von **gleichen** Jobs, die parallel laufen dürfen, alle weiteren kommen in die Warteschlange.	2
maxOverallParallelJobExecutionCount	Integer	Die maximale Anzahl von Jobs insgesamt, die parallel laufen dürfen, alle weiteren kommen in die Warteschlange.	20
maxConcurrentJobInstanceCount	Integer	Die Anzahl der maximal möglichen parallelen Jobs, die generell aufgenommen werden können (Hinweis: im allgemeinen sollte dieser Grenzwert nicht erreicht werden)	100

Parameter	Erlaubte Werte	Beschreibung	Default
maxAllowedStartTimeDelayBeforeDiscard	integer	Zeit in Millisekunden. Job wird verworfen, wenn er um mehr als diese Zeit, sich in der Warteschlange befindet.	60000 (=1 Minute)

In dem Widget "Sysinfo" wird dargestellt, welcher Job sich in der Warteschlange befindet und welcher in Ausführung. Für die Jobs in der Warteschlange ist zusätzlich der Grund angegeben, weswegen ein Job nicht sofort ausgeführt werden konnte:

- **Job Limit** bedeutet die maximale Anzahl an gleichen Jobs ist überschritten,
- **Gesamt Limit** bedeutet die maximale Anzahl an laufenden Jobs ist überschritten.

Jobanzeige im Systeminformations-Widget



Proxy-Konfiguration (Script-Session-Service-Configuration)

Soll der Portal Server über einen Proxy betrieben werden, so muss die Proxy-Adresse im Script-Session Service angegeben werden. Das geschieht wie folgt:

```
de.eoda.dse.core.scriptsession.provider.ScriptSessionServiceProvider: {
  serverRootUri: "https://dse.instance.de/backend"
}
```

Die "serverRootUri" gibt dabei an wie der der Server über einen Proxy zu erreichen ist. Wird der Konfigurationseintrag weg gelassen oder die "serverRootUri" leer gelassen wird auf den Standardpfad (Lokale IP Adresse) zurückgegriffen.

Wird der Proxy über eine verschlüsselte Verbindung betrieben, müssen die Zertifkate für das System gültig sein.

Script Logging

Alle Ausgaben aus einer Scriptsession werden in die Datenbanktabelle core.Scriptlog geschrieben. Die Einträge werden während einer Jobausführung ständig erweitert. Folgende Spalten stehen in der Tabelle zur Verfügung:

Scriptlog Tabelle

Beschreibung des Tabellenschemas der ScriptLog-Tabelle

Spaltenname	Typ	Beschreibung
jobinstance_id	Numerisch	Die Job Instanz ID, die gerade ausgeführt wird
session_id	Text	Session ID aus der ein Log-Eintrag kommt. Hinweis: Es kann sein, dass Fehler oder Einträge geschrieben werden, für die es noch keine Skript-Session gibt.
level	Numerisch	Loglevel: Numerischer Wert zwischen 1: Verbose bis 6: Fatal Error
log	Text	Ein Texteintrag Hinweis: Ein Texteintrag wird in zyklischen Abständen geschrieben und kann mitunter willkürlich gebrochen sein.
number	Numerisch	Ein Sortierungs-Index, wenn ein Text in Chunks gebrochen ist
source	Text	Quelle, die den Log Eintrag auslöst
timestamp	Zeitstempel	Zeitpunkt des Eintrags
qualifier	Numerisch	Typ des Eintrags - siehe Extra Tabelle LogQualifier

LogQualifier

Beschreibung LogQualifier Werte

Quailfier	Datenbank Wert	Beschreibung
TEXT	0	Die Log-Spalte enthält einen allgemeinen Textlog aus der Scriptsession gemäß dem Loglevel
RUNTIME_INFO	1	Die Log-Spalte enthält Informationen zur Scriptlaufzeitumgebung
PACKAGE_LOAD_INFO	2	Ein Paket wurde in der Scriptsession geladen, die Log-Spalte enthält das Paket.
PACKAGE_UNLOAD_INFO	3	Ein Paket wurde in der Scriptsession entladen, die Log-Spalte enthält das Paket.
JOB_START	4	Startzeitpunkt des Jobs (Session id existiert noch nicht)
JOB_FINISH	5	Endzeitpunkt des Jobs (Session id existiert nicht mehr)
JOB_PARAMETER	6	Die Log-Spalte enthält Job Parameter

Quailfier	Datenbank Wert	Beschreibung
JOBINSTANCE_PARAMETER	7	Die Log-Spalte enthält Job Instanz Parameter
JOB_FIELD	8	Die Log-Spalte enthält Job Felder
JOB_SCHEDULED	9	Zeitpunkt zu dem der Job geplant wurde
JOB_CANCELLED	10	Zeitpunkt zu dem der Job abgebrochen wurden
SCRIPT_SESSION_START	20	Zeitpunkt zu dem eine Scriptsession gestartet wurde
SCRIPT_SESSION_STOP	21	Zeitpunkt zu dem eine Scriptsession beendet wurde
NODE_CONTENT_ID	22	In der Log-Spalte wird die Nodecontent id, die in der Scriptsession zur Ausführung kommt, geschrieben

CyclicEvaluationLog Tabelle

Für den vereinfachtenen Zugriff auf die Logs der Skriptausführung steht die Tabelle sp.CyclicEvaluationLog zu Verfügung.

Automatisiertes Löschen älterer Logeinträgen

Um Speicherplatz zu sparen, können die Logeinträge aus dem Scriptlog und dem CyclicEvaluationLog in regelmäßigen Abständen gelöscht werden. Dazu steht ein ActionHandler "scriptlog/cleanup" zur Verfügung, der in der JobServiceKonfiguration wie folgt angewendet werden kann.

```
de.eoda.dse.core.job.provider.JobServiceProvider: {
  actions: [
    "topic 'scriptlog/cleanup' \n
    userName 'admin' \n
    cronExpression '0 0 5 * * ? *' \n
    olderthan 24"
  ]
}
```

Durch den Parameter `cronExpression` wird der Zeitpunkt der zyklischen Ausführung des Löschvorgangs in UTC konfiguriert.

Der Parameter `userName` enthält den Namen des Benutzers, der für die Ausführung verwendet werden soll. Standardmäßig ist das der Benutzer "admin". Alternativ kann entweder ein bestehender Benutzer oder ein speziell angelegter Benutzer verwendet werden.

Der angegebene Benutzer braucht keinen funktionierenden Login und muss auch nicht aktiv sein. Es ist jedoch notwendig, dass ihm eine Rolle mit vollen Berechtigungen zugewiesen wird, z.B. SYSTEM_ADMIN.

Mit dem Parameter `olderthan` wird der Zeitraum in Stunden angegeben, wie lange Texteinträge

(Qualifier: TEXT) in der Scriptlog-Tabelle aufbewahrt werden sollen.

Beispiel-Einträge für cronExpression:

cronExpression	Erklärung
cronExpression '0 0 5 * * ? *'	Täglich um fünf Uhr in der Früh
cronExpression '0 55 * * * ? *'	Stündlich fünf Minuten vor einer vollen Stunde
cronExpression '0 0 5 ? * MON *'	Immer montags fünf Uhr in der Früh

Löschen von Systembenachrichtigungen

Mit jeder Ausführung eines Jobs werden Statusinformationen an den jeweiligen Verantwortlichen versandt. Diese automatisch generierten Benachrichtigungen können über folgende Konfiguration regelmäßig gelöscht werden. Für die Bedeutung der einzelnen Eigenschaften, siehe "Löschen von älteren Logeinträgen

```
de.eoda.dse.core.job.provider.JobServiceProvider: {
  actions: [
    "topic 'message/cleanup' \n
    userName 'admin' \n
    cronExpression '0 0 5 * * ? *' \n
    olderthan 24"
  ]
}
```

Vorfilter-Konfiguration

Das Vorfilter-Feature kann über den Konfigurationseintrag explizit aktiviert bzw. deaktiviert werden. Standardmäßig ist das Feature aktiviert.

Beispiel für die Konfiguration des Vorfilter-Services mit Aktivierung

```
de.eoda.dse.portal.filter.prefilter.provider.PreFilterServiceProvider: {
  active: true
}
```

Mit der Einführung des Vorfilters in **Version 1.5** muss die Konfiguration mindestens den folgenden Eintrag zur Aktivierung des Services enthalten:

```
de.eoda.dse.portal.filter.prefilter.provider.PreFilterServiceProvider: {}
```

Wenn der Eintrag fehlt, startet die Anwendung nicht, da der Service als sicherheitskritisch eingestuft wurde.

Standardwert für das Abbrechen von Datenbankabfragen

Datenbankabfragen, die durch eine DataIO ausgelöst wurden, können standardmäßig abgebrochen werden, sofern diese nicht explicit über das YUNAML-Tag `<cancellable>false</cancellable>` als nicht-Abbrechbar konfiguriert wurden. Dies geschieht automatisch, zum Beispiel beim Verlassen eines Dashboards, die derzeit auf das Ergebnis der Datenbankabfrage wartet.

Sollen Datenbankabfragen standardmäßig nicht abgebrochen werden können, kann dies über den folgenden Eintrag in der Service-Konfiguration (`config.yaml`) definiert werden:

```
de.eoda.dse.portal.cancellation.provider.CancellationServiceProvider: {
  defaultCancelable: false
}
```

Git Repository anbinden

Ein Git-Repository kann angebunden werden um Skripte aus Git in YUNA benutzen zu können. Aktuell kann nur ein einzelnes Git-Repository angebunden werden.

Im Scriptmanager werden alle Dateien aus diesem Repository zur Auswahl bereitgestellt, deren Dateiname auf ".r", ".R" oder ".py" endet.

Servicekonfiguration

Beispiel für die Konfiguration des Git-Service in der config.yaml

```
de.eoda.dse.portal.gitrepo.provider.GitRepoServiceProvider: {
  repositoryStorageDirectory: "/tmp/eoda/yuna/repositories",
  requestTimeout: 180,
  checkoutTimeout: 600
}
```


Es ist üblicherweise nicht notwendig die Standardkonfiguration anzupassen.

Beschreibung der Konfigurationsparameter

Parameter	Typ	Default	Beschreibung
repositoryStorageDirectory	String	"/tmp/eoda/yuna/repositories"	Pfad unter dem das Repository abgelegt wird. Wird automatisch angelegt, falls noch nicht vorhanden.
requestTimeout	Integer	180	Zeit in **Sekunden** nach der eine Anfrage von YUNA an Git abgebrochen wird. Der Wert sollte nicht über 300 liegen.
checkoutTimeout	Integer	600	Zeit in **Sekunden** nach der das Klonen eines Git Repositories abgebrochen wird.

Schritt für Schritt EInrichtung eines Git-Repositories:

Git-Origin über REST anlegen

Bevor Skripte in YUNA aus einem Git-Repository ausgewählt werden können, muss zunächst über die REST-API eine Git-Origin angelegt werden.

Eine Git-Origin ist ein Objekt, das alle nötigen Informationen zur Anbindung eines Git-Repositories an YUNA enhtält.

POST	/backend/de.eoda.dse.portal.gitrepo.rest.origin/
Body	*Json Beispiel* `{ "name": "testRepo", "pvKey": "-----BEGIN RSA PRIVATE KEY-----\nMyPrivateKey\n-----END RSA PRIVATE KEY-----", "repositoryAdress": "git@gitlab.local.eoda.de:DSE/testrepositoryforjgit.git", "branchReference": "refs/heads/myBranchName", "description": "repository to test with" }`
	Das Feld ***branchReference*** muss nicht mit angegeben werden. Fehlt das Feld wird standardmäßig der Branch ***master*** verwendet.
	Der Private-Key muss mit dem **Verfahren RSA** im **PEM Format** generiert werden. Beispiel für die Generierung eines enstsprechenden Schlüsselpaars mit OpenSSH: `ssh-keygen -t rsa -m PEM` Alle Zeilenumbrüche im Private-Key müssen durch `\n` ersetzt werden.
Fehler	**Code:** 400 Die angegebene ID existiert nicht
	Code: 400 Es gab ein Problem mit der Verbindung zum Git-Repository
	Code: 400 ***Malformed input or input contains unmappable characters*** Die Zeichenkodierung des YUNA-Servers ist fehlerhaft. Dies wird durch Sonderzeichen in den Dateinamen im Repository ausgelöst. Das Problem kann wie folgt behoben werden: 1. Angelegte Git-Origin wieder löschen via DELETE-Request 2. Auf dem YUNA Server die korrekte Kodierung sicherstellen: `export LC_CTYPE=de_DE.UTF-8` 3. Git-Origin erneut via POST-Request anlegen
	Code: 500 Es gab ein Problem bei dem Zugriff auf das Dateisystem

Erfolg	**Code:** 200 **Typ:** Application/Json **Inhalt:** gitOrigin Die Git-Origin wurde angelegt

Ein Skript aus Git in YUNA verwenden

Wurde die Git-Origin korrekt angelegt, können über den in Yuna integrirten Skriptmanager auch Scripte aus dem konfigurirten Git-Repository ausgewählt werden. Sollten wider Erwarten keine Skripte angezeigt werden, liegt dies zumeist an einer fehlerhaften Git-Origin.

Es wird in der Regel nur die aktuellste Version des jeweiligen Skripts aus Git angezeigt. Ältere Versionen stehen nur zur Verfügungm wenn diese bereits früher ausgewählt wurden und dann im Repository aktualisiert.

Git Skripte

Aus Git importierte Skripte sind in YUNA nicht editierbar.

Weitere Endpunkte

Abrufen der Git-Origin

GET	/backend/de.eoda.dse.portal.gitrepo.rest.origin/
Erfolg	**Code:** 200 **Typ:** Application/Json **Inhalt:** gitOrigin

Aktualisieren einer Git-Origin

PUT	/backend/de.eoda.dse.portal.gitrepo.rest.origin/{id}
URL-Parameter	id - die ID der Git-Origin die aktualisiert werden soll
Body	*Json Beispiel* `{ "name": "updatedTestRepo", "pvKey": "-----BEGIN RSA PRIVATE KEY-----\nMyPrivateKey\n-----END RSA PRIVATE KEY-----", "repositoryAdress": "git@gitlab.local.eoda.de:DSE/testrepositoryforjgit.git", "branchReference": "refs/heads/myBranchName", "description": "repository to test with" }` Das Feld ***branchReference*** muss nicht mit angegeben werden. Fehlt das Feld wird standardmäßig der Branch ***master*** verwendet.
Erfolg	**Code:** 200 **Typ:** Application/Json **Inhalt:** gitOrigin Die Git-Origin wurde aktualisiert

Fehler	**Code:** 400 Die angegebene ID existiert nicht
	Code: 400 Es gab ein Problem mit der Verbindung zum Git-Repository
	Code: 500 Es gab ein Problem bei dem Zugriff auf das Dateisystem

Löschen einer Git-Origin

DELETE	/backend/de.eoda.dse.portal.gitrepo.rest.origin/{id}
URL-Parameter	id - die ID der Git-Origin die gelöscht werden soll
Erfolg	**Code:** 200 **Typ:** Text/Plain **Inhalt:** Die gelöschte ID Die Git-Origin und das dazugehörige lokale Verzeichnis wurde gelöscht
Fehler	**Code:** 400 Die angegebene ID existiert nicht
	Code: 500 Es gab ein Problem bei dem Zugriff auf das Dateisystem

Remember-Me

Beim Anmelden an YUNA kann die Option "angemeldet bleiben" auch bekannt als `remember-me` ausgewählt werden. Dafür wird ein Remember-Me-Cookie im Browser abgelegt, wodurch der Benutzer automatisch angemeldet bleibt und sich nicht jedes mal erneut anmelden muss.

Die Dauer der Gültigkeit für das angemeldet Bleiben beträgt standardmäßig 30 Tage, kann aber wie folgt in der config.yaml konfiguriert werden.

Beispielskonfiguration für das Remember-Me Feature in der config.yaml

```
de.eoda.dse.core.rest.provider.OsgiRememberMeManager: {
   maxAge: 5184000
}
```

Die Einheit für `maxAge` ist Sekunden und gibt das maximale Alter des Remember-Me Cookies an.

Log-Konfiguration

Ohne Konfiguration wird in YUNA und für den Agenten für das Logging eine Standard-Konfiguration verwendet. Dabei werden die Log-Ausgaben auf die Standard-Konsole geloggt. So sind die Ausgaben in den Logs des System-Services enthalten. Außerdem werden für YUNA und den Agenten je zwei Log-Dateien angelegt:

Im **dse.log** werden alle Ausgaben geloggt, die einen Log-Level von *INFO* oder höher besitzten. Sollte das Log Fehler enthalten, werden diese für die Übersichtlichkeit in verkürzter Form in das Log geschrieben.

In das **dse-error.log** werden nur Nachrichten geschrieben, die einen *ERROR* Log-Level haben. Fehlermeldungen werden in voller Länge im Log festgehalten. Mit dem dse-error.log lassen sich Fehler, die den Betrieb von YUNA beeinträchtigen könnten, schnell identifizieren, da nur kritische Fehler mit dem Log-Level *ERROR* geloggt werden.

In der Standard-Konfiguration werden die Log-Dateien rollend geschrieben. Wenn sich das Datum ändert oder die Dateigröße der Log-Datei ein Maximum übersteigt, dann wird die Datei mit einer entsprechenden Benennung archiviert und eine neue Datei wird begonnen. So kann vermieden werden, dass zu große Log-Dateien das Lesen erschweren.

Die Log-Dateien liegen in den folgenden Ordnern:

- für YUNA: */opt/eoda/dse/portal/*
- für den Agenten: */opt/eoda/dse/agent/*

Sollte die Standard-Konfiguration nicht ausreichen, kann eine eigene Konfigurationsdatei geschrieben werden. Die Datei MUSS **logback.xml** (Groß-Klein-Schreibung beachten) heißen und für YUNA im Ordner */opt/eoda/dse/portal/configuration/* bzw. für den Agenten im Ordner */opt/eoda/dse/agent/configuration/* hinterlegt werden.

Bei der Datei handelt es sich um eine Konfigurationsdatei für die Logging-Bibliothek **Logback**. Das Beispiel zeigt eine mögliche Konfiguration. Weitere Konfigurationsmöglichkeiten finden sich in der Logback-Dokumentation. Außerdem werden bei der Installation der RPM-Pakete Beispiele für Logback-Dateien hinterlegt. Sie sind als **logback.xml.example** in den jeweiligen Konfigurationsordnern zu finden.

Wenn eine eigene Logback-Datei hinterlegt wird, muss der System-Service für YUNA bzw. den Agenten neu gestartet werden, damit die Datei angewendet wird.

Es wird empfohlen, dass die Datei mit dem `scan`-Option konfiguriert wird (s. Beispiel). So kann die Konfiguraiton im laufenden Betrieb geändert werden. Ansonsten muss der Service auch bei jeder Änderung neu gestartet werden.

Beispiel für eine Logback-Konfiguration

```
<!-- Datei wurde mit scan konfiguriert. So kann die Konfiguration im laufenden Betrieb angepasst werden. -->
<configuration scan="true" scanPeriod="60 seconds">

    <!-- Konfiguration für den Datei-Logger. -->
    <appender name="FILE_INFO" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>dse.log</file>
        <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
            <fileNamePattern>dse.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
            <maxFileSize>100MB</maxFileSize>
            <maxHistory>60</maxHistory>
            <totalSizeCap>20GB</totalSizeCap>
        </rollingPolicy>
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} %-5level %logger{36}:%line - %msg%ex{short}%n</pattern>
        </encoder>
    </appender>

    <!-- Konfiguration des Loggers für den Service. -->
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
```

```
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%.15thread] %-5level %logger{36}:%line - %msg%ex{full}%n</pattern>
        </encoder>
    </appender>

    <!--
        Standard Log-Level. Mit dieser Einstellung werden alle Einträge mit dem
        Level "INFO" oder kritischer geloggt.
    -->
    <root level="INFO">
        <appender-ref ref="FILE_INFO" />
        <appender-ref ref="STDOUT" />
    </root>
</configuration>
```

4.1.4. Portal (dse.conf.xml)

Verbindung zur Datenbank

Minimale Beispielkonfiguration der Datenbankverbindungen:

Auszug aus /opt/eoda/dse/portal/configuration/dse.conf.xml.example

```
<!-- Database server connection configuration -->
<!-- do not change id="default", id is required -->
<dbservice id="default">
    <url>jdbc:sqlserver://mssql.hostname.example;applicationName=DSE-Portal-Default</url>
    <driver>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver>
    <name>DatabaseName</name>
    <user>DatabaseUserName</user>
    <password>SuperSecretDatabasePassword</password>
    <initialSize>20</initialSize>
    <maxTotal>100</maxTotal>
    <type>mssql</type>
</dbservice>

<!-- Database server connection configuration for R jobs -->
<!-- do not change id="spconnectserver", id is required -->
<dbservice id="spconnectserver">
    <url>jdbc:sqlserver://mssql.hostname.example;applicationName=DSE-Portal-Spconnect</url>
    <driver>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver>
    <name>DatabaseName</name>
    <user>AndererUser</user>
    <password>OtherSuperSecretPassword</password>
    <initialSize>10</initialSize>
    <maxTotal>50</maxTotal>
    <type>mssql</type>
</dbservice>
```

Verschlüsselte Verbindung

Zum Aufbau einer verschlüsselten Verbindung muss in der dse.conf.xml im Element `<url>` der

<dbservice> die Verschlüsselung durch Anhängen von `trustServerCertificate=true `eingeschaltet werden.

Beispiel für die Verschlüsselund der default Verbindung connection

```
    <url>jdbc:sqlserver://mssql.hostname.example;applicationName=DSE-Portal-
Default;encrypt=true;trustServerCertificate=true;</url>
```

Definition und Nutzung

Um eine Verbindung zur Datenbank zu ermöglichen, müssen die Einstellungen in der dse.conf.xml unter dem 'dbservice'-Tag hinterlegt werden. Dabei ist es erforderlich eine id anzugeben. Die id's können folgende Werte annehmen:

Id	Beschreibung	Erforderlich
default	Diese Id ist zwingend notwendig. Hier werden alle Verbindungen konfiguriert, welche nicht explizit angegeben werden.	✓
mssql	Konfiguriert den Datenbankzugriff des Portals	
spconnect		
spconnectserver	Konfiguriert den Datenbankzugriff für Skripte, die von den Agenten ausgeführt werden.	✓

Folgende Optionen können pro Datenbankverbindung (dbservice id) definiert werden:

Tag	Beschreibung	Erforderlich
connectionProperties	https://commons.apache.org/proper/commons-dbcp/configuration.html	
driver	https://commons.apache.org/proper/commons-dbcp/configuration.html unter dem Parameter 'driverClassName' zu finden	✓
initalSize	https://commons.apache.org/proper/commons-dbcp/configuration.html	
maxIdle	https://commons.apache.org/proper/commons-dbcp/configuration.html	
maxTotal	https://commons.apache.org/proper/commons-dbcp/configuration.html	
minIdle	https://commons.apache.org/proper/commons-dbcp/configuration.html	
minTotal	https://commons.apache.org/proper/commons-dbcp/configuration.html	
name	https://commons.apache.org/proper/commons-dbcp/configuration.html unter Parameter 'defaultCatalog' zu finden	✓
password	https://commons.apache.org/proper/commons-dbcp/configuration.html	✓

Tag	Beschreibung	Erforderlich
type	Gibt den Datenbanktyp an. Kann ausschließlich die Werte 'mssql' oder 'mysql' enthalten.	✓
url	https://commons.apache.org/proper/commons-dbcp/configuration.html	✓
user	https://commons.apache.org/proper/commons-dbcp/configuration.html	✓

Datenbank Objekte

Sind auf dem Datenbankserver ausschließlich zwei Datenbanken mit dem Namen PortalDB und DataDB zu finden, werden mit der unten angegebenen Konfiguration alle Datenbankabfragen (alle, bis auf die im Dashboard definierten Abfragen) an die Portal Tabelle geschickt. Ausgenommen davon sind zusätzlich die Abfragen nach der AdminMsg-Tabelle. Diese werden nun an das Datenbankobjekt [DataDB].[custom].[CustomAdminMessage] gestellt.

Beispiel der Nutzung des Tags aus /opt/eoda/dse/portal/configuration/dse.conf.xml

```
<!-- Database name configuration. The DSE has to know the names of your databases.-->
<!-- These settings are optional, the configuration below represent the default values. -->
<!-- To configure modify the values and uncomment -->
<dbservice-tablereference>
    <portaldb>PortalDB</portaldb>
    <default-datadb>DataDB</default-datadb>
</dbservice-tablereference>
```

RAAS

Auszug aus /opt/eoda/dse/portal/configuration/dse.conf.xml.example

```
<!-- Configuration for the name of the installed R connector package -->
<!-- This setting is optional, the configuration below represents the default value. -->
<raas>
  <connectpackage>dseconnect</connectpackage>
</raas>
```

Tag	Beschreibung	Default
connectpackage	Legt fest, ob dseconnect oder spconnect genutzt werden soll.	dseconnect

Debug

Um Fehlerquellen besser identifizieren zu können, gibt es einen Debug Modus, den man über die *dse.conf.xml* aktivieren kann. Dazu muss der folgende Tag eingetragen werden:

```
<debug>
    <enabled>true</enabled> <!-- default: false -->
```

```
</debug>
```

Verhalten im aktivierten Debug-Modus

Ist der Debug-Modus aktiviert werden die aus dem Portal abgesetzten Queries und die im Dashboard definierten Query-Table-Pathes mit an das Frontend übergeben.

4.1.5. Agent

Konfiguration der virtuellen Maschine

Genau wie im Portal können die Java Memory Options gesetzt werden: (vgl. hierzu Java Virtual Machine).

```
# Example for a memory configuration file of an agent. This file must be readable
# by the application user and must be placed in the configuration directory of the agent
# (default: /opt/eoda/dse/agent/configuration/).
DSE_MEM_OPTS="-Xmx64g"
```

Agentenkonfiguration (config.yaml)

Konfiguration eines Agenten, der Ausführungsumgebungen für R & Python zur Verfügung stellt

```
# Server connection configuration
de.eoda.dse.core.agent.provider.AgentProviderServiceProvider:
  # Configuration of the servers the agent connects to
  core.servers:
    - "http://yuna-portal-backend:8082"
  # optional configuration of the communication port
  ecf.generic.server.port: 59595

# The logging configuration is evaluated once the OSGi logging service is started
de.eoda.dse.core.configuration.provider.LogConfigurator: {
  rootLogLevel: 'WARN',
  de.eoda.dse.core: 'INFO'
}

# R Agent configuration
# NOTE: the R connectivity-package is installed automatically on start up.
de.eoda.dse.core.agent.r.provider.RAgentServiceProvider: {
  name: "R-agent",
  # The connectivity package installation is forced every time the agent is started (default is false)
  forceConnectivityInstallation: true,
  # configures a local libray path (like the R_LIB_SITE environment variable) for the
  # connectivity package installation. Required, if the user running the the agent
  # has no rights to install to the global R library path
  r.libs.site: "/opt/eoda/dse/agent/r-libs"
}

# Python Agent configuration
# NOTE: the python connectivity-package is installed automatically on start up.
de.eoda.dse.core.agent.python.provider.PythonAgentServiceProvider: {
```



```
30   name: "pythonagent",
31   # The connectivity package installation is forced every time the agent is started (default is false)
32   forceConnectivityInstallation: true,
33   # sets the python executable if more than one version is installed
34   executable: ["python3"]
35 }
```

Konfiguration des Backends mit dem der Agent sich verbindet

Im Abschnitt des AgentProviderServiceProvider wird immer mindestens ein Backendserver angegeben, mit dem der Agent sich verbindet.

Zur Konfiguration stehen folgende Parameter zur Verfügung.

Parameter	Beschreibung	Default
core.servers	Liste von URL's verfügbarer Portal services, mit denen der Agent sich verbinden kann	-
ecf.generic.server.port	Kommunikationsport über den der konfigurierte Portal Service mit dem Agenten kommunizieren kann.Wenn der Default in Ihrem Netzwerk durch die Firewall blockiert wird, können sie optional einen anderen Port konfigurieren.	3282

Log-Konfiguration

Auch für den Agenten erfolgt die Konfiguration der Logausgaben erfolgt über die Klasse de.eoda.dse.core.configuration.provider.LogConfigurator in der Konfigurationsdatei des Agenten (siehe Konfigurationsbeispiel in Zeile 10).

Zu den Konfigurationsmöglichkeiten LogConfigurator vergleiche die Beschreibung der Log-Konfiguration des Portal-Service.

Konfiguration der AgentServiceProvider

Automatische Installation der Connectivity Pakete

Die von den Agenten benötigten connectivity Pakete werden seit der Version 2.0.0 zusammen mit dem Agenten ausgeliefert und beim erstmaligen Start des Agenten lokal entpackt und automatisch installiert. Die Paketquellen der aktuellen connectivity Paketversionen werden dabei in das Unterverzeichnis `connectivity` des Installations-Verzeichnis des Agenten abgelegt.

Bei der Installtion des Agenten mittels RPM-Paket findet man die Connectivity-Paketquellen nach dem ersten Start des Agenten im Verzeichnis: `/opt/eoda/dse/agent/connectivity` in entsprechend benannten Unterverzeichnissen je nach Ausführungsumgebung.

Konfigurationsparameter

Es stehen unterschiedliche Parameter für den RAgentServiceProvider und den PythonAgentServiceProvider zur Verfügung.

Parameter	Beschreibung	Werte	Default	RAgent	PythonAgent
name	Optionaler Name zur einfacheren Identifikation des Agenten	beliebige Zeichenkette	-	✓	✓
forceConnectivityInstallation	Erzwingt die Installation der mit dem Agentenservice ausgelieferten connectivity Paketversion bei jedem Neustart des Agenten. Durch die erwungene Aktualisierung wird die Kompatibilität der installierten Connectivity-Paketversion mit dem aktuellen Agenten bei Updates sichergestellt.	true, false	false	✓	✓
r.libs.site	Angabe eines lokalen Bibliothek-Pfades in den das Connectivity-Paket installiert wird. Der Pfad zum Verzeichnis wird ggf. mit angelegt. Es müssen Schreibrechte für den Benutzer existieren, unter dem der Agenten-Prozess gestartet wird. Idealerweise wählen Sie ein Unterverzeichnis im Installations-Pfad des Agenten, z.B. "/opt/eoda/dse/agent/r-libs"	Zeichenkette	-	✓	✗
executable	Wenn mehrere Python Versionen auf einem System installiert sind und die Defaultversion **nicht** verwendet werden soll, kann das auszuführende Executable angegeben werden. Welche Default_Version aktiv ist kann man mittels des Befehls "update-alternatives --config python" herausfinden.	Name des Executable in Anführungszeichen und eckigen Klammern	["python"]	✓	✓

RAgentServiceProvider

Die Angabe des Parameter "**r.libs.site**" ist immer dann zwingend erforderlich, wenn

der der Benutzer mit dem der Agenten Service gestartet wird, keine Schreibrechte auf die globale Bibliothek der lokalen R Installation hat.

4.1.6. Configstore

Der Configstore ist eine Datenbanktabelle, in der einige Einstellungen getätigt werden können.

Key	Beschreibung	Default	Wertebereich	Beispiel
AnalysisFilter	Gibt die Standard FilterId an, welcher bei Analysen im Issue und Job verwendet werden soll.	2	Ganze Zahlen größer 0. **Wichtig**: Ein Filter mit der angegebenen ID muss definiert sein.	2
config.filter.config.filter1	Link zur View auf der der Filter mit der Id 1 bearbeitet werden kann			dse_Device_Basic_Info
config.filter.config.filter2	Link zur View auf der der Filter mit der Id 2 bearbeitet werden kann			
config.filter.filter1.friendlyname	Ersetz den Namen 'filter1' durch einen lesbaren Namen			
config.filter.filter2.friendlyname	Ersetz den Namen 'filter2' durch einen lesbaren Namen			

config.localization.friendlyname	Legt die Anzeigenamen der aktivierten Sprachen fest.	[]	Eine Liste im JSON-Format mit den Schlüsseln *languageKey*, *displayName* und *active*.
			Die Werte von *languageKey* müssen jeweils von einer hochgeladenen oder einer Standard Sprache stammen.
				<pre>[
				  {
				  "languageKey":"de_DE",
				  "displayName":"Deutsch",
				  "active":true
				  },
				  {
				  "languageKey":"en_US",
				  "displayName":
				"Englisch",
				  "active":true
				  }
				]</pre>

Key	Beschreibung	Default	Wertebereich	Beispiel
config.localization.preferred	Definiert die ausgewählte Sprache, wenn ein Nutzer zum ersten Mal das Portal öffnet.	de_DE	Hier kann ein Name einer hochgeladenen oder einer Standard Sprache eingetragen werden.	en_US
dateSubFilter.defaultRelativeFilter	Standardwert für Zeitbereichsfilter		currentDay, currentWeek, currentMonth, currentYear, lastDays_1, lastDays_7, lastDays_30, lastWeeks_13, lastWeeks_26, lastWeeks_52	
filterMenu.skipResolveRequiredWarning	Verhindert, dass beim Doppelklick auf deaktivierte Filterkategorien das Auflösen des Filters bestätigt werden muss.	false	true, false	
infobutton.additionalentry.link	Link zu einer beliebigen PortalView, die zum Beispiel aktuelle Contentänderungen beinhaltet			`#/dse_content_changelog`
map.tileLayer.url	Hier wird die URL des Kachelserver für das Karten-Widget angegeben			`\https://\{s}.eoda.tileserver/\{z}/\{x}/\{y}.png`
message.active	Aktiviert die Benachrichtigungsfunktion	false	true, false	

Key	Beschreibung	Default	Wertebereich	Beispiel
message.unreadMessageRefreshIntervall	Intervall in Sekunden mit dem der Indikator für ungelesene Nachrichten aktualisiert wird. Ein höherer Wert sorgt für langsamere Reaktionszeiten und geringere Systemlast durch eine geringere Anzahl HTTP-Abfragen.	60	Ganze Zahlen größer 0	
portal.name	Wenn im Portal eine Email an den Support gesendet wird, wird dieser Name im Betreff angezeigt.	CM-Portal	Beliebiger Text	
portal.theme.default	Referenz auf die ID des aktuell zu verwendenden Themes. Wird automatisch über den Themes Manager gepflegt und sollte nicht manuell editiert werden.			
showIssuesAsLink	Gibt an, ob die Issues in einem bestimmten Filtermodal als Link angezeigt werden, oder nicht. Hintergrund: Über diesen Weg kamen Nutzer an Sachverhalte, die sie eigentlich nicht sehen durften.	false	true, false	

Key	Beschreibung	Default	Wertebereich	Beispiel
showJobsAsLink	Gibt an, ob die Jobs in einem bestimmten Filtermodal als Link angezeigt werden, oder nicht. Hintergrund: Über diesen Weg kamen Nutzer an Jobs, die sie eigentlich nicht sehen durften.	false	true, false	
tableReloadCount	Definiert die Anzahl von ngTable-Abfragen, bevor ein Reload der Portal-View erzwungen wird.	10	Ganze Zahlen größer 0	
viewConfiguration.issue	Link zum Issue Manager	#/common/issue	Links zu Sichten, auf denen das *issuedirective*-Widget definiert ist. Benutzerdefinierte Sichten müssen über das Dashboard definiert werden.	`#/dse_#/dse_Issue_Manager`
viewConfiguration.issueList	Link zur Liste alle Issues	#/common/issuelist	Links zu Sichten, auf denen eine Liste aller Issues zu finden ist. Benutzerdefinierte Sichten müssen über das Dashboard definiert werden.	`#/dse_Issue_List`

Key	Beschreibung	Default	Wertebereich	Beispiel
viewConfiguration.issueResult	Default-Link zum Ergebnis eines Sachverhalts. Wenn ein neuer Issue erstellt wird, wird der hier definierte Wert als Standardwert für die Sicht zum Ergebnis des Sachverhalts gesetzt. +Dieser kann dann für jeden Issue individuell angepasst werden.	#/common/issueresult	Links zu Sichten, auf denen das Ergebnis eines Sachverhalts aufgeführt wird. Benutzerdefinierte Sichten müssen über das Dashboard definiert werden.	`#/dse_Issue_Result`
viewConfiguration.issueStatusHistory	Link zur Seite über die Historie eines Issues	#common/issueStatushistory		
viewConfiguration.job	Link zum Job Editor	#/common/job	Links zu Sichten, auf denen das *cejobdirective*-Widget definiert ist. Benutzerdefinierte Sichten müssen über das Dashboard definiert werden.	`#/dse_Cyclic_Evaluation_Job_Manager`
viewConfiguration.jobList	Link zur Liste alle Jobs	#/common/joblist	Links zu Sichten, auf denen eine Liste aller Jobs zu finden ist. Benutzerdefinierte Sichten müssen über das Dashboard definiert werden.	`#/dse_Job_List`

Key	Beschreibung	Default	Wertebereich	Beispiel
viewConfiguration.jobResult	Link zum Ergebnis eines Jobs. Wenn ein neuer Issue erstellt wird, wird der hier definierte Wert als Standardwert für die Sicht zum Ergebnis eines Jobs des Sachverhalts gesetzt. Dieser kann dann für jeden Issue individuell angepasst werden.	#/common/jobresult	Links zu Sichten, auf denen das Ergebnis eines Jobs aufgeführt wird. Benutzerdefinierte Sichten müssen über das Dashboard definiert werden.	`#/dse_Cyclic_Evaluation_Job_Result`

4.1.7. Java Virtual Machine

/opt/eoda/dse/portal/configuration/java_memory.example

```
# Example for a memory configuration file. This file must be readable
# by the excuting user and must be placed in the configuration directory of the application
# (default: /opt/eoda/dse/portal/configuration/java_memory).
DSE_MEM_OPTS="-Xmx12g"
```

Definition und Nutzung

In der Standardkonfiguration ist der nutzbare Speicher (Heap Space) auf 4GB begrenzt. Existiert diese Datei nicht oder ist sie nicht lesbar für den Systembenutzer "eoda-dse" wird der Standardwert genutzt. Um den Arbeitsspeicher entsprechend zu erhöhen, muss der Wert des Parameters "Xmx" entsprechend angepasst werden.

4.1.8. env.json

Die Datei "env.json" im Frontend-Verzeichnis, standardmäßig zu finden unter `/opt/eoda/dse/portal-frontend/config/env.json`, dient

1. Zur Konfiguration der Backend-Verbindung für das Frontend
2. Zur Konfiguration von Optionen, die auch ohne angebundenes Backend konfiguriert werden können müssen.

Parametername	Beschreibung	Default
apiUrl	Host-Adresse des YUNA-Backend-Servers	-
baseUrl	Pfad unter dem das YUNA-Backend gehostet wird	/backend/
UIShowversionData	Steuert ob die Versionsinformationen in Kopf- und Fußbereich der Anwendung angezeigt werden	true

Parametername	Beschreibung	Default
UIShowRefTagSwitch	Steuert ob Administratoren das Eingabefeld für Referenz-Tags angezeigt wird	true
UIShowInfoMenu	Steuert ob das Infomenü angezeigt wird	true
UIHideSupportFromNonAdmins	Steuer ob der YUNA-Support im Infomenü für nicht-Admins versteckt wird	false

4.2. Änderungen und Erweiterungen beim Upgrade von Version 0.53.2 auf 1.0

4.2.1. DatabaseReference

Der Java-Enum DatabaseTableReference wurde zu DatabaseReference umbenannt.

Die Nomenklatur der Tabellen-Enumerations wurde von beispielsweise __TBL_ADMINMSG zu TABLE_ADMINMSG umbenannt.

4.2.2. dse.config.xml Änderungen

Die Tablellen-Keys haben sich in Folge der Umbenennungen der Tabellen-Enums geändert.

Das folgende Beispiel ändert den Pfad der TABLE_ISSUE vom Standardpfad auf [CM-DataDB].[xxx].[MyIssue].

```
<dbservice-tablereference>
    <table id="TABLE_ISSUE">
        <database>CM-DataDB</database>
        <schema>xxx</schema>
        <name>MyIssue</name>
    </table>
</dbservice-tablereference>
```

4.2.3. Liste der Tabellenschlüssel alt gegen neu

Liste der Namensänderungen, klicken zum expandieren

Tabellenschlüssel alt	Tabellenschlüssel neu	Datenbank	Schema	Tabelle
__TBL_ADMINMSG	TABLE_ADMINMSG	PORTAL	PORTAL	AdminMsg
__TBL_CHANGEHISTORY	TABLE_CHANGEHISTORY	PORTAL	PORTAL	ChangeHistory
__TBL_ENTITIYBASE	TABLE_ENTITIYBASE	PORTAL	PORTAL	EntityBase
__TBL_EVALUATIONINFO	TABLE_EVALUATIONINFO	PORTAL	PORTAL	EvaluationInfo

Tabellenschlüssel alt	Tabellenschlüssel neu	Datenbank	Schema	Tabelle
__TBL_EVALUATIONINFOPARAMETER	TABLE_EVALUATIONINFOPARAMETER	PORTAL	PORTAL	EvaluationInfoParameter
__TBL_EVALUATIONJOB	TABLE_EVALUATIONJOB	PORTAL	PORTAL	EvaluationJob
__TBL_EVALUATIONJOBRESULT	TABLE_EVALUATIONJOBRESULT	PORTAL	PORTAL	EvaluationJobResult
__TBL_EVALUATIONJOBRESULTLABEL	TABLE_EVALUATIONJOBRESULTLABEL	PORTAL	PORTAL	EvaluationJobResultLabel
__TBL_EVALUATIONJOBRESULTMULTIENTITYBASE	TABLE_EVALUATIONJOBRESULTMULTIENTITYBASE	PORTAL	PORTAL	EvaluationJobResultMultiEntityBase
__TBL_EVALUATIONJOBRESULTTOFILESTORAGE	TABLE_EVALUATIONJOBRESULTTOFILESTORAGE	PORTAL	PORTAL	EvaluationJobResultToFileStorage
__TBL_EVALUATIONJOBRESULTVALUE	TABLE_EVALUATIONJOBRESULTVALUE	PORTAL	PORTAL	EvaluationJobResultValue
__TBL_EVALUATIONJOBSTATUS	TABLE_EVALUATIONJOBSTATUS	PORTAL	PORTAL	EvaluationJobStatus
__TBL_EVALUATIONJOBTYPE	TABLE_EVALUATIONJOBTYPE	PORTAL	PORTAL	EvaluationJobType
__TBL_FILESTREAMDATASTORAGE	TABLE_FILESTREAMDATASTORAGE	PORTAL	PORTAL	FileStreamDataStorage
__TBL_FILTERINFO	TABLE_FILTERINFO	PORTAL	PORTAL	FilterInfo
__TBL_FILTERSTORAGE	TABLE_FILTERSTORAGE	PORTAL	PORTAL	FilterStorage
__TBL_ISSUE	TABLE_ISSUE	PORTAL	PORTAL	Issue
__TBL_ISSUEDEVICEINFO	TABLE_ISSUEDEVICEINFO	PORTAL	PORTAL	IssueDeviceInfo
__TBL_ISSUEDEVICEINFOLINK	TABLE_ISSUEDEVICEINFOLINK	PORTAL	PORTAL	IssueDeviceInfoLink
__TBL_ISSUEDEVICERATING	TABLE_ISSUEDEVICERATING	PORTAL	PORTAL	IssueDeviceRating
__TBL_ISSUEDEVICESTATUS	TABLE_ISSUEDEVICESTATUS	PORTAL	PORTAL	IssueDeviceStatus
__TBL_ISSUEINFO	TABLE_ISSUEINFO	PORTAL	PORTAL	IssueInfo
__TBL_ISSUEINFOTYPE	TABLE_ISSUEINFOTYPE	PORTAL	PORTAL	IssueInfoType
__TBL_ISSUEINFOTYPEGROUP	TABLE_ISSUEINFOTYPEGROUP	PORTAL	PORTAL	IssueInfoTypeGroup
__TBL_ISSUESTATUS	TABLE_ISSUESTATUS	PORTAL	PORTAL	IssueStatus

Tabellenschlüssel alt	Tabellenschlüssel neu	Datenbank	Schema	Tabelle
__TBL_ISSUETYPE	TABLE_ISSUETYPE	PORTAL	PORTAL	IssueType
__TBL_ISSUEUSER	TABLE_ISSUEUSER	PORTAL	PORTAL	IssueUser
__TBL_ISSUESTATUS	TABLE_ISSUESTATUS	PORTAL	PORTAL	IssueStatus
__TBL_LOCALIZATION	TABLE_LOCALIZATION	PORTAL	PORTAL	Localization
__TBL_LOCALIZATION_LANGUAGE	TABLE_LOCALIZATION_LANGUAGE	PORTAL	PORTAL	LocalizationLanguage
__TBL_LOCALIZATION_NAMESPACE	TABLE_LOCALIZATION_NAMESPACE	PORTAL	PORTAL	LocalizationNamespace
__TBL_SCRIPT	TABLE_SCRIPT	PORTAL	PORTAL	Script
__TBL_SCRIPTINFO	TABLE_SCRIPTINFO	PORTAL	PORTAL	ScriptInfo
__TBL_THEMES	TABLE_THEMES	PORTAL	PORTAL	Themes
__TBL_CHANGELOGHISTORY	TABLE_CHANGELOGHISTORY	PORTAL	SP	ChangeLogHistory
__TBL_CONFIGSTORE	TABLE_CONFIGSTORE	PORTAL	PORTAL	ConfigStore
__TBL_CYCLICEVALUATIONLOG	TABLE_CYCLICEVALUATIONLOG	PORTAL	SP	CyclicEvaluationLog
__TBL_DATAID	TABLE_DATAID	PORTAL	SP	DataID
__TBL_DATAID_META	TABLE_DATAID_META	PORTAL	SP	DataIDMeta
__TBL_DATAID_META_ASSOC	TABLE_DATAID_META_ASSOC	PORTAL	SP	DataIDMetaAssoc
__TBL_DATAID_META_FIELD	TABLE_DATAID_META_FIELD	PORTAL	SP	DataIDMetaField
__TBL_DATAID_META_FIELD_PARAM	TABLE_DATAID_META_FIELD_PARAM	PORTAL	SP	DataIDMetaFieldParam
__TBL_DATAID_META_TYPES	TABLE_DATAID_META_TYPES	PORTAL	SP	DataIDMetaTypes
__TBL_DATATYPE	TABLE_DATATYPE	PORTAL	SP	DataType
__TBL_DEFAULTFILTER	TABLE_DEFAULTFILTER	PORTAL	SP	DefaultFilter
__TBL_FILTER	TABLE_FILTER	PORTAL	SP	Filter
__TBL_FILTERHIERARCHY	TABLE_FILTERHIERARCHY	PORTAL	SP	FilterHierarchy
__TBL_FILTERMENU	TABLE_FILTERMENU	PORTAL	SP	FilterMenu
__TBL_FILTERQUERYASSOC	TABLE_FILTERQUERYASSOC	PORTAL	SP	FilterQueryAssoc
__TBL_FILTERROLEASSOC	TABLE_FILTERROLEASSOC	PORTAL	SP	FilterRoleAssoc

Tabellenschlüssel alt	Tabellenschlüssel neu	Datenbank	Schema	Tabelle
__TBL_GRID	TABLE_GRID	PORTAL	SP	Grid
__TBL_LOOKUPQUERY	TABLE_LOOKUPQUERY	PORTAL	SP	LookupQuery
__TBL_PERMISSION	TABLE_PERMISSION	PORTAL	SP	Permission
__TBL_QUERY	TABLE_QUERY	PORTAL	SP	Query
__TBL_QUERYASSOC	TABLE_QUERYASSOC	PORTAL	SP	QueryAssoc
__TBL_ROLE	TABLE_ROLE	PORTAL	SP	Role
__TBL_ROLEPERMISSIONS	TABLE_ROLEPERMISSIONS	PORTAL	SP	RolePermissions
__TBL_TILEPARAMS	TABLE_TILEPARAMS	PORTAL	SP	TileParams
__TBL_USER	TABLE_USER	PORTAL	SP	User
__TBL_USERROLES	TABLE_USERROLES	PORTAL	SP	UserRoles
__TBL_VIEW	TABLE_VIEW	PORTAL	SP	View
__TBL_WIDGETDEPENDENCY	TABLE_WIDGETDEPENDENCY	PORTAL	SP	WidgetDependency
__TBL_DATATABLEWHITELIST	TABLE_DATATABLEWHITELIST	PORTAL	PORTAL	DataTableWhiteList
__VW_USERINFO	TABLE_USERINFO	PORTAL	CUSTOM	vwUserInfo
__VW_EVALUATIONLIST	TABLE_EVALUATIONLIST	PORTAL	PORTAL	vwEvaluationList
__SP_DEACTIVATEJOBS	STOREDPROCEDURE_DEACTIVATEJOBS	PORTAL	PORTAL	sp_DeactivateJobs
__FN_CALCULATESCRIPTVERSION	FUNCTION_CALCULATESCRIPTVERSION	PORTAL	DBO	fn_CalculateScriptVersion
__TBL_RESULTRATINGACTION	TABLE_RESULTRATINGACTION	PORTAL	DEV	ResultRatingAction
__TBL_RESULTRATINGACTIONTYPE	TABLE_RESULTRATINGACTIONTYPE	PORTAL	DEV	ResultRatingActionType
__TBL_RESULTRATINGCOMMENT	TABLE_RESULTRATINGCOMMENT	PORTAL	DEV	ResultRatingComment
__TBL_RESULTRATINGMARKER	TABLE_RESULTRATINGMARKER	PORTAL	DEV	ResultRatingMarker
__TBL_RESULTRATINGREFERENCE	TABLE_RESULTRATINGREFERENCE	PORTAL	DEV	ResultRatingReference
__TBL_RESULTRATINGREFERENCETYPE	TABLE_RESULTRATINGREFERENCETYPE	PORTAL	DEV	ResultRatingReferenceType
__TBL_RESULTRATINGVALIDATION	TABLE_RESULTRATINGVALIDATION	PORTAL	DEV	ResultRatingValidation

4.2.4. Explizite Konfiguration des R-Paket Namens

Im Block **raas** kann der Name des Connect-Package deklariert werden je nach dem ob **dseconnect** oder **spconnect** installierte wurde. Als Standard-Wert oder wenn nicht deklariert wird **spconnect** verwendet.

Langfristig soll das Paket spconnect durch dseconnect abgelöst werden. Als Übergang dient dieser Konfigurationseintrag.

Beispielkonfiguration

```
<raas>
    <version>1.0</version>
    <vendor>eoda GmbH</vendor>
    <vendor_url>https://www.eoda.de</vendor_url>
    <connectpackage>dseconnect</connectpackage>
    <debuguser>false</debuguser>
    <debuguserlogin>test@test</debuguserlogin>
    <debuguserpassword>test</debuguserpassword>
    <localdata>/var/raasimages/</localdata>
    <spconnectversion>45</spconnectversion>
    <spconnect>/opt/virgo/rpackges/spconnect_0.4.tar.gz</spconnect>
    <cron>true</cron>
 </raas>
```

4.2.5. Erweiterungen im R-Paket dseconnect / spconnect

Die Pfade der Portal-System-Tabellen kann über spconnect / dseconnect abgefragt werden. Dieser soll dann für die Abfragen von Tabellen verwendet werden.

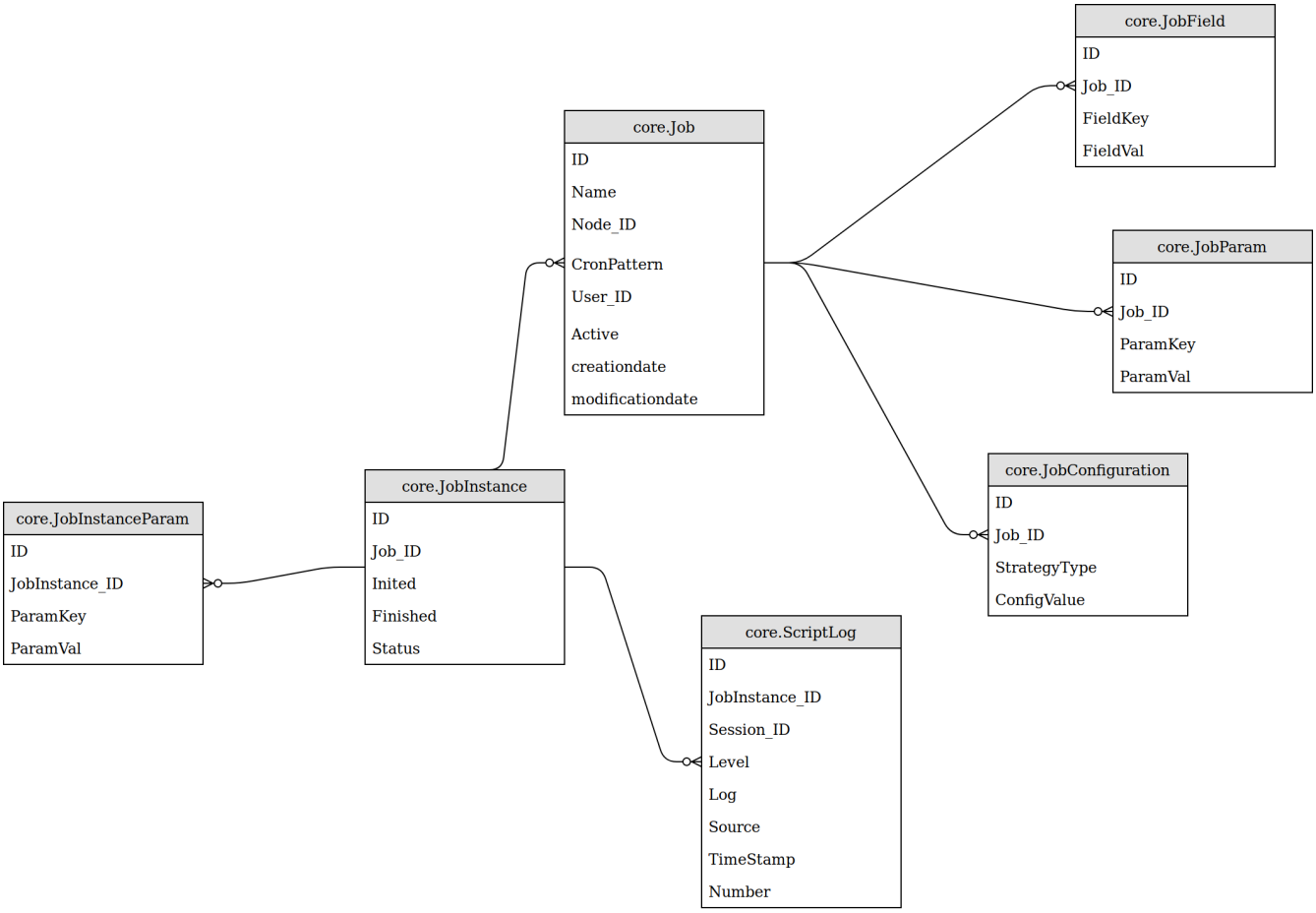
Beispiel:

```
vector <- spconnect::getTablePath("TABLE_ISSUE")
database <- vector[1]
schema <- vector[2]
table <- vector[3]
```

In Zukunft sollen die System-Tabellen nicht mehr fest im Code hinterlegt sein.

4.2.6. Dokumentation der neuen Tabellen im Schema core

Überblick über die neuen Tabellen zur Jobausführung



Auf die neuen Tabellen im Schema `core` sollte generell nicht direkt zugegriffen werden. Für das Auslesen und Änderungen an den Tabelleninhalten stehen REST-Services zur Verfügung.

core.Job

Die Tabelle `Job` im Schema `core` enthält Informationen zu Jobs und ersetzt die Tabelle `EvaluationJob` im Schema `portal`, die aus Kompatibilitätsgründen in `EvaluationJob_old` umbenannt wurde.

Spalte	Beschreibung
ID	Identifier des Jobs
Name	Name des Jobs
CronPattern	Informationen zum zyklischen Ausführen, String repräsentation eines Quartz Cron Ausdrucks
Node_ID	
User_ID	ID des Benutzers der den Job angelegt hat
creationdate	Zeitpunkt zu dem der Job erstellt wurde
modificationdate	Zeitpunkt an dem der Job zuletzt geändert wurde
active	1 wenn der Job aktiv ist

core.ScriptLog

Log enthält Chunks aus der R-Ausführungsumgebung

Level haben folgende Bedeutung:

Level	Bedeutung	Erklärung
1	Verbose	R-Code bevor er in die R-Session geht
2	Debug	Zur zukünftigen Verwendung
3	Info	R console output
4	Warn	Zur zukünftigen Verwendung
5	Error	Fehler aus der R Ausführung
6	Fatal	Zur zukünftigen Verwendung
10	Runtime Info	Zur zukünftigen Verwendung
11	Package load info	Pakete die in die R-Session geladen werden
12	Package unload info	Packete die entladen werden

Source gibt an wer den Logchunk geschrieben hat:

Source	Bedeutung
r-agent	R-Session initialisierung
<node-id>	Node ID die gerade in Ausführung ist

Number ist eine aufsteigende Nummer um die Reihenfolge zu gewährleisten.

core.JobField

in der Tabelle sind zusätzliche Job Felder abgelegt

4.2.7. Logging aus der R-Ausführungsumgebung

Die Tabelle CyclicEvaluationLog wird aus Gründen der Abwärtskompatibilität weiterhin befüllt.

4.2.8. Kundenspezifische Datenbank Views

Bisher konnten auf Tabellen wie

- `[sp].[User]`
- `[portal].[EvaluationJob]`

direkt über Queries aufgerufen werden. Mit der Umstellung auf DSE 1.0 wurden diese Tabellen jedoch durch Core Tabellen ersetzt.

Für eine Abwärtskompatibilität existieren nun Datenbank Views für das Dashboard, die identisch aufgebaut sind wie die bisherigen Tabellen.

Datenbank Views die bisher solche Tabellen referenziert haben, sollten mittels folgenden Befehls

aktualisiert werden:

```
EXECUTE sp_refreshview N'SCHEMA.VIEWNAME';
```


Erfordert die ALTER-Berechtigung für die Sicht und die REFERENCES-Berechtigung für CLR-benutzerdefinierte (Common Language Runtime) Typen und XML-Schemaauflistungen, auf die durch die Sichtspalten verwiesen wird.

4.2.9. Änderungen in der Job-Ausführung

In der Konfigurationsdatei config.yaml kann angegeben werden wie viele Instanzen eines Jobs gleichzeitig laufen dürfen. Siehe:

```
de.eoda.dse.core.job.provider.JobServiceProvider: {
  maxParallelJobExecutionCount: 1
}
```

Die Anzahl bezieht sich dabei sowohl auf manuell als auch auf zyklisch gleichzeitig laufenden Jobs. Das heißt pro Job dürfen maxParallelJobExecutionCount Instanzen gleichzeitig laufen. Dies bedeutet dass von Job A n Instanzen und von Job B zusätzlich n Instanzen gestartet werden können. Wird die Anzahl der maximal laufenden Instanzen überschritten, so wird ein weiterer Start eingeplant sobald eine Instanz fertig wird.

4.3. Neustart von YUNA

4.3.1. Schritt für Schritt Anleitung:

Um das YUNA Portal neu zu starten muss man sich auf dem entsprechenden Server mit seinem Benutzerkonto über SSH anmelden.

1. **DataScience Portal Server herunterfahren**

   Führen Sie folgenden Befehl auf der Kommandozeile aus:

   ```
   sudo service eoda-dse-portal stop
   ```

2. **DataScience Portal Server starten**

   Führen Sie folgenden Befehl auf der Kommandozeile aus:

   ```
   sudo service eoda-dse-portal start
   ```

3. **DataScience Agent herunterfahren**

   Führen Sie folgenden Befehl auf der Kommandozeile aus:


```
sudo service eoda-dse-agent stop
```

4. **DataScience Agent starten**

   Führen Sie folgenden Befehl auf der Kommandozeile aus:

```
sudo service eoda-dse-agent start
```

4.4. Administration

4.4.1. Monitoring

Das Monitoring von YUNA ist mit Prometheus oder einer kompatiblen Software möglich. Dafür stellt YUNA Prometheus-Schnittstellen bereit, mit denen interne Variablen abgerufen werden können. Prometheus ruft diese Schnittstelle in regelmäßigen zeitlichen Abständen ab und speichert die Werte in eine Zeitreihendatenbank. Die Werte können dann mit Prometheus selbst oder mit anderen Visualisierungsplattformen (z.B. Grafana) ausgewertet werden.

Es stehen zwei Prometheus Endpunkte zu Verfügung.

JVM Metrics

Endpunkt: `/backend/prometheus/metrics`

Folgende Variablen werden über die Java Prometheus Exporter bereitgestellt.

Name	offizielle Beschreibung
jvm_memory_bytes_used	Used bytes of a given JVM memory area.
jvm_memory_bytes_committed	Committed (bytes) of a given JVM memory area.
jvm_memory_bytes_max	Max (bytes) of a given JVM memory area.
jvm_memory_bytes_init	Initial bytes of a given JVM memory area.
jvm_memory_pool_bytes_used	Used bytes of a given JVM memory pool.
jvm_memory_pool_bytes_committed	Committed bytes of a given JVM memory pool.
jvm_memory_pool_bytes_max	Max bytes of a given JVM memory pool.
jvm_memory_pool_bytes_init	Initial bytes of a given JVM memory pool.
jvm_buffer_pool_used_bytes	Used bytes of a given JVM buffer pool.
jvm_buffer_pool_capacity_bytes	Bytes capacity of a given JVM buffer pool.
jvm_buffer_pool_used_buffers	Used buffers of a given JVM buffer pool.
jvm_info	JVM version info
jvm_gc_collection_seconds_count	Time spent in a given JVM garbage collector in seconds.
jvm_threads_current	Current thread count of a JVM
jvm_threads_daemon	Daemon thread count of a JVM

Name	offizielle Beschreibung
jvm_threads_peak	Peak thread count of a JVM
jvm_threads_started_total	Started thread count of a JVM
jvm_threads_deadlocked	Cycles of JVM-threads that are in deadlock waiting to acquire object monitors or ownable synchronizers
jvm_threads_deadlocked_monitor	Cycles of JVM-threads that are in deadlock waiting to acquire object monitors
jvm_threads_state	Current count of threads by state
jvm_classes_loaded	Number of classes that are currently loaded in the JVM
jvm_classes_loaded_total	The total number of classes that have been unloaded since the JVM has started execution
process_cpu_seconds_total	Total user and system CPU time spent in seconds.
process_start_time_seconds	Start time of the process since unix epoch in seconds.
process_open_fds	Number of open file descriptors.
process_max_fds	Maximum number of open file descriptors.
process_virtual_memory_bytes	Virtual memory size in bytes.
process_resident_memory_bytes	Resident memory size in bytes.
jvm_memory_pool_allocated_bytes_total	Total bytes allocated in a given JVM memory pool. Only updated after GC, not continuously.

Core Metric

Endpunkt: `/backend/prometheus/coremetrics`

Folgende Variablen geben Informationen über den aktuellen Zustand von YUNA.

Name	Beschreibung
count_of_db_connections	Anzahl der offenen Datenbankverbindungen

Damit die Variable exportiert wird, muss in der Konfiguration eine Datasource namens "prometheus" mit Administratorrechten eingetragen sein.

Beispiel für den Eintrag in der .config.yaml

```
{
   osgi.jdbc.driver.class:
com.microsoft.sqlserver.jdbc.SQLServerDriver,
   serverName: localhost,
   portNumber: "1433",
   databaseName: "master",
   dataSourceName: prometheus,
   user: SA,
   password: "ihr_passwort",
 }
```

Name	Beschreibung
activesession_count	Anzahl der offenen Sessions; d.h. Anzahl der angemeldeten Benutzer
registered_agent_count	Anzahl registrierter Agenten
running_jobs	Anzahl der laufenden Jobs

5. Das YUNA Portal

5.1. Aufbau und Funktionen des Portals

In diesem Bereich der Dokumentation des YUNA Portals möchten wir Ihnen vorstellen, wie das Portal aufgebaut ist, welche Funktionalitäten das Portal für Sie bereit stellt und wie Sie sich am Portal anmelden.

Da die Inhalte und das Design des Portals für jede Instanz frei definiert werden können und sich die darstellbaren und nutzbaren Inhalte des Portals benutzer- oder rollenspezifisch unterscheiden können, stellen wir Ihnen auf den folgenden Seiten die grundlegenden Funktionen des Portals dar.

5.1.1. Portal aufrufen und sich anmelden

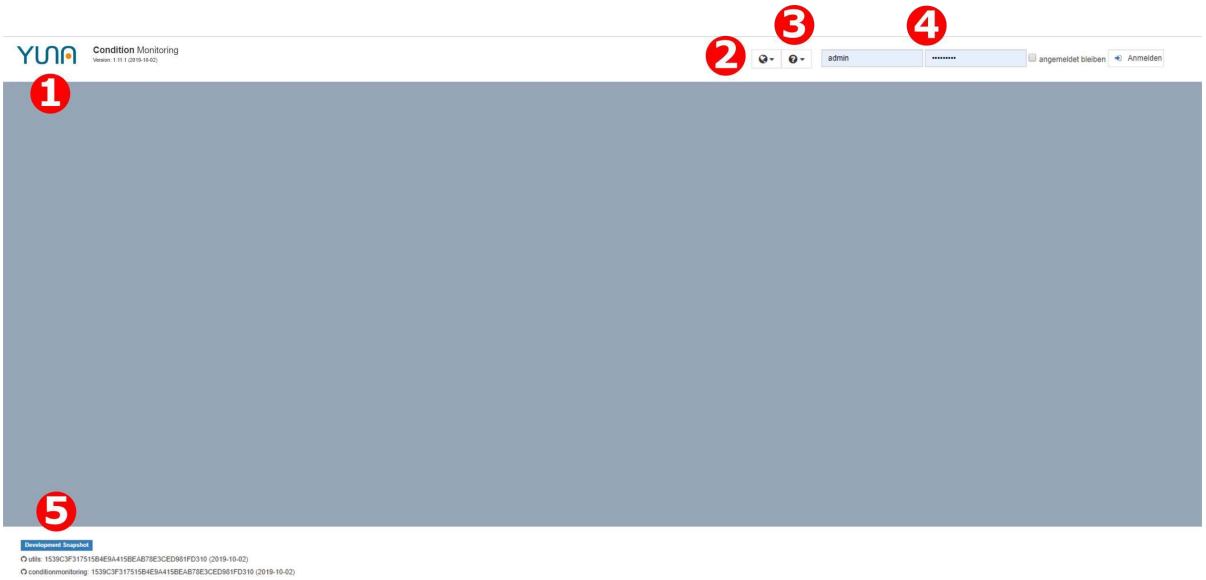
Das Portal aufrufen

Um das YUNA Portal aufzurufen benötigen Sie einen Webbrowser wie Chrome, Firefox, den Internet Explorer. Das Portal können Sie dort erreichen, indem Sie in die Adresszeile Ihres Browsers die URL (Adresse) Ihres Portals eingeben. Diese Adresse kann sich zwischen unterschiedlichen Portalinstanzen unterscheiden und wird Ihnen von Ihren Portalverantwortlichen mitgeteilt.

Beispielhaft könnte die Adresse folgendermaßen aussehen: `http://dsp-demo.eoda.de/portal`

Als User am Portal anmelden

Sobald Sie das Portal aufgerufen haben gelangen Sie auf die Login-Seite.

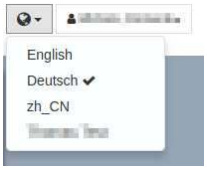


1. Versionsnummer von YUNA
2. Button zur Sprachauswahl
3. Links zum Changelog, Support, Infocenter und den YUNA Tools
4. Eingabefelder für Benutzername und Passwort zum Login
5. In der Fußzeile werden ggf. Zusatzinformation für Entwickler angezeigt

Benutzer, die im Portal durch einen Administrator angelegt wurden, können sich mit ihren LDAP-Anmeldeinforationen am Portal anmelden.
Falls Sie Ihre Zugangsdaten nicht mehr kennen oder Probleme bei der Anmeldung haben wenden Sie sich bitte an Ihren Portalverantwortlichen.

Sprache auswählen

Usern des YUNA Portals steht die Option zur Verfügung, die Anzeigesprache des Portals frei zu wählen. Möglich ist die Auswahl der Anzeigesprache durch Klick auf den Button mit dem Weltkugel-Icon links neben dem Feld für die Eingabe des Usernamens bzw. des an gleicher Stelle angezeigten Dropdown-Menüs, sofern der User bereits eingeloggt ist.



User können alle Sprachen auswählen, die von den Portal-Betreibern definiert und freigeschaltet wurden. Sofern Sie sich als Nutzer eine weitere Sprache wünschen, so wenden Sie sich bitte an Ihren Portalbetreiber.

5.1.2. Dashboard: Inhalte im Portal

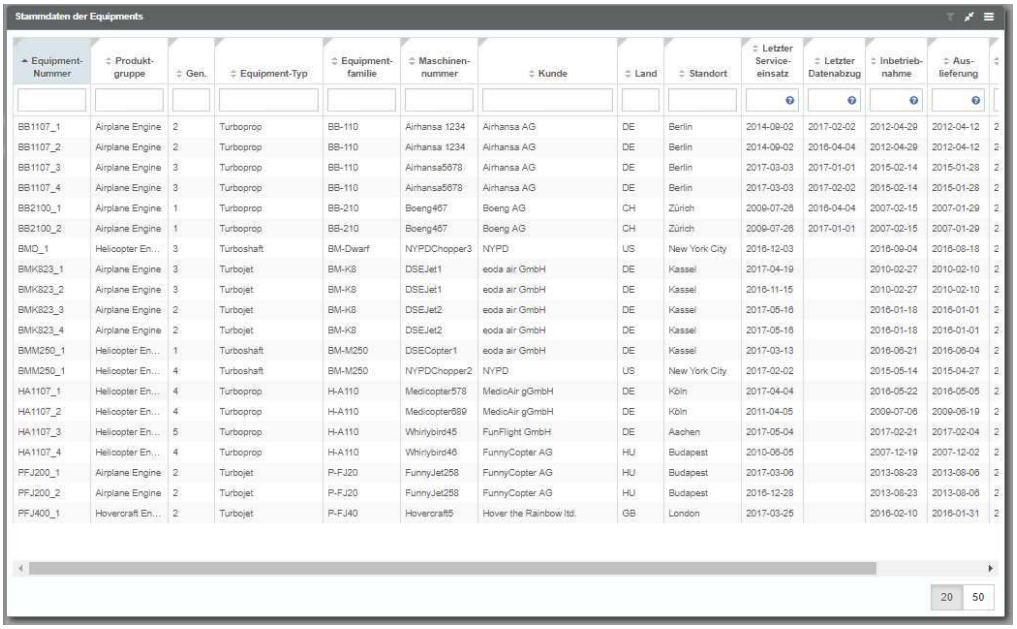
Das YUNA Portal stellt für seine Benutzer Inhalte auf dem Dashboard dar und bietet Interaktionsmöglichkeiten mit diesen. Diese Inhalte werden beispielsweise über Widgets repräsentiert, die in Portal Dashboards angezeigt werden. Die Darstellung und Steuerung von Inhalten erfolgt über Widgets, Filter und DataIDs. Beispiele für Inhalte auf dem Dashboard sind u.a.:

- Portal Views,
- Queries,
- Widgets,
- Widget-Titel oder Widget-Eigenschaften wie "sortierbar" oder "filterbar".

Beispiel: So könnte das Dashboard aufgebaut sein

Die dargestellten Inhalte auf einem Dashboard sind abhängig von den Benutzerrechten bzw. der Benutzerrolle. So kann z.B. eine Sicht (Portal View) für einen Benutzer sichtbar sein, für einen anderen dagegen nicht oder eine Tabelle für einen Benutzer sortierbar und für einen anderen nicht sortierbar sein.

Beispiel : Darstellung von Daten in einem Tabellen Widget



Neben den Dashboard Inhalten kennt das Portal noch Rohdaten und Funktionen als wesentliche Konzepte. Unter Rohdaten werden Daten wie zum Beispiel Messwerte der hinterlegten Anlagen verstanden. Diese werden entweder direkt mit Hilfe der Widgets angezeigt oder in irgendeiner Art vorverarbeitet, z.B. durch Analyse-Skripte analysiert und verdichtet. Funktionen im Sinne des Portals sind Funktionalitäten einzelner Widgets wie beispielsweise das Sortieren von Spalten im Tabellen-Widget oder auch widgetübergreifende Funktionalitäten wie Filter und Funktionalitäten im Rahmen von Workflows.

5.1.3. Widgets und (Portal) Views

Im Portal können Anwender über das Dashboard Inhalte in Form von Widgets und Portals Views verwenden.Als Dashboard Developer erstellen und pflegen sie diese Widgets und Views für die Anwender Ihrer YUNA-Instanz.Im Kontext von YUNA wurden einige Views und Widgets geschaffen, die dazu dienen, Systemadministratoren einen Überblick über die das Portal, die Portalnutzung sowie die User.

Widgets

Ein Widget ist ein Dashboard-Modul, das im Grid - dem Rahmen einer Portal View - angeordnet wird.

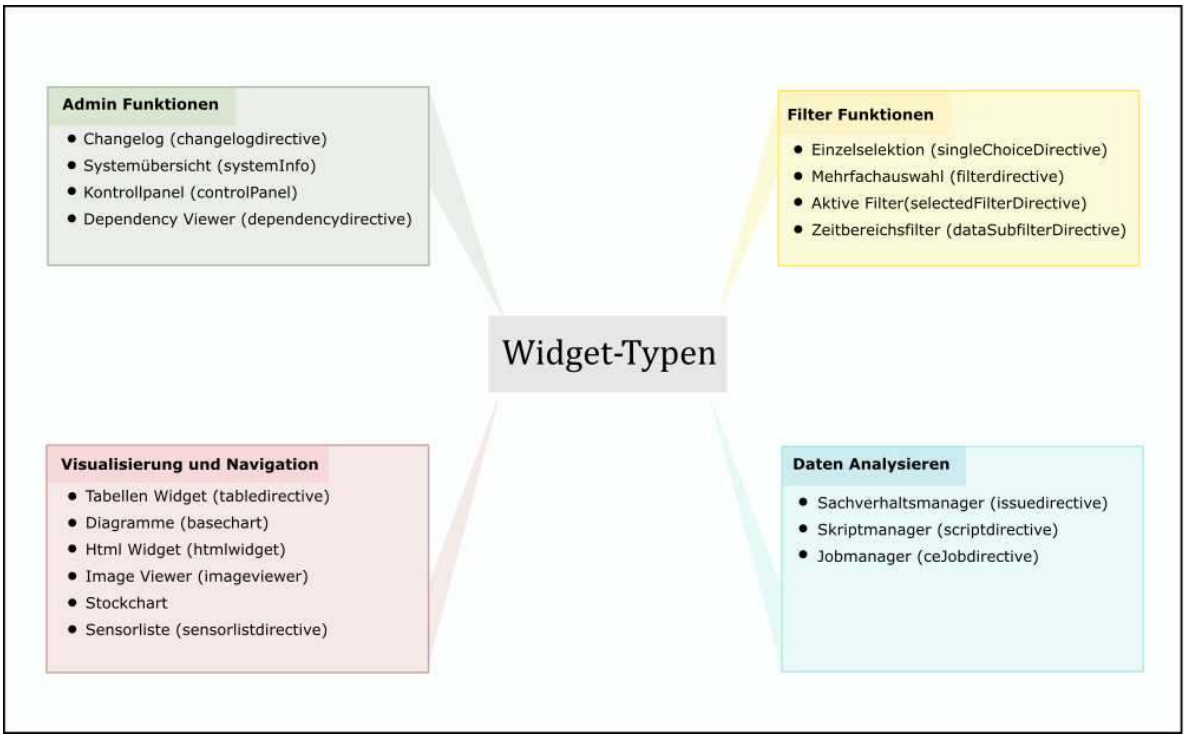
Ein Widget besteht im Wesentlichen aus zwei Elementen:

1. Seiner Konfiguration, also der Art und Weise der Darstellung (Widget-Typ, Name des Widgets, Größe, Position, Icons, Farben, ...) und einer
2. Data-ID, die auf den Inhalt referenziert, der im Widget dargestellt werden soll.

Zwischen den Widgets können Abhängigkeiten bestehen. So kann der Inhalt eines Tabellen-Widgets beispielsweise abhängig sein von der Auswahl in einem Filter-Widget oder ein Diagramm von einem Einzelselektions-Widget.

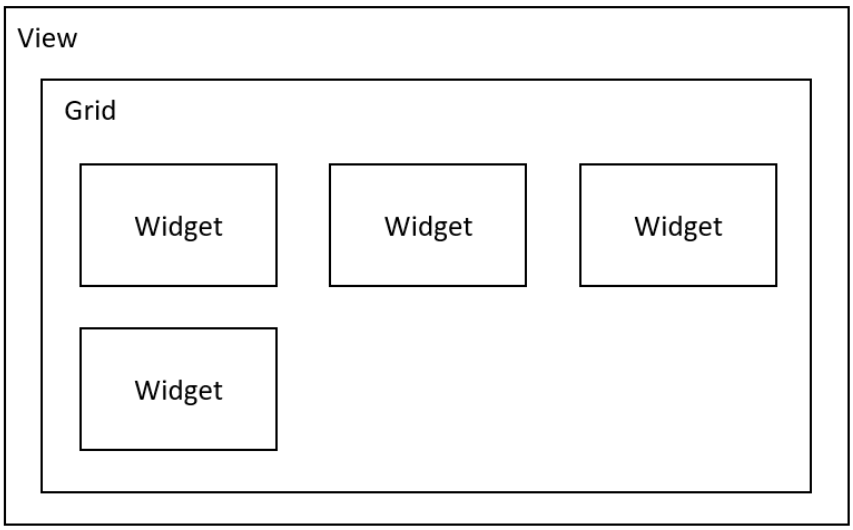
Verfügbare Widget-Typen

Beispiele für Widget-Typen sind unter anderem das Diagramm-Widget, das Filter-Widgets, das Tabellen-Widget oder der Dependency Viewer. In der nachstehenden Grafik finden Sie eine Übersicht über die derzeit verfügbaren Widget-Typen und ihren funktionalen Kategorien:

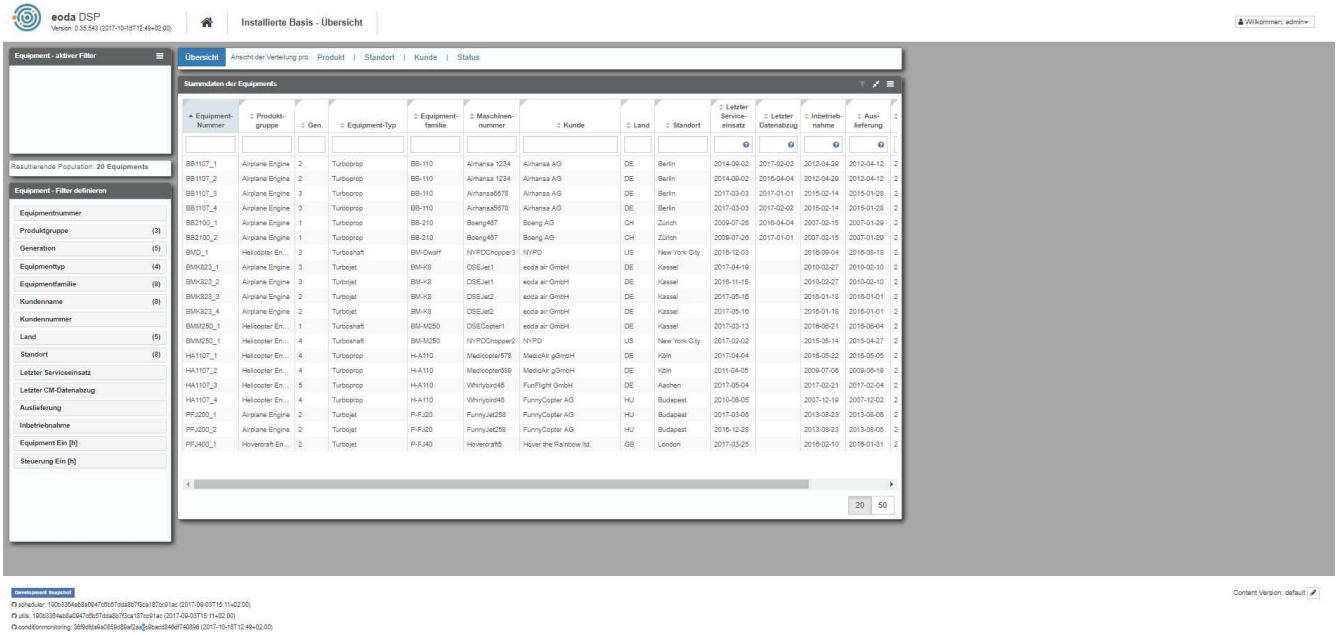


Portal Views

Eine Portal View ist eine Seite, die 1 bis n Widgets enthalten kann. Eine Seite (Portal View) hat dabei ein Grid als Projektionsfläche für Widgets. Eine View hat Eingeschaften wie den Seitennamen, eine Beschreibung oder Benutzerberechtigungen. Das Grid selbst besitzt keine Eigenschaften oder Funktionen und dient lediglich als Raster, in dem Widgets angeordnet werden.



Jede View besitzt eine individuelle Adresse, die über die Adresszeile des Webbrowsers erreichbar ist. Grundlegend ist eine Portal View also vergleichbar mit einer (Kind-)Seite einer Webseite, die Sie im Internet besuchen.



Der obige Screenshot zeigt beispielhaft die Portal View "Installierte Basis" des YUNA Portalss. Sie besteht aus unterschiedlichen Widgets und ist in diesem Fall über die individuelle URL *beispielurl/portal/dsp_InstBasis* erreichbar.

Widgets und Portal Views können selbst erstellt und vielfältig gestaltet werden. Dies bedeutet, dass das Portal bei unterschiedlichen Unternehmen - oder sogar unterschiedlichen Benutzern eines Unternehmens - verschieden aussehen kann.

5.1.4. Allgemeine Eigenschaften von Widgets

Zur Erstellung eines Widgets müssen der Widget-Typ, die Position des Widgets im Grid sowie die Größe definiert werden.

Widget-Typ und Name

Der Widget-Typ wird über das Feld "**widgettype**" bestimmt, z.B. `tabledirective` oder `filterdirective`. Der Widget-Typ definiert die Eigenschaften und das Verhalten des Widgets. Der Name eines Widges wird über das Feld **name** definiert. Über seinen Namen kann das Widget im Dashboard angesprochen und gesourct werden.

```
<widget name="WidgetNameZumSourcen">
    <widgettype>selectedfilterdirective</widgettype>
</widget>
```

```
{
    "widgettype": "selectedfilterdirective"
}
```

Position

Die Position eines Widgets kann durch das Feld **position** durch Angaben für die für die X- und Y-Achsenposition angepasst werden. Hierbei gilt immer, dass eine Einheit 100px im Grid entspricht (Der Positionswert 1 entspricht also am Bildschirm 100 Pixel).

```
<position>
  <x>1</x>
  <y>0</y>
</position>
```

```
"position": { "position": {"x": 1, "y": 0}},
```

Die Positionierung des Elements auf der X-Achse wird direkt übernommen. Beim Verändern des Wertes von Y wird man nicht immer einen Unterschied erkennen. Die Y-Achsenwerte werden nur berücksichtigt, wenn mehrere Widgets positioniert werden.

Überschneidungen von Widget-Positionen

Bei Überschneidungen oder gar Weglassen der Koordinaten kann es zu unerwünschtem Verhalten kommen. Wenn das erste Element eine Dimension von 3x2 Einheiten hat, muss das zweite Element mit einer Position von 2 auf der Y-Achse gesetzt werden, um es unterhalb des ersten darzustellen. Genauso verhält es sich für ein drittes Element, das rechts neben dem ersten dargestellt werden soll und daher einen Positionswert von minbdestens 3 Einheiten auf der X-Achse erhalten muss.

Aktuell sind die X- und Y-Werte vertauscht. Für die Definition von Dashboard Inhalten muss also der gewünschte Y-Wert zuerst eingetragen werden, der gewünschte Wert für X danach.

Größe

Breite und Höhe eines Widgets werden durch das Feld **size** definiert. Zum Anpassen der Größe eines Widgets können jeweils Angaben für die Ausdehnung entlang der X- und der Y-Achse gemacht werden. Das Feld **size** muss angegeben werden und mindestens eine Dimension von 1x1 haben um das Element korrekt darzustellen zu können. Auch hier gilt, dass eine Einheit 100px im Grid entspricht

```
<size>
    <x>3</x> <!-- Breite des Widget in Einheiten -->
    <y>2</y> <!-- Höhe des Widget in Einheiten -->
<size>
```

```
"size": { "x": 3, "y": 2 }
```

Das in obigen Beispiel konfigurierte Widget beansprucht innerhalb des gesamten Grids 300px in der Breite und 200px in der Höhe. Dabei ist zu beachten, dass der tatsächlich für die Darstellung vewendete Bereich innerhalb des Grids der Gesamtbreite abzüglich 28px für Padding bzw. Margin beträgt.

Mehrere Widgets gleichzeitig erzeugen

YUNA ML Das folgende Beispiel erzeugt 3 unterschiedliche Widgets mit definierter Größe und Position:

```
<xml>
    <widget>
        <widgettype>selectedfilterdirective</widgettype>
        <position>
           <x>0</x>
           <y>0</y>
        </position>
        <size>
            <x>3</x>
            <y>2</y>
        </size>
    </widget>
    <widget>
        <widgettype>filterdirective</widgettype>
        <position>
                <x>2</x>
                <y>0</y>
        </position>
        <size>
            <x>3</x>
            <y>6</y>
        </size>
    </widget>
    <widget>
        <widgettype>tabledirective</widgettype>
        <position>
                <x>0</x>
                <y>3</y>
        </position>
        <size>
            <x>15</x>
            <y>8</y>
        </size>
    </widget>
</xml>
```

```
{
    "widgettype": "selectedfilterdirective",
    "position": { "position": {"x": 0, "y": 0}},
    "size": { "x": 3, "y": 2 }
},
{   "widgettype": "filterdirective",
    "position": { "position": {"x": 2, "y": 0}},
    "size": { "x": 3, "y": 6 }
},
```

```
{   "widgettype":"tabledirective",
    "position":{"position":{"x": 0, "y": 3}},
    "size":{"x":15,"y":8}
}
```

Datenanbindung

Viele Widgets können mit einer parametrisierten Datenanbindung arbeiten. Sie beziehen Daten über eine DataID oder setzen Parameter in die URL, die wiederum Reaktionen bei anderen Widgets auslösen können. Die DataID stellt eine Verknüpfung mit einer QueryBuilder-Abfrage her, die ggf. die URL-Parameter als Abfrageparameter aufnehmen kann.

```
<dataID>qy_infoforsingleeqipment</dataID>
<urlParam>VersionID</urlParam>
<triggerParams>
    <mandatory>
        <list>id</list>
        <list>filter2</list>
    </mandatory>
    <optional>
        <list>startdate</list>
        <list>enddate</list>
    </optional>
</triggerParams>
```

```
{
  "dataID": "qy_infoforsingleequipment",
  "urlParam": "VersionID",
  "triggerParams": {
    "mandatory": ["id", "filter2"],
    "optional": ["startdate", "enddate"]
  },
  "triggerOptions": {}
}
```

Übersicht über die Konfigurationsparameter für Datenanbindungen:

Feld	Mögliche Werte	Default	Beschreibung
dataID	String	""	Datenherkunft, die auf eine Datenbankabfrage verweist.
urlParam	String	""	URL-Parameter, der durch das Widget gesetzt wird, z. B. in einer Singlechoice-Direktive.
loadOnInit	true | false	true	Widgetfunktionen, die von URL-Parametern abhängig sein können, werden auch ohne Vorhandensein der Parameter initial ausgeführt.

Feld	Mögliche Werte	Default	Beschreibung
triggerParams (mandatory / optional)	Array [String]	[]	• **mandatory trigger params:** URL-Parameter bei dessen Änderung das Widget reagiert. Es müssen alle mandatory Trigger-Parameter vorhanden sein, um eine Aktion wie z. B. eine Abfrage ausführen zu können. • **optional trigger params:** URL-Parameter bei dessen Änderung das Widget reagiert, sofern alle mandatory parameter vorhanden sind.

Weitere allgemeine Widget-Funktionen

Für jedes Widget können Eigenschaften definiert werden, die das Verhalten und die Funktionen beeinflussen.

Widget ein-/ausklappen

Die Funktion ein Widget einzuklappen und wieder aufzuklappen kann beim Endbenutzer zu einem besserem Überblick verhelfen. Das Feld "appearance" kann wie folgt konfiguriert werden:

```
<appearance>
    <enlargeableY>true</enlargeableY>
    <enlargedY>true</enlargedY>
</appearance>
```

```
"appearance": {
    "enlargeableY": true,
    "enlargedY": true
}
```

Einstellmöglichkeiten zum Ein-/Ausklappen von Widgets

Feld	Mögliche Werte	Beschreibung
enlargeableY	true, false	Bei *true* kann das Widget ein- bzw. ausgeklappt werden
enlargedY	true	Aktueller Zustand: ausgeklappt; Default-Wert
	false	Aktueller Zustand: eingeklappt

Titelzeile

Die Titelzeile bzw. caption eines Widgets beinhaltet den angezeigten Titel des Widgets und kann weitere Funktionalitäten wie ein Kontextmenü, Exportfunktionen o.ä. bereitstellen. Sie kann mit dem Feld "caption" erzeugt und konfiguriert werden.

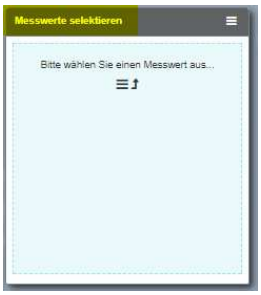
Mit **"show"** bestimmt man, ob die Leiste ein-/ausgeblendet wird. Dem Feld **"label"** kann ein beliebiger Text zugewiesen werden, der anschließend in der Titelzeile angezeigt wird.

Example 1. Anzeige der Titelzeile mit vorgegebem Text

```
<caption>
    <show>true</show>
    <label>Messwerte selektieren</label>
</caption>
```

```
"caption": {
    "show": true,
    "label": "Messwerte selektieren"
}
```

In der Titelzeile wird nun der Text "Messwerte selektieren" angezeigt:



Einstellmöglichkeiten zur Sichtbarkeit der Titelzeile

Feld	Mögliche Werte	Beschreibung
show	false	Titelzeile ist ausgeblendet. Default-Wert.
show	true	Titelzeile ist eingeblendet
label	Text	Beliebiger Text. Mindestens 1 Zeichen (kann nicht aus nur 1 Leerzeichen bestehen)

Weitere Optionen für die Titelzeile

Export

Um die Funktion zum Exportieren einer Tabelle ein- oder auszuschalten kann das Widget im XML wie folgt definiert werden:

```
<caption>
    <show>true</show>
    <label>Messwerte selektieren</label>
    <export>true</export>
</caption>
```

```
"caption": {
    "show": true,
    "label": "Messwerte selektieren",
    "export": true
}
```

Einstellmöglichkeiten für Tabellenexport

Feld	Mögliche Werte	Beschreibung
export	false	Export-Buttons werden ausgeblendet.
	true	Export-Buttons werden eingeblendet. Default-Wert.

Kontext-Menü

Das Kontext-Menü ist ein umfangreiches Werkzeug um Widget spezifische Optionen anzubieten, welche vom Anwender in der Widget Caption mit einem Klick erreicht werden können. Es bietet die Möglichkeit diverse Funktionen auszuführen und Links zu beliebigen Seiten zu öffnen und kann individuell definiert und konfiguriert werden. Im folgenden Artikel wird beschrieben wie Sie ein Kontext-Menü anlegen, Buttons hinzufügen und die Darstellung beeinflussen können.

Das Kontext-Menü ist u.A. beim Tabellen-Widget und "Aktivierte Filter"-Widget bereits vordefiniert und bietet Optionen zur Verwaltung der Filter bzw. zum Exportieren der Tabellen. Diese können jedoch ebenfalls individuell konfiguriert werden.

Kontextmenü erzeugen

Da das Kontext-Menü ein Bestandteile des Widget-Caption ist, muss die caption auch im Widget aktiviert sein (show: true). Das Kontext-Menü wird durch hinzufügen der Eigenschaft **menu** konfiguriert, in dem alle Eigenschaften des Menüs definiert werden. Damit das Kontextmenü angezeigt wird muss die Eigenschaft **show** auf true gesetzt werden.

```
<caption>
    <show>true</show>
    <label>Messwerte selektieren</label>
    <export>true</export>
    <menu>
        <show>true</show>  <!-- default of menu show is false -->
    </menu>
</caption>
```

```
"caption": {
    "show": true,
    "label": "Meswerte selektieren",
    "menu": {
        "show": true
    }
}
```

Nun steht Ihnen in der Caption das Kontextmenü zur Verfügung. Es hat jedoch noch keinen Inhalt. Nach erfolgreicher Aktivierung des Kontextmenüs erscheint rechts in der Titelzeile der folgende Button:  .

Kontext-Menü befüllen

Der Inhalt des Menüs kann mit mehreren Elementen von jeweils unterschiedlichen Typen befüllt werden, welche dem Typ entsprechend eine andere Funktion erfüllen oder die Darstellung verändern.

Optionen hinzufügen

Um das Kontextmenü mit eigens definierten Einträgen zu befüllen, wird innerhalb des Elements **menu** der Eintrag **option** hinzugefügt, der eine Liste von einem oder mehreren Menüeintragen sein kann.

In diesen **option**-Listenelementen können alle Einstellungen für die Kontextmenü-Einträge vorgenommen werden, welche die Darstellung und Funktion beeinflussen. Die zur Verfügung stehenden Optionen finden sich in der folgenden Tabelle.

Einstellmöglichkeiten für Kontextmenüeeinträge

Feld	Mögliche Werte	Beschreibung
name	"Text"	Bezeichnung für die Option muss angegeben und innerhalb der Widget-Definition eindeutig sein.
type	"link", "function", "popup", "label", "divider"	Bestimmt den Typen der Option und dadurch ihre Funktion bzw. Darstellung. Sollte kein Typ angegeben sein, wird die Option nicht angezeigt. Für die Typen "link", "function" und "popup" wird die Funktionalität in einem eigenen Objekt definiert. Im nächsten Abschnitt gehen wir auf die verschiedenen Typen und ihre Konfigurationsmöglichkeiten näher ein.
icon	Name des fa-icons"	Definiert ein Icon, welches vor dem Button-Label angezeigt wird. Hier wird die gewünschte Fontawesome-Klasse eingetragen. [TIP] ==== Derzeit ist die Nutzung von FontAwesome-Icons bis einschließlich Version 4.7 gewährleistet. ====
show	true, false	Button wird ein- bzw. ausgeblendet. Default Wert: false.
disabled	true, false	Für `disabled: true` wird der Button verblasst angezeigt, kann jedoch nicht angeklickt werden. Default Wert: false.
tooltip	"Text"	Beliebiger Text, der über dem Button als Hinweis erscheint, sobald man mit der Maus drüber fährt. Falls nicht angegeben wird kein Tooltip angezeigt.

Die verschiedenen Options-Typen

Typ	Funktion
link	Erzeugt einen Button, welcher beim Klicken eine beliebige URL öffnet
function	Erzeugt einen Button, welcher beim Klicken eine Funktion ausführen kann
popup	Erzeugt einen Button, welcher beim Klicken ein Popup-Fenster mit beliebigem Inhalt öffnet
label	Erzeugt ein Text-Label für Einträge bzw. Buttons im Kontextmenü, welches zur Kategorisierung mehrerer Buttons verwendet werden kann
divider	Trennelement

Der Typ Link

```
<option>
    <list>
        <type>link</type>
        <icon>fa-question</icon>
        <label>Hilfe zum Widget</label>
        <show>true</show>
        <link>
            <url>help/widget</url>
            <extern>true</extern>
        </link>
    </list>
</option>
```

```
"option": [{
    "type": "link",
    "icon": "fa-question",
    "label": "Hilfe zum Widget",
    "show": true,
    "link": {
        "url": "help/widget",
        "extern": true
    }
}]
```

Mit diesen Einstellungen erzeugen Sie einen Button des Typs "link", welcher zu einem Pfad auf dem eigenem Server verlinkt und in einem neuen Tab geöffnet wird.

Einstellmöglichkeiten für Links

Feld	Mögliche Werte	Beschreibung
url	"URL"	URL zu welcher der Link verweisen soll.
extern	true, false	Bei true wird der Link in einem neuen, bei false im selben Tab des Browsers geöffnet. Der default Wert ist false.

Der Typ Popup

Eintrag, der ein Popup mit selbst definiertem Inhalt öffnet:

```
<option>
    <list>
        <type>popup</type>
        <label>Hilfe zum Widget</label>
        <show>true</show>
        <popup>
            <header>Das Filter-Widget</header>
            <body>Dieses Widget dient zur...</body>
        </popup>
    </list>
</option>
```

```
"option": [{
    "type": "popup",
    "label": "Hilfe zum Widget",
    "show": true,
    "popup": {
        "header": "Das Filter Widget",
        "body": "Dieses Widget dient zur..."
    }
}]
```

Einstellmöglichkeiten für Popups

Feld	Mögliche Werte	Beschreibung
header	"Text"	Hier kann ein beliebiger Text stehen, welcher als Titel des Popups verwendet wird.
body	"Text"	Hier Kann ein beliebiger Text stehen, welcher im Dashboard-Bereich des Popups angezeigt wird.

Hinweis zum Styling von Popups

Innerhalb eines Popups können Bootstrap sowie alle im CSS definierten Klassen verwendet werden.
Zusätzliche Style-Angaben sind hingegen nicht möglich.

Der Typ Function

Eintrag, der eine vordefinierte Funktion ausführt:

```
<option>
    <list>
        <type>function</type>
        <label>Führe Funktion aus</label>
```

```
        <show>true</show>
        <function>
            <widget>contextmenu</widget>
            <name>testFunction</name>
            <param>
                <list>test1</list>
                <list>test2</list>
                <list>test3</list>
            </param>
        </function>
    </list>
</option>
```

```
"option": [{
    "type": "function",
    "label": "Führe Fuktion aus",
    "show": true,
    "function": {
        "widget": "contextmenu",
        "name": "testFunction",
        "param": ["test1", "test2", "test3"]
    }
}]
```

Einstellmöglichkeiten für Funktionen

Feld	Mögliche Werte	Beschreibung
widget	"contextmenu", "selectedfilter"...	Hier wird der Name des Widgets angegeben, aus dem eine Fuktion ausgeführt werden soll.
name	"testFunction", "saveFilter"...	Name der Funktion, die ausgeführt werden soll. Auf Groß- und Kleinschreibung muss geachtet werden. Im nächsten Abschnitt werden die möglichen Funktionen aufgelistet.
param	[Array of Strings]	Beliebige Parameter die der Funktion mitgeliefert werden.

Registrierte Funktionen

Es können auch Funktionen ausgeführt werden, die für das jeweilige Widget vordefiniert und im Kontext-Menü registriert wurden.

Selected Filter: ("widget": "selectedfilter")

Einstellmöglichkeiten für Registrierte Funktionen

Funktion	Beschreibung	Parameter
saveFilter	Speichert den aktuellen Filter.	keine
saveASFilter	Speichert den aktuellen Filter unter neuem, beliebigem Namen.	keine
loadFilter	Lädt einen beliebigen, gespeicherten Filter.	keine

Funktion	Beschreibung	Parameter
clearFilter	Löscht alle ausgewählten Filter.	keine

Der Typ Label

EIn einfacher Texteintrag, als Kategorie-Überschrift:

```
<option>
    <list>
        <type>label</type>
        <label>Filteroptionen</label>
        <show>true</show>
    </list>
</option>
```

```
"option": [{
    "type": "label",
    "label": "Filteroptionen",
    "show": true
}]
```

Der Typ Divider

Ein Divider als Trennelement:

```
<option>
    <list>
        <type>divider</type>
        <show>true</show>
    </list>
</option>
```

```
"option": [{
    "type": "divider",
    "show": true
}]
```

Beispiel für ein komplettes Kontext-Menü

Auch Widgets mit vordefinierten Kontextmenüs können individuell konfiguriert und erweitert werden. Das folgende Beispiel erzeugt ein Kontextmenü für ein Tabellenwidgets:

```
<xml>
    <widget name="Tabelle">
    <widgettype>tabledirective</widgettype>
```

```
        <position>
                <x>5.5</x>
                <y>3</y>
        </position>
        <size>
            <x>13</x>
            <y>5</y>
        </size>
        <caption>
            <show>true</show>
            <label>Tabellen-Widget mit kontextmenü</label>
            <export>true</export>
            <menu>
                <show>true</show>
                <option>
                    <list>
                        <type>label</type>
                        <label>Tabellenoptionen</label>
                        <name>Widgetoptions</name>
                        <show>true</show>
                    </list>
                    <list>
                        <type>link</type>
                        <icon>fa-question</icon>
                        <label>Hilfe zum Kontextmenü</label>
                        <show>true</show>
                        <link>
                            <url>https://confluence.eoda.local/pages/viewpage.action?pageId=15696020</url>
                            <extern>true</extern>
                        </link>
                        <tooltip>Confluence Artikel zum Konfigurieren des Kontext-Menüs</tooltip>
                    </list>
                    <list>
                        <type>function</type>
                        <name>testfunction</name>
                        <icon>fa-refresh</icon>
                        <label>Testfunktion</label>
                        <show>true</show>
                        <function>
                            <widget>contextmenu</widget>
                            <name>testFunction</name>
                            <array_param_1>test1</array_param_1>
                            <array_param_2>test2</array_param_2>
                            <array_param_3>test3</array_param_3>
                        </function>
                    </list>
                    <list>
                        <type>popup</type>
                        <name>tableproperties</name>
                        <icon>fa-info</icon>
                        <label>Tabellen Eigenschaften</label>
                        <show>true</show>
                        <popup>
                            <header>Eigenschaften</header>
                            <body>Die Tabelle zeigt Ihnen...</body>
                        </popup>
                    </list>
                </option>
            </menu>
```

```
        </caption>
    </widget>
</xml>
```

```
{
  "caption":
    {
    "show": "true",
    "label": "Tabellen-Widget mit Kontextmenü",
    "menu": {
      "show": true,
      "option": [{
          "name": "widgetoptions",
          "type": "label",
          "label": "Tabellenoptionen",
          "show": true
      },{
        "name": "widgethelp",
        "type": "link",
        "icon": "fa-question",
        "label": "Hilfe zum Kontextmenü",
        "show": true,
        "link": {
          "url": "https://confluence.eoda.local/pages/viewpage.action?pageId=15696020",
          "extern": true
        },
        "tooltip": "Confluence Artikel zum konfigurieren des Kontext-Menüs"
      },{
        "name": "testfunction",
        "type": "function",
        "icon": "fa-refresh",
        "label": "Testfunktion",
        "show": true,
        "function": {
          "widget": "contextmenu",
          "name": "testFunction",
          "param": ["Test1", "Test2", "Test3"]
        }
      },{
        "name": "tableproperties",
        "type": "popup",
        "icon": "fa-info",
        "label": "Tabellen Eigenschaften",
        "show": true,
        "popup": {
          "header": "Eigenschaften",
          "body": "Die Tabelle zeig Ihnen..."
          }
        }
```

```
        ]
      }
    },
    "position":
    {
      "position": [0,0]
    },
    "size":
    {
      "x":5,
      "y":2
    },
    "widgetname" : "tabledirective"
}
```

Ansicht des oben konfigurierten Kontextmenüs in Yuna

5.1.5. Triggerparameter für Widgets

Widgets dienen zur Darstellung und Manipulation von Daten, Analysen und Ergebnissen im Portal. Welche Daten in die jeweiligen Widgets geladen werden ist abhängig von Parametern, die die jeweiligen Widgets aus der URL entnehmen. Auf welche Parameter einzelne Widgets reagieren wird bei der Dashboard-Definition der einzelnen Widgets festgelegt durch Eintragungen im Feld **triggerParams**. Dies bedeutet auch, dass unterschiedliche Widgets (unabhängig vom Widget-Typ) auf verschiedene Parameter reagieren oder nicht reagieren können.

Im nachfolgenden Beispiel wird ein Widget vom Typ Stockchart (developmentstockchart) erzeugt, das von 5 URL-Parametern abhängig ist: `sensor`, `equi`, `startdate`, `enddate` und `selectedRelativePeriod`. Diese Triggerparameter werden durch andere Widgets oder aktive Filter erzeugt und an die URL angehangen. Allerdings müssen nicht alle Parameter in der URL vorhanden sein, damit die Datendarstellung im Widget funktioniert. So stellt die Auswahl einer relativen Zeitperiode von beispielsweise 52 Wochen (selectedRelativePeriod) einen Widerspruch zur Festlegung eines festen Start- und Enddatums dar, weshalb von der Datumsauswahl (DateSubFilterDirective) nur entweder die relative Zeitperiode oder das Start- und Enddatum an die URL weitergegeben wird bzw. werden.

YUNA ML Stockchart-Definition

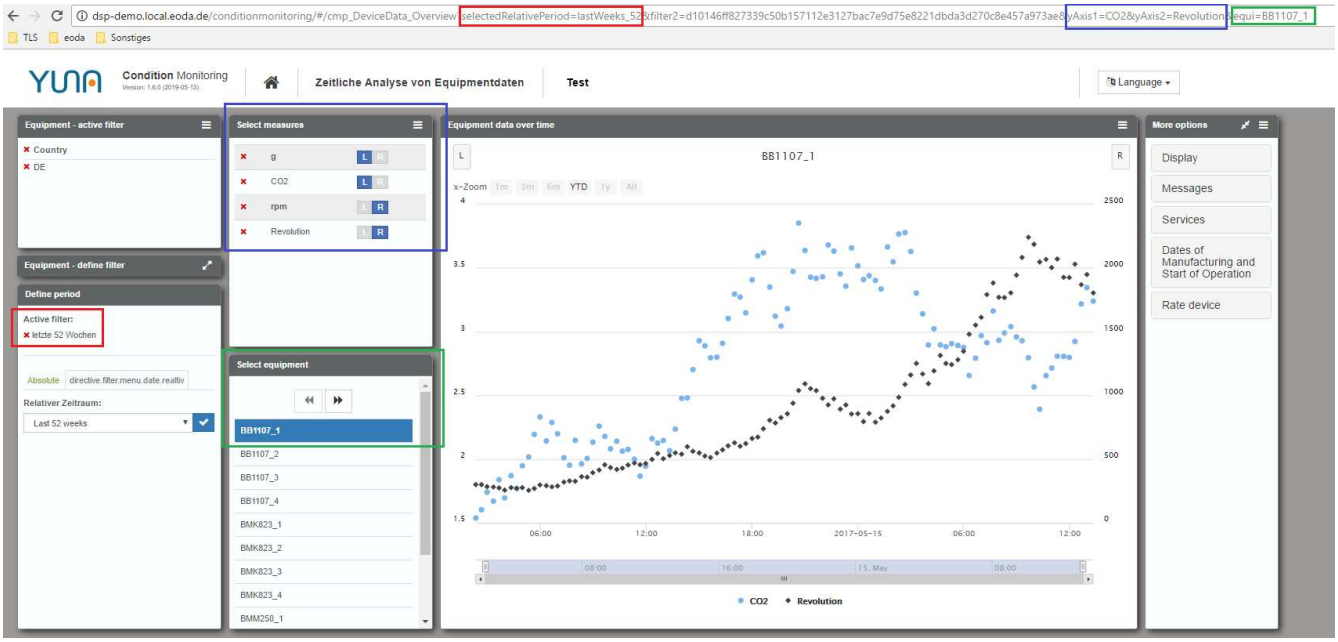
```
<position>
    <position>
        <x>2</x>
        <y>6</y>
    </position>
</position>
```

```
<size>
    <x>10</x>
    <y>7.4</y>
</size>
<widgettype>developmentstockchart</widgettype>
<dataID>qy_DevelopmentStockchart</dataID>
<triggerParams>
    <mandatory>
        <list>sensor</list>
        <list>equi</list>
    </mandatory>
    <optional>
        <list>startdate</list>
        <list>enddate</list>
        <list>selectedRelativePeriod</list>
    </optional>
</triggerParams>
<caption>
    <show>true</show>
    <label>Equipment data over time</label>
</caption>
```

JSON

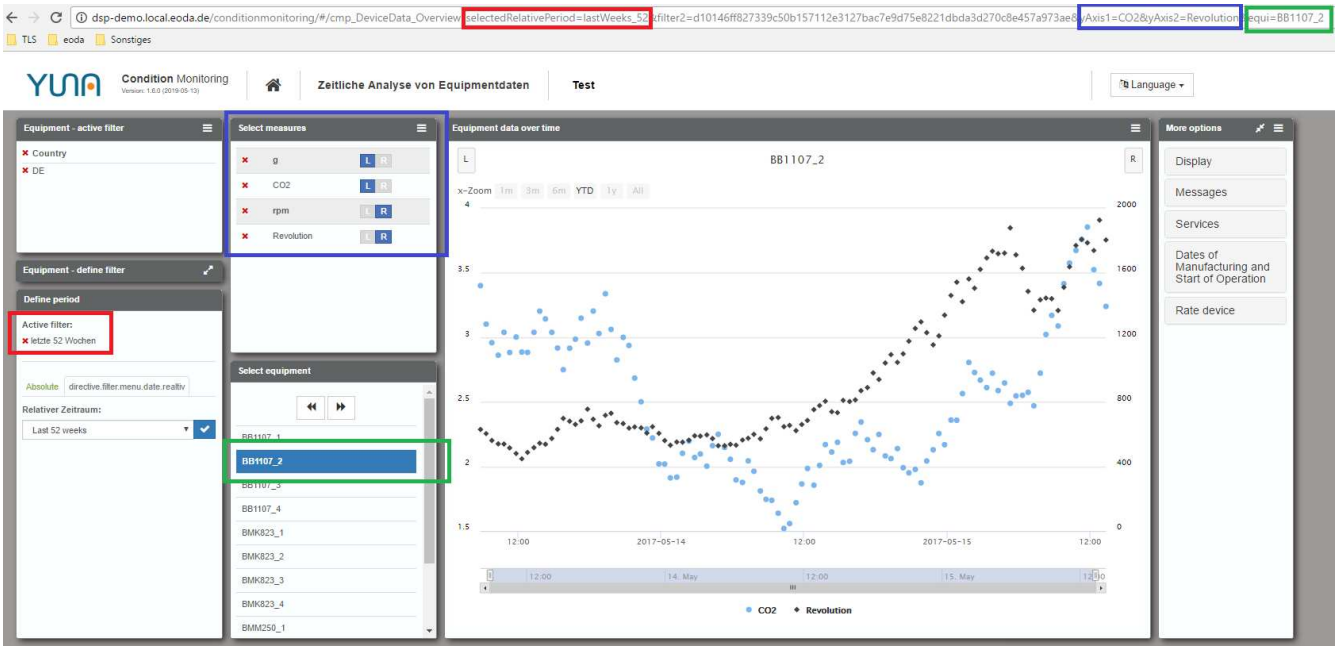
```
{
  "position": {
    "position": [
      2,
      6
    ]
  },
  "size": {
    "x": 10,
    "y": 7.4
  },
  "widgettype": "developmentstockchart",
  "dataID": "qy_DevelopmentStockchart",
  "triggerParams": {
    "mandatory": ["sensor", "equi"]
    "optional": ["startdate", "enddate", "selectedRelativePeriod"]
  },
  "caption": {
    "show": true,
    "label": "Equipment data over time",
  }
}
```

Das folgende Bild zeigt die Zusammenhänge zwischen den genannten Triggerparametern des Widgets und den im Stockchart dargestellten Daten:



Das Stockchart reagiert auf die Triggerparameter und stellt Sensordaten in Abhängigkeit der ausgewählten Equipmentnummer (Triggerparameter `equi`, grün umrandet), der ausgewählten Betrachtungsperiode (Triggerparameter `selectedRelativePeriod`, rot umrandet) sowie den ausgewählten Sensordaten (Triggerparameter `sensor, blau umrandet) dar. Ändert sich durch Eingaben des Anwenders (z.B. durch Auswahl eines anderen Equipments oder einer anderen Zeitperiode) etwas an den Parametern in der URL, so ändert sich auch die Darstellung bzw. der Inhalt des Stockcharts.

Im folgenden Bild wurde ein anderes Equipment ausgewählt, was eine Änderung des URL-Parameters `equi` zur Folge hat, wodurch sich auch die im Stockchart dargestellten Daten ändern.



Trigger-Verhalten

Die Trigger-Parameter müssen in **optional** (Optionale Parameter) und **mandatory** (Zwingend notwendige Parameter) unterteilt werden, um die verschiedenen Ausprägungen des Trigger-Verhaltens eines Widgets abbilden zu können. Grundsätzlich gilt, dass ein Widget die Abfrage der Daten erst startet, sobald alle **mandatory**-Parameter in der URL-Vorhanden sind.

Im folgendem Beispiel wird das Trigger-Verhalten der Widgets mit unterschiedlich definierten Trigger-Parametern verdeutlicht:

Widget A

```
<triggerParams>
    <mandatory>
        <list>Param1</list>
    </mandatory>
    <optional>
        <list>Param2</list>
    </optional>
</triggerParams>
```

Widget B

```
<triggerParams>
    <mandatory>
        <list>Param1</list>
    </mandatory>
    <optional>
        <list>Param3</list>
    </optional>
</triggerParams>
```

Widget C

```
<triggerParams>
    <mandatory>
        <list>Param1</list>
        <list>Param3</list>
    </mandatory>
</triggerParams>
```

Widget A

```
"triggerParams": {
    "mandatory": [ "Param1" ],
    "optional": [ "Param2" ]
}
```

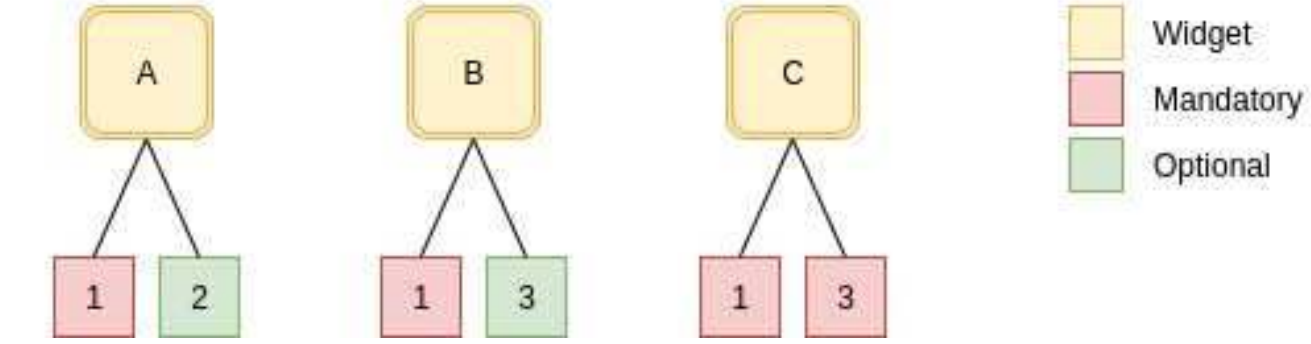
Widget B

```
"triggerParams": {
    "mandatory": [ "Param1" ],
    "optional": [ "Param3" ]
}
```

Widget C

```
triggerParams: {
    "mandatory": [ "Param1",
"Param3" ]
}
```

```
http://portalurl/ViewName?Param1=value1&Param2=value2
```



Sollte nun in der oben gezeigten URL durch den User eine Änderung an `Param1` vorgenommen werden, würden Widget A und B eine neue Abfrage starten, da hier alle **mandatory**-Parameter vorhanden sind. Bei einer Änderung von `Param2` würde nur Widget A eine neue Abfrage senden. Widget C wird somit in keinem der beiden Fälle neu getriggert, da dort `Param3` als **mandatory** -Parameter definiert wurde, der jedoch nicht in der URL vorhanden ist.

Diese Art der Definition verhindert illegale und unerwünschte Abfragen an das Backend, wodurch die Performance des Systems deutlich weniger beeinträchtigt wird.

Als zusätzliche Performance-Unterstützung, wurde durch die Module http-request-cancellation und den request-collector ein Cancel-Mechanismus implementiert. Diese prüfen, ob unterschiedliche Widgets in einer View die gleichen Daten aus dem Backend abfragen möchten und sorgen dafür, dass eine Abfrage nur einmal gestellt wird und deren Datenrückgabe an alle anfragenden Widgets verteilt wird. Gleichzeitig wird geprüft, ob eine angestoßene Abfrage in der laufenden Session bereits aktiv ist, um diese doppelte Abfrage anschließend zu canceln.

5.1.6. Filterfunktionen

Einleitung

Ein Filter gibt eine Menge ausgewählter Daten aus einer Datenquelle zurück. Andere Widgets einer View können auf einen Filter "hören". Diese Funktionalität basiert auf den verschiedenen Typen von Filtern und ihrer Hierarchie: Primärfilter, Sekundärfilter, Subfilter und (lokale) Tabellenfilter.

Ein Filter bzw. die gefilterten Daten sind z.B. die Grundlage für einen Issue-Workflow.

Funktionen von/für Filter(n)

- Neu anlegen
- Filter laden
- Unscharfe Suche
- Vorfilter
- Freitext Suche
- Update
- Filter löschen

Filtertypen

Primärfilter

Der Primärfilter ist die erste Filterebene für die Daten der Installierten Basis. Ausgewählte Primärfilter werden in der URL als eindeutiger Hashwert gespeichert und können so anderen Benutzern durch Weitergabe eines Links zu exakt dieser Portal View zugänglich gemacht werden. So ist im aktuellen YUNA Portal der einzig vorhandene Primärfilter die Abfrage der Stamm-, Sensor- und Meldungsdaten der Equipments, die anschließend in der Installierten Basis angezeigt werden.

Sekundärfilter

Ein Sekundärfilter ist hierarchisch einem Primärfilter unterstellt und von diesem abhängt. Er basiert demnach auf den durch den Primärfilter vorgefilterten Daten und wird zur weiteren logischen Einschränkung der anzuzeigenden Anlageninformationen genutzt.

Ein Sekundärfilter kann beispielsweise in der Filterverwaltung erstellt und anschließend im Portal geladen werden. Alternativ können die Stammdaten in den einzelnen Portal Views über das Filterwidget (filterdirective) gefiltert werden. Dort werden in einem Akkordeonmenü sämtliche Filterkriterien aufgeführt und anhand einer danebenstehenden Zahl in Klammern die Anzahl der Ausprägungen/Auswahlmöglichkeiten des jeweiligen Filterkriteriums angezeigt (z.B. bedeutet „Land (15)“, dass 15 Länder in den Daten hinterlegt sind und zur Filterung ausgewählt werden können).

Der ausgewählte Sekundärfilter wird im Widget Aktive Filter (selectedfilterdirective) angezeigt. Sekundärfilter werden ebenfalls in der URL als Hashwerte gespeichert und ermöglichen es so, dass die ausgewählten Filtereinstellungen über das Versenden von Links an weitere Benutzer weitergegeben werden können.

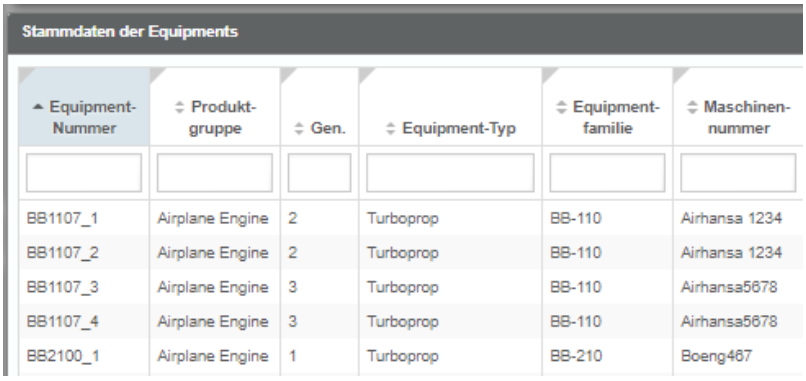
Subfilter

Mit einem Subfilter lassen sich Informationen zu einzelnen Anlagen anzeigen, die zuvor durch Primär- und Sekundärfilter ausgewählt wurden. Hierzu erscheinen die nach der vorherigen Filterung nicht herausgefilterten Anlagen in einer Liste (Einzelselektion (singlechoicedirective)) und können dort ausgewählt werden. Nach der Auswahl werden die ausgewählten Informationen zu der im Subfilter selektierten Anlage in den zugehörigen Widgets (Charts oder Tabellen) dargestellt.

(lokale) Tabellenfilter

Innerhalb einer Tabelle lässt sich der Inhalt weiter durch einen Tabellenfilter filtern. Hierzu kann in einer Spalte eines angezeigten Tabellenwidgets je nach Spaltentyp nach bestimmten Begriffen oder Werten gesucht werden. Eine Liste der nutzbaren Suchparameter findet sich im Dokubereich zu Spaltendefinition des Tabellenwidgets.

Tabellenfilter lassen sich nicht speichern. Somit muss die Filterung nach jedem Neuladen der Portal View erneut durchgeführt werden. Dies bedeutet auch, dass Tabellenfilter bei der Weitergabe von URLs nicht inbegriffen sind.

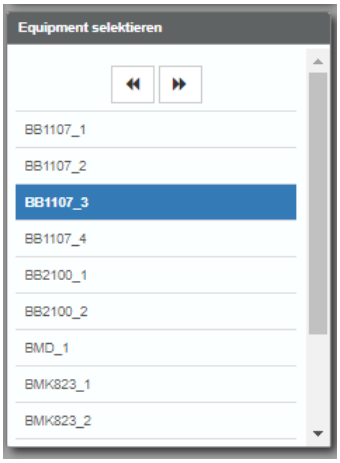


Selektionsmethoden

Die zu filternden Daten können auf unterschiedliche Weise ausgewählt werden. Es existieren folgende Selektionsmethoden:

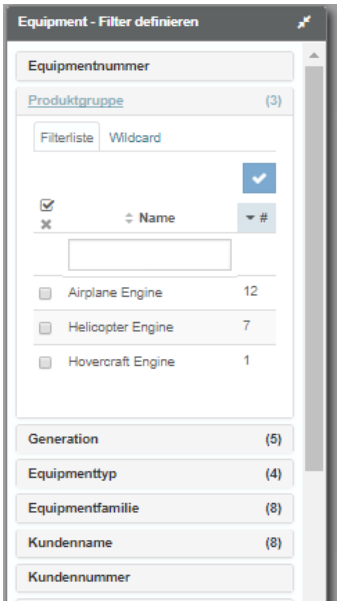
Einzelauswahl (SingleChoicedirective)

Bei einer Einzelselektion werden jeweils nur die Daten zu einer ausgewählten Einheit (z.B. ein Gerät) angezeigt, die zuvor aus einer Liste ausgewählt und durchgeschaltet werden kann.



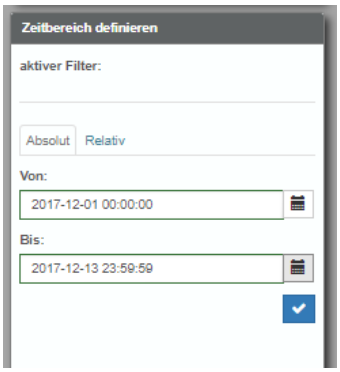
Mehrfachauswahl (filterdirective)

Bei der Mehrfachselektion kann ein Benutzer aus einer Liste mehrere Elemente auswählen, indem er die zu den Listenelementen gehördenden Checkboxen anklickt.



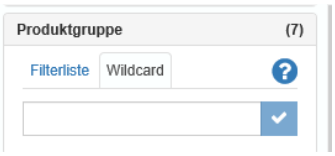
Bereichsauswahl (datesubfilterdirective)

Eine weitere Möglichkeit zur Auswahl und Filterung von Daten besteht in der Angabe von Zeitpunkten oder Zeiträumen.



Wildcard

Eine zusätzliche Option zur Filterung bzw. Abfrage von Daten besteht in der Nutzung des Wildcard-Felds. In diesem Feld kann gemäß der SQL-Syntax frei eine SQL-Abfrage erstellt werden, sofern sie nicht gegen definierte Abfrage-Restriktionen verstößt.



Filter kopieren und referenzieren:

Die Analyse von Frage- und Problemstellungen im YUNA Portal erfolgt in der Regel nicht über die gesamte Datenbasis der installierten Basis, sondern über Subsets (z.B. einzelne Equipment-Familien). Es kann sein, dass sich Filter über den Lebenszyklus einer Analyse verändern, indem

einzelne Equipments oder Equipmenttypen zur Filterpopulation hinzugefügt oder daraus entfernt werden. Aus diesem Grund kann bei der Erstellung von Sachverhalten und Jobs ausgewählt werden, ob ein dort zu verwendender Filter kopiert oder ob darauf referenziert wird.

Im Fall einer Kopie wird im weiteren Verlauf des Sachverhalt-Workflows für die Population die Bezeichnung "feste Definition" angezeigt. Das bedeutet, dass die für den Sachverhalt gewählte Population dauerhaft bestehen bleibt - auch, wenn anschließend jemand den Populationsfilter überarbeitet.

Wird die Option "Filter referenzieren" gewählt, so bleibt weiterhin die Populationsbezeichnung sichtbar und flexibel. Wird also nach der Wahl der Sachverhaltspopulation etwas am Filter geändert, wird diese Änderung automatisch auch auf die Sachverhaltspopulation angewendet.

Aktuelle Umsetzung:

Population eines Jobs

Die Population, über die eine automatisierte Auswertung zu einem Sachverhalt in Form eines Jobs erfolgt, ergibt sich immer (wie bereits umgesetzt) aus der Schnittmenge der Population des Sachverhalts und der Population des Jobs.

Grundsätzlich

Für die Definition der gültigen Population eines Jobs werden folgende zwei Möglichkeiten benötigt:

1. Feste Definition der Population im job
2. Referenz auf einen gespeicherten Filter als Population des Jobs mit automatischem Update

Eine weitere denkbare Möglichkeit ohne aktuellen ‚konkreten Bedarf ist:

3. Referenz auf einen gespeicherten Filter als Population des Jobs mit Update nach manueller Bestätigung

Umsetzung

Möglichkeit 1 und 2 werden zeitnah schrittweise umgesetzt.Möglichkeit 3 wird erst bei konkretem Bedarf angegangen, da als Voraussetzung für die Umsetzung im Gegensatz zu den anderen beiden Möglichkeiten ein komplexeres Bedienkonzept benötigt und auch von der Implementierung deutlich komplexer ist.

Details zu Möglichkeit 1 -"Feste Definition der Population im Job"

- Im ersten Umsetzungsschritt besteht die Möglichkeit einen gespeicherten Filter auszuwählen. Die Filterkriterien des gewählten Filters werden fest mit dem Sachverhalt verknüpft. Filterinfos wie der Name o.ä. werden nicht verknüpft.
- Im zweiten Umsetzungsschritt wird die Möglichkeit geschaffen, die Population über die gewohnten Filterkriterien direkt aus dem Formular zum Job heraus zu ändern.
- Die Definition der Population eines Jobs ist in den AdHoc-Analysen nicht über das bekannte Filterkonzept "Gespeicherte Filter laden" erreichbar.
- Eine Änderung der Population ist nur durch den Verantwortlichen des Jobs (nicht durch die Assistenten) im Job selbst möglich.
- Für eine Änderung der Population ist eine Deaktivierung und nach der Speicherung der

Änderungen eine erneute Freigabe des Jobs nötig.

Details zu Möglichkeit 2 -"Referenz auf einen gespeicherten Filter als Population des Jobs mit automatischem Update"

- Im ersten Umsetzungsschritt besteht die Möglichkeit einen gespeicherten Filter auszuwählen. Es wird eine Referenz auf den gewählten Filter mit dem Job verknüpft. Änderungen an dem Filter wirken sich automatisch auf den Job aus.
- Im zweiten Umsetzungsschritt werden die entsprechenden Jobverantwortlichen über die Änderung per Mail informiert. Die Auswirkung erfolgt weiterhin automatisch.
- Eine Änderung der Population auf Grund einer Änderung am referenzierten Filter hat keine Auswirkung auf den Status des Jobs.

Generell gilt:

- Ein gespeicherter Filter, auf den eine Referenz besteht, kann nicht gelöscht werden. Beim Versuch, einen solchen Filter zu löschen, wird der Anwender darauf hingewiesen, in welchen Sachverhalten / Jobs eine Referenz auf den entsprechenden Filter besteht und wer für diese verantwortlich ist. Wie in einem solchen Fall weiter verfahren wird, liegt in der Verantwortung der beteiligten Anwender.
- In der Filterverwaltung und im "Filter laden"-Dialog wird zu jedem Filter als weitere Information angezeigt, ob eine Referenz auf einen Filter besteht. Wenn ja, sieht der Nutzer in einer Detailsicht, in welchen Sachverhalten / Jobs eine Referenz auf den entsprechenden Filter definiert ist.

Weitere Filtereigenschaften

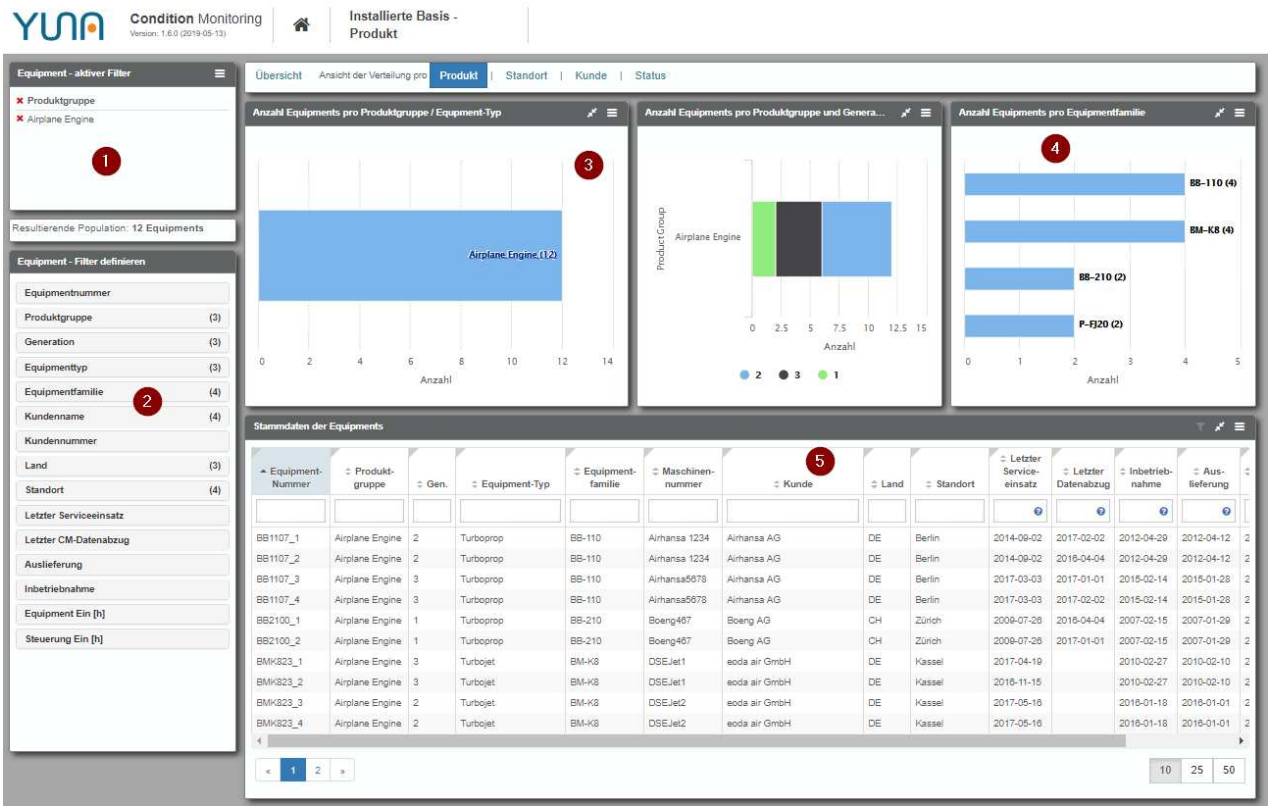
Filter können zudem weitere Eigenschaften aufweisen. So können sie beispielsweise **global** sein, also allen Benutzern des YUNA Portals zugänglich gemacht werden. Andernfalls sind sie *lokale bzw. eigene Filter *, also nur für den jeweiligen Benutzer selbst sicht- und nutzbar.

Weiterhin können Filter **statisch** sein. Dies bedeutet, dass die Auswahlmöglichkeiten eines Filters auf einer fixen Listen von Filterelementen beruhen, die sich nicht automatisch anpasst, sobald neue Elemente zu einer Tabelle hinzugefügt werden, auf die sich der Filter bezieht. Die Liste der Filterelemente muss bei statischen Filtern von einem Nutzer aktiv gepflegt werden, um Elemente hinzuzufügen, zu entfernen oder zu ändern. Das Gegenstück zu statischen Filtern sind **dynamische** Filter. Die Elemente eines dynamischen Filters ändern sich automatisch, sobald neue Elemente hinzukommen und die Filterbedingung zutrifft.

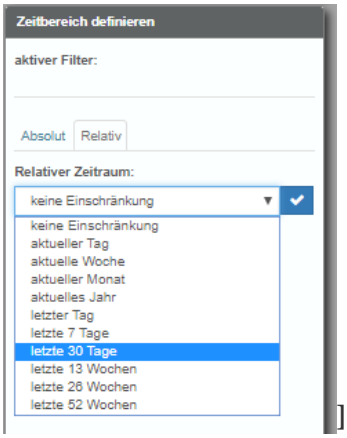
Wenn nicht eindeutig Equis angeben werden, ist ein Filter dynamisch, d.h. der Filter „Kunde = Bosch“ zeigt immer die zum Zeitpunkt der Abfrage vorhandene Equis mit dem Kunde Bosch.

Zudem können Filter **relativ oder absolut** wirken. Ein absoluter Filter bezieht sich immer auf ein genau definiertes Element oder einen genau definierten Bereich (z.B. 01.01.2017 - 15.01.2017). Ein relativer Filter ist dagegen beispielsweise die Angabe eines Zeitraums wie "die letzten 30 Tage". Diese Filterangabe liefert jeden Tag aufs neue andere Daten, da sich jeden Tag die Datenbasis ändert.

Der folgende Screenshot zeigt die View "Installierte Basis" mit dem voreingestellten Filter "Produktgruppe - Airplane Engine" - im Widget Aktive Filter (1) zu erkennen. Der ausgewählte Filter wurde im eigentliche Filter-Widget (2) definiert.



Durch Ausklappen der einzelnen Elemente lassen sich die Einstellungen sichtbar machen:



]

Die Diagramm-Widgets (3) und (4) und das Tabellen-Widget (5) stellen die gefilterten Daten dar.

5.1.7. Data ID - Definition der Daten

Was ist eine Data ID?

Eine Data_ID referenziert eindeutig auf Dashboard Inhalte wie Querys, Bilder oder Skripte. Beispiele hierfür sind z.B. das ChangeLog (changeLogHistory), die Query zur View "Widget-Abhängigkeiten" (qy_WidgetDependency) oder die Query zur Installierte Basis (qy_InstBasis_Overview).

In der Tabelle sp.DataID werden die Data_IDs mit weiteren Informationen wie einer Role_ID und einer DataType_ID verknüpft und diese Kombinationen mit einer DataID_ID versehen. Im Rahmen von Datenabfragen in Widgets lässt sich so anhand der DataID_IDs eindeutig regeln, welche Benutzerrollen auf welchen Dashboard Inhalt zugreifen dürfen.

Die Struktur der DataID in der Tabelle sp.DataID

Wert	Beschreibung
ID (= DataID_ID)	Die ID der Data_ID.
Role_ID	Definiert, welche Rolle die Data_ID nutzen darf.
Data_ID	Referenziert auf Dashboard Inhalt, der in verschiedenen Kontexten und aus Sicht verschiedener Benutzerrollen abgerufen werden kann.
DataType_ID	Definiert den Typ der Data_ID (QueryBuilder, StoredProcedure oder R-Skript).

Role_ID

Die Role_ID definiert, welche Benutzerrollen existieren und für Benutzer des YUNA Portals vergeben werden können. Diese Role_IDs sind in der Tabelle sp.Role hinterlegt. Pro Benutzer lassen sich Rollen vergeben, für die ebenfalls Rollenberechtigungen definiert und vergeben werden können.

Beispiele für Benutzerrollen im YUNA-Umfeld, die jeweils eine Role_ID besitzen können:

Role_ID	Bezeichnung
1	registered
2	System_Admin
3	Quality_Manager
4	Service_SCC
5	After_Sales
6	Development
7	Data_Analyst
8	Executive_Board
9	Controlling
10	Development_AdHoc
11	Service_NCC
12	EES_SoftwareTest

DataType_ID

Es gibt derzeit drei Typen von DataTypes, die in der Tabelle sp.DataType verwaltet und mit einer DataType_ID versehen werden:

DataType_ID	DataType	fest in sp.DataTypehinterlegt
1	QueryBuilder	ja
2	StoredProcedure	ja
3	RScript	nein

DataID_ID

Um deutlich zu machen, wie die Nutzung der DataID_ID funktioniert, soll das folgende kurze Beispiel genutzt werden:

DataID_ID	Role_ID	Data_ID	DataType_ID
1	7	changeLogHistory	1
2	2	changeLogHistory	1
3	2	qy_WidgetDependency	1

Ein Benutzer mit der Role_ID 7 (z.B. "Data Analyst") dürfte demnach auf das ChangeLog vom Datentyp QueryBuilder zugreifen. Diese Kombination hat die DataID_ID 1 zugewiesen bekommen.

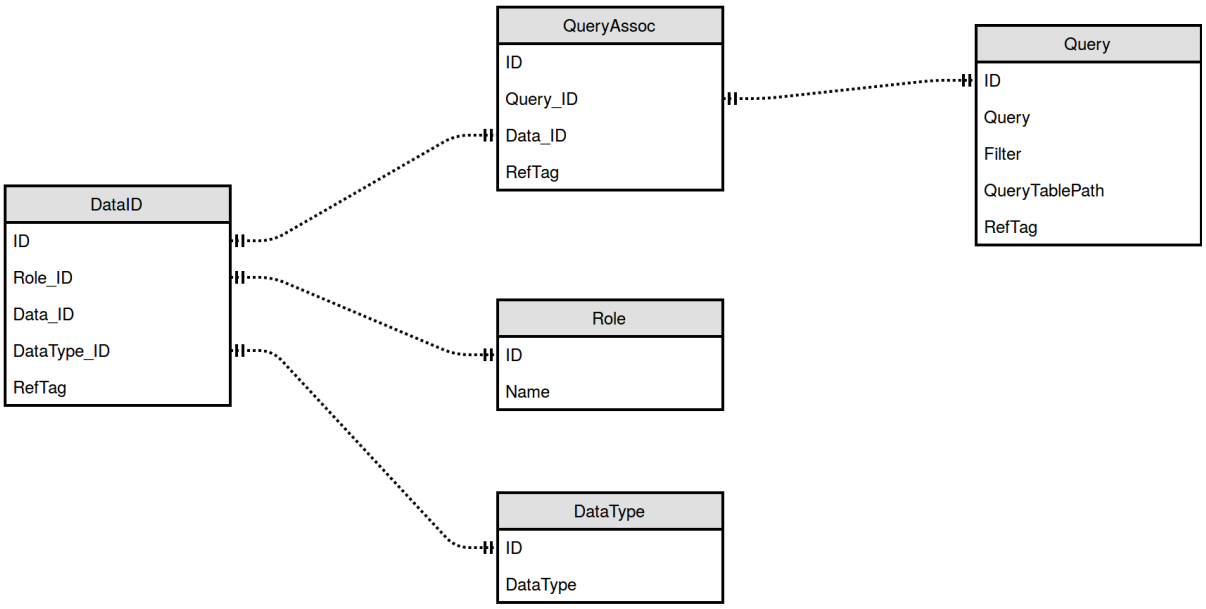
Ein Benutzer mit der Role_ID 2 (z.B. "Systemadministrator") dürfte in diesem Beispiel auf das ChangeLogHistory vom Typ QueryBuilder und auf die Query qy_WidgetDependency zugreifen, die mit den DataID_IDs 2 und 3 versehen wurden.

Nutzung der Data_ID

Im Kontext des YUNA Portals können mithilfe der Data_IDs bzw. der DataID_IDs (Kombination aus Data_ID, Role_ID und Datatype_ID) eindeutige Datenabfragen zur Einbindung in Widgets erstellt werden. Hierzu werden die DataID_IDs (aus sp.DataID) in der Tabelle sp.QueryAssoc mit Query_IDs (aus der Tabelle sp.Query) verknüpft, um eine QueryAssoc_ID zu erzeugen, die eine eindeutige, rollenspezifische und datenspezifische Datenabfrage erlaubt.

QueryAssoc_ID	Query_ID	DataID_ID
1	15	3

Schematischer Aufbau der DataID



Query.Filter:boolean entscheidet ob ein Filter angewandt werden kann.
Query.QueryTablePath:String Join für einen den Filter

Metadaten

Optional können zu einer DataID_ID Metadaten angegeben werden.

Metadaten-YunaML-Struktur-Überblick

```
<data>
 <meta>
  <description>
   <![CDATA[Hier kann ein die DataID_ID beschreibender Text angegeben werden.]]>
  </description>
  <fields>
   <!-- Liste an Feldern zur Beschreibung von erweiterten Datentypen im DSP-Context -->
  </fields>
 </meta>
</data>
```

```
<data>
 <meta>
  <description>
   <![CDATA[Hier könnte Ihre DataID-Beschreibung stehen!]]>
  </description>
  <fields>
   <field>
    <name>LastServiceMission</name>
     <type>
      <name>ZonedTimestamp</name>
       <params>
        <param>
         <name>timezone</name>
          <value>America/Belize</value>
        </param>
        <param>
         <name>absolute</name>
          <value>true</value>
        </param>
       </params>
     </type>
   </field>
  </fields>
 </meta>
</data>
```

Erweiterte Datentypen

Im Rahmen der Metadaten können Spalten als Träger erweiternder Datentypen gekennzeichnet werden. Diese Funktionalität wird über <field>-Tags im Listen-Tag <fields> definiert. Innerhalb von <field> wird die referenzierte Spalte mit dem Tag <name> und der entsprechende Typ über den

Container-Tag <type> festgelegt. Innerhalb von <type> wird über <name> der zu verwendende Typ ausgewählt. Ist der ausgewählte Typ parametrisierbar, so können die entsprechenden Parameter über den Listen-Tag <params>, und dessen Kind-Tag <param> eingestellt werden. Innerhalb eines <param>-Tags wird über <name> der entsprechende Parameter ausgewählt und mittels <value> ein Wert für diesen angegeben.

YUNA ML Aufbau von Field-Tags

```
<field>
    <name>
        <!-- Selektor zur Auswahl der zu beschreibenden Spalte -->
    </name>
    <type>
        <!-- Angabe des erweiternden Datentypes -->
        <name>
            <!-- Selektor zur Auswahl des erweiternden Datentypes -->
        </name>
        <params>
            <!-- Optionale Liste zur Konfiguration des ausgewählten Datentypes -->
            <param>
                <!-- Angabe eines Parameters -->
                <name>
                    <!-- Selektor zur Auswahl des zu konfigurierenden Parameters -->
                </name>
                <value>
                    <!-- Angabe des Parameter-Wertes -->
                </value>
            </param>
        </params>
    </type>
</field>
```

ZonedTimestamp

Der Metadatentyp ZonedTimestamp ermöglicht für Zeitstempel zum einen die nachträgliche Definition einer Ursprungs-Zeitzone, zum anderen kann hier auch direkt definiert werden, ob eine Darstellung des Zeitstempels in Lokalzeit zulässig ist.

Parameter	Wert	Default-Wert	Beschreibung
timezone	Valide ZeitzonenIds	UTC	Legt die Ursprungs-Zeitzone der Zeitstempel fest.
absolute	true/false	false	Wird absolute auf den Wert true gesetzt, so werden entsprechende Zeitstempel nicht in Lokalzeit umgerechnet.

User

Der Metadatentyp User erlaubt es eine UsererId mit dem entsprechenden DisplayName ('{lastname}, {firstname} ({username})') zu ersetzen, sodass im Portal konsistent der Displayname

angezeigt und von allen Komponenten verwendet wird. Das bedeutet, dass alle Funktionen, die auf die mit diesem Metadatentyp versehenen Daten zugreifen, bereits vorverarbeitete Daten verwenden. So kann, obwohl aus der Datenbank eine UserID zurückgeliefert wird, alle Funktionalität auf Basis des Displaynamen ausgeführt werden: Z.B. kann nach dem Displaynamen gefiltert werden, der Displayname kann als Spalteninhalt beim Tabellenexport exportiert werden.

Translatable

Der Metadatentyp Translatable wird genutzt, um eine Spalte als Übersetzbar zu kennzeichnen. Damit die Daten einer Spalte übersetzt werden können, muss die durch die DataID zurückgegebene Tabelle zusätzlich eine Spalte enthalten, deren Name der markierten Spalte mit dem Suffix 'TK' entspricht und die zu den Daten gehörenden Translation-Keys enthält.

Beispiel für die Verwendung des Metadatentyps 'Translatable'

Datenbanktabellen:

DSE-DataDB.data.engine

ID	Type	TypeTK
1	airplane	data.engine.airplane
2	helicopter	data.engine.helicopter
3	airplane	data.engine.airplane

Übersetzungsdatei (ohne andere Translation-Keys): "deutsch.json"

```
{
    "data.engine.airplane": "Flugzeug"
}
```

DataID

Friendly Name: Der Wert im Element <friendlyName> dient dem eindeutigen Referenzieren auf die Data-ID. Ist ein `friendly Name` angegeben, wird die Data-ID über ihn angesprochen und nicht über den Attribut name aus dem <data> Element.

```
<xml>
  <data name="qy_InstBasis_Overview" roles="System_Admin, AdHoc_Full_Issue">
    <friendlyName>qy_mdt_translatable</friendlyName>
    <type>QueryBuilder</type>
    <path>DSE-DataDB data engine</path>
    <query>
      <![CDATA[
        <QueryBuilder>
            <select>
            <table>DSE-DataDB</table>
            <table>data</table>
            <table>engine</table>
```

```
            <fields>
            <field>
                <name>Type</name>
                <prefix>A</prefix>
            </field>
            <field>
                <name>TypeTK</name>
                <prefix>A</prefix>
            </field>
            </fields>
            <where/>
            <groupby/>
            <orderby/>
            <limit>0</limit>
            <limitoffset>0</limitoffset>
        </select>
        </QueryBuilder>
        ]]>
    </query>
    <meta>
        <fields>
            <field>
                <name>Type</name>
                <type>
                    <name>Translatable</name>
                </type>
            </field>
        </fields>
    </meta>
  </data>
</xml>
```

Mit dieser DataID und den beschriebenen Datenbanktabellen, wird im Portal zu jedem Wert in der Spalte 'TypeTK' versucht ein für die ausgewählte Sprache passender Wert zu ermitteln. Ist dies nicht möglich wird der Wert der Originalspalte 'Type' angezeigt.

<table>
<tr><th>Ausgewählte Sprache</th><th>Deutsch</th><th>Englisch</th></tr>
<tr><td>Angezeigte Tabelle</td><td><table><tr><th>ID</th><th>Type</th></tr><tr><td>1</td><td>Flugzeug</td></tr><tr><td>2</td><td>helicopter</td></tr><tr><td>3</td><td>Flugzeug</td></tr></table></td><td><table><tr><th>ID</th><th>Type</th></tr><tr><td>1</td><td>airplane</td></tr><tr><td>2</td><td>helicopter</td></tr><tr><td>3</td><td>airplane</td></tr></table></td></tr>
<tr><td>Erläuterung</td><td>Im Deutschen liegt nur für den Translation-Key 'data.engine.airplane' eine Übersetzung vor, daher wird für den Translation-Key 'data.engine.helicopter' der ursprüngliche Wert der Spalte 'Type' aus der Tabelle 'DSE-DataDB.data.engine' angezeigt.</td><td>Da für Englisch noch keine Übersetzung definiert ist, werden die Daten der mit dem Metadatentyp 'Translatable' versehenen Spalte angezeigt.</td></tr>
</table>

5.1.8. Global Administrator Message

Es ist möglich, eine globale AdminMessage in dem Portal zu konfigurieren. Diese kann benutzt werden um User über anstehende Updates zu informieren. Diese wird im Header-Bereich des Portals für alle Benutzern angezeigt.

Die Nachricht muss direkt in der Tabelle einer Tabelle [portal].[AdminMsg] in der Portal Datenbank eingetragen werden. Zusätzlich können die Farbe und das Scrollverhalten konfiguriert werden.

	ID	Text	Active	Color	BackgroundColor	Scroll	ChangedAt
1	3	This is a Message	1	#000000	#ffffff	0	2019-02-21 08:27:50.059 +00:00

Konfigurationsoptionen für die Administrator-Nachricht.

Spalte	Beschreibung
Text	anzuzeigende Nachricht
Active	Ob die Nachricht angezeigt werden soll. Bei '0' ist die Nachricht inaktiv, bei '1 ist die Nachricht aktiv.
Color	Zeichenfarbe als html hexadecimal code (https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Colors/Color_picker_tool)
BackgroundColor	Hintergrundfarbe als html hexadecimal code (https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Colors/Color_picker_tool)
Scroll	Bei '0' ist das Scrolling inaktive, bei '1' ist es aktiv. Wenn die Nachrichte so groß ist, dass sie nicht in die verfügbare Breite passt, ist das Scrolling immer aktiv.
ChangedAt	Datum zu dem die Nachricht geändert wurde

Wenn diese Tabelle mehrere Zeilen enthält, wird die letze Nachricht angezeigt.

5.1.9. Definition des Reload Counters für den Table-Widget Memory Leak Workaround

Hintergrund des Workarounds

Aufgrund von Memory-Leaks in einer für das Tabellen-Widget verwendeten Bibliothek (ngTable) wurde ein Workaround eingebaut, der dafür sorgt, dass das Portal nach einer gewissen Anzahl von Aufrufen von Portal-Elementen, in denen ngTable verwendet wird (Table-Widget und Single Choice Directive), beim nächsten Aufruf neu geladen wird. So kann vermieden werden, dass ein häufiger Aufruf von Sichten mit Tabellen und Einzelselektionen den Speicherbedarf des Portals im Browser in die Höhe treibt und so das System verlangsamt.

Definition des Counters:

Die Einstellung des Counters zur erfolgt direkt in der Datenbank:

DB-Tabelle	Parametername	Eigenschaft	Mögliche Werte	Default
portal.ConfigStore	tableReloadCount	Anzahl von ngTable-Abfragen, bevor ein Reload der Portal-View erzwungen wird.	alle positiven ganzzahligen Werte ab 1	10

Wirkweise des Workarounds:

Wird beispielsweise in der Datenbank der Parameter `tableReloadCount` auf den Wert 5 gesetzt, so erlaubt das Portal 5 Abfragen von Portalelementen, die ngTable beinhalten (z.B. 5 Table-Widgets in der Portal-View), bevor die Portal-View komplett neu geladen wird.

Wenn der Parameter `tableReloadCount` nicht anders definiert wurde, erhält er den Default-Wert 10. Das heißt, bei der 11. Abfrage an ngTable-Elemente würde die PortalView neu geladen.

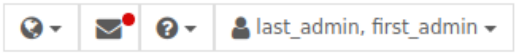
5.1.10. YUNA-Vorfilter

Grundlagen

Vorfilter können nur von einem Administrator angelegt und aktualisiert werden. Mithilfe eines Vorfilters werden die angezeigten Daten für einzelne Benutzer vorab gefiltert. Über das Filter-Symbol in der Menüleiste kann ein Benutzer erkennen ob ein Vorfilter aktiv ist. Er kann aber nicht erkennen welche Daten gefiltert werden.

Status des Vorfilters

Das Feature Vorfilter verfügt über drei Status. In der Menübar ist erkennbar, in welchem Status sich das Feature befindet:

Status	Bedeutung	
Deaktiviert	Das Feature Vorfilter ist über die Konfiguration deaktiviert.In der Menübar erscheint kein Filter-Symbol.	
Inaktiv	Es wurden **keine** Vorfilter für diesen Benutzer definiert.In der Menüleiste erscheint ein ausgegrautes Filter-Symbol.	
Aktiv	Es wurde mindestens ein Vorfilter für diesen Benutzer definiert. In der Menüleiste erscheint ein blaues Filter-Symbol.	

Verwendung

Vorfilterservice konfigurieren

Um das Feature der Vorfilter nutzen zu können, muss zunächst der Service aktiviert werden.Die Konfiguration des Vorfilterservice ist hier beschrieben: Vorfilter-Konfiguration

Vorfilter definieren

Vorfilter können über eine **Stored Procedure** oder eine **REST-Schnittstelle** konfiguriert werden.

Ein Vorfilter wird immer für eine Kombination aus Benutzername und Filter-ID definiert. Er kann eine Kombination aus mehreren existierenden Filter sein und wird über die Filter-Hashes der Filter referenziert. Die Filter eines Vorfilters werden untereinander mit **ODER** verknüpft.

Ist für den angemeldeten Benutzer ein Vorfilter eingetragen, so wird dieser bei jeder Anwendung eines Filters an die Datenbankabfrage angehangen.

Example 2. Vorfilterverknüpfung

Wir setzen voraus, das folgende vier Filter definiert sind :
NameFilter: userName = Mustermann
*AgeFilter1_*: age = 18
AgeFilter2: age = 30
AgeFilter3: age = 60

Für den *Benutzer1* besteht der **Vorfilter** aus *AgeFilter1*, *AgeFilter2* und *AgeFilter3*.

Wenn für den *Benutzer1* eine Datenbankabfrage ausgeführt wird, auf die der *NameFilter* angewandt werden soll, wird zusätzlich der definierte Vorfilter angehangen.

Syntaktisch

```
SELECT *
FROM User
WHERE userName = Mustermann
AND (age = 18
  OR age = 30
  OR age = 60)
```

Es werden alle Personen mit dem Namen "Mustermann" die entweder 18, 30 oder 60 Jahre alt sind aus der Tabelle geladen.

Anlage / Aktualisierung von Vorfilter

Über die Stored Procedure

Die Stored Procedure ***sp_insertOrUpdatePrefilter*** hat drei Parameter:

Parameter	Type	Beschreibung
@username	nvarchar	Der Loginname des Benutzers, für den der Vorfilter angelegt werden sollen.
@filterID	bigint	Die Id eines existierenden Filters, für den der Vorfilter hinzugefügt werden sollen.
@filterHashes	nvarchar	Eine JSON-Liste mit den Hashes der Filter, die für den Benutzer und den angegebenen Filter als Vorfilter definiert werden sollen.

Ist in der Datenbank für den angegebenen Benutzer und den angegebenen Filter bereits ein Vorfilter eingetragen, wird dieser beim Aufruf der Stored Procedure überschrieben. Im Falle von ungültigen Parametern werden entsprechende SQL-Fehlermeldungen generiert.

Beispiel für den Aufruf der Stored Procedure zur Definition eines zusammengesetztenVorfiltersd für den Benutzer MyUser und den Filter mit der ID 2

```
EXEC portal.sp_insertOrUpdatePrefilter
    @username = 'MyUser'
   ,@filterID = 2
   ,@filterHashes = '["96bd5c2dc96a64135c1dd0dac6475ab5e5e833b1e33fa7aa9cdcba35b9a3896a"
                     ,"6b2739096042f02055f32cab50db8516dfe8b10836acb5ca6a0ce9a69ad01969"]';
GO
```

Über die Rest-Schnittstelle

Das von der restschnittstelle genutzte Objekt für Vorfilter enthält die ID des Benutzers, die ID des Filters und eine Liste von Filter-Hashes, die den Vorfilter ausmachen.

Abrufen aller Vorfilter

GET	backend/de.eoda.dse.portal.filter.prefilter.rest/
Erfolg	**Code:** 200 **Typ:** Application/Json **Inhalt:** PreFilter Liste
Kein Ergebnis	**Code:** 204

Abrufen aller einzigartigen Vorfilter

GET	backend/de.eoda.dse.portal.filter.prefilter.rest/unique
Erfolg	**Code:** 200 **Typ:** Application/Json **Inhalt:** Liste der einzigartigen PreFilter Hashes pro Filter ID
Kein Ergebnis	**Code:** 204

Abrufen aller Vorfilter für den aktuell angemeldeten Benutzer

GET	backend/de.eoda.dse.portal.filter.prefilter.rest/hasfilter
Erfolg	**Code:** 200 **Typ:** Application/Json **Inhalt:** PreFilter Liste
Kein Ergebnis	**Code:** 204
Fehler	**Code:** 404 Filter-Service ist nicht aktiv

Abrufen eines Vorfilters für eine Filter ID und einen Benutzer

GET	backend/de.eoda.dse.portal.filter.prefilter.rest/{filterId}/user/{userId}
URL-Parameter	**filterId:** Die ID des Filters, dessen Vorfilter abgerufen werden soll. **userId:** Die ID des Benutzers, dessen Vorfilter abgerufen werden soll.

Erfolg	**Code:** 200 **Typ:** Application/Json **Inhalt:** PreFilter Liste

Anlegen/Aktualisieren eines Vorfilters

POST	backend/de.eoda.dse.portal.filter.prefilter.rest/{filterId}/user/{userId}
URL-Parameter	**filterId:** Die ID des Filters, für den der Vorfilter angelegt werden soll. **userId:** Die ID des Benutzers, für den der Vorfilter angelegt werden soll.
Body	Eine Json-Liste von Filter-Hashes `["96bd5c2dc96a64135c1dd0dac6475ab5e5e833b1e33fa7aa9cdcba35b9a3896a","07e6a59b4319f9a68729289851f955eb03be13fa3f4b86920a093a9902da1301"]`
Erfolg	**Code:** 200 **Typ:** Application/Json **Inhalt:** PreFilter Der Vorfilter wurde angelegt/aktualisiert
Fehler	**Code:** 400 Ein leerer Body wurde an den Endpunkt gesendet.

Löschen eines Vorfilters

DELETE	backend/de.eoda.dse.portal.filter.prefilter.rest/{filterId}/user/{userId}
URL-Parameter	**filterId:** Die ID des Filters, für den der Vorfilter angelegt werden soll. **userId:** Die ID des Benutzers, für den der Vorfilter angelegt werden soll.
Erfolg	**Code:** 200 **Typ:** Application/Json **Inhalt:** PreFilter Der Vorfilter wurde gelöscht

5.1.11. Datasource

Was ist die Datasource?

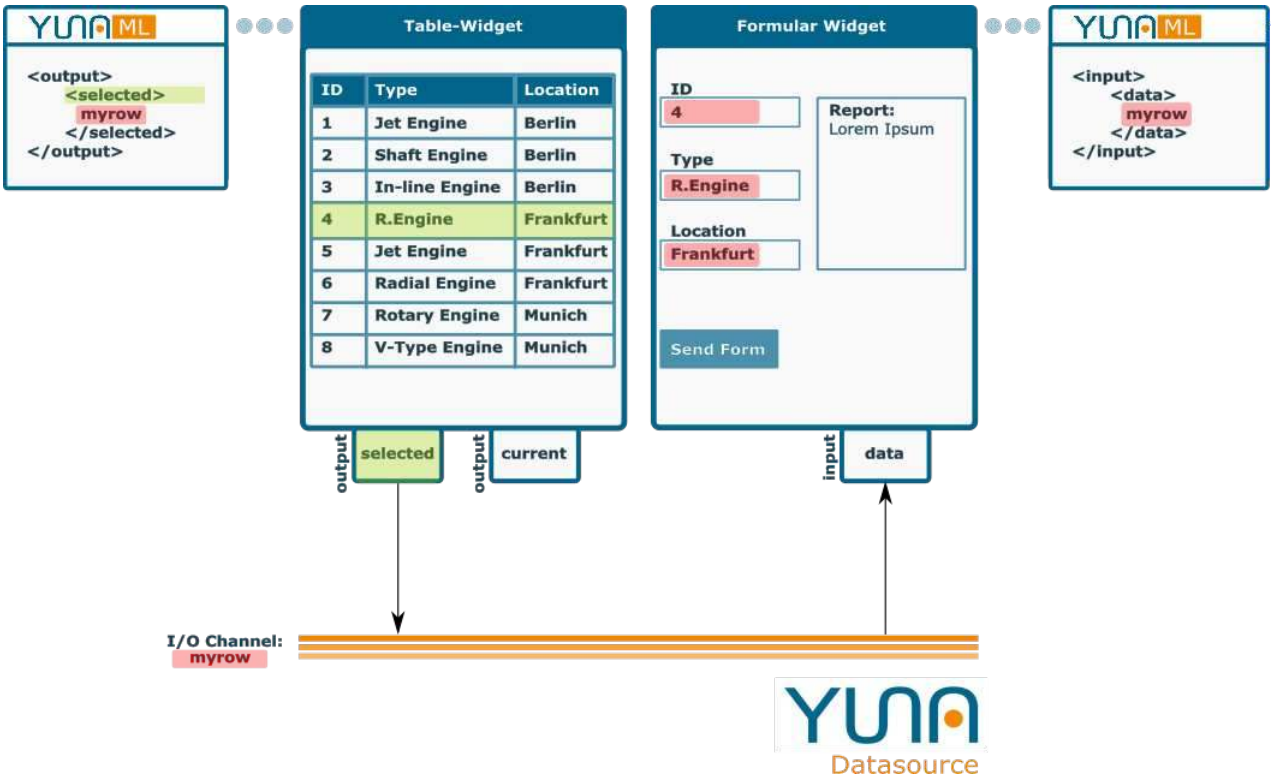
Die Datasource ist ein Konzept mit dem Widgets untereinander Daten austauschen. Der Mechanismus besteht aus zwei zentralen Elementen:

1. **Inputs oder Outputs** sind an Widgets oder IO-Providern fest definiert (in der Regel, das Formularwidget stellt hier eine Ausnahme dar) und stellen Daten bereit bzw. konsumieren sie. Die Ausgänge stellen immer einen bestimmten Datentyp bereit, die ausschließlich mit Eingängen des selben Datentyps verknüpft werden können. Die Datentypen können aus der Dokumentation der einzelnen Widgets entnommen werden.
2. **Channel oder Kanäle** werden über YUNAML definiert und stellen eine oder mehrere Verbindungen zwischen Ein- und Ausgänge her. Ein Channel benötigt immer einen Identifier in Form eines einzigartigen Namens.

Ein Channel erlaubt grundsätzlich mehrere Verknüpfungen. Mehrere Widgets können Daten in den Channel einspielen oder auslesen.

Example 3. **Verknüpfung von Tabellen und Formular-Widget mittels Datasource**

Das nachfolgende Beispiel zeigt, wie der Ausgang **selected** eines Tabellen-Widgets mit Hilfe des Channels **myrow** mit dem Eingang **data** des Formular-Widgets verknüpft werden kann.



TableWidget

```
        <widget name="table_default">
            <cols>
                <list>
                    <field>ID</field>
                ....
                </list>
                ....
            </cols>
            <outputs>
                <selected>myrow</selected>
            </outputs>
            ....
            <widgettype>tabledirective</widgettype>
        </widget>
```

FormWidget

```
        <widget name="IV_default">
            ....
```

```
            <inputs>
                <data>myrow</data>
            </inputs>
            <widgettype>formwidget</widgettype>
        </widget>
```

Folgende Widgets und IOProvider stellen Datasource Inputs- und Outputs bereit:

Widget	Input-/Output-Channel		
TableWidget	**Typ**	**Name**	**Beschreibung**
	Output	current	Die über die Spaltenheader gefilterten und sortierten Daten der Tabelle.
	Output	selected	Die aktuell ausgewählten Zeilen der Tabelle.
	Output	selectedIndex	Der aktuell ausgwählte Zeilenindex der Tabelle.
	Input	data	Die in der Tabelle darzustellenden Daten. Diese müssen in einem vorgegebenen Format vorliegen, das derzeit nur durch die Output-Channel anderer Tabellenwidgets und des DataID-Provider bereitgestellt.
	Input	selectionIndex	Der in der Tabelle auszuwählende Zeilenindex.
ImageViewer	**Typ**	**Name**	**Beschreibung**
	Output	current	Der aktuell ausgwählte Bildindex.
	Input	hash	Nimmt Tabellendaten entgegen, die Hashes enthalten, die Bilder referenzieren. Die Tabellendaten müssen Spalte 'Hash' enthalten, diese muss aber nicht zwangsweise angezeigt werden.
	Input	src	URLs oder Data-URIs zur Darstellung im ImageViewer.
	Input	current	Der im ImageViewer auzuwählende Bildindex.
DataGridWidget	**Typ**	**Name**	**Beschreibung**
	Output	current	Die über die Spaltenheader gefilterten und sortierten Daten des DataGrids.
	Output	selected	Die aktuell ausgewählten Zeilen des DataGrids.
	Output	selctedIndex	Der aktuell ausgewählte Zeilenindex des DataGrids.
	Output	cellEdited	Die Änderung an der aktuellen Zelle.
	Input	data	Die im DataGrid darzustellenden Daten.
	Input	selectionIndex	Der im DataGrid auszuwählende Zeilenindex.

Widget	Input-/Output-Channel		
FormularWidget	**Typ**	**Name**	**Beschreibung**
	Output	submit	Stellt nach Klick auf den Submit-Button Daten aus dem Formularwidget für den Outputchannel bereit.
	Input	*(beliebig)*	Nimmt jegliche Daten entgegen. Ein Formularwidget kann über beliebig viele Input-Channel mit unterschiedlichen Namen verfügen.

Das Datasource Konzept bietet keine Deeplink Funktionalität

5.1.12. Benachrichtigungen

Benachrichtigungen sind zur Zeit in einer Evaluationsphase. Das Feature muss daher über den Eintrag "message.active" im Configstore aktiviert werden.

Verwendungszweck

Benachrichtigungen einen schnellen und einfachen Weg zu Autausch von Informationen innerhalb von YUNA bereit.

Den Messenger aufrufen

Das Benachrichtigungs-Icon in der YUNA-Kopfleiste ermöglicht aus jedem Dashboard den direkten Zugriff auf die Benachrichtigungen. Wenn neue Nachrichten vorliegen, wird dies durch einen Indikator angezeigt. Der Indikator wird in festen Intervallen geupdatet.

Nachrichteneingang

Der Nachrichteneingang zeigt alle Nachrichten, die ein Benutzer empfangen hat. Die Liste kann nach Titel, Sender und Zeitstempel der Nachricht gefiltert werden um schnell bestimmte Nachrichten zu finden

Nachrichten löschen

Wenn eine Nachricht ausgewählt ist, kann sie aus dem Nachrichteneinagng gelöscht werden. Beachten sie, dass das Löschen einer Nachricht nicht rückgängig gemacht werden kann. Das Löschen einer Nachricht beeinflusst andere Benutzer nicht.

Nachrichtenausgang

Der Nachrichteneingang zeigt alle Nachrichten, die ein Benutzer gesendet hat. Die Liste kann nach Titel, Empfängern und Zeitstempel der Nachricht gefiltert werden. Im Gegensatz zum Nachrichteneingang können Nachrichten hier nicht gelöscht werden.

Neue Nachrichten

Benutzer können Nachrichten an andere Benutzer mithilfe des Formulars für neue Nachrichten senden. Das Formular stellt eine einfache Auswahl der möglichen Empfänger bereit. Die Felder 'Titel' und 'Empfänger' müssen befüllt werden, der Nachrichteninhalt ist optional. Nach dem

Absenden der Nachricht kann sie im Nachrichtenausgang des Senders geöffnet werden Eine gesendete Nachricht kann nicht verändert werden.

Solange das Formular für neue Nachrichten geöffnet ist, kann das Fenster nicht durch Klicken neben das Fenster oder Benutzung der ESC-Taste geschlossen werden um Datenverlust zu vermeiden.

Informationen für YUNA-Administratoren

Das Intervall mit dem der Indikator für ungelesene Nachrichten aktualisiert wird, kann im Configstore konfiguriert werden. Der Standardwert beträgt 60 Sekunden.

Über einen Konfigurationseintrag in der config.yaml können die automatisch durch das System versandten Nachrichten regelmäßig aufgeräumt werden. Siehe "Löschen von Systembenachrichtigungen" unter Script Logging

Informationen für Benutzer der Nachrichten-REST-API

Benachrichtigungen unterstützen aktuell die Felder 'title' und 'text' des Nachrichteninhalts. Alle anderen Felder werden ignoriert. Der 'title' wird als einfacher Text interpretiert, während der 'text' unter Verwendung von HTML formatiert werden kann.

5.1.13. REST-API für Nachrichten

Verwendungszweck

Die Nachrichten-REST-API stellt einfach Methoden zum Senden und Arufen von Nachrichten in YUNA bereit. Nachrichten unterstützen ein flexibles Inhalts-Format. Dadurch kann sie einfach auf verschiedene Anwendungsfälle und die Anbindung externer Systeme angepasst werden.

Authentifizierung

Cookie-basierte Authentifizierung und HTTP-Basic-Authentifizierung werden derzeit unterstützt. Letztere wird aktuell empfohlen, sofern eine sichere Verbindung, wie zum Beispiel durch Verwendung von HTTPS, für die Abfragen verwendet wird.

Nachrichten-Objekte

Schema

```
{
    "id": integer,
    "senderId": integer,
    "receiverIds": [integer],
    "sentAt": {
        "epochSecond": integer,
        "nano": integer
    },
    "readAt": {
        "epochSecond": integer,
```



```
        "nano": integer
    },
    "content": {
        "propKey1": "string",
        "propKey2": "string",
        "propKey3": "string",
        ...
    }
}
```

Erläuterung

- *id*: Die ID der abgerufenen Nachricht.
- *senderId*: Die Benutzer-ID des Senders der Nachricht.
- *receiverIds*: Die Benutzer-IDs der Empfänger der Nachricht
- *sentAt*: Der Zeitpunkt zu dem die Nachricht gesendet wurde.
- *readAt*: Der Zeitpunkt zu dem die Nachricht zum ersten Mal vom aktuellen Benutzer abgerufen wurde. Wird nie gesetzt, wenn der abrufende Benutzer nicht Empfänger der Nachricht ist.
- *content*: Ein Key-Value-Objekt, das die Inhalte der Nachricht enthält.

Wichtiger Hinweis: Werte für Inhalts-Eigenschaften werden standardmäßig nicht zurückgegeben. Alle Endpunkte die Nachrichten-Objekte zurückgeben, unterstützen URL-Parameter, über die gesteuert werden kann, welche Inhalts-Werte zurückgegeben werden. Dies ist notwendig, um das versehentliche Abrufen großer Datenmengen zu verhindern und das initiale Betrachten der Nachrichten-Objekte zu erlauben.

Beispiel

```
{
  "id": 5,
  "senderId": 1,
  "receiverIds": [2, 3],
  "sentAt": {
    "epochSecond": 1573808131,
    "nano": 587000000
  },
  "readAt": {
    "epochSecond": 1573808133,
    "nano": 200000000
  },
  "content": {
    "title": "Lorem Ipsum",
    "text": null
  }
}
```

Endpunkte

Methode	URL	Beschreibung
GET	`/unread/count`	Anzahl ungelesener Nachrichten abrufen
GET	`/inbox`	Alle vom Nutzer empfangenen sichtbaren Nachrichten abrufen
GET	`/outbox`	Alle vom Nutzer empfangene Nachrichten abrufen
POST	`/`	Eine neue Nachricht senden
POST	`/{messageId}`	Eine bestimmte Nachricht abrufen
POST	`/{messageId}/hide`	Eine empfangene Nachricht verstecken

Anzahl ungelesener Nachrichten abrufen

Gibt die Anzahl ungelesener Nachrichten zurück, die ein Benutzer empfangen hat. Nachrichten zählen als ungelesen, solange die Nachricht nicht explizit über die API abgerufen wurde.

GET	`/de.eoda.dse.portal.message/unread/count`
Erfolg	**Code:** *200* **Typ:** *application/json* **Inhalt:** *integer*
Fehler	**Code:** *401* Wird zurückgegeben, wenn der Benutzer nicht authentifiziert ist

Beispielaufruf:

```
curl -X GET https://<yuna-backend-url>/de.eoda.dse.portal.message/unread/count -u myUsername
```

Antwort:

```
2
```

Alle vom Nutzer empfangenen sichtbaren Nachrichten abrufen

Gibt alle von dem Benutzer empfangenen Nachrichten zurück, die nicht versteckt wurden.

GET	`/de.eoda.dse.portal.message/inbox`
URL Parameters	*returnAllContentValues=(boolean)* :Ob alle Inhalts-Werte zurückgegeben werden sollen *content=(string)* : Spezifischer Nachrichten-Inhalt für den der Wert zurückgegeben werden soll[1]
Erfolg	**Code:** *200* **Typ:** *application/json* **Inhalt:** `[Message, Message, ⋯]`[2]
Fehler	**Code:** *401* Wird zurückgegeben, wenn der Benutzer nicht authentifiziert ist

Anmerkungen:
[1]: Kann mehrfach angegeben werden um die Werte mehrerer ausgewählter Inhalte abzurufen. (z.B.: `/de.eoda.dse.portal.message/inbox?content=title&content=text` um die Werte der Eigenschaften 'title' und 'text', aber keiner anderen Inhalts-Felder, abzurufen.
[2]: Weiterführende Informationen finden Sie im Abschnitt über Nachrichten-Objekte.

Beispielaufruf:

```
curl -X GET https://<yuna-backend-url>/de.eoda.dse.portal.message/inbox -u myUsername
```

Antwort:


```
[
  {
    "id": 1,
    "senderId": 1,
    "receiverIds": [3],
    "sentAt": {
      "epochSecond": 1573808131,
      "nano": 587000000
    },
    "readAt": {
      "epochSecond": 1573809247,
      "nano": 200000000
    },
    "content": {
      "text": null,
      "title": null
    }
  },
  {
    "id": 5,
    "senderId": 2,
    "receiverIds": [3, 4, 8],
    "sentAt": {
      "epochSecond": 1543821524,
      "nano": 600000000
    },
    "readAt": null,
    "content": {
      "text": null,
      "title": null
    }
  }
]
```


Alle vom Benutzer empfangene Nachrichten abrufen

Gibt alle vom Benutzer versendeten Nachrichten zurück.

GET	`/de.eoda.dse.portal.message/outbox`
URL Parameters	*returnAllContentValues=(boolean)* :Ob alle Inhalts-Werte zurückgegeben werden sollen *content=(string)* : Spezifischer Nachrichten-Inhalt für den der Wert zurückgegeben werden soll[1]
Erfolg	**Code:** *200* **Typ:** *application/json* **Inhalt:** `[Message, Message, ⋯]`[2]
Fehler	**Code:** *401* Wird zurückgegeben, wenn der Benutzer nicht authentifiziert ist

Anmerkungen:
[1]: Kann mehrfach angegeben werden um die Werte mehrerer ausgewählter Inhalte abzurufen. (z.B.: `/de.eoda.dse.portal.message/inbox?content=title&content=text` um die Werte der Eigenschaften 'title' und 'text', aber keiner anderen Inhalts-Felder, abzurufen.
[2]: Weiterführende Informationen finden Sie im Abschnitt über Nachrichten-Objekte.

Beispielaufruf:

```
curl -X GET https://<yuna-backend-url>/de.eoda.dse.portal.message/outbox?returnAllContentValues=true -u myUsername
```

Antwort:


```
[
  {
    "id": 1,
    "senderId": 1,
    "receiverIds": [3],
    "sentAt": {
      "epochSecond": 1573808131,
      "nano": 587000000
    },
    "readAt": {
      "epochSecond": 1573809247,
      "nano": 200000000
    },
    "content": {
      "text": "Lorem Ipsum",
      "title": "Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore..."
    }
  }
]
```


Eine neue Nachricht senden

Versendet eine neue Nachricht. Eine Nachricht kann mit beliebigem Inhalt (Eigenschaft *content*) an eine beliebige Anzahl Nutzer gesendet werden (Eigenschaft *reveiverIds*). Beachten sie, dass der Messenger in der YUNA-Oberfläche aktuell nur die Inhalts-Eigenschaften *title* und *text* unterstützt.

POST	`/de.eoda.dse.portal.message/`
Body:	**Typ:** *application/json* **Schema:** [1] `{ receiverIds: [integer], content: { "propKey1": "string", ⋯ } }`
Erfolg	**Code:** *200* **Typ:** *application/json* **Inhalt:** *MessageID (integer)*
Fehler	**Code:** *401* Wird zurückgegeben, wenn der Benutzer nicht authentifiziert ist

Anmerkungen:
[1]: Alle anderen Eigenschaften werden ignoriert, unabhängig davon ob sie Teil des Nachrichten-Objekt-Schemas) sind.

Beispielaufruf:

```
curl -X POST https://<yuna-backend-url>/de.eoda.dse.portal.message/ -H "Content-Type: application/json" -u myUsername -d @"lorem_ipsum.json" -u myUsername
```

Body:

```
{
  "receiverIds": [3],
  "content": {
    "text": "Lorem Ipsum",
    "title": "Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore..."
  }
}
```

Antwort:

```
5
```

Eine bestimmte Nachricht abrufen

Ruft bestimmte Nachrichten ab. Benutzer können nur Nachrichten abrufen, die sie selbst gesendet oder empfangen haben. Beim ersten Abruf einer Nachricht wird die Eigenschaft *readAt* gesetzt.

POST[1]	`/de.eoda.dse.portal.message/{messageId}`
URL Parameters	*returnAllContentValues=(boolean)* :Ob alle Inhalts-Werte zurückgegeben werden sollen *content=(string)* : Spezifischer Nachrichten-Inhalt für den der Wert zurückgegeben werden soll[2]
Body:	- [1]
Erfolg	**Code:** *200* **Typ:** *application/json* **Inhalt:** *Message*

POST[1]	`/de.eoda.dse.portal.message/{messageId}`
Erfolg	**Code:** *204* Wird zurückgegeben, wenn der Benutzer weder Sender oder Empfänger der Nachricht ist oder die Nachricht nicht existiert.
Fehler	**Code:** *401* Wird zurückgegeben, wenn der Benutzer nicht authentifiziert ist

Anmerkungen:
[1]: Diese Abfrage setzt die Eigenschaft *readAt* der abgerufenen Nachricht wenn sie vorher den Wert *null* hatte. Da der Server-Zustand verändert wird, wird ein POST-Request verwendet.
[2]: Kann mehrfach angegeben werden um die Werte mehrerer ausgewählter Inhalte abzurufen. (z.B.: `/de.eoda.dse.portal.message/inbox?content=title&content=text` um die Werte der Eigenschaften 'title' und 'text', aber keiner anderen Inhalts-Felder, abzurufen.\

Beispielaufruf:

```
curl -X POST https://<yuna-backend-url>/de.eoda.dse.portal.message/5 -u myUsername
```

Antwort:

```
{
  "id": 5,
  "senderId": 1,
  "receiverIds": [3],
  "sentAt": {
    "epochSecond": 1573808131,
    "nano": 587000000
  },
  "readAt": {
    "epochSecond": 1573809247,
    "nano": 200000000
  },
  "content": {
    "text": null,
    "title": null
  }
}
```

Eine empfangene Nachricht verstecken

Zeigt eine Nachricht nicht mehr im Nachrichteneingang eines Benutzers an.

POST	`/de.eoda.dse.portal.message/{messageId}/hide`
Body:	- [1]
Erfolg	**Code:** *200* **Typ:** *application/json* **Inhalt:** *boolean* : Ob eine Nachricht versteckt wurde.

POST	`/de.eoda.dse.portal.message/{messageId}/hide`
Fehler	**Code:** *401* Wird zurückgegeben, wenn der Benutzer nicht authentifiziert ist

Anmerkungen:
[1]: Die Nachricht wird unabhängig von dem übergebenen Nachrichten-Body versteckt.

Beispielaufruf:

```
curl -X POST https://<yuna-backend-url>/de.eoda.dse.portal.message/5/hide -u myUsername
```

Antwort:

```
true
```

5.2. Inhalte für das Dashboard erstellen

Dashboard Inhalte werden im Kontext von YUNA bei der Erstellung in YunaML definiert. Anschließend wird der erzeugte Content über das Tool dseDep auf die Datenbank deployed, von der das YUNA Portal seine Daten bezieht. Beim Deployment in die Datenbank werden Queries als YunaML abgelegt (im Schema QueryBuilder), anderer Content wird von spDep ins JSON-Format überführt und so in die Datenbank geschrieben.

Im Rahmen dieser Dokumentation für Dashboard Developer wird zunächst der YunaML-Code zur Erstellung des Dashboard Inhalts aufgeführt, gefolgt von einer Wiedergabe des JSON-Codes, der durch dseDep erzeugt und in der Datenbank abgelegt wird. Desweiteren werden viele Beispiele durch Screenshots ergänzt, um den Output des YunaML greifbarer zu machen.

In den folgenden Abschnitten finden sich Informationen zu Tools und Vorgehensweisen für die Erstellung und das Deployment von Dashboards.

5.2.1. Tools für Dashboard Developer

Für die Erstellung und Bearbeitung von YunaML-Code stehen Ihnen unterschiedliche Tools zur Unterstützung zur Verfügung, z.B.:

- Visual Studio Code oder
- Notepad++.

Beispiel 1: Visual Studio Code

Laden Sie sich zunächst Visual Studio Code herunter und installieren Sie es. Nach der Installation sollten Sie eine XML-Erweiterung wie z.B. den XML Formatter einbinden, um sich die Arbeit zu erleichtern. So erkennt die XML-Erweiterung das XML-Schema, färbt Code ein und rückt zusammenhängenden Code in Tabs ein.

Beispiel 2: Notepad++

Laden Sie sich Notepad++ herunter und installieren Sie es. Nach der Installation sollten Sie sich eine XML-Erweiterung einbinden, um sich die Arbeit zu erleichtern. So erkennt die XML-Erweiterung das XML-Schema, färbt Code ein und rückt zusammenhängenden Code in Tabs ein.

5.2.2. Query Builder

Um Inhalte im Portal anzuzeigen, muss definiert werden, welche Daten von wo aus der Datenbank abgerufen werden sollen. Dies geschieht über die Verwendung von sogenannten Queries. Im Kontext von YUNA-ML werden alle Dashboard Inhalte, also auch Queries, über XML definiert. Da die Datenbank jedoch SQL-Statements benötigt, muss die XML-Definition übersetzt werden.

Query Builder

Der QueryBuilder ist eine XML-Meta-Sprache für MSSQL. Verwendet wird der QueryBuilder im CMP zum Erstellen von

- Abfragen
- Security-Aspekten wie der Verhinderung von DropTable o.ä.

YUNA ML Allgemeine Struktur einer SELECT-Query

```
<xml>
    <data name="template_qy_NameOfDataID" roles="System_Admin, AdHoc_Full_Issue">
        <!-- Setting the dataID -->
        <friendlyName>template_qy_NameOfDataID</friendlyName>
        <!-- StoredProcedure/QueryBuilder -->
        <type>QueryBuilder</type>
        <!-- Use filter on this query -->
        <filter>false</filter>
        <!-- Optional Path: Database Scheme Table -->
        <path/>
        <!-- Data-Query build with the QueryBilder (also see "QueryBuilder" in the Content Developer Guide) -->
        <query>
            <![CDATA[
                <QueryBuilder>
                    <select>
                        <table>CM-DataDB</table>
                        <table>data</table>
                        <table>DataTable</table>
                        <fields>
                            <asteriskfield />
                        </fields>
                        <where/>
                        <orderby/>
                        <limit>0</limit>
                        <limitoffset>0</limitoffset>
                    </select>
                    <dictionary/>
                </QueryBuilder>
            ]]>
        </query>
    </data>
</xml>
```

SELECT-Field Typen

Name	Beispiel	SQL	Erklärung

field	`<field>`
	`<name>Spaltenname</name>`
	`<prefix>Tabellenalias</prefix>`
	`<as>Spaltenalias</as>`
	`</field>`
		SELECT *	Wählt alle Spalten aus.

distinctfield	`<distinctfield>`
	`<field>`
	`<name>Spaltenname</name>`
	`<prefix>A</prefix>`
	`<as>Spaltenalias</as>`
	`</field>`
	`</distinctfield>`
		SELECT Spaltenname	Wählt bestimmte Spalten aus. Optional ist der <prefix>. <prefix> wird beim Auftreten mehrerer Tabellen nacheinander als A, B, C, ... automatisch definiert.

asteriskfield	`<asteriskfield/>`	SELECT DISTINCT Spaltenname	Gibt eine Liste unterschiedlicher Werte zurück, in der keine Duplikate der selektierten Spalte mehr vorhanden sind. Mehrere <field>-Angaben sind möglich.

countfield	`<countfield>`
	`<name>Spaltenname</name>`
	`<prefix>A</prefix>`
	`<as>Anzahl</as>`
	`</countfield>`
		SELECT COUNT(Spaltenname)	Anzahl der Zeilen ohne NULL-Werte in der Spalte. Die Angabe eines Namens für <as> ist obligatorisch.

countdistinctfield	`<countdistinctfield>`
	`<name>Spaltenname</name>`
	`<prefix>A</prefix>`
	`<as>EquiCount</as>`
	`</countdistinctfield>`
		SELECT COUNT DISTINCT(Spaltenname)	Gibt die Anzahl unterschiedlicher Werte einer selektierten Spalte zurück.

maxfield	`<maxfield>`
	`<name>Spaltenname</name>`
	`<prefix>A</prefix>`
	`<as>Maximum</as>`
	`</maxfield>`
		SELECT MAX(Spaltenname)	Maximum der Spalte. Die Angabe eines Namens <as> ist obligatorisch.

Name	Beispiel	SQL	Erklärung
minfield	`<minfield><name>Spaltename</name><prefix>A</prefix><as>Minimum</as></minfield>`	SELECT MIN(Spaltenname)	Minimum der Spalte. Die Angabe eines Namens <as> ist obligatorisch.
staticfield	`<staticfield><as>stringexample</as><staticValue class="java.lang.String">abcdefg</staticValue></staticfield><staticfield><as>numberexample</as><staticValue class="java.lang.Integer">123456</staticValue></staticfield>`	SELECT "abcdefg" AS stringexample SELECT 123456 AS numberexample	Statische Werte für eine Tabellenspalte
concatfield	`<concatfield><as>Message</as><innerfield><name>Spaltenname1</name><prefix>A</prefix></innerfield><innerstaticfield><staticValue class="java.lang.String"/></innerstaticfield><innerfield><name>Spaltenname2</name><prefix>A</prefix></innerfield></concatfield>`	SELECT CONCAT(Spaltenname1, ' ', Spaltenname2)	Bildet die SQL-CONCAT-Funktion ab. Als Felder der Funktion sind innerfield (analog zu field) und innerstaticfield (analog zu staticfield) möglich. Voraussichtlich ab Version 0.14.

TABLE

Die eindeutige Angabe von Tabellen erfolgt über die Zusammensetzung aus Datenbankname, Schema und Tabellennamen.

Name	Beispiel	SQL	Erklärung
table	`<table>Datenbank</table><table>Schema</table><table>Name</table>`	FROM [Datenbank].[Schema].[Name]	Gibt die Tabelle an, auf die sich die Abfrage bezieht.

JOIN

Im XML steht derzeit der INNER JOIN zur Verfügung.

Name	Beispiel	SQL	Erklärung
innerjoin	<innerjoin> <a> <name>SpalteTabelleA</name> <prefix>A</prefix> <name>SpalteTabelleB</name> <prefix>B</prefix> <table>DatenbankTabelleB</table> <table>PräfixTabelleB</table> <table>NameTabelleB</table> <as>B</as> </innerjoin>	INNER JOIN [TabelleB] AS B ON A.SpalteTabelleA = B.SpalteTabelleB	<innerjoin> lassen sich aneinander hängen. <a> gibt die erste Spalte nach dem 'ON' an, die zweite.
leftjoin	s.o.	LEFT JOIN [TabelleB] AS B ON A.SpalteTabelleA = B.SpalteTabelleB	(ab Version 0.12)
rightjoin	s.o.	RIGHT JOIN [TabelleB] AS B ON A.SpalteTabelleA = B.SpalteTabelleB	(ab Version 0.14)

WHERE

Liste der Vergleichsoperatoren

Innerhalb von <where></where> können Bedingungen angegeben werden.

Name	Beispiel	SQL	Erklärung
equal	<equal> <field> <name>Spaltenname</name> <prefix>A</prefix> </field> <value class="java.lang.String" >Wert</value> </equal>	Spaltenname = Wert	Gibt die Tabellenzeilen zurück, die die Bedingung erfüllen. Werte für das Attribut class in <value>: "java.lang.String", "java.lang.Integer", "java.lang.Boolean"
notequal	s.o.	Spaltenname != Wert	s.o.
less	s.o.	Spaltenname < Wert	s.o.
lessequal	s.o.	Spaltenname ⇐ Wert	s.o.
greater	s.o.	Spaltenname > Wert	s.o.

Name	Beispiel	SQL	Erklärung
greaterequal	s.o.	Spaltenname >= Wert	s.o.
like	<like> <field> <name>Spaltenname</name> <prefix>A</prefix> </field> <value class="java.lang.String" >%search%</value> </like>	Spaltenname LIKE "%seach%"	Wildcards sind vor dem Suchbegriff, danach oder beides möglich.
null	s.o.	Spaltenwert IS NULL	Tabellenzeilen, in denen der Spaltenwert NULL ist.
notnull	s.o.	Spaltenname IS NOT NULL	Tabellenzeilen, in denen der Spaltenwert nicht NULL ist.
in	<in> <field> <name>Spaltenname</name> <prefix>A</prefix> </field> <value> <object class="java.lang.String">Wert1</objec t> <object class="java.lang.String">Wert2</objec t> <object class="java.lang.String">Wert3</objec t> </value> </in>	Spaltenname IN (Wert1, Wert2, Wert3)	Beachten: Attribut class gehört zu <object>, nicht zu <value> wie an anderer Stelle.

Weitere Operatoren

Zusätzlich besteht mit ID die Möglichkeit, triggerParams aus der URL in die Abfrage einzubeziehen, die <value> ersetzen. Weiterhin können die Vergleichoperatoren logisch verknüpft werden.

Name	Beispiel	SQL	Erklärung
ID	<less> <field> <ID>triggerParam</ID> <name>Spaltenname</name> <prefix>A</prefix> </field> </less>	Spaltenname < triggerParam	Die Abfrage übernimmt einen der im Widget definierten triggerParams aus der URL. Somit können die <value> bei den Vergleichsoperatoren durch variable Werte ersetzt werden.

Name	Beispiel	SQL	Erklärung

and	`<equal>...</equal>`
	`<and/>`
	`<less>...</less>`
		Spaltename1 = Wert1 AND Spaltenname2 < Wert2	Logisches UND von WHERE-Bedingungen

or	`<equal>...</equal>`
	`<or/>`
	`<less>...</less>`
		Spaltename1 = Wert1 OR Spaltenname2 < Wert2	Logisches ODER von WHERE-Bedingungen

not	`<not/>`
	`<like>...</like>`
	`<and/>`
	`<not/>`
	`<equal>...</equal>`
		NOT Spaltenname1 LIKE "%search%" AND NOT Spaltenname2 = Wert2	Logisches NICHT von WHERE-Bedingungen (ab Version 0.12)

Reservierte Key Words für Backend generierte Values

Name	Beispiel	SQL	Erklärung

UserID	`<where>`
	`  <equal>`
	`  <field>`
	`  <ID>UserID</ID>`
	`  <name>ID</name>`
	`  <prefix>A</prefix>`
	`  </field>`
	`  <value class="java.lang.Integer">0</value>`
	`  </equal>`
	`</where>`
		-	Das Backend übergibt die UserID (des aktuell eingeloggten Users).

RoleID	`<where>`
	`  <equal>`
	`  <field>`
	`  <ID>RoleID</ID>`
	`  <name>ID</name>`
	`  <prefix>A</prefix>`
	`  </field>`
	`  <value class="java.lang.Integer">0</value>`
	`  </equal>`
	`</where>`
		-	Das Backend übergibt die RoleID (des aktuell eingeloggten Users).

GROUP BY

Name	Beispiel	SQL	Erklärung
groupby	<groupby> <field> <name>Spaltenname1</name> <prefix>A</prefix> </field> </groupby>	GROUP BY Spaltenname1, Spaltenname2	Gruppiert nach Spalten. Sinnvoll für Gruppierungsfunktionen countfield, maxfield, minfield.

ORDER BY

Name	Beispiel	SQL	Erklärung
orderby	<orderby> <field order="ASCENDING"> <name>Spaltenname1</name> <prefix>A</prefix> </field> </orderby>	ORDER BY Spaltenname1 ASC, Spaltenname2 DESC	Sortiert nach Werten der Spaltennamen. Gültige Werte for das Attribut *order* sind "ASCENDING" oder "DESCENDING".

LIMIT

Name	Beispiel	SQL	Erklärung
limit	<limit>n</limit>	SELECT TOP n <MSSQL> LIMIT n <MYSQL>	Liefert die ersten n Werte einer Abfrage
limitoffset	<limitoffset>m</limitoffset>	LIMIT n, m <MYSQL>	Liefert n Werte einer Abfrage, beginnend mit dem m-ten Wert

(NOLOCK)

MSSQL-spezifisch. Sperrung anderer Abfragen während der Ausführung.

Abbrechen von Datenbankabfragen

Über den Query-Builder definierte Abfragen werden üblicherweise abgebrochen, wenn ein Benutzer während der Durchführung der Abfrage das aufrufende Dashboard schließt, wird diese abgebrochen. Soll dies nicht nicht passieren, kann dies entweder global über die Service-Konfiguration oder für eine einzelne Query konfiguriert werden:

Name	Beispiel	Erklärung
cancellable	<cancelable>false</cancelable>	Verhindert das Abbrechen der definierten Datenbankabfrage

Beispiel für das Setzen des <cancelable>-Tags

```
<xml>
```

```
    <data name="qy_query_with_disabled_cancellation"
        roles="System_Admin, AdHoc_Full_Issue">
        <friendlyName>qy_DataQueryIntegrationTest_ExecWithDelay_CancelableFalse</friendlyName>
        <type>StoredProcedure</type>
        <filter>true</filter>
        <path>DataDB data</path>
        <params>
            <cancelable>false</cancelable>
        </params>
        <query><![CDATA[
        <QueryBuilder>
              <exec>
                <procedure>DSE-DataDB</procedure>
                <procedure>data</procedure>
                <procedure>sp_nameOfStoredProcedure</procedure>
            </exec>
        </QueryBuilder>
        ]]></query>
    </data>
</xml>
```

EXEC-Query. Aufruf einer Stored Procedure

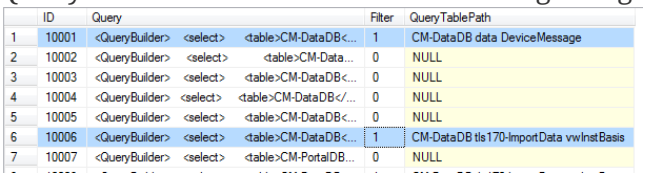
Name	Beispiel	SQL	Erklärung
exec			Aufruf einer Stored Procedure mit Parametern.

Nutzung von Filtern und aus dem Backend gesetzten Parametern

Name	Beispiel	SQL	Erklärung
exec	<QueryBuilder> <exec> <procedure>CM-PortalDB</procedure> <procedure>dev</procedure> <procedure>sp_Issue_EquiList</procedure> <parameters> <parameter> <id>filter2</id> <parameter>InstBasis</parameter> <type>VARCHAR</type> </parameter> <parameter> <id>UserID</id> <parameter>UserID</parameter> <type>INT</type> </parameter> </parameters> </exec> </QueryBuilder>	Beispielaufruf einer Stored Procedure mit Filter und User ID DECLARE @InstBasis VARCHAR(MAX) DECLARE @UserID INT EXEC [CM-PortalDB].[dev].[sp_Issue_EquiList] @InstBasis = N'(([InstBasis].[CommissioningDate] >= '2016-09-14' AND [InstBasis].[CommissioningDate] <= '2016-10-15') AND ([InstBasis].[ProductGroup] IN (N'TruDisk')))', @UserID = 31	Aufruf einer Stored Procedure mit Filtern und User ID.

Falls die Spalten, auf die der Filter wirkt, nicht vorhanden sind, wird die Basistabelle des Filters (z.B. die Installierte Basis) über die definierten Felder mit der Tabelle, auf der gefiltert werden soll, gejoined:

1. **Eintrag in [CM-PortalDB].[sp].[Query]** mit
   Filter = 1
   QueryTablePath = Tabelle auf die der angehangene Filter wirken soll



	ID	Query	Filter	QueryTablePath
1	10001	<QueryBuilder> <select> <table>CM-DataDB<...	1	CM-DataDB data DeviceMessage
2	10002	<QueryBuilder> <select> <table>CM-Data...	0	NULL
3	10003	<QueryBuilder> <select> <table>CM-DataDB<...	0	NULL
4	10004	<QueryBuilder> <select> <table>CM-DataDB</...	0	NULL
5	10005	<QueryBuilder> <select> <table>CM-DataDB<...	0	NULL
6	10006	<QueryBuilder> <select> <table>CM-DataDB<...	1	CM-DataDB tls170-ImportData vwInstBasis
7	10007	<QueryBuilder> <select> <table>CM-PortalDB...	0	NULL

2. **Eintrag in [CM-PortalDB].[dev].[FilterAssocTMP]**
   Query_ID = ID der Query, an die ein Filter angehangen werden soll
   Filter_ID = ID des Filters, der an die Query angehangen werden soll
   FilterField = Feld des Filters für den JOIN
   QueryResultField = Feld des Querys für den JOIN

Im Query-XML wird definiert:

- über *<filter>true</filter>*, dass ein Filter angehangen wird;
- über *<path> tabelle </path>*, auf welcher Tabelle gefiltert wird
- und über z.B. den folgenden XML-Code, über welche Felder der JOIN für den Filter erfolgt, sofern dieser auf einer anderen Tabelle definiert ist als das Query:

```
<data>
...
   <FilterAssociation_2>
      <Filter_ID>1</Filter_ID>
         <FilterField>EquipmentNo</FilterField>
         <QueryResultField>EquipmentNo</QueryResultField>
   </FilterAssociation_2>
</data>
```

Filter werden automatisch vom Backend als B,C, ... (in der Reihenfolge, wie sie angehängt werden = FilterAssociation) gejoint - d.h. wenn in der Query noch ein weiterer Join vorhanden ist, muss diese Tabelle mit einem anderen Prefix versehen werden.

EXEC-Query: Reservierte Key Words für Parameter

Name	Beispiel	SQL	Erklärung
filterX	<parameter> <id>filter2</id> <parameter>InstBasis</parameter> <type>VARCHAR</type> </parameter>	**Beispielaufruf einer Stored Procedure mit Filter und User ID** DECLARE @InstBasis VARCHAR(MAX) ... @InstBasis = N'(([InstBasis].[CommissioningDate] >= '2016-09-14' AND [InstBasis].[CommissioningDate] <= '2016-10-15') AND ([InstBasis].[ProductGroup] IN (N'TruDisk')))'...	Ein Filter kann aus dem Frontend übergeben werden. Dafür wird der Hash (der Value) im backend aufgelöst und der zu MSSQL konvertiert String wird dann an die Stored Procedure übergeben
UserID	<parameter> <id>UserID</id> <parameter>UserID</parameter> <type>INT</type> </parameter>	Beispielaufruf einer Stored Procedure mit Filter und User ID DECLARE @UserID INT ... @UserID = 31	UserID stellt eine gesonderte Funktion bereit. Im Backend wird diesem Parameter die UserID übergeben.
currentLanguage	<parameter> <id>currentLanguage</id> <parameter> languageParam</parameter> <type>VARCHAR</type> </parameter>	Beispielaufruf einer Stored Procedure ... DECLARE @language VARCHAR(20) = @languageParam ...	Die aktuell ausgewählte Sprache eines Benutzers wird bei Abfragen als Parameter 'currentLanguage' an die DataId übergeben. Dieser kann genutzt werden, um abhängig von dem Parameter verschiedene Spalten mit Übersetzungen anzuzeigen.

5.2.3. Stored Procedures

Über Stored Procedures können Dashboard Developer sich Datenabfragen aus der Datenbank erleichtern

```
<xml>
```

```
    <!--
                    Copyright (c) 2017 eoda GmbH
           All Rights Reserved, see LICENSE.TXT for further detail

           More information about configuring this template
           can be found in the Content Developer Guide:
           "Data_ID" and "QueryBuilder"
-->
    <data name="template_qy_NameOfDataID" roles="System_Admin, AdHoc_Full_Issue">
        <!-- Setting the dataID -->
        <friendlyName>template_qy_NameOfDataID</friendlyName>
        <!-- StoredProcedure/QueryBuilder -->
        <type>StoredProcedure</type>
        <!-- Use filter on this query -->
        <filter>false</filter>
        <!-- Optional Path: Database Scheme Table -->
        <path/>
        <!-- Data-Query build with the QueryBilder (also see "QueryBuilder" in the Content Developer Guide) -->
        <query>
            <![CDATA[
                <QueryBuilder>
                    <exec>
                        <procedure>CM-PortalDB</procedure>
                        <procedure>data</procedure>
                        <procedure>sp_NameOfTheStoredProcedure</procedure>
                        <parameters>
                            <parameter>
                                <id>equi</id>
                                <parameter>EquipmentNo</parameter>
                                <type>VARCHAR</type>
                            </parameter>
                        </parameters>
                    </exec>
                </QueryBuilder>
            ]]>
        </query>
    </data>
</xml>
```

5.2.4. Filter anlegen

Filter können die Ergebnismenge einer DataID-Abfrage einschränken. Sie können über das Filtermenü-Widget und das Filterauswahl-Widget während der Benutzung des Portals konfiguriert. Damit die Filter in den Widgets verfügbar sind, müssen sie in YUNAML bei der Dashboardentwicklung definiert werden. Dabei erhält jede Filter-Definition eine eigene *Filter-ID*, über die die Filter in den Widgets referenzierbar sind.

Die Filter-ID wird außerdem von anderen Widgets, wie dem Tabellen-Widget, als Triggerparameter verwendet. Die Angabe als *TriggerParam* lautet dabei *filter* + Filter-ID. Also lautet zum Beispiel für die Filter-ID *2* der TriggerParam *filter2*.

Beispiel einer Filter-Definition

```
<xml>
    <FilterEntry id="2">
        <FilterTablePath>CM-DataDB data DeviceBasicInfo</FilterTablePath>
        <FilterMenuContainer roles="System_Admin, AdHoc_Full_Issue">
```



```
            <FilterMenu>
                <FilterType>LIST</FilterType>
                <Field>EquipmentNo</Field>
                <Caption>Equipmentnummer</Caption>
                <Position>1</Position>
                <Count>false</Count>
            </FilterMenu>
        </FilterMenuContainer>
    </FilterEntry>
</xml>
```

YUNAML-Eigenschaften der Filter-Definition

Eigenschaften der Filter-Definition

YUNAML-Tag	Beschreibung	Beispiel

FilterEntry	Start der Filter-Definition. Über das Attribut *id* wird die ID des Filters festgelegt. Die ID muss eine ganze positive Zahl sein.	`<FilterEntry id="2">`
		`  ...`
		`</FilterEntry>`

FilterEntry > FilterTablePath	Gibt die Datenbank-Tabelle an, auf die sich der Filter angewendet wird. Datenbank, Schema und Tabelle werden getrennt durch Leerzeichen angegeben, also:
	
	Datenbank Schema Tabelle
	
	Außerdem wird die Information verwendet, um bei der Anwendung des Filters die Tabelle über einen JOIN mit dem SELECT-QueryBuilder zu verknüpfen, falls sich der Filter auf eine andere Tabelle als die Tabellen des SELECT-QueryBuilders bezieht.
		`<FilterTablePath>`
		`  YUNA-DataDB data DeviceBasicInfo`
		`</FilterTablePath>`

FilterEntry > FilterMenuContainer	Legt die Konfiguration des Filtermenüs fest, über das die Filter auf den Dashboards gesetzt werden können. Der YUNAML-Tag enthält eine Auflistung von *FilterMenu* -Einträgen (siehe FilterMenu-Definition). Über das Attribut *roles* kann festgelegt werden, für welche Benutzergruppen das Filtermenü zur Verfügung steht.	`<FilterMenuContainer roles="Default">`
		`  ...`
		`</FilterMenuContainer>`

Beim anlegen eines neuen Filters muss beachtet werden, dass die Filter-ID ist unabhängig vom RefTag ist. Außerdem kann der *FilterTablePath* nicht durch wiederholtes Deployment ersetzt werden.

Definition der FilterMenu-Einträge

YUNAML-Tag	Beschreibung	Beispiel
Allgemeingültige YUNAML-Tags		
FilterMenu	Start des Filtermenü-Eintrags.	`<FilterMenu> ... </FilterMenu>`
FilterMenu > FilterType	Typ des Filtermenü-Eintrags. Gültige Werte sind *LIST*, *DATE*, *CATEGORY* und *METRIC*. Mehr Informationen zu den verschiedenen Filter-Typen enthält dieser Abschnitt.	`<FilterType>LIST</FilterType>`
FilterMenu > Field	Das Tabellen-Feld, auf das dieser Filtermenü-Entrag angewendet wird.	`<Field>EquipmentNo</Field>`

FilterMenu > Caption	Die Überschrift, die im Filtermenü angezeigt wird. Kann ein String oder eine Übersetzung mit TKey sein.	`<Caption> Equipmentnummer </Caption>`
		oder
		`<Caption> <TKey> content.filter2.equino </TKey> <Default> Equipmentnummer </Default> </Caption>`

FilterMenu > Position	Die Position des Filtermenü-Eintrags. Muss eine ganze, positive Zahl sein.	`<Position>1</Position>`
FilterMenu > Count	Für *CATEGORY*-Einträge kann der Wert auf *true* gesetzt werden, um die Anzahl der aktuell in der Kategorie verbleibenden Einträge anzuzeigen. **Für die anderen Filtertypen muss der Wert auf *false* gesetzt werden.**	`<Count>false</Count>`
YUNAML-Tags für den *CATEGORY*-Typ		
FilterMenu > TKeyField	Ein zusätzliches Tabellenfeld, dass die Übersetzungsschlüssel für die Kategorieeinträge bereit stellt. *Optional.*	`<TKeyField> ProductGroupTK </TKeyField>`

YUNAML-Tag	Beschreibung	Beispiel
FilterMenu > LookupQuery	Ein SELECT-QueryBuilder über den alternativ die Kategorieeinträge abgerufen werden können. Das erste Feld der Ergebnismenge entspricht dabei dem Wert, der ohne LookupQuery aus dem mit <*Field*> abgerufene Tabellenfeld geladen wird. Der zweite, optionale Wert der Ergebnismenge wird als TKey für den Kategorieeintrag verwendet. *Optional.*	<LookupQuery> <![CDATA[<QueryBuilder> ... </QueryBuilder>]]> </LookupQuery>

Die Filter-Typen

Für die Filtermenü-Einträge gibt es vier verschiedene Filtertypen: *LIST*, *DATE*, *CATEGORY* und *METRIC*. Sie bieten unterschiedliche Optionen bei der Konfiguration der Filter auf den Dashboards und sind für verschiedene Datentypen gedacht.

LIST-Filter

Im *LIST*-Filter kann eine Liste von Werten angegeben werden. In der Ergebnismenge der Datenabfrage müssen die Werte der entsprechenden Tabellenspalte in der Liste der Werte vorkommen. Alternativ kann eine Wildcard angegeben werden. So wird z.B. über die Wildcard *DE%* alle Ergebnisse abgerufen, bei denen die Werte den entsprechenden Tabellenspalte mit *DE* beginnen.

DATE-Filter

Über den *DATE*-Filter lassen sich Daten und Uhrzeiten filtern. Dafür können im Filter ein Start- und/oder Endzeitpunkt angegeben werden, um einen Zeitraum für die Daten festzulegen. Alternativ kann ein relativer Zeitraum, wie z.B. *Die letzten 7 Tage*, gewählt werden.

CATEGORY-Filter

Der *CATEGORY*-Filter bietet eine Auswahl aller möglichen Werte der zugeordneten Tabellenspalte. Duplikate in den Werten werden zu einer Option zusammengefasst. Bei der Verwendung des Filters können einzelne Werte an- und abgewählt werden. Alternativ kann, wie beim *LIST*-Filter, eine Wildcard angegeben werden.

METRIC-Filter

Der *METRIC*-Filter dient zur Einschränkung von Daten, die aus ganzen Zahlen bestehen. Im Filter lassen sich über einen minimalen und/oder maximalen Wert die Ergebnisse der Datenabfrage einschränken.

Analyse-Filter festlegen

Für das System muss ein existierender Filter als Analyse-Filter festgelegt werden, damit dieser in den R-Skripten, im Sachverhalt und in den Jobs verwendet werden kann. Hierzu muss in der Datenbank ein Eintrag zur Konfiguration gesetzt werden.

Gehen Sie hierzu in Ihre Datenbank in das Schema "portal" und fügen Sie der Tabelle "ConfigStore" einen Eintrag hinzu. Die Spalte "Key" muss hierzu auf "AnalysisFilter" und die Spalte "Value" mit

der Filter-ID gesetzt werden.

```
INSERT INTO [DB].[portal].ConfigStore([Key], Value)
VALUES('AnalysisFilter', '2');
```

Filter-Konfigurations-Views konfigurieren

Im Portal können Sie Sichten erstellen, die zur Zusammenstellung eines neuen Filters dienen. Diese sind aus dem "Filter laden"-Popup und der Filterverwaltung zu erreichen. Damit das Portal weiß, welche Sicht für welchen Filter zuständig ist, müssen diese in der Datenbank festgelegt werden.

Gehen Sie hierzu in Ihre Datenbank in das Schema "portal" und fügen Sie der Tabelle "ConfigStore" einen Eintrag hinzu. Die Spalte "Key" muss hierzu auf "portal.filter.config.filter2" (Wobei 2 für die jeweilige Filter-ID steht) und die Spalte "Value" auf den Link zur Sicht im Portal gesetzt werden.

```
INSERT INTO [DB].[portal].ConfigStore([Key], Value)
VALUES('config.filter.config.filter2', 'dsp_configurefilter_2');
```

Sie können jedem Filter einen "Friendly"-Name vergeben, der für den Anwender leichter zu verstehen ist, als wenn ein Filter nur mit "Filter2" angezeigt wird. Hierzu können Sie im "ConfigStore" einen neuen Eintrag erstellen mit dem Key "portal.filter.filter2.friendlyname" (Wobei 2 für die jeweilige Filter-ID steht) und dem Value, in dem Sie eine beliebige Bezeichnung eintragen.

```
INSERT INTO [DB].[portal].ConfigStore([Key], Value)
VALUES('config.filter.filter2.friendlyname', 'Analytics-Filter');
```

Sobald Sie in der Filterverwaltung auf "Neuen Filter anlegen" klicken erscheint nun in einem Dialogfenster eine auswahlliste mit den jewiligen Filtersichten. Sollte der Friendlyname nicht gesetzt sein, wird die jeweilige Sicht in der Auswahlliste im Portal z.B. wie folgt dargestellt: filter2 (Link:dsp_configurefilter_2)

YUNA ML Beispielcode für einen Filter

YUNA ML Beispielcode für die Einbindung des Filters in einem Datenaufruf mit Stored Procedure (z.B. für die Nutzung Lokalisierung via Stored Procedure)

Wichtig für das nachfolgende Beispiel sind die Inhalte im <parameters>-Tag. Zum einen der Parameter **<id>currentLanguage</id>** in Verbindung mit dem zugehörigen Parameter der Stored Procedure, der im darunterliegenden <parameter>-Tag beschrieben ist. Zum anderen müssen zur Filterung die entsprechenden Tags **<id>filter2</id>** mit parameter InstBasis gesetzt und in der Stored Procedure verwendet werden.

```
<xml>
  <data name="qy_InstBasis_Overview" roles="System_Admin, AdHoc_Full_Issue">
    <friendlyName>qy_InstBasis_Overview</friendlyName>
    <type>StoredProcedure</type>
```



```
    <filter>true</filter>
    <path>DSE-DataDB data</path>
    <query>
      <![CDATA[<QueryBuilder>
        <exec>
          <procedure>DSE-DataDB</procedure>
          <procedure>data</procedure>
          <procedure>sp_GetTableDataAccordingToLanguageParameter</procedure>
          <parameters>
            <parameter>
              <id>currentLanguage</id>
              <parameter>languageParameter</parameter>
              <type>VARCHAR</type>
            </parameter>
            <parameter>
              <id>filter2</id>
              <parameter>InstBasis</parameter>
              <type>VARCHAR</type>
            </parameter>
          </parameters>
        </exec>
      </QueryBuilder>]]>
    </query>
    <meta>
        <fields>
            <field>
                <name>ProductGroup</name>
                <type>
                    <name>Translatable</name>
                </type>
            </field>
            <field>
                <name>MachineType</name>
                <type>
                    <name>Translatable</name>
                </type>
            </field>
        </fields>
    </meta>
  </data>
</xml>
```

Es folgt die Definition einer passenden Stored-Procedure "sp_GetTableDataAccordingToLanguageParameter" für den Abfruf von Daten aus einer Tabelle in Abhängigkeit der eingestellten Sprache und des konfigurierten Filters. Dabei wird in Abhängigkeit von der eingestellten Sprache die Spalte "Location" (bei de_DE) oder "Location_en" (alle anderen Sprachen) zurückgegeben.

Damit der Rückgabewert der Prozedur zum definierten Dashboard Inhalt passt, muss

die Spalte "Location_en" via Alias als "Location" an das Backend zurückgegeben werden.

Die folgende Stored-Procedure-Definition zeigt, wie das auszuführende SELECT-Statement in Abhängigkeit der Sprache ausgewählt und der Filter `InstBasis` in der WHERE-Klausel des Statements angehangen wird.

```
CREATE PROCEDURE [data].sp_GetTableDataAccordingToLanguageParameter
    @languageParameter varchar(20),
    @InstBasis nvarchar(MAX)
AS
  BEGIN

  DECLARE @sql nvarchar(MAX);
  DECLARE @sqlPartFilter NVARCHAR(MAX) = ' ';


  IF @InstBasis IS NOT NULL
  BEGIN
    SET @sqlPartFilter =  ' WHERE ' + @InstBasis  ;
  END

  IF @languageParameter = 'de_DE'
    SET @sql = 'SELECT ProductionStartDate,
        ProductGroup,
        ProductGroupTK,
        ProductionCounter,
        Location
    FROM [DSE-DataDB].[data].DeviceBasicInfo AS InstBasis' + @sqlPartFilter
  ELSE
    SET @sql = 'SELECT ProductionStartDate,
        ProductGroup,
        ProductGroupTK,
        ProductionCounter,
        Location_en AS Location
    FROM [DSE-DataDB].[data].DeviceBasicInfo AS InstBasis' + @sqlPartFilter
END

EXEC sp_executesql @sql
WITH RESULT SETS UNDEFINED;
```


Ein Filtername kann pro Benutzer immer nur einmal vergeben werden (→ also nicht zwei Filter mit gleichem Namen als privat und global)

Für jeden Filter wird in der Datenbank ein Filterverantwortlicher Nutzer abgelegt (der Ersteller des Filters). Wird dieser Benutzer aus der Datenbank gelöscht, so werden auch alle von diesem Benutzer angelegten Filter unbrauchbar.

Solche "unbrauchbar gemachten" Filter können jedoch anderen Benutzern zugewiesen

werden. Hierbei gilt jedoch auch die oben genannte Einschränkung, dass der neue filterverantwortliche Benutzer keine zwei Filter mit gleichem Namen haben kann.

5.2.5. Views anlegen

Sämtliche Inhalte des Portals werden anhand von Widgets in Portal Views dargestellt. Beim Anlegen einer View können mehrere DInge definiert werden:

```
<xml>
<!--
                    Copyright (c) 2017 eoda GmbH
          All Rights Reserved, see LICENSE.TXT for further details
-->
  <view name="template_view" roles="System_Admin, AdHoc_Full_Issue">
    <!-- View-Title -->
    <caption>View-Template</caption>
    <!-- View-Description -->
    <description>Description of the View</description>
    <!-- Link to a document -->
    <userdocu>file://server/file.pdf</userdocu>
    <!-- List of the Widgets. Uncomment it to display it. -->
    <!-- <widget source="template_widget_Basechart" /> -->
    <!-- <widget source="template_widget_Changelog" /> -->
    <!-- <widget source="template_widget_Dependency" /> -->
    <!-- <widget source="template_widget_Filter" /> -->
    <!-- <widget source="template_widget_HTML" /> -->
    <!-- <widget source="template_widget_ImageViewer" /> -->
    <!-- <widget source="template_widget_Issue" /> -->
    <!-- <widget source="template_widget_Job" /> -->
    <!-- <widget source="template_widget_Landingpage" /> -->
    <!-- <widget source="template_widget_Scriptmanager" /> -->
    <!-- <widget source="template_widget_SelectedFilter" /> -->
    <!-- <widget source="template_widget_Sensorlist" /> -->
    <!-- <widget source="template_widget_Singlechoice" /> -->
    <!-- <widget source="template_widget_Statetile" /> -->
    <!-- <widget source="template_widget_Stockchart" /> -->
    <!-- <widget source="template_widget_StockchartOption" /> -->
    <!-- <widget source="template_widget_SystemInfo" /> -->
    <!-- <widget source="template_widget_Table" /> -->
  </view>
</xml>
```

Info	Bedeutung
view name	Frei wählbarer Name der Portal View
roles	Namen der Nutzerrollen, welche diese View sehen dürfen
caption	Beschreibung der View / Überschrift
list	Auflistung aller Widgets, die in der View eingebunden werden sollen

5.2.6. YunaML zur Definition von Dashboard Inhalten

Dashboard Inhalte werden im Kontext von YUNA in YunaML definiert und anschließend in JSON umgewandelt, um so in der Datenbank abgelegt zu werden.

Durch die Nutzung von YunaML

- können einzelne Elemente wiederverwendet werden. Sourcing ermöglicht es, YunaML-Elemente nur an einer Stelle editieren zu müssen, die Änderung aber an vielen Stellen gleichzeitig wirken zu lassen.
- ist eine Versionsverwaltung des Dashboards möglich
- ist es möglich, definierten Dashboard Inhalt parallel über mehrere Instanzen zu deployen
- können einfach Kommentare in den Dashboard-Code geschrieben werden.

UTF-8 Kodierung

YUNAML Dateien müssen in UTF-8 Kodierung abgespeichert werden.

Funktion	Definition	betroffene Widgets

Definition des Widget-Typs	`<widget>`
	`  <widgettype>tabledirective</widgettype>`
	`</widget>`
		• alle

Definition der Position des Widgets im Grid	`<position>`
	`  <x>0</x>`
	`  <y>0</y>`
	`</position>`
		• alle

Definition von Arrays	`<array>`
	`  <list>`
	`  <!-- Listenelement 1 -->`
	`  </list>`
	`  <list>`
	`  <!-- Listenelement 1 -->`
	`  </list>`
	`</array>`
		• alle

Definition der Spaltenbreite von Tabellen	`<cols>`
	`  <list>`
	`  ...`
	`  <width>100</width>`
	`  <style>`
	`  ...`
	`  </style>`
	`  ...`
	`  </list>`
	`</cols>`
		Tabellen-Widget

Funktion	Definition	betroffene Widgets
Triggerparameter zur Steuerung von Widgets bzw. deren Inhalten	<triggerParams> <mandatory> <list>sensor</list> <list>equi</list> </mandatory> <optional> <list>startdate</list> <list>enddate</list> <list>selectedRelativePeriod</list> </optional> </triggerParams>	Alle Widgets

Dashboard Inhalte definieren

YUNA-ML Objekt mit Array zur Definition der Objekteigenschaften

```
<Objekt>
    <list>Name des ersten Listenelements</list>
    <list>Name des zweiten Listenelements</list>
    <list>Name des dritten Listenelements</list>
</Objekt>
```

YUNA-ML Objekt mit fest benannten Eigenschaften

XML

```
<Objekt>
    <attribut1>...</attribut1>
    <attribut2>...</attribut2>
</Objekt>
```

Source

Um Definitionen in YUNA-ML, die an mehreren Stellen des Portals eingesetzt werden sollen (z.B. einheitliche Kontextmenüs der Widgets im Portal), wiederzuverwenden, wird das Attribut **source** verwendet. Dadurch wird das "gesourcete" Element quasi im aktuellen XML eingebettet und kann dort referenziert werden. Verschachteltes `sourcing` ist möglich.

Beispiel:

```
<label source="labeldef" />
```

```
<labeldef>Mein Label</labeldef>
```

analog:

```
<wasweisich name="labeldef">Mein Label</wasweisich>
```

Die gesourcten Elemente müssen sich dabei auf der höchsten Ebene im XML befinden.
Also:

```
<xml>
  <meinedefinition>daten</meinedefinition>
</xml>
```

- "widget snippets/definitionen" sollen in der Datei widget.xml angelegt werden
- links werden in dem file Global/glb_Link_Prefixes.xml definiert

Übersetzung von Dashboard Inhalten

Damit ein im Rahmen der Dashboard-Entwicklung definierter Text übersetzt werden kann, muss ein entsprechender Übersetzungsschlüssel definiert werden. Wie der folgenden Tabelle entnommen werden kann, ist es möglich entweder nur den Übersetzungsschlüssel zu definieren oder zusätzlich einen Standardwert zu setzen, der verwendet wird, wenn in der ausgewählten Sprache keine Übersetzung für diesen Schlüssel verfügbar ist.

Werden Dashboard Inhalte mit Übersetzungen via dsedep deployt, werden die Standardwerte in der Portal-Datenbank gespeichert und es ist möglich eine Übersetzungsdatei zu generieren.

YUNA XML Verhalten je nach XML-Definition

	XML-Definition	Verhalten

1.	`<label>`
	`  Tabellenüberschrift`
	`</label>`
		Da kein Übersetzungsschlüssel definiert ist, wird unabhängig von der gewählten Sprache als Label "Tabellenüberschrift" angezeigt.

2.	`<label>`
	`  <tkey>content.table.header</tkey>`
	`</label>`
		Da ein Übersetzungsschlüssel, jedoch kein Standartext definiert ist, wird als Label immer der zu dem Übersetzungsschlüssel gehörende Wert der aktuell ausgewählten Sprache angezeigt. Ist keine Übersetzung vorhanden, wird der Übersetzungsschlüssel angezeigt.

3.	`<label>`
	`  <default>Tabellenüberschrift</default>`
	`  <tkey>content.table.header</tkey>`
	`</label>`
		Da sowohl Übersetzungsschlüssel als auch Standardtext definiert sind, wird immer der Wert der aktuell ausgewählten Übersetzung angezeigt. Ist in der aktuell ausgewählten Übersetzung kein Wert mit dem entsprechenden Übersetzungsschlüssel hinterlegt, wird der Standard-String, also "Tabellenüberschrift" angezeigt.

Beispiel: Widget mit übersetztem Titel

```
<xml>
  <widget>
    <position>
      <y>8.4</y>
      <x>0</x>
    </position>
    <size>
      <x>3</x>
      <y>5.7</y>
    </size>
    <widgettype>singlechoicedirective</widgettype>
```

```
    <caption>
      <show>true</show>
      <label>
        <default>Standard-Widgetüberschrift</default>
        <tkey>content.view1.singlechoicedirective.label</tkey>
      </label>
    </caption>
    <dataID>qy_EquipmentList</dataID>
    <urlParam>equi</urlParam>
    <triggerParams>
      <mandatory>
        <list>filter2</list>
      </mandatory>
    </triggerParams>
  </widget>
</xml>
```

5.2.7. Widgets

YUNA stellt diverse Typen von Widgets zur Verfügung, durch welche die unterschiedlichsten Sichten für die verschiedensten Aufgaben realisiert werden können.

Ein Widget besteht im Wesentlichen aus drei Elementen:

1. Seiner allgemeinen Konfiguration, also der Art und Weise der Darstellung (Widget-Typ, Größe, Position, etc.),
2. seiner Data_ID, die auf den Inhalt referenziert, der im Widget dargestellt werden soll und
3. Seiner type-spezifischen Konfiguration (Spalten bei Tabellen, oder der Text eines Html-Widgets).

In diesem Abschnitt soll es zunächst um eine genauere Betrachtung der ersten beiden Punkte gehen, während in den Unterabschnitten für jeden Widgettype auf den dritten Punkt eingegangen wird.

Beispiel zur Definition eines Widgets

```
<widget>
    <widgettype>Name_eines_Widgets</widgettype>
    <position>
        <x>0</x> <!-- Die obere linke Ecke des Widgets ist auf der oberen linken Ecke der Grid -->
        <y>0</y>
    </position>
    <size>
        <x>3</x> <!-- Das Widget ist 300px breit -->
        <y>2</y> <!-- und 200px hoch -->
    <size>
    <dataID>qy_infoforsingleeqipment</dataID>
    <triggerParams>
        <mandatory>
            <list>id</list>
            <list>filter2</list>
        </mandatory>
        <optional>
```

```
            <list>startdate</list>
            <list>enddate</list>
        </optional>
    </triggerParams>
    <caption>
        <label>Messwerte selektieren</label>
    </caption>
    <appearance>
        <enlargeableY>true</enlargeableY>
        <enlargedY>true</enlargedY>
    </appearance>
</widget>
```

Feld	Möglicher Wert	Beschreibung	Default	Notwendig
caption		Die Titelzeile eines Widgets.		✗
position		Definiert die Position des Widgets im Grid. Der Nullpunkt des Grids ist in der oberen linken Ecke. Das Raster des Grid ist in 100px schritten unterteilt.		✓
position > x	Integer >= 0	Position der oberen linken Ecke des Widgets auf der X Achse des Grid.	0	✓
position > y	Integer >= 0	Position der oberen linken Ecke des Widgets auf der Y Achse des Grid.	0	✓
size		Definiert die Größe des Widgets im Grid. Das Raster des Grid ist in 100px Schritte unterteilt.		✓
size > x	Integer >= 1	Gibt die Breite des Widgets auf dem Grid an.	1	✓
size > y	Integer >= 0	Gibt die Höhe des Widgets auf dem Grid an.	1	✓
widgettype	Ein Widget-Typ	Muss den Typ eines von YUNA bereitgestellten Widgets beinhalten. Mögliche Werte und damit einhergehende Konfigurationsmöglichkeiten werden in dem nachfolgenden Kapitel Widget-Typen behandelt.		✓
dataID	Text	Eine Referenz zu einer definierten Data-ID.		✗

Feld	Möglicher Wert	Beschreibung	Default	Notwendig
appearance > enlargeableY	true, false	**Wert** / **Erläuterung** true: Das Widget kann reduziert und erweitert dargestellt werden. Wenn es reduziert ist, wird nur die Titelzeile dargestellt. false: Das Widget wird immer vollständig dargestellt	false	✗
appearance > enlargedY	true, false	**Wert** / **Erläuterung** true: Das Widget wird beim initialen aufrufen der View erweitert dargestellt false: Das Widget wird beim initialen aufrufen der View reduziert dargestellt	false	✗
allowFullscreen	true, false	**Wert** / **Erläuterung** true: Das Widget kann in den Vollbildmodus überführt werden false: Das Widget kann nicht im Vollbildmodus dargestellt werden	false	✗

Caption

Die Titelzeile bzw. caption eines Widgets beinhaltet den angezeigten Titel des Widgets und kann weitere Funktionalitäten wie ein Kontext-Menü, Exportfunktionen o.ä. bereitstellen. Sie kann mit dem Feld *caption* erzeugt und konfiguriert werden.

Das Kontext-Menü ist ein umfangreiches Werkzeug um Widget spezifische Optionen anzubieten, welche vom Anwender in der Widgetcaption mit einem Klick erreicht werden können. Es bietet die Möglichkeit diverse Funktionen auszuführen und Links zu beliebigen Seiten zu öffnen. Das Kontext-Menü ist u.A. beim Tabellen-Widget und "Aktivierte Filter"-Widget bereits vordefiniert und bietet Optionen zur Verwaltung der Filter bzw. zum Exportieren der Tabellen. Diese können trotzdem individuell konfiguriert werden.

Beispiel zur Definition einer Caption

```
<caption>
    <label>Tabellen-Widget mit kontextmenü</label>
    <menu>
        <option>
            <list>
                <type>label</type>
                <label>Tabellenoptionen</label>
                <name>Widgetoptions</name>
```

```
            </list>
            <list>
                <type>link</type>
                <icon>fa-question</icon>
                <label>Hilfe zum Kontextmenü</label>
                <link>
                    <url>https://confluence.eoda.local/pages/viewpage.action?pageId=15696020</url>
                    <extern>true</extern>
                </link>
                <tooltip>Confluence Artikel zum Konfigurieren des Kontext-Menüs</tooltip>
             </list>
             <list>
                <type>function</type>
                <name>testfunction</name>
                <icon>fa-refresh</icon>
                <label>Testfunktion</label>
                <function>
                    <widget>contextmenu</widget>
                    <name>testFunction</name>
                    <array_param_1>test1</array_param_1>
                    <array_param_2>test2</array_param_2>
                    <array_param_3>test3</array_param_3>
                </function>
             </list>
             <list>
                <type>popup</type>
                <name>tableproperties</name>
                <icon>fa-info</icon>
                <label>Tabellen Eigenschaften</label>
                <popup>
                    <header>Eigenschaften</header>
                    <body>Die Tabelle zeigt Ihnen...</body>
                </popup>
            </list>
        </option>
    </menu>
</caption>
```

Feld	Mögliche Werte	Beschreibung	Default	Notwendig
label	Text	Gibt den Text an, welcher in der Caption des Widgets angezeigt werden soll.		✓
<menu><option><list> ... </menu></option></list>				
disabled	true, false	Deaktiviert (true) bzw. aktiviert (false) den Menü Eintrag. Kann nicht in Optionen vom Typ 'label' oder 'divider' benutzt werden.	false	✗

Feld	Mögliche Werte	Beschreibung	Default	Notwendig
icon	fa-icons	Definiert ein Icon, welches vor dem Label angezeigt wird. Hier wird die gewünschte Fontawesome-Klasse eingetragen. Derzeit ist die Nutzung von FontAwesome-Icons bis einschließlich Version 4.7 gewährleistet. Kann nicht in Optionen vom Typ 'label' oder 'divider' benutzt werden.		✗
label	Text	Gibt den im Menü Eintrag angezeigten Text an. Kann nicht in Optionen vom Typ 'divider' benutzt werden.		✓ in jedem Menü Eintrag
name	Text	Bezeichnung für die Option muss angegeben und innerhalb der Widget-Definition eindeutig sein.		✓ in jedem Menü Eintrag
tooltip	Text	Beliebiger Text, der über dem Button als Hinweis erscheint, sobald man mit der Maus drüber fährt. Kann nicht in Optionen vom Typ 'label' oder 'divider' benutzt werden.		✗
type	"link" "function" "popup" "label" "divider"	`link`: Erzeugt einen Button, welcher beim Klicken eine beliebige URL öffnet. `funktion`: Erzeugt einen Button, welcher beim Klicken eine Funktion ausführt. `popup`: Erzeugt einen Button, welcher beim Klicken ein Popup-Fenster mit beliebigem Inhalt öffnet. `label`: Erzeugt ein Text-Label, welches zur Kategorisierung mehrerer Buttons verwendet werden kann. `divider`: Erzeugt ein Trennelement.		✓ in jedem Menü Eintrag
<type>popup</type>				
body	Text	Inhalt des Popups		✓
header	Text	Titel des Popups		✓
<type>link</type>				
url	Text	URL, auf welche der Link verweist.		✓

Feld	Mögliche Werte	Beschreibung	Default	Notwendig
extern	true, false	Definiert, ob es sich um ein Link auf eine Seite innerhalb der Anwendung handelt (false) oder ob es ein Link auf eine externe Seite ist (true). Ist das Field auf 'true' gesetzt, wird der Link immer in einem neuen Tab geöffnet, andernfalls immer im gleichen Tab.	false	✗

Datenbankabfragen definieren

Damit ein Widget auch eigene Daten anzeigen kann, muss in dessen Definition angegeben werden, welche Daten angezeigt werden sollen. Dazu müssen, zusätzlich zum Widget selbst, auch entsprechende Datenbankabfragen definiert werden. Dies geschieht in einem '*data*'-Tag. Ein Widget, welches die folgende Query in seinem '*dataID*'-Tag unter dem Namen *name_of_this_query* referenziert, würde alle Spalten und Einträge der Tabelle [DataBase].[schema].table als Daten erhalten.

Beispiel

```
<data name="name_of_this_query" roles="admin, registered">
    <type>QueryBuilder</type>
    <query>
        <![CDATA[
            <QueryBuilder>
                <select>
                    <table>DataBase</table>
                    <table>schema</table>
                    <table>table</table>
                    <fields>
                        <asteriskfield/>
                    </fields>
                </select>
            </QueryBuilder>
        ]]>
    </query>
</data>
```

Attribut	Mögliche Werte	Beschreibung	Notwendig
name	Text ohne Leerzeichen	Der Name dieses Data-Objekts der vom Widget genutzt wird, um auf dieses Objekt zu referenzieren. Der Name muss unbedingt eindeutig sein.	✓
roles	Durch Komma separierte Liste	Liste aller Rollen, für die diese Abfrage gestartet werden darf.	✓

Feld	Mögliche Werte	Beschreibung	Notwendig
filter			
friendlyName	Text		
path	Durch Leerzeichen separierte Liste		
query	Text	Definiert die Abfrage unter Verwendung der QueryBuilder Sprache.	✓
type	QueryBuilder | StoredProcedure	Gibt den Typ der Abfrage an	✓

5.3. Dashboard Inhalte deployen

Für das Deployment des YunaML-Codes des zuvor erstellten Inhalts in die Datenbank wird das Tool **dsedep** verwendet.

dsedep ist ein Tool zur Erstellung, zur Verwaltung und zum Deployment von Dashboard Inhalten wie Widgets und Views. Während die CM-PortalDB die jeweilig aktuelle/deployte Version enthält, können über die YunaML-Dateien verschiedene Varianten Content verwaltet werden. Die Bearbeitung von Content sollte in den YunaML-Dateien erfolgen, so dass sich folgender Ablauf ergibt:

1. Initiale Extraktion der Datenbank-Struktur mit dsedep zum Erstellen der ersten Version der YunaML-Datei(en) 1.0.
2. Bearbeiten der YunaML-Datei(en) und Erzeugen der nächsten Version 2.0.
3. Deployen von Version 2.0 mit dsedep auf die Datenbank/das Portal.
4. Bearbeiten der YunaML-Datei(en) und Erzeugen der nächsten Version 3.0.
5. Deployen von Version 3.0 mit dsedep auf die Datenbank/das Portal.

dsedep unterscheidet zwischen den Entitäten Views (Widgets), DataID und Filtern. Für jede Entität lassen sich eigene YunaML-Dateien erzeugen oder auch eine große YunaML-Dateien für alle drei Entitäten.

Ab Version 0.53 werden im Zuge des Dashboard Inhalt-Deployments auch Translation-Keys und ihre Defaults in die Übersetzungstabellen der Datenbank persistiert.

Ab Version 0.49 wird das Tool dseDep verwendet. Für alle vorherigen Versionen gilt die Nutzung des Tools spDep.

In Portalversionen kleiner als 0.49 muss in allen nachfolgenden Beispielen dseDep durch spDep zu ersetzen!

5.3.1. dsedep herunterladen

Die jeweils aktuelle Version von dsedep wird Ihnen über die RPM-Paketeausgeliefert und kann aus der Anwendungsoberfläche Heraus als Tool downgeloaded werden. So wird sichergestellt, dass Ihnen bei Anpassungen jeweils die korrekte Version von dsedep vorliegt.

5.3.2. Nutzung von dsedep

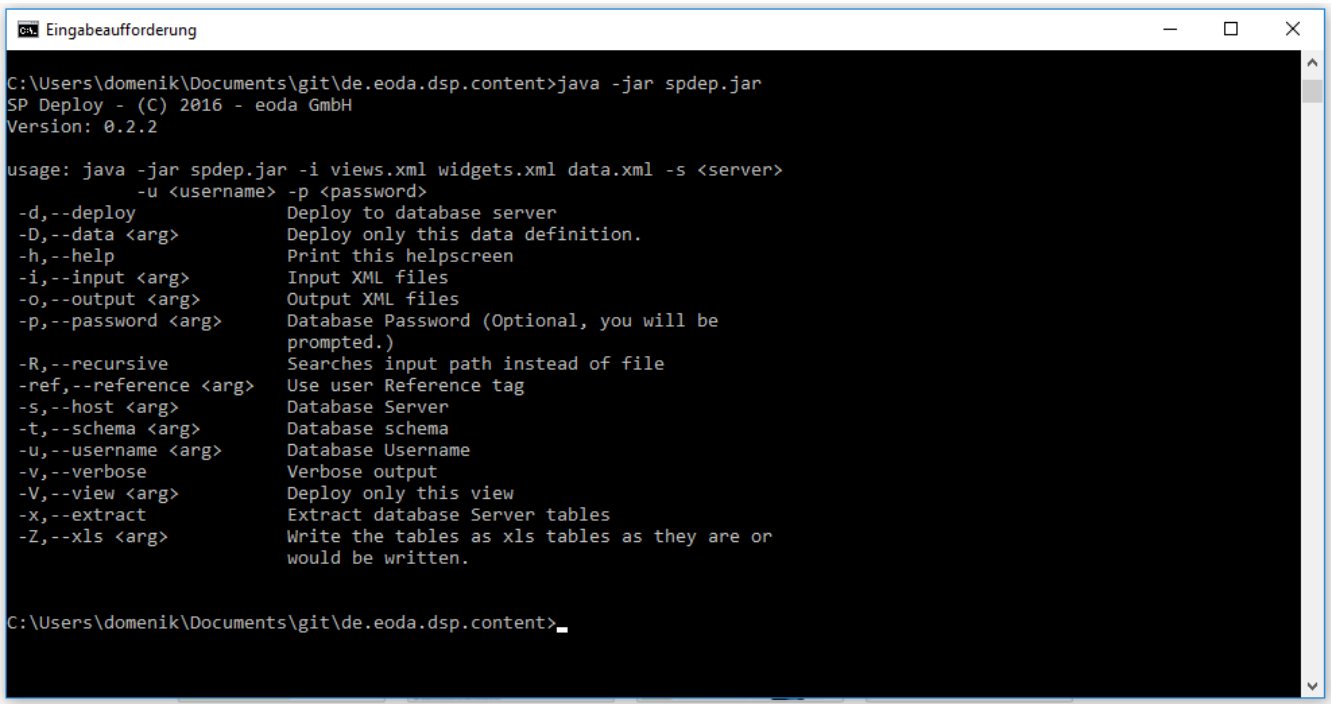
dsedep wird über die Eingabeaufforderung (Command Line Interface) aufgerufen und gesteuert. Ob Sie die Eingabeaufforderung Ihres Betriebssystems oder ein entsprechendes Tool eines Drittanbieters nutzen spielt dabei keine Rolle.

dseDep sollte sich im gleichen Ordner wie der zu deployende Dashboard Inhalt oder eine Ebene darüber befinden, da in dseDep der Ordner des zu deployenden Inhalt definiert wird.

Zunächst müssen Sie sich zum Speicherort von dsedep navigieren, um die Datei dort ausführen zu können. Nutzen Sie hierfür die Konsolenbefehle, die Ihnen zur Verfügung stehen:

Befehl	Bedeutung	Beispiel	Funktion
cd Ordnername	Wechselt in das Verzeichnis Ordnername	cd git	Wechselt ausgehend vom aktuellen verzeichnis in das angegebene Unterverzeichnis. Im Beispiel wird der Unterordner "git" im Ordner "documents" geöffnet.
cd ..	Wechselt in den übergeordneten Ordner	cd ..	Wechselt in den übergeordneten Ordner zurück
dir (windows) ls (Linux)	Verzeichnisinhaltsanzeige	dir	Listet die Inhalte des gewählten Ordners aus.

Nachdem Sie zum Speicherort der dsedep.jar navigiert sind, geben Sie zum Aufrufen von dsedep in die Kommandozeile den Befehl **java -jar dsedep.jar** ein. Daraufhin erscheint die folgende Ansicht, in der Sie alle zur Verfügung stehenden Funktionen von dsedep und deren Kommandos vorfinden:



Befehlskürzel	Befehlsname	Bedeutung / Funktion
-d	deploy	Deploy to database server
-clean	clean	Cleans database Server tables (deletes the content from tables, not the tables themself)
-D	data <arg>	Deploy only this data definition
-h	help	Print this helpscreen
-i	input <arg>	Input XML files / Folder of content files to be deployed
-lang	language <arg>	Generate translation file (.json) in the current working directory containing translation key-value pairs for the specified language
-o	output <arg>	Output XML files
-p	password <arg>	Database password (optional, you will be prompted)
-R	recursive	Searches input path instead of file
-ref	reference <arg>	Use user reference tag
-s	host <arg>	Database server (IP adress)
-t	schema <database> <sp-schema> <portal-schema>	Database schema. Up to three strings: Database, 'sp'-Schema, 'portal'-Schema.Must be specified in this order. Only database is required.
-u	username <arg>	Database username
-v	verbose	Verbose output
-V	view <arg>	Deploy only this view
-x	extract	Extract database server tables
-Z	xls <arg>	Write the tables as xls as they are or would be written

Zur Nutzung der dsedep-Befehle ist folgendes Schema anzuwenden:

```
java -jar dsedep.jar <Befehlskürzel> <Wert für Befehlskürzel> ⋯
```

Beispiele

Deployment mit Referenz-Tag zur Nutzung von Dashboard-Versionen:

Befehl mit Platzhaltern

```
java -jar dsedep.jar -i <path-to-content> -R -u <username> -p <password> -s <server-name-or-address> -t <schema> -d -ref
<ref-tag-name>
```

Befehl mit Werten

```
java -jar dsedep.jar -i Content/ -R -u ServerAdmin -p TollesPasswort12345 -s 192.168.0.1 -t CM-PortalDB sp -d -ref test
```

Abruf der Datenbank als Excel-File

```
java -jar Tools/dsedep.jar -i Content/ -R -u Username -p Password -s 123.456.789.0  -t Schema -Z
```

Beispiel für ein Deployment:

```
java -jar dsedep.jar -d -i views.xml widgets.xml data.xml -t CM-PortalDB sp -s 123.456.789.0 -u Username -p 123456
```

Deployment der gesamten XML-Definition in die DB:

```
java -jar dsedep.jar -i Dateiname.xml -u Username -p Password -s 123.456.789.0 -t Schema -d
```

(Rekursives) Deployment aller XML-Definitionen (innerhalb des Verzeichnisses und aller Unterordner)

```
java -jar dsedep.jar -i  Dateiname.xml -R -u Username -p Password -s 123.456.789.0 -t Schema -d
```

Deployment einer einzelnen View in die DB:

```
java -jar dsedep.jar -i *.xml -u Username -p Password -s 123.456.789.0 -t Schema -d -V viewname (aus view.xml)
```

Deployment einer einzelnen Datendefinition (DataID) in die DB:

```
java -jar dsedep.jar -i *.xml -u Username -p Password -s 123.456.789.0 -t Schema -d -D friendlyname (aus data.xml)
```

Beispiel für das Löschen der zu einem Referenz-Tag gehörenden YUNAML-Inhalte:

```
java -jar dsedep.jar -s 123.456.789.0 -t Schema -u Username -p Password --clean -ref demo
```

Achtung bei Verwendung des Befehls --clean

Mit dem Befehl "clean" können Inhalte unter einem Referenz-Tag gelöscht werden. Wird kein Referenz-Tag angegeben, werden ALLE YUNAML-Inhalte gelöscht, unabhängig davon, ob sie über einen Referenz-Tag verfügen oder nicht! Dieser Vorgang kann nicht rückgängig gemacht werden und muss nicht erneut bestätigt werden.

Generierung einer Übersetzungsdatei (.json) aus den im Dashboard Inhalt und in der Datenbank definierten Translation-Keys

```
java -jar dsedep.jar -i Dateiname.xml -u Username -p Password -s 123.456.789.0 -t Schema sp portal -lang deutsch -d
```

→ Erzeugt die Datei 'deutsch.json' im aktuellen Arbeitsverzeichnis. Diese bildet die Sicht des Nutzers im Portal ab und enthält dementsprechend alle Übersetzungen, die dem Nutzer bei der ausgewählten Sprache angezeigt werden

Änderung des Verhaltens für Übersetzungen

- Translation-Keys aus dem Dashboard Inhalt, die nicht über einen Standardwert verfügen, werden mit dem Translation-Key als Wert versehen.
- Da die Translation-Keys mit Standardwerten in die Datenbank geschrieben werden, wird bei jedem Deployment Inhalt-Standard-Übersetzung in der Datenbank überschrieben (⚠ Übersetzungen die nicht auf den im Dashboard definierten Standardwerten basieren, sind hiervon nicht betroffen). **Diese fungiert für alle Sprachen unter jedem Referenz-Tag als Rückfallwert**.
- Bei der Verwendung von dsedep wird der Dashboard-Developer informiert , welche Translation-Keys noch nicht in der Datenbank vorhanden sind.

Beim Aufruf der dsedep.jar ist der Pfad für die YunaML-Input-Files zu beachten. Beim Deployen ist die Abfrage yes/no mit 'yes' zu beantworten.

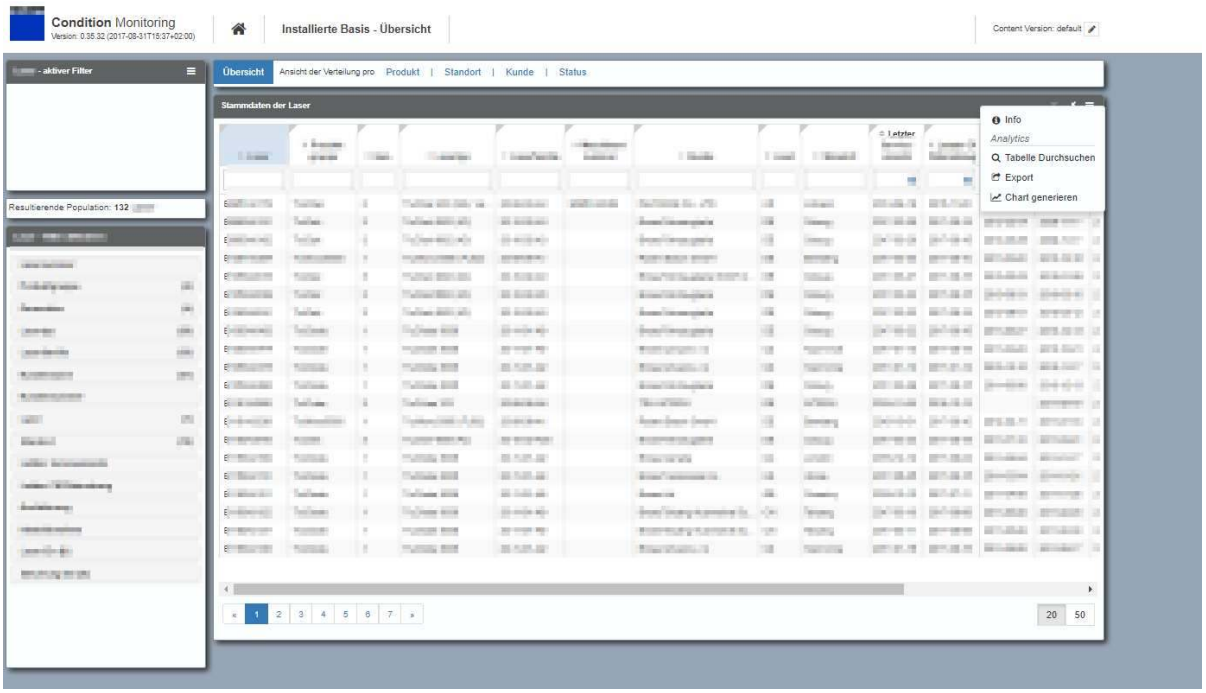
5.3.3. Referenz-Tags: Unterschiedliche Dashboard-Versionen

Das YUNA Portal bietet die Möglichkeit, unterschiedliche Versionen der Dashboard Inhalte gleichzeitig zu erstellen und anschließend aufrufen zu können.

Bei der Nutzung des Portals können Benutzer eine Dashboard-Version auswählen, um sich genau diese Version des Dashboards anzeigen zu lassen. Standardmäßig wird die Dashboard-Version "default" geladen und kann im oberen Bereich des Portals geändert werden.

Möchten Sie eine alternative Version des Dashboards betrachten, so geben Sie den von Ihnen gewünschte Referenz-Tag als Versionsnamen ein und klicken auf den Haken. Das Portal wird anschließend in der Dashboard-Version des neuen Referenz-Tags geladen und angezeigt. Die nutzbaren Referenz-Tags für die verfügbaren Dashboard-Versionen werden von den Dashboard-Developern vergeben.

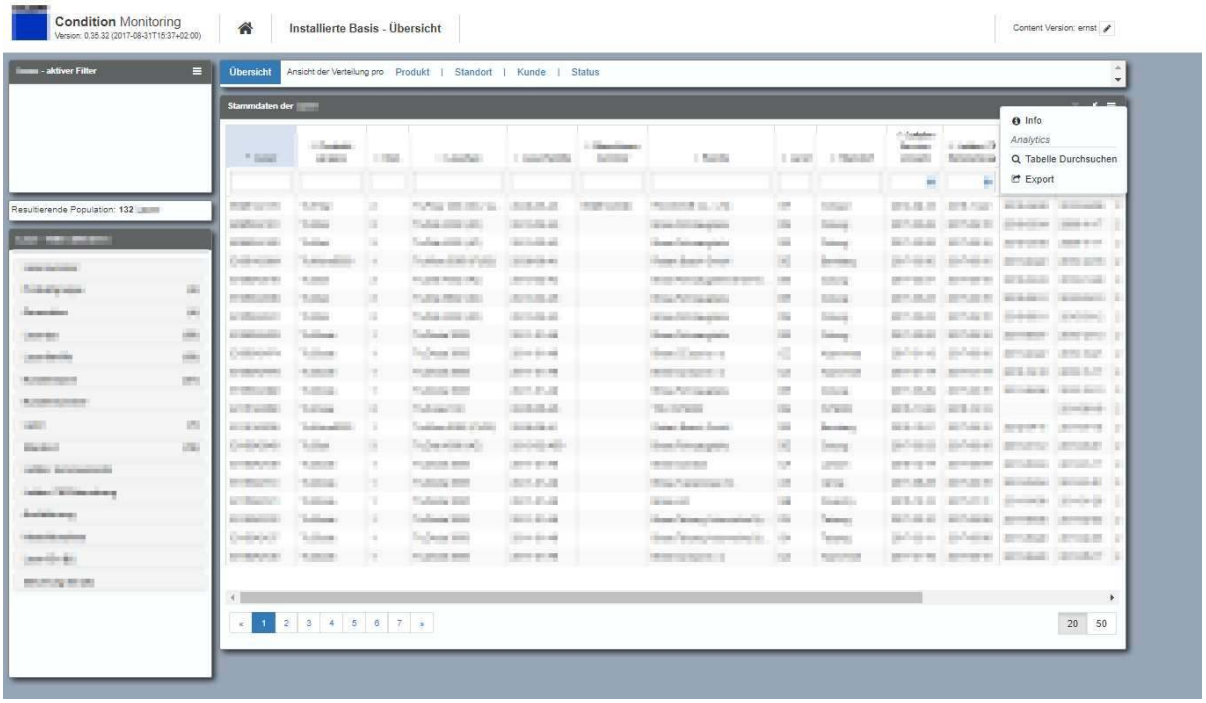
Beispiel



Hier hat ein Nutzer, in der Dashboard-Version "default" zum Beispiel die Möglichkeit, die Option

"Chart generieren" zu benutzen, um sich unterschiedliche Diagramme zu den in der Tabelle enthaltenen Daten erzeugen zu lassen.

Ändert man nun die Dashboard-Version, durch die Eingabe des Referenz-Tags "ernst", so sind nur die Funktionen verfügbar, die unter diesem Referenz-Tag definiert wurden. Im Vergleich zur Dashboard-Version "default" ist in der Dashboard-Version "ernst" z.B. die Funktion Charts zu generieren nicht mehr verfügbar.

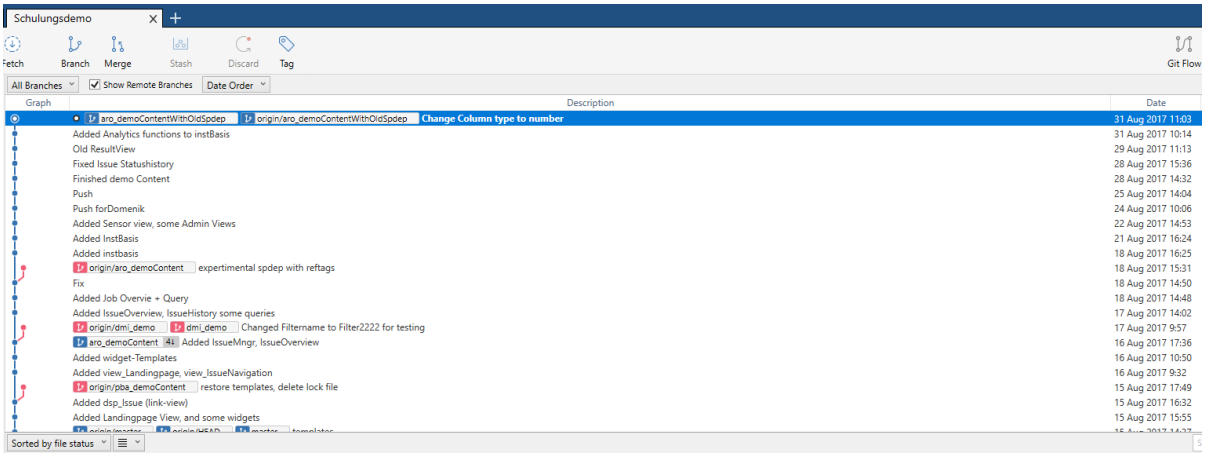


5.3.4. Referenz-Tags: Parallele Entwicklung des Dashboards

Um mehreren Dashboard Developern die Möglichkeit zu bieten, gleichzeitig Dashboard Inhalte zu erstellen und zu deployen, ohne die erzeugten Inhalte gegenseitig zu überschreiben, gibt es mehrere Möglichkeiten.

Git

Im Alltag hat sich bislang die Nutzung von Git als praktikabel und sinnvoll herausgestellt. So können mehrere Entwickler das Dashboard weiterentwickeln,vorläufig in ihren eigenen Branches ablegen und schließlich in gemeinsame Branches mergen, um so einen neuen gemeinsamen Entwicklungsstand zu erzeugen.



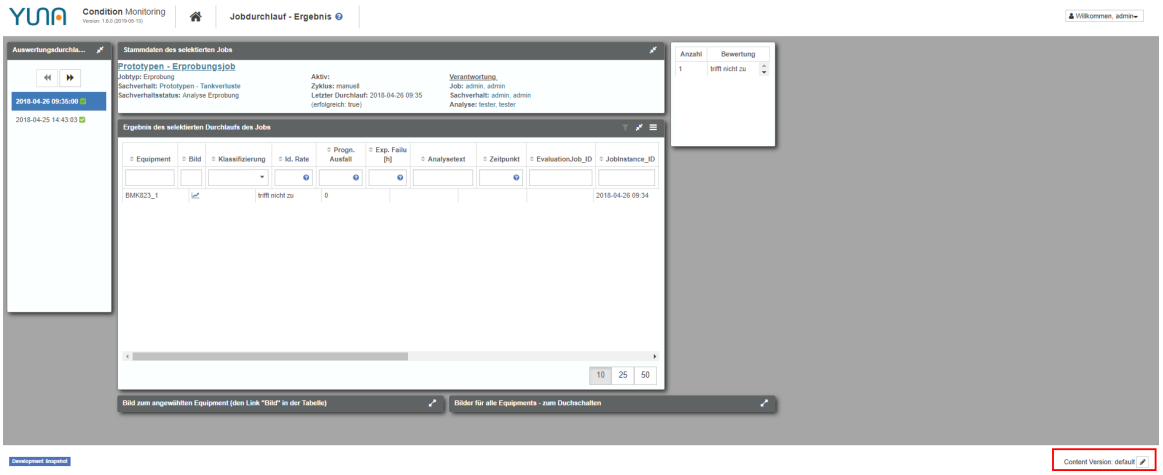
Referenz-Tags

Eine weitere Methode zur parallelen Entwicklung des Dashboards kann über Referenz-Tags realisiert werden. Bei dieser Methode gibt der Dashboard Developer beim Deployment mit dseDep über Parameter **-ref** ein Referenztag an, das an den Datenbankeintrag in der Tabelle [CM-PortalDB].[sp].[Grid] angehängt wird. So können Dashboard Inhalte auch in unterschiedlichen Versionen mehrmals deployed werden, um die Änderungen direkt testen zu können, ohne dass die für User erreichbaren Views im Portal verändert werden müssen.

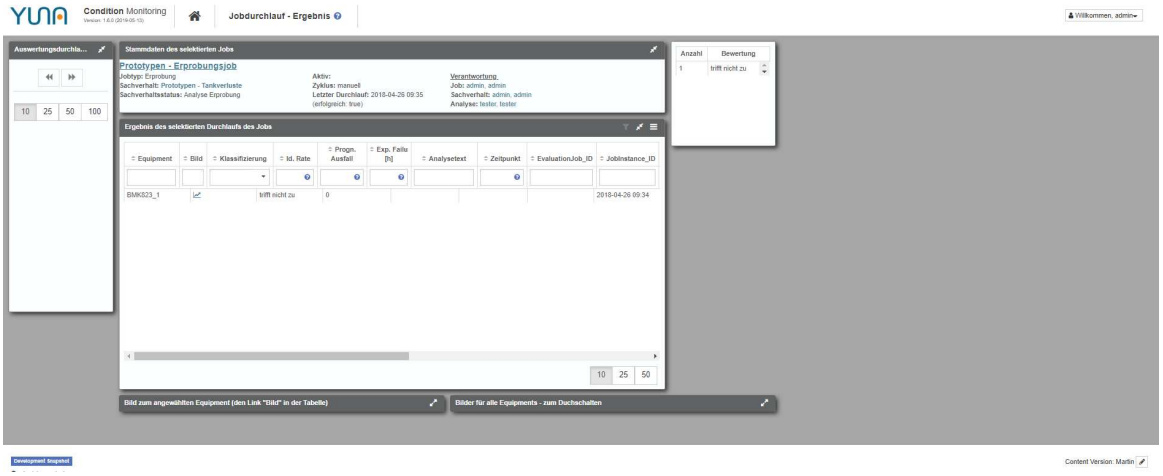
Beim Aufruf des Portals wird Standardmäßig die Dashboard-Version "default" geladen. Diese kann unten rechts im Portal geändert werden. Hierzu geben Sie einfach das von Ihnen gewünschte Referenz-Tag ein und klicken auf den Haken. Das Portal wird anschließend in der Dashboard-Version des neuen Referenz-Tags geladen und angezeigt.

Die Nutzung von Referenz-Tags ist nur für Portal-User der Rolle System_Admin möglich! Reguläre User haben keinen Zugang zu dieser Funktion, da sie keine Dashboard Inhalte entwickeln sondern das Portal nur konsumieren.

1. Standard-Dashboard (unter dem Referenz-Tag "default"):



2. Wechsel auf das Referenz-Tag "Martin", um die Version "Martin" zu laden und Anpassungen am Dashboard zu testen.
3. Unter dem Referenz-Tag "Martin" ist bei der Dashboard-Definition des Widgets links wohl ein Fehler unterlaufen, weshalb die Jobdurchläufe nicht mehr angezeigt werden. Die Dashboard Inhalte des regulären Portals bleiben gleichzeitig für alle anderen User nutzbar.



4. Wechsel auf das Referenz-Tag "Test", um die Version "Test" zu laden.

5. Ansicht des Dashboards unter Referenz-Tag "Test". Wie man sieht wurde unter diesem Referenz-Tag keiner oder fehlerhafter Dashboard Inhalt deployt, weshalb hier nichts angezeigt wird.

6. Das Referenz-Tag kann wieder entfernt werden, um wieder das originale, öffentlichen Dashboard anzuschauen.

5.3.5. Datenbanknutzer

Zum Deployen von Dashboards muss zuvor ein Datenbanknutzer angelegt werden.

Dieser benötigt folgende Rechte:

R=Lesen, W=Schreiben

SP-Schema	Rechte
DataID	R/W
DataIDMeta	R/W
DataIDMetaAssoc	R/W
DataIDMetaField	R/W
DataIDMetaFieldParam	R/W
DataIDMetaTypes	R
DataType	R
Grid	R/W
Query	R/W
QueryAssoc	R/W

SP-Schema	Rechte
Filter	R/W
FilterHierarchy	R/W
FilterMenu	R/W
FilterQueryAssoc	R/W
FilterRoleAssoc	R/W
LookupQuery	R/W
Role	R
View	R/W

Portal-Schema	Rechte
Localization	R

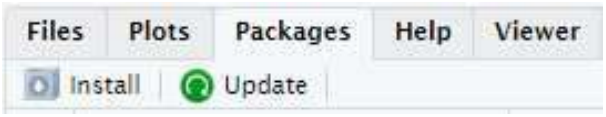
5.4. dseonnect in RStudio

Um die Konnektivität zum YUNA Backend zu gewährleisten muss das R-Paket dseonnect in RStudio installiert und aktiviert werden.

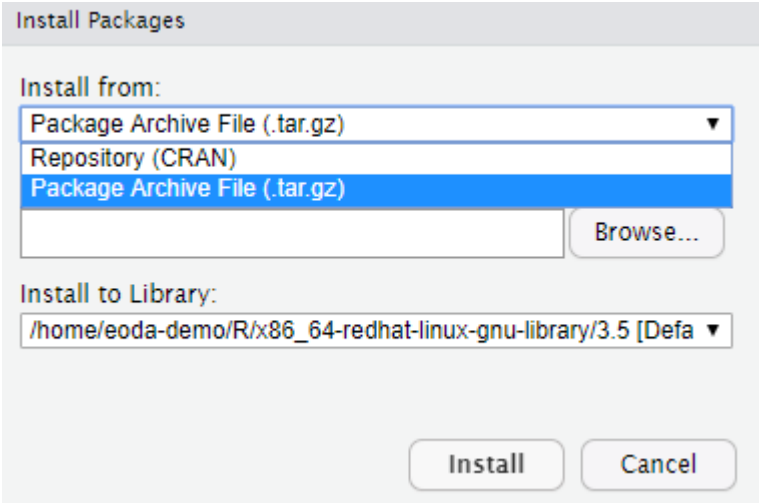
Das Paket erhalten Sie von Ihrem zuständigen Systemadministrator.

5.4.1. dseonnect installieren

1. Auf Packages klicken



2. Auf Install klicken
3. Package Archive File auswählen



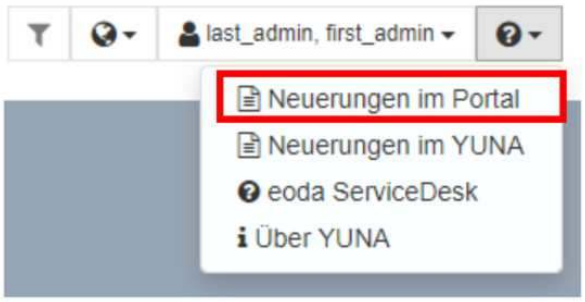
4. Package Archive File auswählen
5. Auf Browse klicken

5.5. Widget Typen

5.5.1. Changelog (changelogDirective)

Das ChangeLog ist eine View in YUNA, in der Änderungen, Bugfixes und neue Features angezeigt werden.

Es kann über den Info Button oben rechts über das Dropdown-Menü "Neuerungen im Portal" erreicht werden.



Das Changelog besteht derzeit aus einem Einzelselektions-Widget, in der die jeweilige Version ausgewählt werden kann, für die die Änderungen betrachtet werden sollen. Abhängig von der Einzelselektion sind zwei Widgets vom Typ ChangelogDirective, die die Änderungen am Portal bzw. dem Dashboard für die ausgewählte Version anzeigen.

Ein Changelog anlegen

Um ein Changelog im Portal einzubinden, müssen mehrere Widgets angelegt und miteinander verknüpft werden. Die einzelnen Schritte werden im Folgenden aufgeführt:

Schritt 1: View anlegen

Zuerst muss eine View angelegt werden, in der ihre Widgets eingebunden werden

Schritt 2: SingleChoiceDirective anlegen

Damit im Changelog Informationen zu unterschiedlichen Versionen angezeigt werden können, muss ein SingleChoiceDirective angelegt werden. Dies dient schließlich dazu, durch die einzelnen Versionen im Changelog durchzuklicken.

Schritt 3: Die ChangeLogDirective(s) anlegen, die die Versionsänderungen beinhalten

Die eigentlichen Informationen zu den Versionsänderungen werden im Changelog-Widget selbst dargestellt. Es können auch mehrere Changelogs in der View eingebunden werden, um beispielsweise einerseits Informationen zu Code-Änderungen und andererseits zu Änderungen am Dashboard getrennt darzustellen:

```
<xml>
  <!--
                    Copyright (c) 2017 eoda GmbH
          All Rights Reserved, see LICENSE.TXT for further details
```

```
          More information about configuring this template
          can be found in the Content Developer Guide:
          "Changelog (changelogdirective)"
-->
  <widget name="template_widget_Changelog">
    <!-- Position from left top  -->
    <position>
        <!-- y -->
        <y>0</y>
        <!-- x -->
        <x>0</x>
    </position>
    <!-- Size of the Widget -->
    <size>
      <x>7</x>
      <y>6</y>
    </size>
    <!-- Caption: title over the widget and additional features (Contextmenue..) -->
    <caption>
        <!-- Display caption -->
        <show>true</show>
        <!-- Title of the Widget -->
        <label>Changelog-Template</label>
    </caption>
    <!-- Name of the WidgetType -->
    <widgettype>changelogdirective</widgettype>
    <!-- URL-Paramters to listen on for triggering the widget -->
    <triggerParams>
      <list>VersionID</list>
    </triggerParams>
    <!-- Log-Type (ChangeLog/ContentLog) -->
    <contentKey>ChangeLog</contentKey>
  </widget>
</xml>
```

Wichtig ist die Verknüpfung der ChangelogDirectives mit dem SingleChoiceDirective. Dies geschieht, indem für die SingleChoiceDirective ein "urlParam" definiert wird, also ein Parameter, der vom SingleChoiceDirective bei der Auswahl einer Version an die URL weitergegeben wird. Im Anschluss wird für die konsumierenden ChangeLogDirectives über die Funktion triggerParams definiert, dass sie auf den vom SingleChoiceDirective erzeugten URL-Parameter hören sollen.

5.5.2. Dashboard-Übersicht

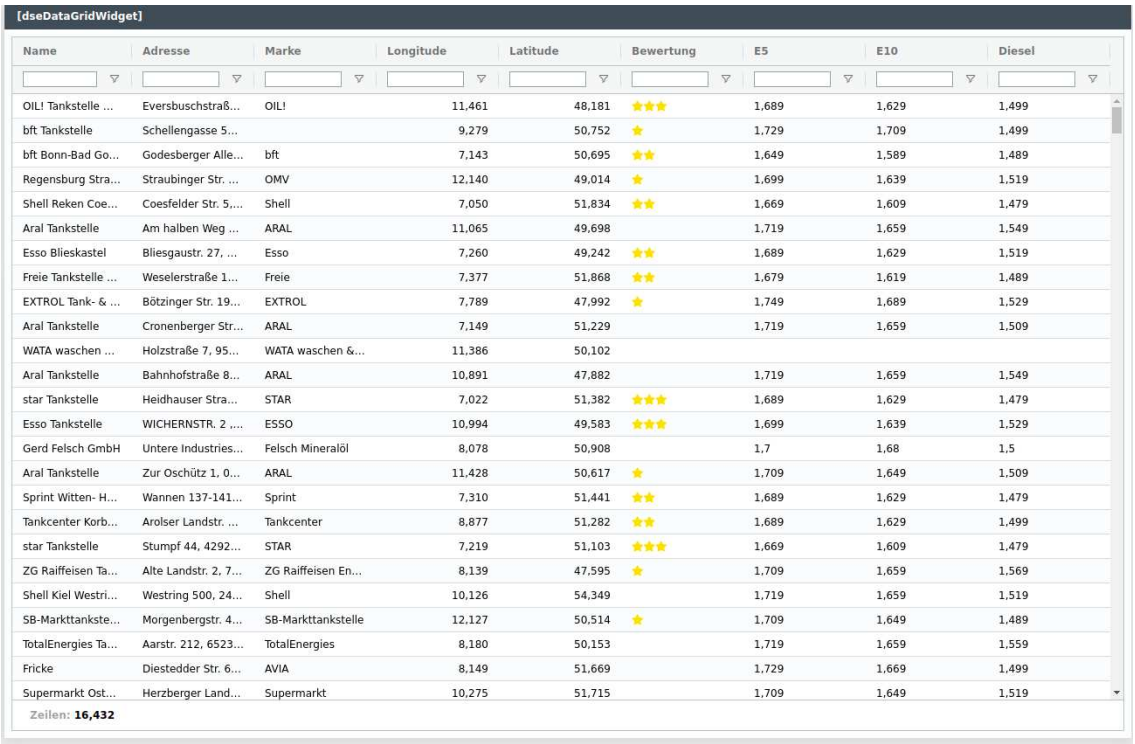
Das Dashboard-Übersicht-Widget zeigt eine Liste aller für den aktuell eingeloggten Benutzer verfügbaren Dashboards.

Definition einer Dashboard-Übersicht in YUNAML

XML

```
<xml>
  <widget>
    <!-- Position from left top  -->
    <position>
        <x>0</x>
        <y>0</y>
    </position>
    <!-- Size of the Widget -->
    <size>
      <x>8</x>
      <y>8</y>
    </size>
    <widgettype>dashboardOverviewWidget</widgettype>
  </widget>
</xml>
```

5.5.3. DataGrid-Widget



Das DataGrid-Widget erlaubt die effiziente Darstellung von großen Datenmengen und löst damit das Tabellen-Widget ab. Das DataGrid beruht auf der Bibliothek AG Grid und erbt daher viele Konzepte von AG Grid, wobei wesentliche Teile der Komplexität abstrahiert werden, um die Benutzung zu vereinfachen.

Grundlegendes:

Der folgende Abschnitt wird grundlegende Konzepte des DataGrid-Widgets beleuchten wie

beispielsweise die Möglichkeiten das Layout zu verändern oder auch der Umgang mit Filterung bzw. Sortierung. Im weiteren Verlauf des Dokuments wird erklärt werden wie man das DataGrid-Widget in YunaLang definiert.

Kontextmenü:

Das Kontextmenü wird mit einem Rechtsklick auf das DataGrid geöffnet und bietet Aktionen wie unter anderem die Möglichkeit die Informationen aus dem DataGrid zu exportieren. Die einzelnen Optionen werden im Laufe dieser Dokumentation noch genauer erklärt werden. Wichtig ist, dass das Kontextmenü in Abhängkeit von eingebunden Modulen entsprechend mehr oder weniger viele Aktionen darstellt. Aktionen, die immer angezeigt werden, sind die Funktionen zum Kopieren der DataGrid-Inhalte und das Zurücksetzen des Layouts.

Spaltenmenü:

Wenn der Nutzer über einen Spaltenkopf hovert, wird ein kleines Hamburger-Menü sichtbar, das bei einem Klick das Spaltenmenü öffnet. Nachdem der Nutzer auf das Icon geklickt hat, wird das Spaltenmenü sichtbar. Das Menü bietet drei Panele. Standardmäßig wird das Panel zum Filtern der Spalte geöffnet. Das Filtern wird in der nächsten Sektion genauer erläutert.

Wenn der Nutzer auf das mittlere Icon des Menüs klickt, kommt er auf die Auswahl möglicher Aktionen für eine Spalte. Die erste Aktionen "Spalte anpinnen" erlaubt die Spalte rechts oder links im DataGrid festzusetzen. Darunter findet sich die Sektion zur automatischen Einstellung der Größe der gewählten Spalte oder aller Spalten. Die Größe des DataGrids wird sich durch diese Aktion automatisch dem verfügbaren Platz anpassen. Die Aktion "Spalten zurücksetzen" ermöglicht, dass falls es Änderungen an dem Layout der Spalten etwa bei der Reihenfolge oder der Größe gibt, das standardmäßige Layout wieder hergestellt wird.
Zuletzt wird mit "Vollständige Spalte auswählen" die gesamte Spalte ausgewählt, was beispielsweise nützlich ist, wenn man die ganze Spalte exportieren möchte. Diese Aktion ist allerdings nur verfügbar, wenn man das rangeSelecion-Modul einbindet. Im letzten Panel des Spaltenmenüs kann der Nutzer beliebige Spalten ausblenden.

Filterung:

Das DataGrid-Widget bietet vielfältige Möglichkeiten Spalten zu filtern und zu sortieren, die an den Typen der Spalte gebunden sind. Eine Spalte vom Typ <<number,number> ermöglicht beispielsweise - wie man es erwarten würde - das Filtern und Sortieren auf numerischen Werten. Je nach Spaltentyp kann der Nutzer Bedingungen für die Filterung einer Spalte wählen und sie mit den boolschen Operatoren "UND" beziehungsweise "ODER" verknüpfen.

Sortierung:

Die Daten einer Spalte können aufsteigend oder absteigend sortiert werden. Bei einem Klick auf den Spaltenkopf kann die Sortierung der entsprechenden Spalte geändert werden. standardmäßig ist eine Spalte unsortiert. Beim ersten Klick auf den Spaltenkopf wird die Spalte aufsteigend sortiert, bei einem weiteren Klick wird die Spalte absteigend sortiert und bei einem dritten Klick auf den Spaltenkopf kehrt die Spalte in den unsortierten Ausgangszustand zurück.

Layout ändern und zurücksetzen:

Das Layout des DataGrid kann angepasst werden, beispielsweise kann die Reihenfolge der Spalten über Drag&Drop geändert werden. Ebenfalls möglich ist die Anpassung der Spaltengröße und über das Spaltenmenü können einzelne Spalten versteckt werden. Diese Änderungen am Layout werden

im Browser-Speicher persistent gespeichert.

Unter dem Punkt "Raster zurücksetzen" im Kontext-Menü kann das ursprüngliche Layout des DataGrids wieder hergestellt werden. Die im Browser gespeicherten Einstellungen werden entfernt.

YunaLang Definition:

Im Folgenden wird zunächst beispielhaft eine vollständige Definition des DataGrid-Widgets aufgeführt. Danach finden Sie eine Erklärung der einzelnen Optionen für die Definition.

```
datagridWidget ExampleDataGrid {
    size: (3, 3)
    position: (0, 0)
    inputs: [
        data <- provider
    ]

    columnDefs: [
        text {
            field: "DatabaseTextField"
            headerName: "TextFieldName"
        },
        number {
            field: "DatabaseNumberField"
            numberFormatOptions: {
                numberStyle: percent
            }
        },
        date {
            field: "DatabaseDateField"
            outputFormat: "dd-mm-yyyy"
        }
    ]
    modules: [
        csvExport,
        excelExport,
        rangeSelection,
        charts,
        statusBar {
            total: false
            selection: true
            aggregation: true
            filtered: true
        }
    ]
    licenseKey: "exampleKey"
}
```

Option	Beschreibung	Beispiel
inputs	Hier kann ein Inputchannel definiert werden. Über einen Inputchannel können Daten an das Data-Grid-Widget übergeben und dargestellt werden. Weitere Informationen: Datasource	`inputs: [ data <- provider ]`
outputs	Hier kann ein Outputchannel definiert werden.Über einen Outputchannel können Daten vom Tabellen-Widget an andere Inputchannel übergeben werden. Weitere Informationen: Datasource	`outputs: [ provider <- selected ]`
columnDefs	Über columnDefs können Spaltendefinitionen angegeben werden, die bestimmen wie im DataGrid die entsprechenden Daten in den Spalten dargestellt werden sollen. Mögliche Spaltentypen sind: text, number, date und template.	`columnDefs: [ text { field: "Field" headerName: "FieldName" }, number { field: "NumberField" numberFormatOptions: { maximumSignificantDigits: 2 } }, date { field: "DateField" outputFormat: "dd-mm-yyyy" } ]`
modules	Über Module kann die Grundfunktionialität des DataGrids erweitert werden. Informationen zu den zur Verfügung stehenden Modulen finden in der Modulbeschreibung. Das Einbinden von Modulen erfordert das Setzen eines validen Lizenzschlüssel.	`modules: [ csvExport, excelExport, charts ]`
additionalOptions	Erlaubt die Übergabe von Optionen, die direkt an die dem DataGrid zugrunde liegende Komponente von AG Grid übergeben werden. Anmerkung: Die `additionalOptions` werden als JSON-String angegeben, so dass YunaLang hier keine Unterstützung anbietet. Die zur Verfügung stehenden Optionen sind in der AG Grid-Dokumentation beschrieben. Es ist ratsam die enstprechenden Optionen aus einer JSON-Datei zu laden, statt sie als JSON-String direkt anzugeben.	`additionalOptions: loadFile "additionalOptions.json"`
licenseKey	Ein AG Grid Lizenzschlüssel	`licenseKey: "Key"`

Module

Das DataGridWidget erbt die modulare Architektur von Ag Grid und stellt Module zur Verfügung, die eingebunden werden können, um die Funktionalität des DataGrid-Widgets zu erweitern.

Das Einbinden von modulen geschieht in YunaLang über modules.

charts

Das charts-Modul stellt im Kontext-Menü das Erzeugen von Graphen über einer RangeSelection bereit. Das charts-Modul setzt also voraus, dass das rangeSelection-Modul eingebunden ist.

Das Modul `charts` bietet keine Optionen.

Beispiele:

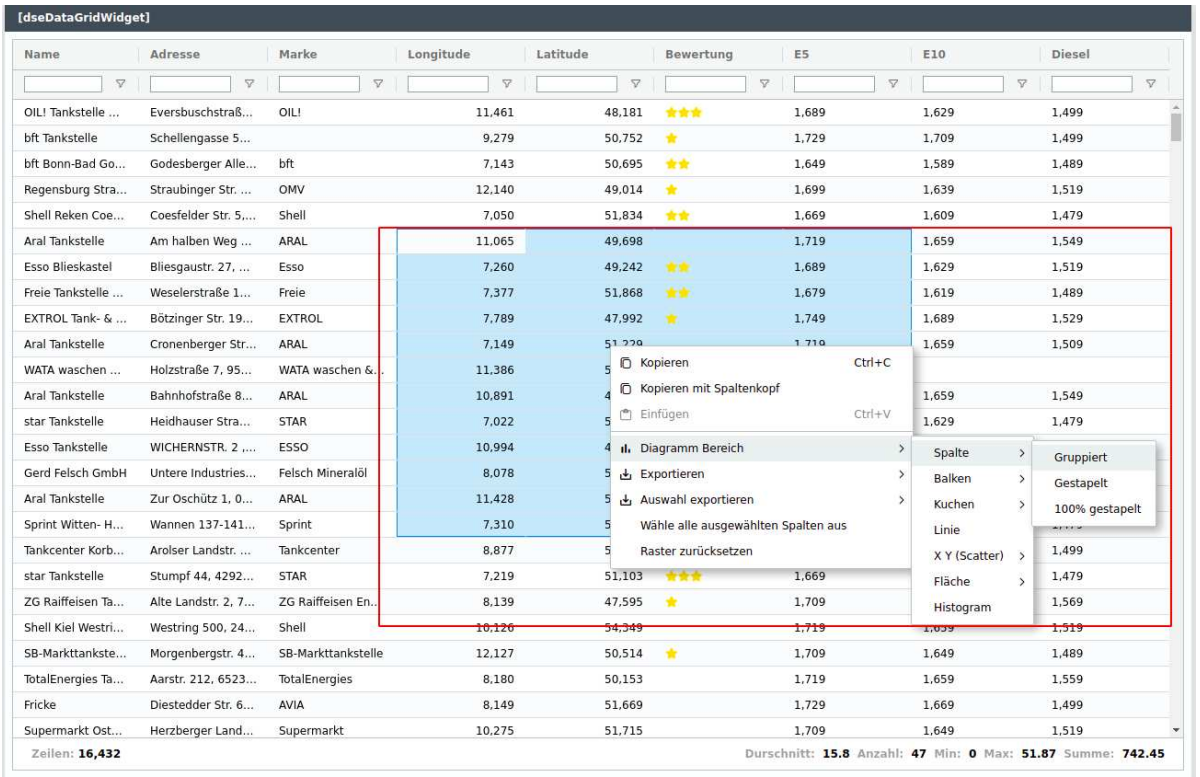
```
modules: [
    charts,
]
```

Oder auch:

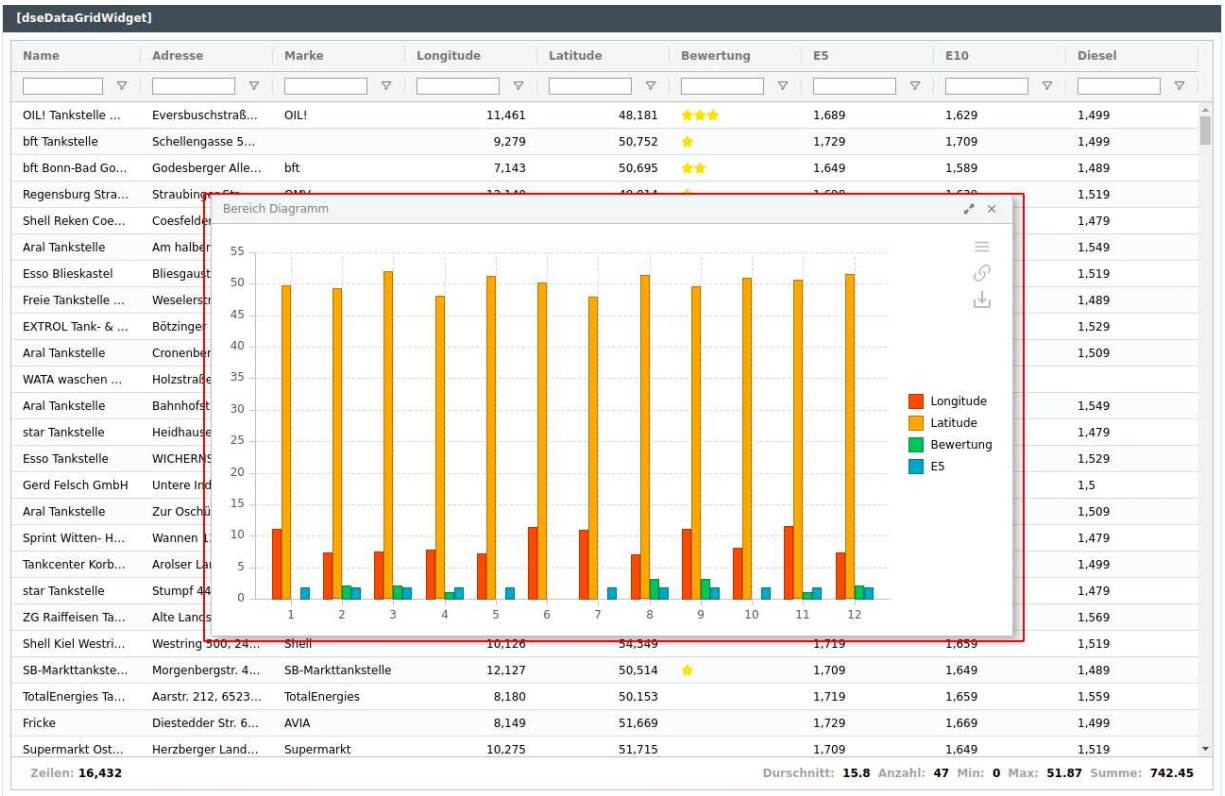
```
modules: [
    charts {}
]
```

Graph erzeugen:

1. Das nachfolgende Bild zeigt wie über eine rangeSelecion ein Graph erzeugt werden kann.



2. Der resultierende Graph aus dem ersten Bild.



Name	Adresse	Marke	Longitude	Latitude	Bewertung	E5	E10	Diesel
OIL! Tankstelle ...	Eversbuschstraß...	OIL!	11,461	48,181	★★★	1,689	1,629	1,499
bft Tankstelle	Schellengasse 5...		9,279	50,752	★	1,729	1,709	1,499
bft Bonn-Bad Go...	Godesberger Alle...	bft	7,143	50,695	★★	1,649	1,589	1,489
Regensburg Stra...	Straubinger Str.	OMV	12,148	48,814	★	1,699	1,639	1,519
Shell Reken Coe...	Coesfelder							1,479
Aral Tankstelle	Am halber							1,549
Esso Blieskastel	Bliesgaust							1,519
Freie Tankstelle ...	Weselerstr							1,489
EXTROL Tank- & ...	Bötzinger							1,529
Aral Tankstelle	Cronenber							1,509
WATA waschen ...	Holzstraße							
Aral Tankstelle	Bahnhofst							1,549
star Tankstelle	Heidhause							1,479
Esso Tankstelle	WICHERN							1,529
Gerd Felsch GmbH	Untere Ind							1,5
Aral Tankstelle	Zur Oschü							1,509
Sprint Witten- H...	Wannen 1							1,479
Tankcenter Korb...	Arolser La							1,499
star Tankstelle	Stumpf 44							1,479
ZG Raiffeisen Ta...	Alte Lands							1,569
Shell Kiel Westri...	Westring 500, 24...	Shell	10,126	54,349		1,719	1,659	1,519
SB-Markttankste...	Morgenbergstr. 4...	SB-Markttankstelle	12,127	50,514	★	1,709	1,649	1,489
TotalEnergies Ta...	Aarstr. 212, 6523...	TotalEnergies	8,180	50,153		1,719	1,659	1,559
Fricke	Diestedder Str. 6...	AVIA	8,149	51,669		1,729	1,669	1,499
Supermarkt Ost...	Herzberger Land...	Supermarkt	10,275	51,715		1,709	1,649	1,519

Zeilen: **16,432** Durschnitt: **15.8** Anzahl: **47** Min: **0** Max: **51.87** Summe: **742.45**

csvExport

Der Csv-Export funktioniert äquivalent zu dem Excel-Export. Das Exportieren ist über das Kontext-Menü möglich, wobei die ganze Tabelle oder nur eine Range Selection exportiert werden kann. Das Exportieren einer Range Selection erfordert das Einbinden des rangeSelection-Moduls.

Das Modul `csvExport` bietet keine Optionen.

Beispiel:

```
modules: [
    csvExport
]
```

Oder auch:

```
modules: [
    csvExport {}
]
```

excelExport

Der Excel-Export funktioniert äquivalent zu dem Csv-Export. Das Exportieren ist über das Kontext-Menü möglich, wobei die ganze Tabelle oder nur eine Range Selection exportiert werden kann. Das

Exportieren einer Range Selection erfordert das Einbinden des rangeSelection-Moduls.

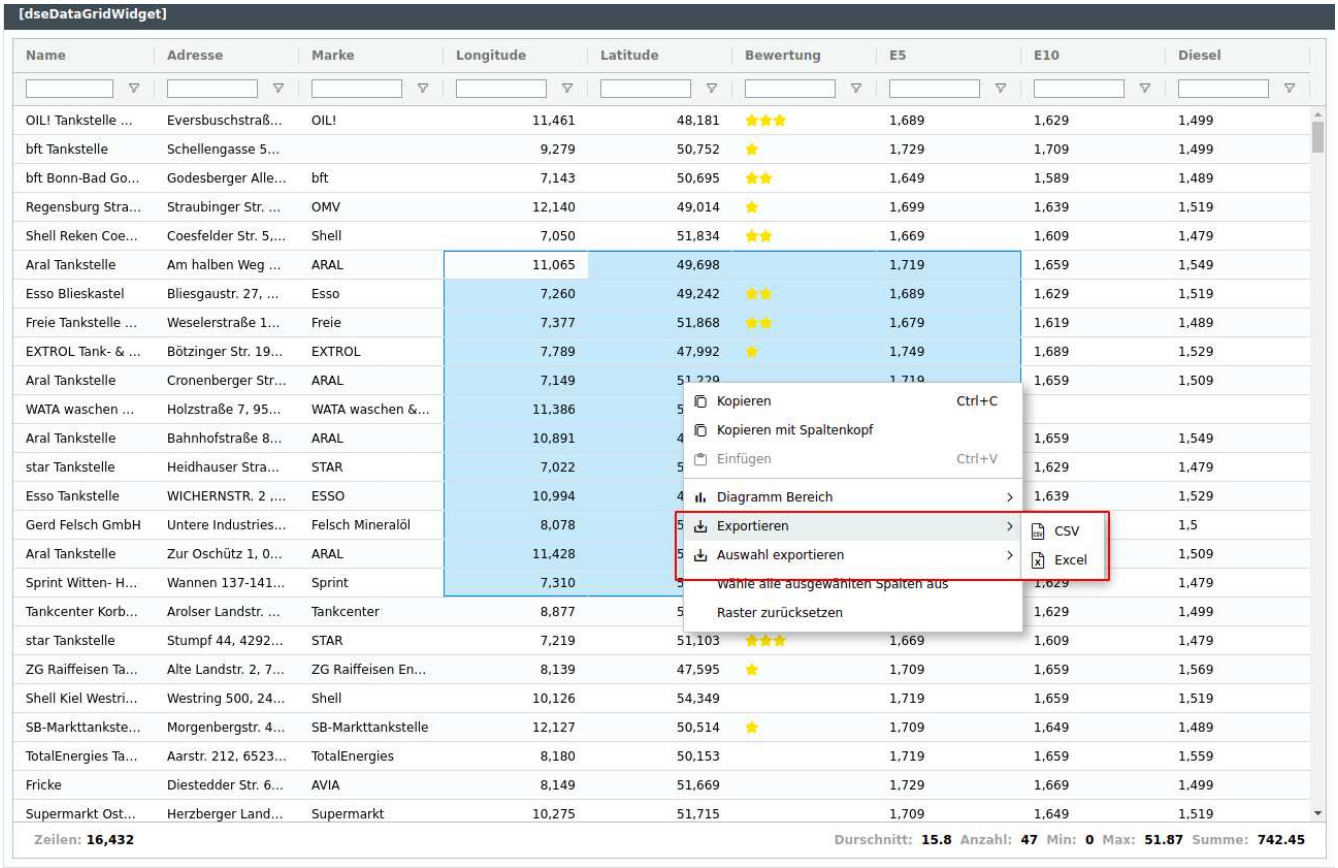
Das Modul `excelExport` bietet keine Optionen.

Beispiel:

```
modules: [
    excelExport
]
```

Oder auch

```
modules: [
    excelExport {}
]
```



rangeSelection

Es ist möglich gewisse Teilbereiche des DataGrid auszuwählen und bestimmte Operationen wie den CSV- beziehungsweise Excelexport oder das Darstellen von Charts auf diesen Teilbereichen durchzuführen. Im Kontextmenü besteht unter dem Punkt "Alle ausgewählten Spalten wählen" die Möglichkeit eine beliebige Auswahl auf die vollständigen Spalten zu erweitern. Wenn man beispielsweise einen Graphen über die vollständigen Daten von zwei Spalten anzeigen möchte, kann man eine willkürliche Selektion auf den zwei Spalten wählen, die nicht alle Zeilen beinhaltet und dann durch den Punkt "Alle ausgewählten Spalten wählen" die Auswahl auf alle Zeilen erweitern.

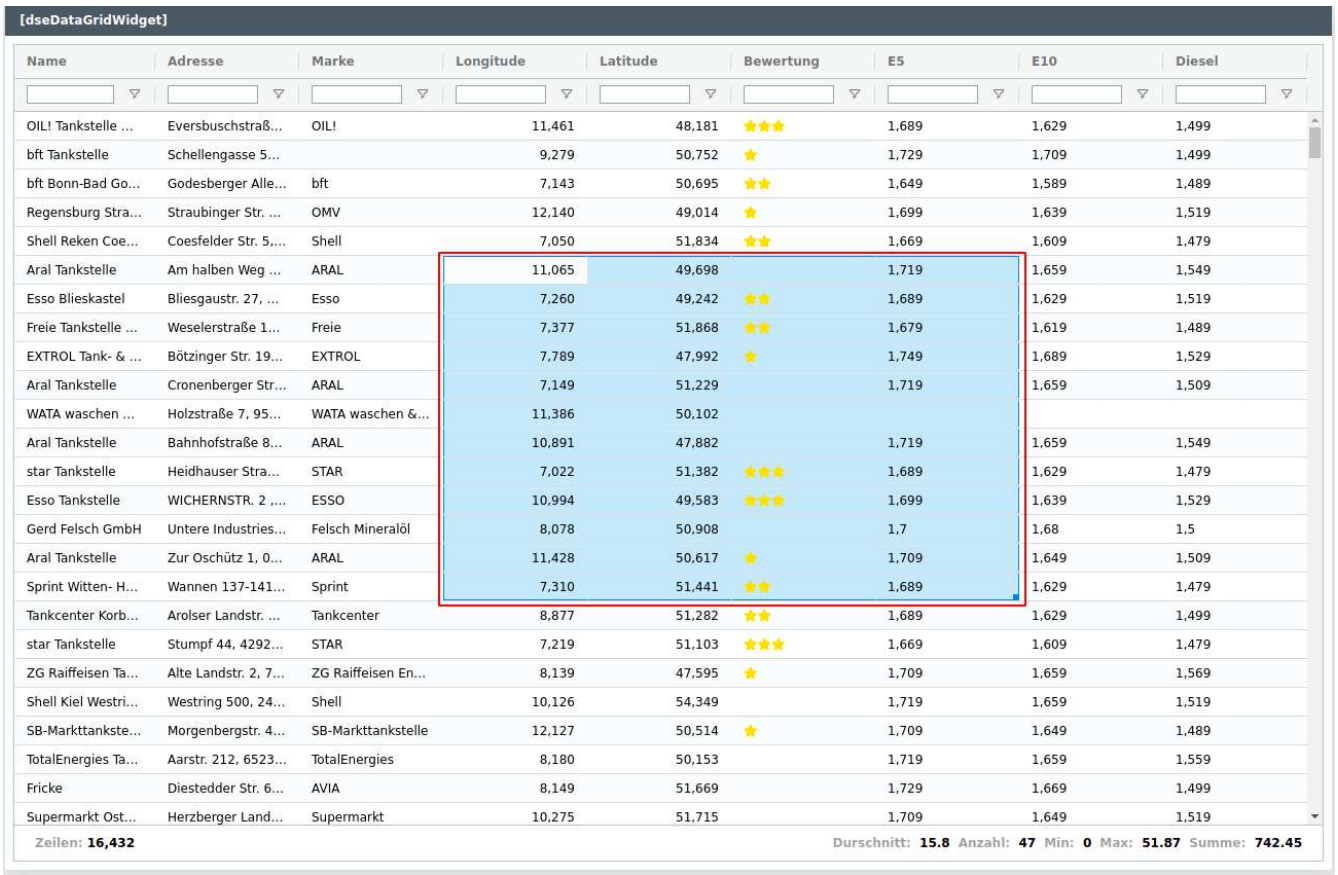
Das Modul `rangeSelecion` bietet keine Optionen.

Beispiel:

```
modules: [
    rangeSelection,
]
```

Oder auch

```
modules: [
    rangeSelection {}
]
```



statusBar

Die Status Bar kann am unteren Ende des DataGrid angezeigt werden und bietet eine Zusammenfassung der Daten im DataGrid.

Das Modul `statusBar` hat folgende Optionen:

Option	Beschreibung	Typ
total	Anzahl der Zeilen im ganzen DataGrid an	boolean, default: false

Option	Beschreibung	Typ
totalAndFiltered	Anzahl der Zeilen im ganzen DataGrid an und Anzahl Zeilen nach Anwenden von Filtern	boolean, Default: true
selection	Anzahl der Selections	boolean, Default: false
aggregation	Informationen über aggregierte Daten einer Range Selection (Durchschnit, Summe, etc.), allerdings nur in Verbindung mit dem Modul rangeSelecion.	boolean, Default: false
filtered	Anzahl der Zeilen nach Anwenden von Filtern	boolean, + Default: false

Beispiel:

```
modules: [
    statusBar {
        total: false
        totalAndFiltered: true
        selection: false
        aggregation: true
        filtered: false
    }
]
```

Spaltentypen

Jeder Spaltentyp bietet folgende grundlegende Basisoptionen:

Option	Beschreibung	Typ
field	Verweis auf die darzustellende Spalte aus der Tabelle.	String (notwendig)
headerName	Anzeigename für die Spalte, wird im Spaltenkopf angezeigt	String
editable	Bestimmt ob die Spalte editierbar ist.	Boolean
cellRules	Eine Liste von Regeln, um die Textfarbe und Hintergrundfarbe der Zelle einer Spalte dynamisch zu stylen.	Liste von CellRule
CellRule → color	Die Textfarbe einer Zelle.	String
CellRule → backgroundColor	Die Hintergrundfarbe einer Zelle.	String
CellRule → rule	Eine Regel, um zu definieren wann die Textfarbe und die Hintergrundfarbe genutzt werden sollen. Innerhalb der Regel kann auf den Wert der Zelle, auf die Daten der aktuellen Zeile und den Index der akuellen Zeile zugegriffen werden. Der Wert der Zelle ist über die Variable x verfübar, die Daten der aktuellen Zeile über data und der aktuelle Index der Zeile über rowIndex. Valide Regeln sind zum Beispiel: - 'x > 20' → der Wert der Zelle muss größer als 20 sein. - 'x > 40 && x < 50' → der Wert der Zelle muss größer als 40 und kleiner als 50 sein. - 'data.someField < 100' → der Wert des someField Feldes innerhalb der aktuellen Zeile muss kleiner 100 sein.	String
additionalOptions	Ähnlich wie auf oberster Definitonsebene kann man auch einer Spalte additionalOptions übergeben, die direkt an die dem DataGrid zugrundeliegende Komponente von AG Grid übergeben werden. Die genauen Optionen, die genutzt werden können, werden von AG Grid dokumentiert. Die Optionen sollten idealerweise aus einer JSON-Datei gelesen werden. Wichtig: Yuna-Lang bietet keine Hilfestellungen für die einzenen Optionen.	String
columnGroupShow	Wenn die Spalte Teil einer Gruppe ist, bestimmt diese Option, ob die Spalte immer angezeigt wird oder nur, wenn die Gruppe auf- oder zugeklappt ist.	`always` (default), `open`, `closed`

Die nun aufgeführten Spaltentypen erweitern die oben angeführten Optionen um ihre jeweils typ-spezifischen Optionen.

date

Der Spaltentyp `date` ermöglicht das Arbeiten mit Datums. Die Basisoptionen werden um die folgenden Optionen erweitert.

Option	Beschreibung	Typ
inputFormat	Stellt die Datenquelle das Datum als Datums-String bereit, kann hier das entsprechende Format angegeben werden, damit das Datum korrekt geparst werden kann.	String
outputFormat	Das im DataGrid darzustellende Format	String
dateLocale	Setzen der Lokalität. Über diese wird, bei nicht gesetztem "outputFormat", das anzuzeigende Datums-Format ermittelt.	String

Beispiel:

```
columnDefs: [
    date {
        field: "Field"
        inputFormat: "mm.dd.yyyy"
        outputFormat: "dd.mm.yyy"
    }
]
```

number

Der Spaltentyp `number` erlaubt das Anzeigen, Sortieren und Filtern von numerischen Daten. Neben den Basisoptionen bietet der Number-Spaltentyp Optionen zum Formattieren der numerischen Werte:

Option	Beschreibung	Typ
numberFormat Options	Implementiert die Teilmenge `currency`, `style`, `minimumIntegerDigits`, `minimumFractionDigits`, `maximumFractionDigits`, `minimumSignificantDigits` und `maximumSignificantDigits` der Intl-NumberFormat-Browser-API. Zu beachten ist, dass die Option `style` mit `numberStyle` definiert wird.	`{ currency: String numberStyle: currency | decimal (default) | percent | unit minimumFractionDigits: Integer [0;20] maximumFractionDigits: Integer [0;20] minimumSignificantDigits: Integer [1;21] maximumSiginificantDigits: Integer [1;21] minimumIntegerDigits: Integer [1;21] }` Hinweis: [x;y] steht für ein Intervall gültiger numerischer Werte.
locale	Setzen der Lokalität.	String

Beispiel:

```
columnDefs: [
    number {
        field: "FieldName"
```

```
        numberFormatOptions: {
            currency: "EUR" // <- Formattiere als Euro
            numberStyle: currency
            maximumFractionDigits: 2
        }
    }
]
```

template

Mit dem template-Spaltentyp kann man ein Handlebars-Template verwenden, um die Zellen einer Spalte zu rendern. Die Basisoptionen werden um folgende Optionen erweitert:

Optionen	Beschreibung	Typ
template	Ein Handlebars template, das in den Zellen der Spalte gerendert werden soll. Innerhalb des Templates kann auf die Daten der Zeile über row zugegriffen werden.	String (required)
getValue	Wenn man Vergleichsoperationen nicht auf den resultierenden HTML-Strings durchführen möchte, sondern auf den tatsächlichen Daten, ist es sinnvoll anzugeben wie der Wert jeder Zelle erhalten weden kann. Die Synatx kann der Dokumentation von AG Grid entnommen werden.	String
filterType	Der Filtertyp, nach dem die Spalte gefiltert werden kann.	text (default), number, date

Beispiel:

```
columnDefs: [
    template {
        field: "Field"
        template: "<div>Hello {{row.name}}</div>"
        getValue: "data.name"
        filterType: text
    }
]
```

text

Der text-Spalten Typ stellt einfachen Text im DataGrid dar und ermöglicht das Filtern und Sortieren auf Textwerten. Der Text-Typ realisiert die Basisoptionen und keine weiteren.

Beispiel:

```
columnDefs: [
    text {
        field: "Field",
```

```
        headerName: "FieldName"
    }
]
```

group

Der Spaltentyp `group` definiert eine Gruppe, die Spalten zusammenfasst. Innerhalb einer Gruppe können alle Spaltentypen vorkommen. Dies schließt `group`-Spalten selbst mit ein, sodass Gruppen verschachelt werden können.

Gruppen enthalten selbst keine Daten, sondern dienen nur dazu Struktur in einem DataGrid zu schaffen. Daher wirken Basisoptionen, die sich auf Daten beziehen (z.B. `field` oder `editable`) nicht auf Gruppen.

Gruppen sind auf- und zuklappbar, wenn an mindestens einem der `children` die Option `columnGroupShow` auf `open` oder `closed` gesetzt ist.

Option	Beschreibung	Typ
children	enthält Spaltendefinitionen, die innerhalb der Gruppe angezeigt werden	Liste von Spaltendefinitionen
marryChildren	children lassen sich durch verschieben der Spalten nicht voneinander trennen	Boolean

Beispiel:

```
columnDefs: [
    group {
        headerName: "Überschrift der Gruppe"
        marryChildren: true
        children: [
            text {
                field: "textFieldName"
            },
            number {
                field: "numberFieldName"
                columnGroupShow: open // Spalte nur zeigen, wenn Gruppe aufgeklappt
            }
        ]
    }
]
```

5.5.4. Diagramme (basechart)

Das Basechart-Widget ermöglicht die Darstellung verschiedener Highcharts-Charttypen.

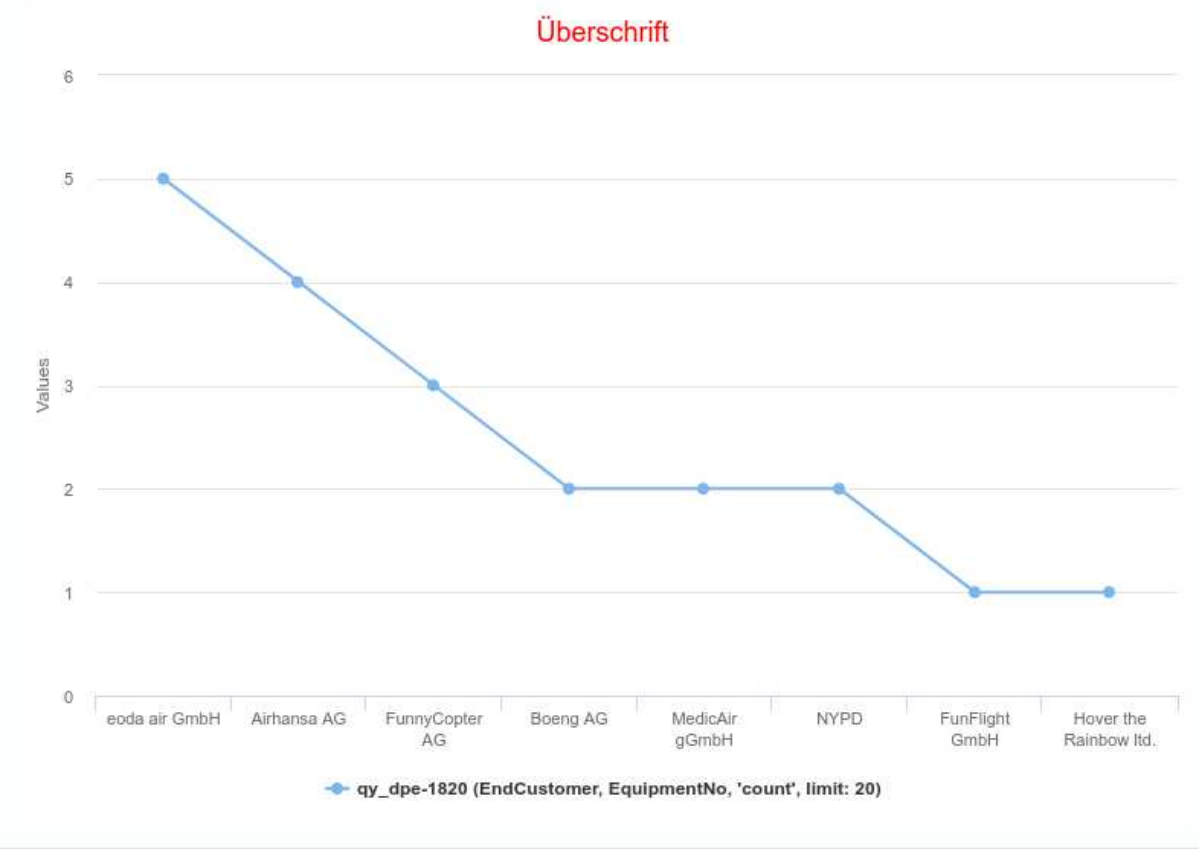
Verfügbare Diagrammtypen

Derzeit verfügbare Highcharts-Charttypen und deren Funktionen finden sich unter http://www.highcharts.com/demo.

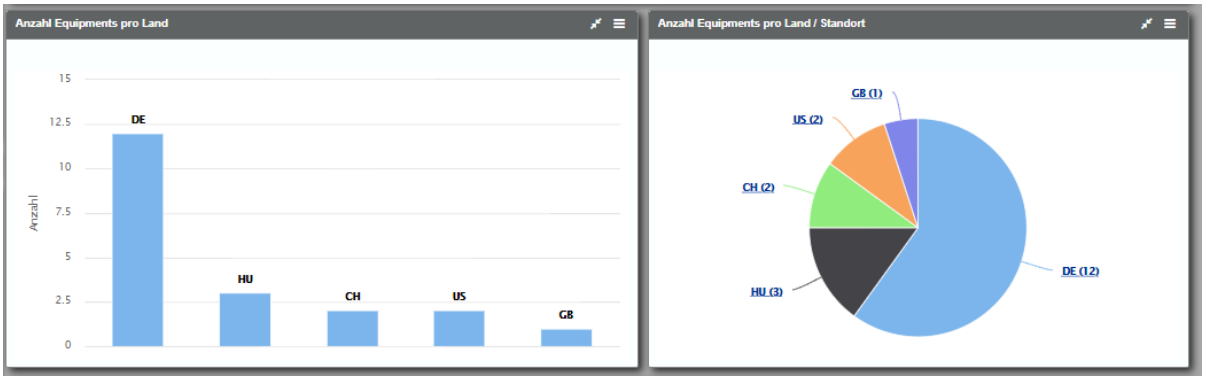
Grundsätzlich verfügbar sind statische und dynamische Grundformen und Varianten von

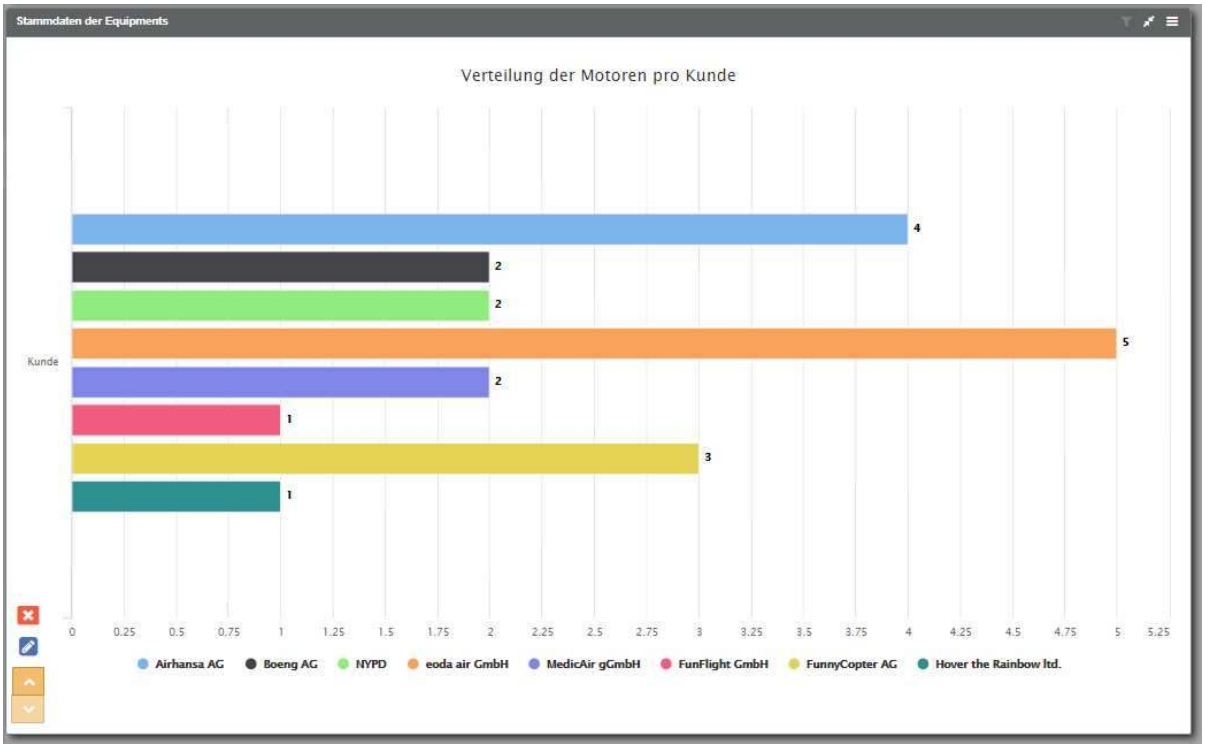
- Linien-Diagrammen
- Flächen-Diagrammen
- Stab- & Balken-Diagrammen
- Kreis- / Torten-Diagrammen
- Punkt- & Blasen-Diagrammen
- Kombinationen aus den obigen Typen
- Dynamic Charts
- 3D-Diagramme
- Heatmaps
- Boxplots
- Netz-Diagramme

Beispiele für Diagramme



[[diagramme_basechart_allgemeines]]





Anlegen eines Diagramms

YUNA XML Chart-typische Angaben

```
<xml>
    <!-- The Basechart enables you to generate different Chart-Types with the same Data-Interface  -->
    <widget name="template_widget_Basechart">
        <!-- Position from left top  -->
        <position>
                <x>0</x>
                <y>0</y>
        </position>
        <!-- Size of the Widget -->
        <size>
            <x>8</x>
            <y>6</y>
        </size>
        <!-- Caption: title over the widget and additional features (Contextmenue..) -->
        <caption>
            <!-- Display caption -->
            <show>true</show>
            <!-- Title of the Widget -->
```

```
            <label>Basechart-Template</label>
        </caption>
        <!-- Name of the WidgetType -->
        <widgettype>basechart</widgettype>
        <!-- URL-Paramters to listen on for triggering the widget -->
        <triggerParams>
            <mandatory>
                <list>equi</list>
            </mandatory>
        </triggerParams>
        <!-- Chart settings (also see: http://api.highcharts.com/highcharts) -->
        <chartSettings>
            <!-- Chart-type -->
            <chart>
                <type>line</type>
            </chart>
            <!-- Chart-title -->
            <title>
                <text>Überschrift</text>
                <style>
                    <color>red</color>
                </style>
            </title>
            <!-- Tooltip appears when hovering a series -->
            <tooltip>
                <pointFormat>
                    <![CDATA[
                      {point.ShortText}<br/>
                      {point.name}: <b>{point.y}</b><br/>
                    ]]>
                </pointFormat>
            </tooltip>
        </chartSettings>
        <!-- Data origin -->
        <chartData>
            <!-- Query to execute -->
            <dataID>qy_NameOfDataID</dataID>
            <!-- Column for the Y-value -->
            <value>OperatingHoursItemOn</value>
            <!-- Column for the X-value -->
            <category>Generation</category>
        </chartData>
    </widget>
</xml>
```

Definierbare Parameter

YUNAML-Tag	Beschreibung	Beispiel

triggerParams	In den *triggerparams* kann definiert werden, auf welche URL-Parameter das Widget reagieren soll.Wenn sich der Wert eines hier angegebenen URL-Parameter verändert, wird das Tabellen-Widget neu geladen.
	
	Sind nicht alle URL-Parameter vorhanden, wird in der Titelleiste ein Ausrufezeichen angezeigt, an dem ein Tooltip dargestellt wird.Der Text im Tooltip ist über *paramsIncompleteInfo* und *paramsIncompleteInfoTkey* frei definierbar.
	
	Weitere Informationen: Triggerparameter
		`<triggerParams>`
		`  <mandatory>`
		`  <list>UrlParameter1</list>`
		`  </mandatory>`
		`</triggerParams>`

paramsIncompleteInfo	Der Text, der hierbei hinterlegt wird, wird als der Standardwert verwendet.Dieser kommt zum Einsatz, wenn die Übersetzung über den Translation-Key fehlgeschlagen ist.
	
	Weitere Informationen: Übersetzbare Inhalte
		`<paramsIncompleteInfo>`
		`  default translation`
		`</paramsIncompleteInfo>`

paramsIncompleteInfoTkey	Hier wird der Translation-Key eingetragen.Über diesen wird aus der Aktuell gewählten Sprache die Übersetzung für den Tooltip angezeigt.
	
	Weitere Informationen: Übersetzbare Inhalte
		`<paramsIncompleteInfoTkey>`
		`  my.translation.key`
		`</paramsIncompleteInfoTkey>`

chartSettings	Die Eigenschaften des Diagramms. Bis auf die Eigenschaft series sind alle Eigenschaften, die in Highcharts definiert werden können, auch als YUNAML definierbar. siehe https://api.highcharts.com/highcharts/6.0.2	`<chartSettings>` `  <chart>` `  <type>line</type>` `  </chart>` `  <title>` `  <text>Überschrift</text>` `  <style>` `  <color>red</color>` `  </style>` `  </title>` `  <tooltip>` `  <pointFormat>` `<![CDATA[{point.ShortText} {point.name}: <b>{point.y}</b> ]]>` `  </pointFormat>` `  </tooltip>` `</chartSettings>`

<table>
<tr><th>YUNAML-Tag</th><th>Beschreibung</th><th>Beispiel</th></tr>
<tr>
<td>chartData

chartData > dataID

chartData > value

chartData > category

chartData > time

chartData > groupBy

chartData > limit

chartData > aggregate

chartData > orderByCategory

chartData > orderByGroup</td>
<td>Verarbeitung und Darstellung der Daten im Diagramm
<table>
<tr><th>Unterelement</th><th>Beschreibung</th></tr>
<tr><td>dataID</td><td>Mithilfe einer DataID kann eine Datenbankabfrage ausgeführt werden. Das daraus resultierende Ergebnis wird dann in der Tabelle dargestellt.</td></tr>
<tr><td>value</td><td>Name der Spalte, die für die Y-Werte verwendet werden soll.</td></tr>
<tr><td>category</td><td>Name der Spalte, die bei kategorischen X-Werten verwendet werden soll.</td></tr>
<tr><td>time</td><td>Name der Spalte, die bei Zeitstempel X-Werten verwendet werden soll. Kann alternativ für category verwendet werden.</td></tr>
<tr><td>groupBy</td><td>Name der Spalte, nach der kategorische Daten gruppiert werden sollen.</td></tr>
<tr><td>limit</td><td>Standardmäßig: 0
Begrenzung der anzuzeigenden Daten auf die ersten n Werte (absteigend nach Werten sortiert). Der Wert 0 bedeutet, dass die Daten nicht begrenzt werden.</td></tr>
<tr><td>aggregate</td><td>Standardmäßig: sum
(Aufsummierung der gruppierten Werte)
Aggregation der durch groupBy gruppierten Werte.</td></tr>
<tr><td>orderByCategory</td><td>Aufsteigende (asc) oder Absteigende (desc) Sortierung der Kategorie-Werte.</td></tr>
<tr><td>orderByGroup</td><td>Aufsteigende (asc) oder Absteigende (desc) Sortierung der gruppierten Werte.</td></tr>
</table>
</td>
<td>

```
<chartData>
  <dataID>qy_my_data_id</dataID>
  <value>OperatingHoursItemOn</value>
  <category>Generation</category>
  <groupBy>Generation</groupBy>
  <limit>100</limit>
  <aggregate>count</aggregate>
  <orderByCategory>
    asc
  </orderByCategory>
  <orderByGroup>desc</orderByGroup>
</chartData>
```

</td>
</tr>
</table>

<table>
<tr><th>YUNAML-Tag</th><th>Beschreibung</th><th>Beispiel</th></tr>
<tr>
<td>chartOptions

chartOptions > asDrilldown

chartOptions > asTimeseries

chartOptions > autoAxesTitles

chartOptions > additionalColumns</td>
<td>Zusätzliche Darstellungsoptionen
<table>
<tr><th>Unterelement</th><th>Beschreibung</th></tr>
<tr><td>asDrilldown</td><td>Standardmäßig: false Wenn true wird die Gruppierung als Drilldown dargestellt.</td></tr>
<tr><td>asTimeseries</td><td>Standardmäßig: false Wenn true wird die X-Achse als Zeitreihe dargestellt Siehe chartData.time</td></tr>
<tr><td>autoAxesTitles</td><td>Standardmäßig: false Wenn true, werden die Titel der Axen automatisch bestimmt.</td></tr>
<tr><td>additionalColumns</td><td>Spaltennamen, die zusätzlich als Eigenschaft eines Punktes hinzugefügt werden sollen. Funktioniert nicht bei gruppierten Daten.</td></tr>
</table>
</td>
<td>

```
<chartOptions>
  <asDrilldown>false</asDrilldown>
  <asTimeseries>false</asTimeseries>
  <autoAxesTitles>true</autoAxesTitles>
  <additionalColumns>
    Generation
  </additionalColumns>
</chartOptions>
```

</td>
</tr>
</table>

Basisdefinition für ein Chart-Widget

```
<xml>
  <widget name ="bch_MeasurementChart">
    <position>
        <y>0</y>
        <x>0</x>
    </position>
    <size>
      <x>8</x>
      <y>5</y>
    </size>
    <widgettype>basechart</widgettype>
    <caption><show>false</show></caption>
    <chartData>
      <dataID>qy_Measurement</dataID>
      <value>Count</value>
      <category>JobName</category>
      <groupBy></groupBy>
      <limit>0</limit>
      <aggregate>sum</aggregate>
    </chartData>
```

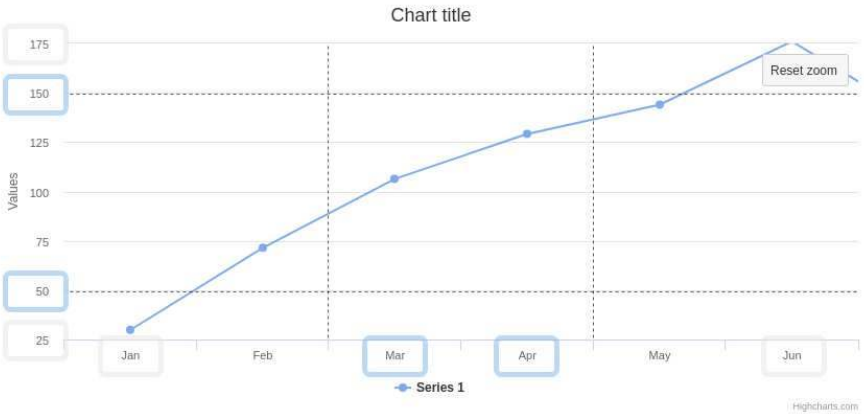
```
    <chartOptions>
      <asTimeseries>false</asTimeseries>
      <asDrilldown>false</asDrilldown>
    </chartOptions>
  </widget>
</xml>
```

Zoom:

Um in einem Charts das Zoomen zu ermöglichen, muss in den `chartSettings` ein `zoomType` angegeben werden, der die zu transformierenden Dimensionen beschreibt (x, y oder xy). Ist der Parameter gesetzt, kann durch eine mit dem Cursor selektierte Fläche der neue Mittelpunkt angegeben und die Transformation der angewendet werden (siehe Demo).

Beispiel für eine XY-Transformation:

(Die Darstellung der Achsen wurde in dieser Abbildung zum besseren Verständnis an manchen Stellen hervorgehoben)



Definition des Zoom-Verhaltens in YUNA-ML

```
<xml>
   ...
   <widget name="my_basechart">
```

```
      ...
      <chartSettings>
         <!-- General chart options -->
         <chart>
            ...
            <!-- Transformable dimensions by dragging the mouse -->
            <zoomType>xy</zoomType>
         </chart>
         ...
      </chartSettings>
      ...
   </widget>
</xml>
```

5.5.5. Einzelselektion (singleChoiceDirective)

Die singleChoiceDirective ist ein Widget-Typ aus dem Bereich der Filter.

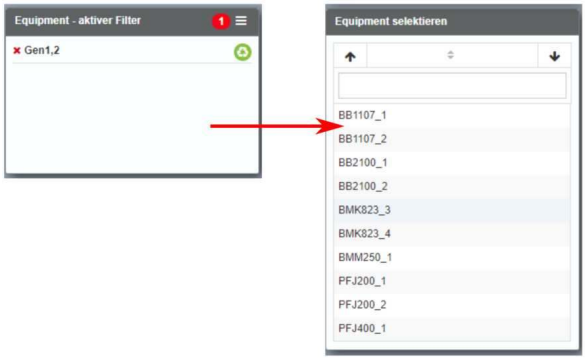
Funktionen

Listenelemente anzeigen lassen und durchschalten

Die Equipments (z.B. Maschinen, Autos, Flugzeuge,...) werden im Widget in einer Liste dargestellt. In der Liste dargestellten Geräte können einzeln ausgewählt werden.

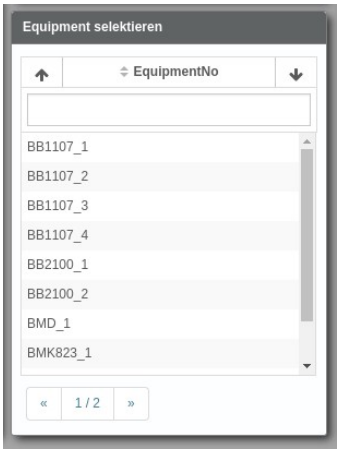
Da immer nur ein Gerät ausgewählt werden kann, spricht man von einer Einzelselektion.

Die Auswahl in der Einzelselektion kann durch einen Primär- oder Sekundärfilter vorgefiltert werden



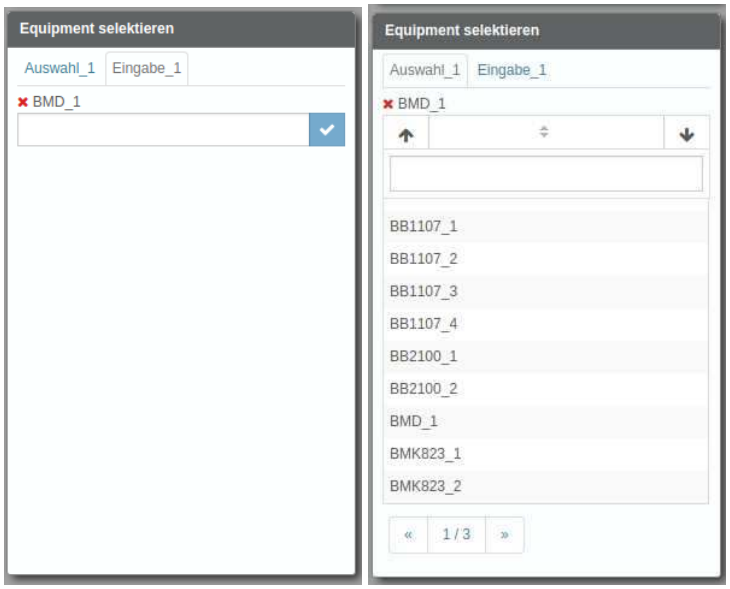
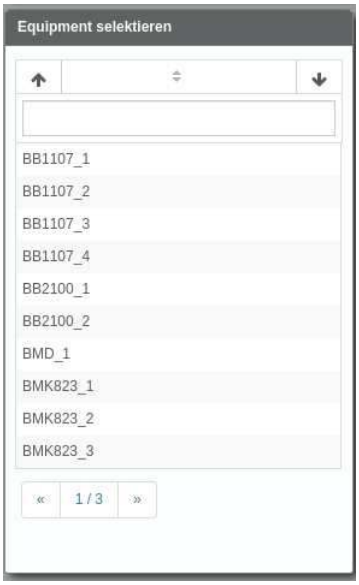
Liste durchsuchen

Bei der Einzelselektion steht dem Nutzer zusätzlich ein Suchfeld zur Verfügung, um ein Equipment direkt anzuwählen.



Tabs & Eingabefeld

Neben der tabellarischen Auswahl können in der Einzelselektion auch Eingabefelder anzeigt werden. Über diese können die gewünschten Werte direkt eingegeben werden. Darüber hinaus ist es möglich beide Darstellungen mit Tabs getrennt darzustellen.

Nutzung von Tabs (Mit Auswahl-Anzeige)	Ohne Nutzung von Tabs
	

Für die SingleChoiceDirective gilt seit Einführung von Tabs folgendes Verhalten:

- Die Funktion "Tabs" ist ein optionales Feature und kann definiert oder weggelassen werden.

> Bei der Nutzung von Tabs wird der unter <defaultValue> gesetzte Wert nicht angezeigt.

- **Wird "Tabs" nicht verwendet**, so wird per default nur die zuvor bekannte Listenansicht inkl. Suchfeld verwendet
- **Wird "Tabs" verwendet,** so hat der Dashboard Developer folgende Möglichkeiten:
  - In "list"-tags werden die einzelnen Tabs definiert
  - Über den Parameter "type" wird der Typ des Tabs definiert (choice für die bekannte Liste inkl. Suchfeld, input für das Eingabefeld)
  - Über den Parameter "label" wird der Anzeigename des Tabs definiert

- Über den Parameter "labelTransId" kann eine Übersetzungs-ID angegeben werden (customer.tablabel.[labelTransId])
- Über den optionalen Parameter "showValue" kann bei Bedarf definiert werden, ob das jeweilige Tab das ausgewählte Element darstellt (für Listenansicht default=false, bei Eingabe default=true)
- Über den optionalen Parameter "readData" kann bei Bedarf definiert werden, ob ein Tab die Filterliste ausliest und somit eine Listenauswahl darstellt (default=true bei Liste mit Suchfeld; default=false bei Eingabefeld).

Definition der Tabs-Funktion in YUNA-ML

```
<tabs>
    <list>
      <label>Auswahl</label>
      <labelTransId>select</labelTransId>
      <type>choice</type>
      <showValue>true</showValue>
    </list>
    <list>
      <label>Eingabe</label>
      <type>input</type>
      <default>true</default>
    </list>
</tabs>
```

Widget anlegen und gestalten

```
<xml>
  <widget name="template_widget_Singlechoice">
    <!-- Position from left top  -->
    <position>
        <x>0</x>
        <y>0</y>
    </position>
    <!-- Size of the Widget -->
    <size>
      <x>3</x>
      <y>5.7</y>
    </size>
    <!-- Caption: title over the widget and additional features (Contextmenue) -->
    <caption>
      <show>true</show>
      <label>Singlechoice-Template</label>
    </caption>
    <!-- WidgetType -->
    <widgettype>singlechoicedirective</widgettype>
    <!-- Data-ID -->
    <dataID>qy_NameOfDataID</dataID>
    <!-- URL-Paramter to trigger on -->
    <triggerParams>
```

```
        <mandatory>
            <list>filter2</list>
            <list>equi</list>
        </mandatory>
    </triggerParams>
    <!-- URL-Parameter to be set by the widget -->
    <urlParam>equi</urlParam>
    <tabs>
      <list>
        <label>Auswahl</label>
        <labelTransId>select</labelTransId>
        <type>choice</type>
        <showValue>true</showValue>
      </list>
      <list>
        <label>Eingabe</label>
        <type>input</type>
        <default>true</default>
      </list>
    </tabs>
  </widget>
</xml>
```

Verfügbare Optionen für die singleChoiceDirective

Feld	Mögliche Werte	Default	Beschreibung
Allgemeine Optionen			
position	Zahlenwerte für x- und y-Koordinaten	0 / 0	Definiert die Position des Widgets im Grid
size	Zahlenwerte für x- und y-Koordinaten	0 / 0	Definiert die Größe des Widgets
Widgettype	singlechoicedirective		Definiert den Widgettyp als Einzelselektions-Widget
dataID	dataID aus der Datenbank		Gibt die Datenherkunft an.
urlparam			Definiert den Parameter, der an die URL übergeben wird, aus der z.B. abhängige Widgets (z.B. Charts) ihre Triggerparameter beziehen
triggerParams (mandatory / optional)	Namen der Parameter	[ ]	Definiert die Parameter, die für die Anzeige der Daten im Widget zuständig sind

Feld	Mögliche Werte	Default	Beschreibung
val			
label	Freitext		
labeltype			Optional: Definiert den Datentyp des Labels
flag	Feld dessen Wert als Flag interpretiert werden soll.		Der Wert 0 wird als '**false**' interpretiert, Der Wert 1 als '**true**'. Wenn der Wert nicht existiert, wird er als '**undefined**' interpretiert.
defaultValue	Freitext		Wird kein anderer Wert aus der URL geladen, setzt das Widget den angegebenen Wert sowohl in der URL, als auch in der Auswahl der Widgets selbst. Bei Verwendung des 'Choice'-Tabs (Default) wird der eingegebene Wert gegen die von der DataID übergebenen Werte validiert.
mandatory	true, false	false	Ist der Wert "true", wird das Widget hervorgehoben, wenn keine Auswahl getroffen wurde.
Caption-Optionen			
show	true, false		Einblenden oder Ausblenden des Widgets
label	Freitext		Definiert den die Überschrift des Widgets
Footer-Optionen			
hideTableFooter	Siehe Tabellen-Widget		
countsOptions	Siehe Tabellen-Widget		
Appearance-Optionen			
enlargeableY	true, false		Definiert, ob das Widget durch einen Button ein- und ausgeklappt werden kann

Feld	Mögliche Werte	Default	Beschreibung
enlargedY	true, false		Definiert, ob das Widget standardmäßig ein- oder ausgeklappt ist ("enlargedY": true heißt, das Widget ist ausgeklappt)
Tab-Optionen			

tabs	Array der angezeigten Tabs	<pre>{
		"tabs" : {
		"label":"default",
		"type":"choice"
		}
		}</pre>
			Definiert die Anzeige von Tabs für verschiedene Auswahloptionen. Das Label dient hierbei als Überschrift für den jeweiligen Tab. Über den Typ kann die Auswahloption gewählt werden (z.Z. *choice* für die tabelarische Darstellung und *input* für die Auswahl über ein Eingabefeld). Der boolsche Parameter "default" ermöglicht eine Vorauswahl welcher Tab beim Laden der Seite ausgewählt ist.
			
			Über den optionalen Parameter "showValue" kann bei Bedarf definiert werden, ob das jeweilige Tab das ausgewählte Element darstellt (für Listenansicht default=false, bei Eingabe default=true).

5.5.6. Erweitertes Kachelwidget (stateTileDirective)

Ziele und Nutzen des Anwenders

Mit Hilfe des Kachelwidgets wird der Status eines Testzyklus dargestellt.

Die Statusanzeige basiert dabei auf Datenbankeinträge die zuvor bereitgestellt wurden.

Dabei liefert der Testzyklus eines der folgenden Ergebnisse:

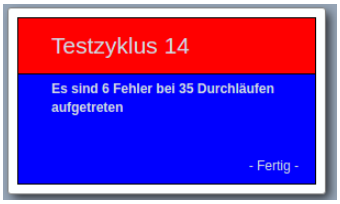
- Success
- Failed
- Aborted

Die Ergebnisse sind in einer Datenbanktabelle hinterlegt und können von dort aus abgerufen werden. Die Auswertung der Ergebnisse soll farblich in der Masterübersicht der jeweiligen Testdaten gekennzeichnet werden und dem Betrachter einen möglichst schnellen Überblick zu dem Testausgang liefern. Folgendes Muster soll als Orientierung zur Umsetzung dienen:

Folgende Farbcodes sind implementiert:

Ergebnis/Status	Farbe	Priorität
Success	Grün	3
Failed	Rot	2
Aborted	Gelb	1

Zusätzlich zur farbliche Markierung werden in dem Kachel-Rumpf Informationen zu den Testzyklen angezeigt:



Aufbau

YUNA ML Es müssen neben den Standardeigenschaften für ein Widget die folgende Angaben gemacht werden (Caption wird üblicherweise weggelassen).

```
<xml>
    <widget>
        <position>
                <x>0</x>
                <y>0</y>
        </position>
        <size>
            <x>8</x>
            <y>6</y>
        </size>
        <widgettype>statetiledirective</widgettype>
        <dataID>qy_StateTile</dataID>
        <queryParams></queryParams>
```

```
        <tile>
            <id>1</id>
            <template>
                <main>
                    Es sind {errorCount} Fehler bei {count} Durchläufen aufgetreten
                </main>
                <footer> </footer>
            </template>
            <params>
                <cyleNo>14</cyleNo>
                <result>failed</result>
                <resultId>1</resultId>
                <current>actual</current>
                <count>35</count>
                <errorCount>6</errorCount>
                <firstError>8</firstError>
            </params>
        </tile>
    </widget>
</xml>
```

Weitere zu spezifiziernde Eigenschaften eines Kachel-Widgets:

Feld	Mögliche Werte	Beschreibung
widgetname	"statetiledirective"	Festlegung des Widgettyps
dataID	"Text"	Datenherkunft
queryParams	Objekt	Optionale Abfrageparameter, sofern in der dataID definiert
tile	Objekt	Festlegung der Kacheleigenschaften

Template

Das Template ist geteilt in Header, Main und Footer. Hier können alternativ zu den Defaulteinstellungen HTML-Vorlagen für die Widgetbereiche angegeben werden:

```
"template": {
    "header": null,
    "main": "Es sind {errorCount} Fehler<br>bei {count} Durchläufen aufgetreten.",
    "footer": null
}
```

Feld	Mögliche Werte	Beschreibung
header, main, footer	"Text"	HTML-Text mit möglichen Parametern in geschweiften Klammern für die entspechenden Bereiche der Kachel. Die Parameter werden bei der Darstellung des Widgets 1. durch die params aus dem Tile-Objekt und 2. durch die Werte der ersten Zeile des Abfrageergebnisses der dataID ersetzt. Unbekannte Parameter bleiben unverändert als Text erhalten.
	null	Default-Wert wird verwendet.

Parameter

Die params ersetzen im ersten Schritt die in den Templates definierten Parameter. Neben den für das Kachelwidget vorgegebenen Standard-Parametern sind weitere beliebige Parameter möglich:

```
params: {
    cycleNo: 14,
    result: 'failed',
    current: 'actual',
    count: 35,
    errorCount: 6,
    firstError: 8
 }
```

Feld	Mögliche Werte	Beschreibung
cycleNo	Nummer	Nummer bzw. ID des Testzyklus
result	"failed", "aborted", "success"	Ergebnis: fehlgeschlagen, abgebrochen oder erfolgreich
current	"past", "actual", "running"	Aktueller Zustand: länger nicht durchgelaufen, in letzter Zeit gelaufen oder derzeit am laufen
count	Nummer	Anzahl der Durchläufe
errorCount	Nummer	Anzahl der Fehler
firstError	Nummer	Durchlauf mit dem ersten Fehler

Optionen

Derzeit ist eine weitere Option möglich:

```
"options": {
    "caseSensitive": false
```

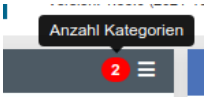
```
}
```

Feld	Mögliche Werte	Beschreibung
caseSensitive	true	Default. Beim Ersetzen der Parameter in den Templates wird zwischen Groß- und Kleinschreibung unterschieden.
	false	Es wird **nicht** zwischen Groß- und Kleinschreibung unterschieden, d.h. "errorCount" wird auch durch "errorcount" oder "ErrorCout" ersetzt.

5.5.7. Filter-Widgets

Aktive Filter (selectedFilterDirective)

Das Aktive-Filter-Widget zeigt aktuell die geladenen Filter gruppiert nach der Kategorie an.Mithilfe der Zahl im roten Kreis ist ersichtlich, wie viele Kategorien unter den Aktiven Filter sind



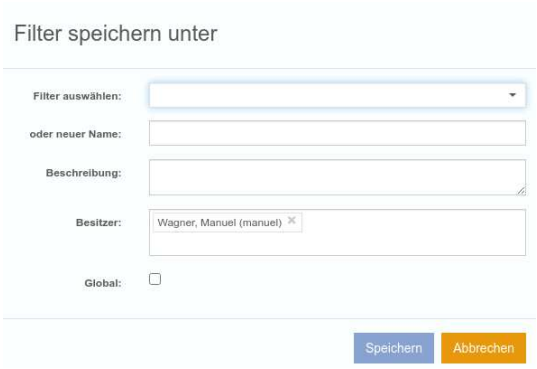
Filteroptionen

Über das Burger-Menü sind folgende Filteroptionen verfügbar

Schaltfläche	Funktion
Filterauswahl zurücksetzen	Setzt die Filterauswahl auf den als Default in der Filterverwaltung eingestellten Filter zurück.
Filter speichern	Speichert den Filter. Meta-Daten wie Name oder Besitzer können nicht verändert werden.
Filter speichern unter	Speichert den Filter unter einem neuen Namen oder überschreibt einen anderen wählbaren Filter.

Schaltfläche	Funktion
Filter laden	Öffnet einen Dialog in dem bestehende Filter gelistet werden.Der aus dieser Liste gewählte Filter wird geladen und daraufhin auf die Daten angewendet.

Filter speichern unter



Unter dem Punkt "Filter speichern unter" kann ein Filter gespeichert werden.

Feld	Bedeutung
Filter auswählen	Falls ein Filter aktualisiert/überschrieben werden soll, kann hier ein existierender gewählt werden
oder neuer Name	Falls ein neuer Filter erstellt werden soll, kann hier ein neuer Name definiert werden
Beschreibung	Platz für eine ausführliche Beschreibung des Filters
Besitzer	Ein oder Mehrere Besitzer. Besitzer haben besondere Berechtigungen, je nach dem ob der Filter global oder Lokal ist. Der Ersteller des Filters wird als Autor eingetragen, hat jedoch keine Berechtigungen an diesem Filter, wenn er nicht als Besitzer eingetragen ist.
Global	**Wert** / **Erläuterung** True: Der Filter ist für jeden Benutzer zu sehen und verwendbar.Jedoch können nur Benutzer, die als Besitzer eingetragen sind, den Filter bearbeiten und löschen. False: Der Filter ist Lokal.Dadurch können nur Benutzer, die als Besitzer eingetragen sind, den filter sehen, verwenden, bearbeiten und löschen

Optionen zum Anlegen einer selectedFilterDirective

Funktion	Mögliche Werte	Bedeutung
filterID	ID des Filters	Definiert den Filter, mit dem das Widget verknüpft werden soll.

Beispielcode für ein "Aktive Filter"-Widget

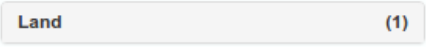
XML

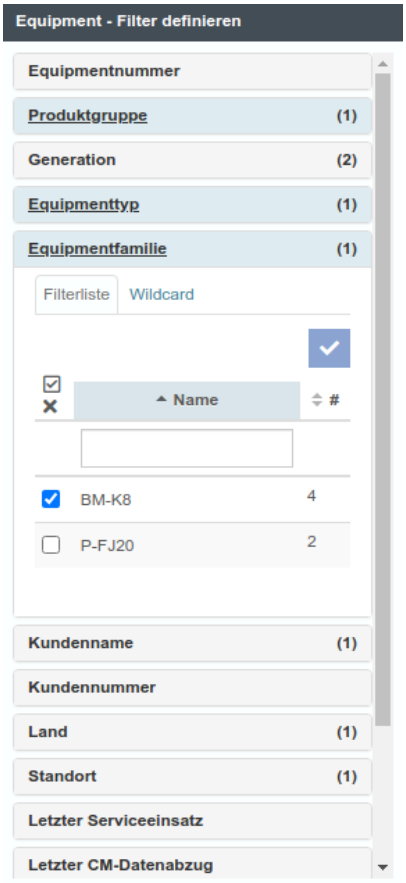
```
<xml>
  <widget name="template_widget_SelectedFilter">
    <!-- Position from left top  -->
    <position>
        <x>0</x>
        <y>0</y>
    </position>
    <!-- Size of the Widget -->
    <size>
      <x>3</x>
      <y>2</y>
    </size>
    <!-- Caption -->
    <caption>
      <show>true</show>
      <label>Selected-Filter-Template</label>
    </caption>
    <!-- Name of the WidgetType -->
    <widgettype>selectedfilterdirective</widgettype>
    <!-- ID of the filter to listen on -->
    <filterID>2</filterID>
  </widget>
</xml>
```

Mehrfachauswahl (filterdirective)

Die Mehrfachauswahl ermöglicht die Auswahl mehrerer Elemente aus einer Liste von Equipments. Durch die Nutzung von Check-Boxen oder durch die Eingabe von Werten in ein Eingabefeld kann ein Filter konfiguriert werden.

Wurde ein Filter konfiguriert, wird dieser in Form eines Hash als URL-Parameter in die URL geschrieben. Widgets, die auf den Filter reagieren sollen, können entsprechend konfiguriert werden (siehe Diagramme: triggerparams). Die Zahl die rechts in einer Filterkategorie kann zwei Bedeutungen haben:

In einer Kategorie sind **ein** oder **mehrere** Werte gewählt		Die Zahl steht für die **ausgewählte** Ausprägung
In einer Kategorie ist **kein** Wert ausgewählt		Die Zahl steht für die **wählbare** Ausprägung



> *Deaktivierte Filterkategorien*
>
> Einzelne Filterkategorien oder Auswahlmöglichkeiten werden deaktiviert, wenn diese Inhalt des aktuell geladenen Filters sind. Um diese zu verändern, muss zunächst der geladene Filter entweder im Aktive Filter (selectedFilterDirective) oder durch Doppelklick auf die entsprechende Kategorie/Auswahlmöglichkeit aufgelöst werden.

Anlegen einer FilterDirective

Parameter

Feld	Möglicher Wert	Beschreibung
filterID	ID des Filters	Definiert den Filter, mit dem das Widget verknüpft werden soll.

Beispielcode für eine Mehrfachauswahl

XML

```
<xml>
  <widget name="template_widget_Filter">
    <!-- Position from left top  -->
    <position>
        <x>0</x>
        <y>0</y>
    </position>
    <!-- Size of the Widget -->
    <size>
```

```
      <x>3</x>
      <y>6</y>
    </size>
    <!-- Caption -->
    <caption>
      <show>true</show>
      <label>Filter-Template</label>
    </caption>
    <!-- Name of the WidgetType -->
    <widgettype>filterdirective</widgettype>
    <!-- ID of the filter to connect with -->
    <filterID>2</filterID>
  </widget>
</xml>
```

Zeitbereichsfilter (dateSubfilterDirective)

Was ist ein Zeitbereichsfilter?

Der Zeitbereichsfilter ermöglicht es Nutzern Daten abzurufen und anzeigen zu lassen, die in einem gewählten Zeitbereich liegen.

Absolute vs. relative Zeitbereiche

Dabei haben Nutzer zwei Möglichkeiten, den Zeitbereich zu definieren: absolut und relativ.

Zeitbereich	Beispiel
Beschreibung	absolut
01.01.2018 - 30.01.2018	Es werden feste Daten angegeben, die als Zeitpunkt oder Zeitraum für die Datenabfrage und -anzeige dienen sollen. Der Zeitraum ist immer abgeschlossen und kann losgelöst vom aktuellen Datum der Datenabfrage vollständig in der Vergangenheit liegen. Ruft der Nutzer am 30.03. die Daten zwischen 01.01. und 30.01.2018 ab, so erhält er auch am 30.03. die Daten, die zwischen 01.01. und 30.01.2018 erzeugt/gespeichert wurden.
relativ	letzte 30 Tage, letzte 52 Wochen,...

Lokale Zeit vs. absolute Zeit

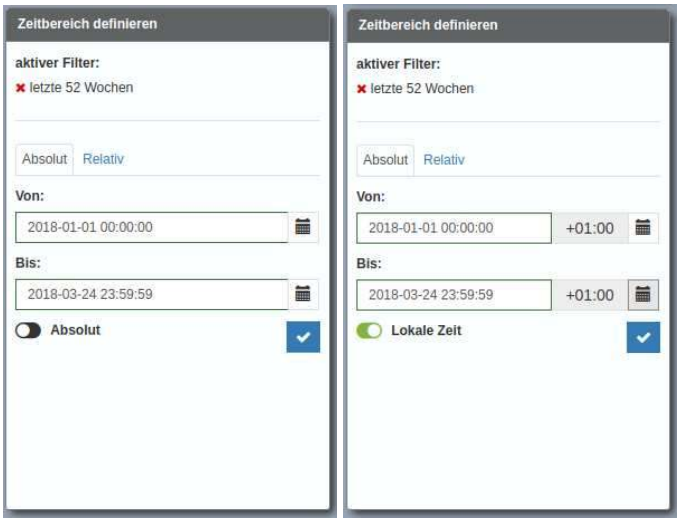
Über einen Umschalter kann zwischen lokaler und koordinierteR Weltzeit (utc) gewählt werden.

Beispiel:

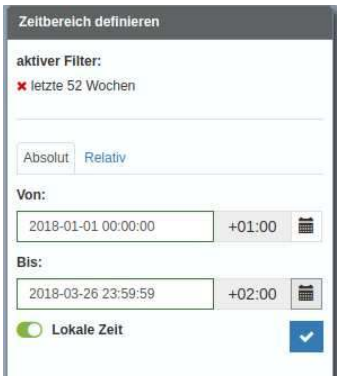
Ein Servicetechniker in Deutschland soll einen Anlagenausfall im Ausland prüfen. Der Kunde im Ausland (z.B. Japan) wird dem Servicetechniker mitteilen, dass in Japan morgens um 10:00 eine Anlage plötzlich nicht mehr richtig funktionierte. Beim Abruf der Sensordaten der Anlage muss der Servicetechniker nun darauf achten, die Daten um 10:00 lokaler Zeit in Japan zu betrachten, nicht nach lokaler Zeit in Deutschland.

Hierfür können der japanische Kunde und der deutsche Servicetechniker beide entweder die absolute, in der Datenbank abgespeicherte Zeit austauschen und abfragen, oder der Servicetechniker in Deutschland kann bei der lokalen Zeit auf die UTC-Zeitverschiebung des japanischen Kunden wechseln und anschließend den gewünschten Zeitbereich des Vorfalls auswählen.

Noch leichter (und weniger fehleranfällig) ist die Weitergabe des Portal-Links zu den relevanten Daten. Über die Deeplink-Funktionalität ist gewährleistet, dass dem Empfänger im Portal die gleichen Daten (inkl. sämtlicher Filterungen) angezeigt werden wie dem Absender.



Die Umstellung zwischen Sommer- und Winterzeit erfolgt automatisch.



Anlegen und definieren eines Zeitbereichsfilters

Struktur des Widgets

```
<xml>
  <widget name="template_dateSubfilter">
    <!-- Position from left top  -->
    <position>
        <x>0</x>
        <y>0</y>
    </position>
    <!-- Size of the Widget -->
```

```
    <size>
      <x>3</x>
      <y>6</y>
    </size>
    <!-- WidgetType -->
    <widgettype>datesubfilterdirective</widgettype>
    <!-- Caption -->
    <caption>
        <!-- Display caption -->
      <show>true</show>
        <!-- Title of the Widget -->
      <label>Zeitbereich definieren</label>
    </caption>
    <!-- Relative-Filter that shall be set initially -->
    <defaultRelativeFilter>lastDays_30</defaultRelativeFilter>
  </widget>
</xml>
```

Parameter

Parameter	Optionen	Default	Beschreibung
timezone	local, utc	utc	Definiert, welche Zeitzone beim initialen Laden der View im Widget genutzt werden soll
tztoggle	true, false	false	Definiert, ob der UTC-Umschalter Nutzbar sein soll oder nicht.
defaultRelativeFilter	currentDay, currentWeek, currentMonth, currentYear, lastDays_1, lastDays_7, lastDays_30, lastWeeks_13, lastWeeks_26, lastWeeks_52		Der angegebene Wert wird beim laden des Widgets gesetzt und schränkt die initial geladenen Daten ein.
mandatory	true, false	false	Ist der Wert "true", wird das Widget hervorgehoben, wenn keine Auswahl getroffen wurde.

Voreinstellung Lokal/ Absolut

Über den Parameter "timezone" kann dem jeweiligen Widget eine Voreinstellung für die Umrechnung der Zeitzone definiert werden. Für die Umrechnung der Zeit in die lokale Zeitzone kann hier 'local' als Werte vergeben werden. Ebenfalls kann auf 'utc' umgestellt werden, um keine Umrechnung der Zeit zu bewirken. 'utc' ist hier die Standardeinstellung, sollte der Parameter undefiniert bleiben.

Zeitzone umschalten

Die Umstellung von Lokal zu Absolut kann im Widget auch vom Portalnutzer direkt im Widget geschehen. Hierzu kann ein Umschalter, sichtbar unterhalb der Eingabefelder, eingestellt werden. Geben Sie hierzu dem Parameter "tzToggle" den wert *true*. Der Standardwert ist hier *false*, sollte der Parameter undefiniert bleiben.

Standard-Filter definieren

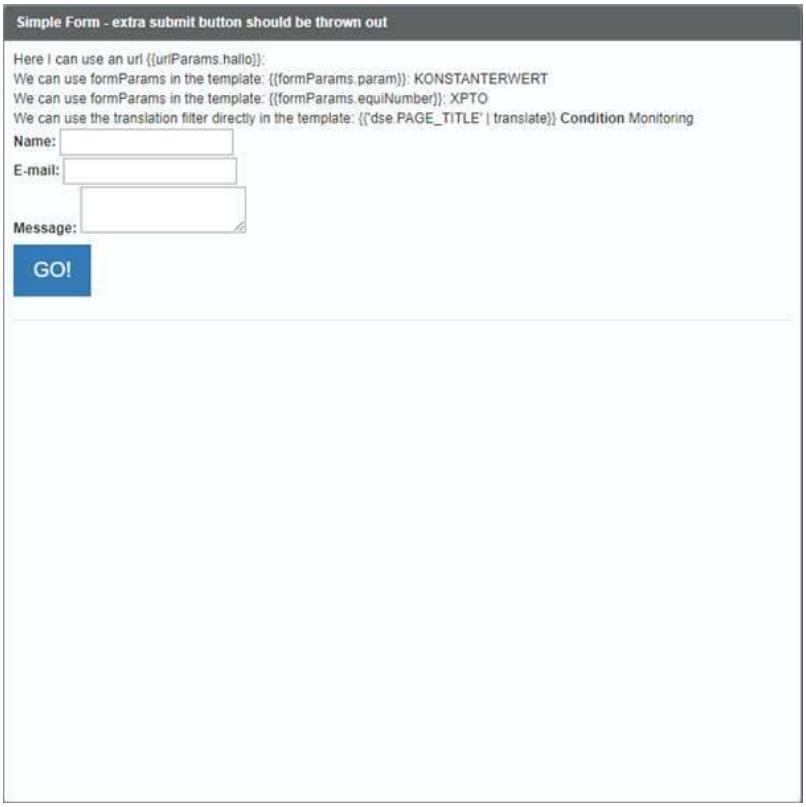
Über den Parameter 'defaultRelativeFilter' kann ein initialer Wert für die relative Filterung gesetzt werden. Wird zum Beispiel der Wert 'lastDays_30' übergeben, schränkt der Filter die dargestellten Daten auf die Daten der letzten 30 Tage ein und verhindert so, dass initial zu große Datenmengen geladen werden.

5.5.8. Formular-Widget

Mit dem Formular-Widget können Formulare zur Dateneingabe erstellt werden.

Die eingegebenen Daten werden beim Absenden als multipart/form-data an einen Endpunkt übergeben.

Beispiel: Ein Formular-Widget im YUNA-Portal



Form Parameter einbinden

Beispiel: Ein Form Parameter eingebunden in das Template

Im formTemplate können Parameter aus dem Tag <formParams> verwendet werden. Diese werden in doppelt geschweiften Klammern geschrieben.

```
<formTemplate>

    <div>We can use formParams in the template:
        \{\{formParams.equiNumber\}\}: {{formParams.equiNumber}}
    </div>

</formTemplate>
```

Beispiel: Auszug formParams tag

Im tag <name> steht der Name des Parameters, im tag <value> der Wert.

```
    <formParams>
        <list>
            <name>param</name>
            <value>KonstanterWert</value>
        </list>
        <list>
            <name>equiNumber</name>
            <value>XPTO</value>
        </list>
    </formParams>
```

Submit Button

Über einen Submit Button können die eingegebenen Daten an einen vordefinierten Endpunkt gesendet werden. Alternativ können die Daten auch über eine StoredProcedure in die DataDB geschrieben werden.

Innerhalb des Submit Buttons kann nur ein Ziel, entweder <endpoint> oder <dataID> verwendet werden. Beide Parameter sind nicht zulässig.

Beispiel: Submit Button mit Übergabe an einen Endpunkt

```
    <submitButton>
        <endpoint>http://www.example.com</endpoint>
        <label>
            <default>GO!</default>
            <tkey>content.MyFormWidget.Submit</tkey>
        </label>
    </submitButton>
```

Beispiel: Submit Button mit Übergabe an eine StoredProcedure

```
    <submitButton>
        <dataID>qy_sp_insertOrUpdatePrefilter</dataID>
        <label>
            <default>GO!</default>
            <tkey>content.MyFormWidget.Submit</tkey>
        </label>
    </submitButton>
```

YUNA XML Beispiel für ein Formular-Widget

```
<widget>
    <caption>
        <label>Simple Form - extra submit button should be thrown out</label>
        <show>true</show>
    </caption>
    <position>
        <x>0</x>
        <y>0</y>
    </position>
    <size>
        <x>7</x>
        <y>7</y>
    </size>
    <widgettype>formwidget</widgettype>
    <outputs>
        <submit>formData</submit>
    </outputs>
    <formTemplate>
<!-- In the form block we define OUR html form elements. We can reference form parameters using double brackets:
{{formParams.param}}. We can use translation keys: {{'dse.translationKey' | translation}} -->
<![CDATA[
<div>Here I can use an url \{\{urlParams.hallo\}\}:{{urlParams.hallo}}</div>
<div> We can use formParams in the template:
\{\{formParams.param\}\}: {{formParams.param | uppercase}}</div>
<div>We can use formParams in the template:
\{\{formParams.equiNumber\}\}: {{formParams.equiNumber}}
</div>
<div>We can use the translation filter directly in the template: \{\{'dse.PAGE_TITLE' | translate\}\} {{'dse.PAGE_TITLE'
| translate}}</div>
<div>
<label for="name">Name:</label>
<input type="text" id="name" name="user_name" value="{{urlParams.name}}">
</div>
<div>
<label for="mail">E-mail:</label>
<input type="email" id="mail" name="user_mail">
</div>
<div>
<label for="msg">Message:</label>
<textarea id="msg" name="user_message"></textarea>
</div>
<input type="submit" value="hello">
]]>
    </formTemplate>
    <!-- the values of the params will replace the given entries in the form CDATA -->
    <formParams>
        <list>
            <name>param</name>
```



```
            <value>KonstanterWert</value>
        </list>
        <list>
            <name>equiNumber</name>
            <value>XPTO</value>
        </list>
    </formParams>
    <submitButton>
        <endpoint>http://www.example.com</endpoint>
        <label>
            <default>GO!</default>
            <tkey>content.MyFormWidget.Submit</tkey>
        </label>
    </submitButton>
</widget>
```

YUNAML-Parameter

Parameter	mögliche Werte	Notwendig	Beschreibung	Beispiel
formTemplate	CDATA-Block	✅	HTML-Template für die Eingabefelder des Formulars ⚠️ Im Template können keine <form>- oder <button>-Elemente genutzt werden. Das umschließende <form>-Element und der Submit-Button werden aus der YUNAML-Definition ermittelt	`<![CDATA[ <label for="name">Name:</label> <input type="text" id="name" name="user_name" value="{{urlParams.name}}"/> <label for="mail">E-mail:</label> <input type="email" id="mail" name="user_mail"> <label for="msg">Message:</label> <textarea id="msg" name="user_message"></textarea> ]]>`
formParams	YUNAML-Listenelemente (<list>-tags) mit Eigenschaften 'name' und 'value'	❌	Parameter, die in der Template-Definition (<formTemplate>) referenziert werden können	`<formParams> <list> <name>param</name> <value> KonstanterWert </value> </list> </formParams>`
submitButton		✅	Definition des Bestätigen-Buttons	

Parameter	mögliche Werte	Notwendig	Beschreibung	Beispiel
submitButton.endpoint	URL		Ziel-Endpunkt, an den die Formulardaten geschickt werden sollen	http://www.example.com
submitButton.dataID	String		DataID, die mit den Formulardaten als Parameter aufgerufen werden soll. Die Formularfelder können unter den vergebenen Namen in der DataID als Parameter verwendet werden.	qy_exampleDataID
submitbutton.label	String		Label des Bestätigen-Buttons	OK

outputs.submit			Stellt nach klick auf den Submit Button Daten aus dem Formularwidget für den Outputchannel "submit" bereit.	`<outputs>`
				`  <submit>formData</submit>`
				`</outputs>`

URL Parameter

Name	Code	Beschreibung
urlParameter	`{{ urlParams.name }}`	Referenziert aktuell in der URL gesetzte Parameter
formParameter	`{{ formParams.name }}`	Referenziert Parameter aus der YUNAML-Definition (<formParams>)
inputchannel	`{{ inputs.channelname }}`	Referenziert Daten, die über einen Inputchannel bereitgestellt werden

Die Notation '\{\{ parameter }}' orientiert sich an AngularJS. Dadurch ist es möglich AngularJS-Filter einzusetzen, die einen eingegebenen Wert weiterverarbeiten.

Beispiele:

In der Template-Definition	Im dargestellten Widget
`{{ 'hallo welt!' | uppercase }}`	HALLO WELT!
`{{ formParams.userId | userDisplayFilter }}`	Nachname, Vorname (username)

Datasource

Über einen I/O Channel können sie Daten aus anderen Widgets in das Formular-Widget übertragen. Dafür muss in der YUNAML Definition des Formular-Widgets ein Input definiert werden. Die gewünschten Daten müssen daraufhin mit Feldern im Formular-Widget verbunden werden. Weitere Informationen über das Datasource-Konzept finden sie hier.

Beispiel:

Über den I/O Channel werden Zeilen aus einem Tabellen-Widget in das Formular-Widget überführt.

1. **Input channel definieren**

```
<widget>
    <inputs>
        <data>myrow</data>
    </inputs>
    <widgettype>formwidget</widgettype>
</widget>
```

2. **Ein Textfeld befüllen (gesamte Auswahl)**

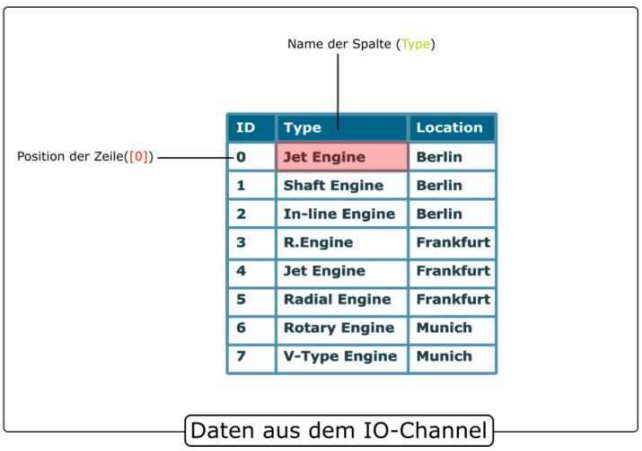
```
<formTemplate>
     <![CDATA[
        <div>
           Type of the Machine: {{ inputs.data }}
        </div>
     ]]>
</formTemplate>
```

Ein Textfeld befüllen (spezifischer Wert)

```
<formTemplate>
     <![CDATA[
        <div>
           Type of the Machine: {{ inputs.data[0].Type}}
        </div>
     ]]>
</formTemplate>
```

In diesem Beispiel wird der Wert "Jet Engine" aus der ersten Zeile aus der Spalte "Type" dargestellt:

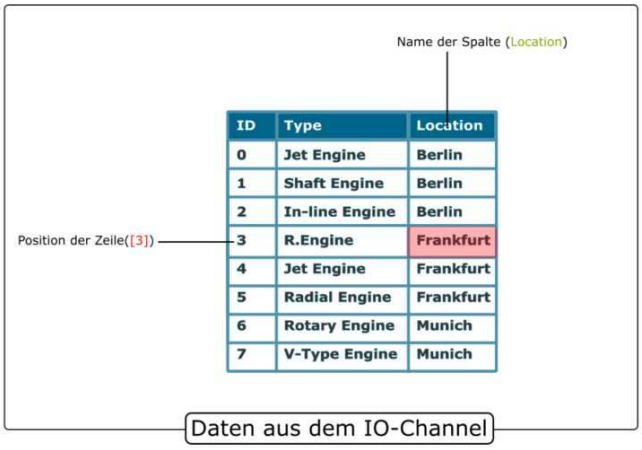
```
{{ inputs.data[0].Type }}
```



ID	Type	Location
0	Jet Engine	Berlin
1	Shaft Engine	Berlin
2	In-line Engine	Berlin
3	R.Engine	Frankfurt
4	Jet Engine	Frankfurt
5	Radial Engine	Frankfurt
6	Rotary Engine	Munich
7	V-Type Engine	Munich

Um den Wert "Frankfurt" aus der vierten Zeile aus der Spalte Location zu erhalten muss die Definition folgendermaßen aussehen:

```
{{ inputs.data[3].Location }}
```



ID	Type	Location
0	Jet Engine	Berlin
1	Shaft Engine	Berlin
2	In-line Engine	Berlin
3	R.Engine	Frankfurt
4	Jet Engine	Frankfurt
5	Radial Engine	Frankfurt
6	Rotary Engine	Munich
7	V-Type Engine	Munich

5.5.9. HTML-Widget (htmlwidget)

Neben den typischen Eigenschaften wie Größe, Position oder Titel können sie im HTML-Widget HTML Code, CSS und JavaScript verwenden. Der HTML Code wird in das Tag <template> geschrieben.

Beispiel: Darstellung eines HTML Widgets im YUNA Portal

Beispiel: Auszug Inhalt des tags <h1>

```
<template>
    <![CDATA[
        <h1>Headline 1</h1>
    ]]>
</template>
```

Im Template können Parameter aus dem Tag <htmlParams> verwendet werden. Diese werden in geschweiften Klammern geschrieben. Die HTML-Parameter können frei definierbarer Text, HTML, CSS oder JavaScript Code sein.

Sobald eine Data_ID angegeben wird, dient ein HTML-Parameter als Verweis auf eine Spalte in der Datenbank-Tabelle, deren Wert übernommen wird.

Beispiel: Auszug HTML Parameter

```
    <htmlParams>
      <list>
        <!-- Name is used to bind the value to the template -->
        <name>label</name>
        <!-- Value of the param -->
        <value>Equipment-Number</value>
      </list>
    </htmlParams>
```

Beispiel: Die HTML Parameter eingebunden in das tag <h2> im Template

```
<template>
  <!-- Accepts HTML. Dynamic values are set with {} (Values could be htmlParams or columns from the dataID) -->
  <![CDATA[<h2>{label} {EquipmentNo}</h2>]]>
</template>
```

Ein DATETIME wird erkannt und seine Formatierung über *format* festgelegt. Unbekannte Parameter bleiben als solche erhalten, NULL-Werte werden durch Leerstrings ersetzt.

Unabhängig von den Html-Parametern können mit der Präfix 'urlParam.' Url-Parameter ausgelesen werden. Ist in der aktuellen Url beispielsweise ein Parameter 'id' gesetzt (?id=1) kann dieser im Template mit \{urlParam.id} ausgelesen werden.

Skriptmodus

Über options.skriptModus kann der Skriptmodus aktiviert werden. Parameter werden in doppelt geschweiften Klammern geschrieben.

```
<options>
    <scriptMode>true</scriptMode>
</options>
```

Übersetzung von Inhalten

Mit Hilfe des Tags <TKEY> können Inhalte im HTML Widget übersetzt werden. Da bei der Übersetzung der komplette Inhalt des Eltern-Tags ersetzt wird, sollte um die Elemente <TKEY> und <DEFAULT> ein eigenes HTML-Tag gelegt werden.

```
<span>
  <TKEY>übersetzung.schlüssel</TKEY>
  <DEFAULT>Standardwert</DEFAULT>
<span>
```

Beispiel für ein HTML-Widget

```
<xml>
  <widget name="template_widget_HTML">
    <!-- Position from left top  -->
    <position>
        <x>0</x>
        <y>0</y>
    </position>
    <!-- Size of the Widget -->
    <size>
      <x>8</x>
      <y>2</y>
    </size>
    <!-- Caption -->
    <caption>
```

```
      <show>true</show>
      <label>HTML-Widget-Template</label>
    </caption>
    <!-- the WidgetType -->
    <widgettype>htmlwidget</widgettype>
    <!-- Query to execute -->
    <dataID>qy_NameOfDataID</dataID>
    <!-- Widget specific settings -->
    <htmlwidget>
      <!-- Template to dispay -->
      <template>
        <!-- Accepts HTML. Dynamic values are set with {} (Values could be htmlParams or columns from the dataID) -->
        <![CDATA[<h2>{label} {EquipmentNo}</h2>]]>
      </template>
      <!-- Array of HTML-Params -->
      <htmlParams>
        <list>
          <!-- Name is used to be bind the value to the template -->
          <name>label</name>
          <!-- Value of the param -->
          <value>Equipment-Number</value>
        </list>
      </htmlParams>
    <options>
        <scriptMode>true</scriptMode>
    </options>
    </htmlwidget>
  </widget>
</xml>
```

Feld	Mögliche Werte	Default	Beschreibung
widgettype	htmlwidget		Gibt den Widget-Typen an.
dataID	dataID aus der Datenbank		Gibt die Datenherkunft an.
triggerParams	triggerParameter aus der URL	optional	Wie bei anderen Widgets
htmlwidget	Objekt		Definiert die Eigenschaften des HTML-Widgets
template	Objekt		Das im Widget darzustellende HTML-Template.
htmlParams	Objekt		Auflistung der Parameter, die im Template dargestellt werden, sofern sie nicht automatisch durch die dataID gesetzt werden.
htmlParams.name	String		Name des Parameters.
htmlParams.value	String oder Zahl	optional	Wert der anstelle des Parameters im Template gesetzt wird.
htmlParams.format	Datumsformat	optional	siehe genDate. Weiterhin sind auch freie Formate möglich, z.B. "dd.MM.yyyy".
options	Objekt		Optionen für das Widget.
options.caseSensitive	true, false	false	Bei den Parametern wird auf Groß- und Kleinschreibung geachtet bzw. ignoriert.

Feld	Mögliche Werte	Default	Beschreibung
options.skriptModus	true, false	false	Schaltet den Skriptmodus ein oder aus. Im SkriptModus erfolgt die Paramemterersetzung so, dass auch in <script>-Tags geschweifte Klammern genutzt werden können.

Besonderheiten für Links im html-Widget

Wird im Template eine href-Adresse definiert, kann mit dem Parameter `tabExchange` beim Öffnen eines Links in einem neuen Tab der Filter des aktuellen Tabs übergeben werden. Zum Beispiel:

```
href='http://localhost:8080/conditionmonitoring/#/cmp_Messages?equi={equipmentNo}&\
{#tabExchange}
```

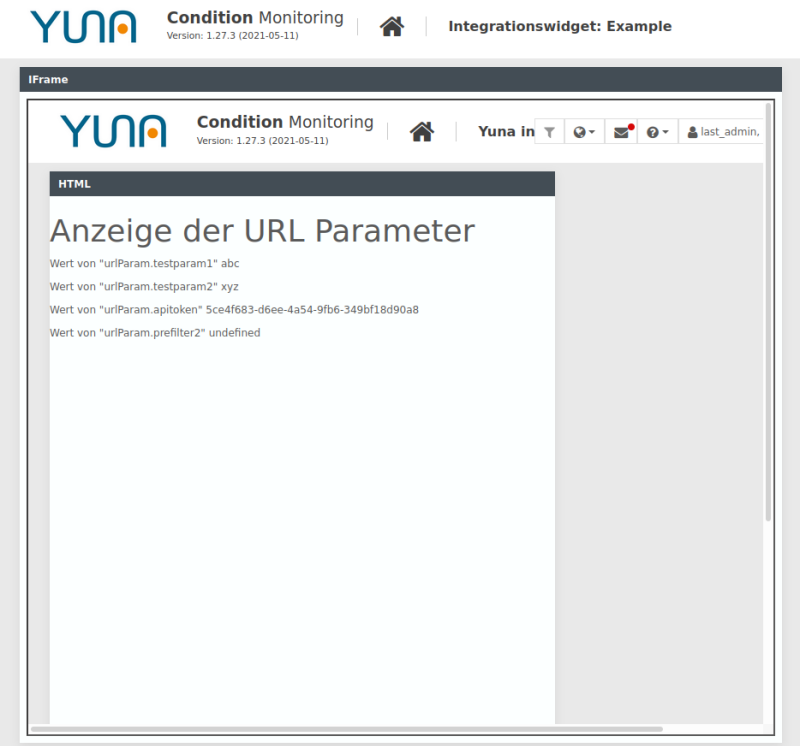
Verwendung von jQuery

Von der Verwendung von jQuery wird im allgemeinen abgeraten, da dies zu Fehlern in der Anwendung führen kann

Es ist möglich im HTML-Widget die JavaScript-Bibliothek jQuery zu nutzen. Diese steht unter dem Namen 'jQuery' zur Verfügung. Die Verwendung von '$' statt 'jQuery' kann zu Fehlern führen und wird nicht empfohlen.

5.5.10. Integration-Widget

Beispiel eines Integation-Widget anhand eines HTML-Widgets in YUNA



Mit dem Integration-Widget können externe Webseiten und Web-Anwendungen in YUNA eingebunden werden.Beispielsweise können dadurch R-Entwickler Shiny-Apps innerhalb eines YUNA-Dashboards verwenden.

YUNAML-Tags

<table>
<tr><th>YUNAML-Tag</th><th>Wert</th><th>Beschreibung</th><th>Default</th><th>Beispiel</th></tr>
<tr>
<td>appendParameters</td>
<td>false</td>
<td>
<table>
<tr><th>Wert</th><th>Erläuterung</th></tr>
<tr><td>false</td><td>Es werden keine Parameter automatisch an die URL angefügt.</td></tr>
<tr><td>system</td><td>An die URL werden automatisch die Parameter apitoken, language, filter<ID> und prefilter<ID> angefügt, Details nächste Tabelle

Beispiel:
https://www.yourHostAdress.xyz/?prefilter2=96bd5c2dc96a64135c1dd0dac6475ab5e5e833b1e33fa7aa9cdcba35b9a3896a,6b2739096042f02055f32cab50db8516dfe8b10836acb5ca6a0ce9a69ad01969 &apiToken=1efef5c5-dbfd-4dd2- a80d-6857322a592f&language=de_DE&filter2=2a0286d94d42d6634fd5592ea85b8005a03d66f7b967e99cd074269ee5a77eff"</td></tr>
</table>
</td>
<td>system</td>
<td><pre><appendParameters>
  false
</appendParameters></pre></td>
</tr>
</table>

Parameter	Bedeutung
apitoken	Token für die Anmeldung bei einer externen App als der aktuell angemeldeter Benutzer.Der Token ist nur für eine einmalige Anmeldung gültig.
language	ID der ausgewählten Sprache
filter<ID>	Die aktuell ausgewählten Filter
prefilter<ID>	Die aktuell gesetzten Vorfilter. Ein Vorfilter kann mehrere Hashes enthalten und wird somit als Liste übergeben.

Zieladresse des Integration-Widgets definieren

Die Zieladresse des Integration-Widgets kann im Tag <url> definiert werden.

Hier können auch URL-Parameter verwendet werden. URL-Parameter werden in doppelt geschweiften Klammern geschrieben.

Example 4. Beispiel für die Defintion der Widget-Adresse

```
<url>
  <![CDATA[
    /#/view_IFrameContent?testparam1=\{urlParams.testparam1}}&testparam2=\{urlParams.testparam2}}
  ]]>
</url>
```

Weiterreichen von Filtern

Einem Integration-Widget können Filter und Vorfilter aus YUNA übergeben werden. Diese können als URL-Parameter definiert werden.

Example 5. Beispiel für die Übergabe von Filtern

```
<url>http://www.example.com/examplepage?filter3=\{\{urlParams.filter3}}</url>
```

Example 6. Beispiel für die Übergabe von Vorfiltern

```
<url>http://www.example.com/examplepage?prefilter7=\{\{filters.prefilter7}}</url>
```

Shiny Authentifizierung

YUNA erzeugt anhand der Login-Daten des Portal-Nutzers einen API Token. Dieser kann über das Integration-Widget als URL-Parameter ausgegeben werden und von Shiny für die Authentifizierung verwendet werden.

Example 7. Beispiel für die Ausgabe eines API Tokens

```
<url>
  /#/view_IFrameContent?testparam1=\{\{urlParams.testparam1}}&amp;testparam2=\{\{urlParams.testparam2}}
&amp;apitoken=\{\{apiToken}}&amp;prefilter2=\{\{filters.prefilter2}}
</url>
```

Die Authentifizierung von Shiny am Yuna Portal erfolg über ein R Script unter Verwendung des R-Paketes dseconnect.

Example 8. Beispiel für den Verbindungsaufbau in R mittels API Token:

```
#dseconnect Paket laden
library(dseconnect)
#apiToken aus der URL holen
apiToken <- getparam("apitoken")
```

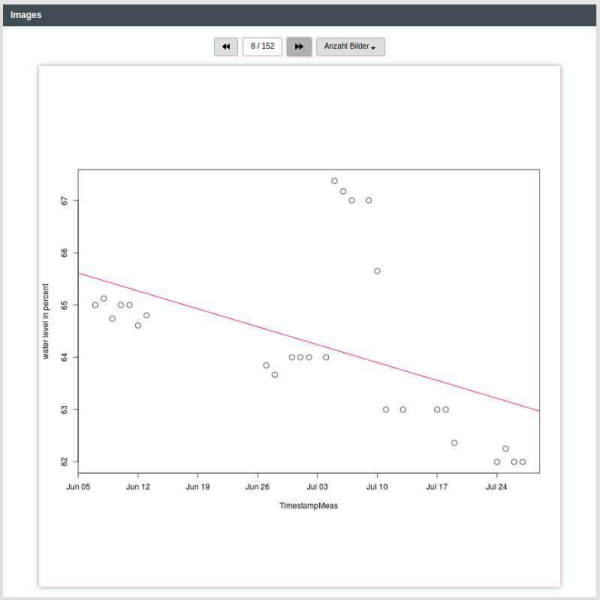
```
#mit dem apiToken am Portal anmelden
dseconnect::connectWithApiKey("http://localhost:8080/backend/" , apiToken)
```

Beispiel für ein Integration-Widget

```
<xml>
  <widget name="widget_IFramewidget">
    <caption>
      <show>true</show>
      <label>IFrame</label>
    </caption>
    <position>
      <x>0</x>
      <y>0</y>
    </position>
    <size>
      <x>14</x>
      <y>8</y>
    </size>
    <widgettype>iframewidget</widgettype>
    <url>
      <![CDATA[
        /#/view_IFrameContent?testparam1={urlParams.testparam1}}&testparam2={urlParams.testparam2}}
      ]]>
    </url>
    <appendParameters>system</appendParameters>
  </widget>
</xml>
```

5.5.11. Imageviewer

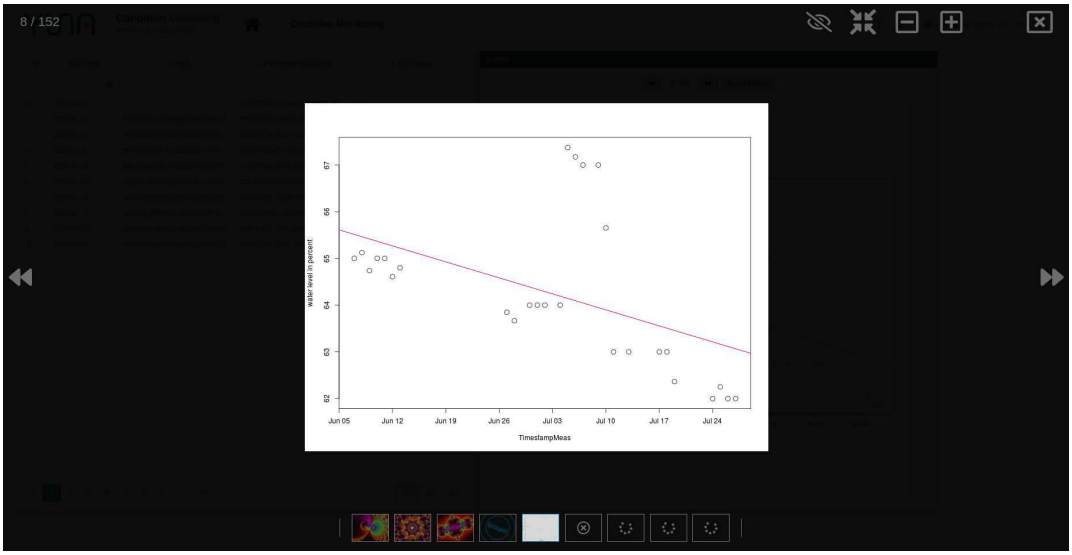
Das Imageviewer-Widget dient der Anzeige von Bildern. Mit dem Widget lassen sich Bilder anzeigen, die - beispielsweise mittels R-Skripten - in der Datenbank abgelegt wurden.



Funktionen im Widget:

- Durch Klick auf das aktuelle Bild, kann dieses im Vollbild-Modus geöffnet werden.
- Um direkt auf eine Seite zu springen, kann auf die aktuelle Seitenzahl geklickt werden
- Anzeige von x Bildern (z.B. 1, 2, 4, 9)
- Durchschalten von Bildern

Funktionen im Vollbildmodus



- In das aktuelle Bild kann über das Mausrad hinein und auch wieder herausgezoomt werden.
- Im gezoomten Zustand kann der Bildausschnitt durch Halten der linken Maustaste und Ziehen verschoben werden.
- Das Drücken der Leertaste ist die einfachste Möglichkeit Zoom und Bildausschnitt anschließend wieder zurückzusetzen.
- Liegt der Mauszeiger auf der Bildvorschau, kann über das Mausrad durch die Bilder geschaltet werden.
- Durch einen Doppelklick in den Bildbereich, werden die Bedienelemente ausgeblendet.

Hotkeys im Vollbild-Modus:

Key	Funktion
Escape	Schließt die Vollbildansicht
a, Pfeiltaste Links	Wählt das vorherige Bild aus
d, Pfeiltaste Rechts	Wählt das nächste Bild aus
w, +, Pfeiltaste Hoch	Hineinzoomen
s, - , Pfeiltaste Runter	Herauszoomen
Space	Zentriert das Bild und wechselt zwischen 1:1 und optimaler Skalierung

YUNA-ML Definition

Anlegen eines Imageviewer-Widgets im Portal

XML

```
<xml>
  <widget name="template_widget_Imageviewer">
    <!-- Position from left top  -->
    <position>
        <x>0</x>
        <y>0</y>
    </position>
    <!-- Size of the Widget -->
    <size>
      <x>6</x>
      <y>5</y>
    </size>
    <!-- Caption -->
    <caption>
      <show>true</show>
      <label>Imageviewer-Template</label>
    </caption>
    <!-- Name of the WidgetType -->
    <widgettype>Imageviewer</widgettype>
    <!-- Query to execute -->
    <dataID>qy_NameOfDataID</dataID>
    <!-- URL-Paramters to listen on for triggering the widget -->
    <triggerParams>
        <mandatory>
            <list>jiid</list>
            <list>equi</list>
        </mandatory>
    </triggerParams>
  </widget>
</xml>
```

Definierbare Angaben und Parameter

Parameter	mögliche Ausprägungen	Bedeutung
Allgemeine Optionen		
widgettype	Imageviewer	Definiert den Widget-Typ als Imageviewer
position	Zahlenwerte für die x- und y-Achse	Definiert die Position des Widgets im Grid
size	Zahlenwerte für die x- und y-Achse	Definiert die Größe des Widgets
dataID		Definiert die im Widget anzuzeigende Datenquelle
triggerparams (optinal / mandatory)		Definiert die Parameter, die für die Anzeige des Bilds im Imageviewer zuständig sind

Parameter	mögliche Ausprägungen	Bedeutung
imagecount		Definiert die Anzahl der gleichzeitig im Widget angezeigten Bilder
Caption-Optionen		
show	true, false	Definiert, ob eine Caption angezeigt wird oder nicht.
label	Freitext	Definiert die Überschrift des Widgets
menu	show: true, false label: Freitext icon: fontawesome-icon tooltip: Freitext type: popup	Definiert Einträge des Caption-Menüs
Appearance-Optionen		
enlargeableY	true, false	Definiert, ob das Widget ein- und ausgeklappt werden kann.
enlargedY	true false	Definiert, ob das Widget beim Aufruf der View aus- oder eingeklappt ist.

Datasource

Über die Datasource (siehe Datasource) können Bilder aus einem IO-Channel in den Imageviewer geladen werden. Dafür muss ein ein Input-Channel in YunaML definiert werden.

Der 'hash'-Input-Channel erlaubt es, Bilder aus der Tabelle FileStreamDataStorage anhand ihres Hashs zu laden.

Beispiel: Hash-Input-Channel am Imageviewer

```
    <inputs>
        <hash>tableData</hash>
    </inputs>
```

Über den 'src'-Input-Channel können URLs oder Data-URIs übergeben werden, aus denen die Bilder geladen werden sollen.

Beispiel: 'src'-Input-Channel am Imageviewer

```
    <inputs>
        <src>tableData</src>
    </inputs>
```

YUNAML View mit Tabellenwidget und Imageviewer

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

```
<xml>
    <view name="dse_ImageviewerDemo" roles="System_Admin, AdHoc_Full_Issue">
        <widget name="IV_table_default">
            <position>
                <x>0</x>
                <y>0</y>
            </position>
            <size>
                <x>8</x>
                <y>8</y>
            </size>
            <widgettype>tabledirective</widgettype>
            <dataID>qy_images</dataID>
            <cols>
                <list>
                    <field>ID</field>
                    <width>50</width>
                    <filter><ID>number</ID></filter>
                </list>
                <list>
                    <field>DateTime</field>
                    <type>genDate</type>
                    <filter><DateTime>date-custom</DateTime></filter>
                    <format>short</format>
                    <width>120</width>
                </list>
            </cols>
            <outputs>
                <current>tableData</current>
            </outputs>
            <generalOptions>
                <addColumns>true</addColumns>
                <skipColumns><list>FileStreamData</list><skipColumns>
                <sortable>true</sortable>
                <filter>true</filter>
            </generalOptions>
        </widget>
        <widget name="IV_default">
            <position>
                <x>8</x>
                <y>0</y>
            </position>
            <size>
                <x>8</x>
                <y>8</y>
            </size>
            <caption>
                <show>true</show>
                <label>Images</label>
            </caption>
```

```
            <inputs>
                <hash>tableData</hash>
            </inputs>
            <widgettype>Imageviewer</widgettype>
        </widget>
    </view>
</xml>
```

Query:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xml>
  <data name="qy_images" roles="System_Admin">
    <friendlyName>qy_images</friendlyName>
    <type>QueryBuilder</type>
    <filter>false</filter>
    <path>DSE-PortalDB portal FileStreamDataStorage</path>
    <query>
      <![CDATA[
        <QueryBuilder>
            <select>
                <table>DSE-PortalDB</table>
                <table>portal</table>
                <table>FileStreamDataStorage</table>
                <as>A</as>
                <fields>
                <field><name>ID</name></field>
                <field><name>Hash</name></field>
                <field><name>FileStreamDataGUID</name></field>
                <field><name>DateTime</name></field>
                <field><name>Extension</name></field>
                </fields>
                <where/>
                <groupby/>
                <orderby/>
                <limit>0</limit>
                <limitoffset>0</limitoffset>
            </select>
            <dictionary/>
        </QueryBuilder>
      ]]>
    </query>
  </data>
</xml>
```

Beispiel: Verknüpfung ImageViewer und Tabelle mit synchronisierten Indeces (YUNALang)

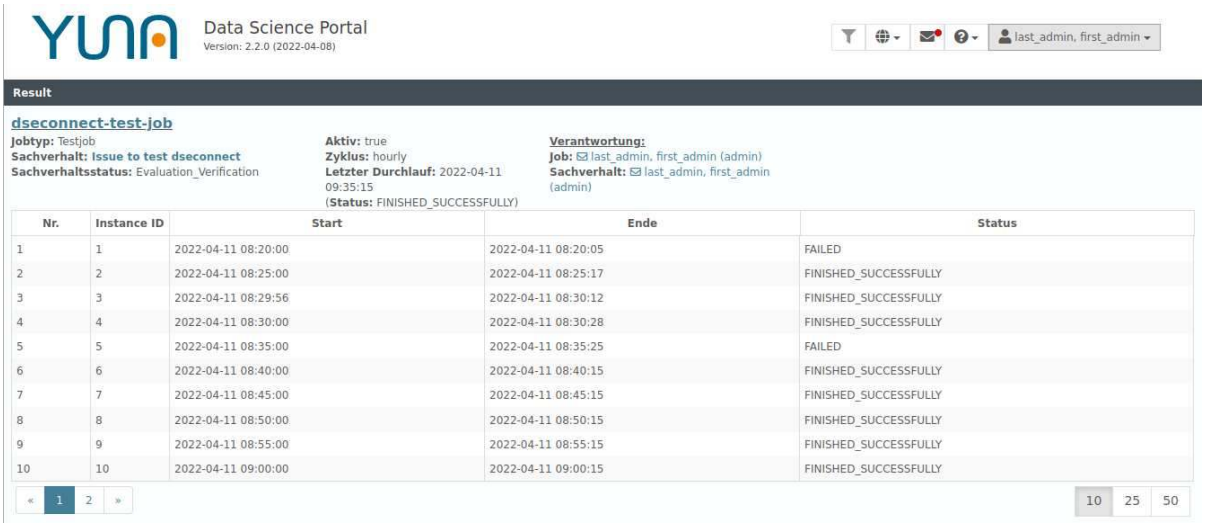
```
dashboard imageviewerDemo for System_Admin {
```

```
    title: "imageviewerDemo"
    widgets: [
        tableWidget {
            position: (0, 0)
            size: (6, 3)
            caption: "Table Widget"
            allowSearch: true
            allowChartAnalysis: true
            allowFullscreen: true
            allowExport: true
            inputs: [
                selectionIndex <- imageViewerSelection,
                data <- data
            ]
            outputs: [
                selectedIndex -> tableSelection,
                current -> tableCurrent
            ]
            linesSelectable: single
            columns: [
                text {
                    field: "Hash"
                    sortable: true
                },
                number {
                    field: "ID"
                    sortable: true
                }
            ]
        },
        imageWidget {
            position: (6, 0)
            size: (4, 6)
            caption: "Imageviewer Widget"
            inputs: [
                current <- tableSelection,
                hash <- tableCurrent
            ]
            outputs: [
                current -> imageViewerSelection
            ]
        }
    ]
    ioProvider: [
        dataIDProvider &provider
    ]

}

Database DSE_DataDB = `DSE-DataDB`
```

```
Database DSE_PortalDB = `DSE-PortalDB`

IOProvider provider = dataIDProvider {
    dataID: qy_allImages
    outputs: [
        output -> data
    ]
}

dataquery qy_allImages for System_Admin  {
    query: {
        SELECT Hash, ID
        FROM &DSE_PortalDB.portal.FileStreamDataStorage
        LIMIT 1000
    }
}
```

5.5.12. JobResult (dseJobResult)

Mit dem JobResult-Widget können Sie die Ergebnisse ihrer Jobausführungen anzeigen. Angezeigt werden Instanz ID, Start, Ende und Status des Jobs.



YUNA ML Beispiel

```
    <xml>
        <widget>
            <position>
                <x>3</x>
                <y>0</y>
            </position>
            <size>
                <x>11</x>
                <y>7.5</y>
            </size>
            <caption>
```

```
            <show>true</show>
            <label>Job-Ergebnisse</label>
        </caption>
        <widgettype>dseJobResult</widgettype>
        <urlParams>
            <jobid>myJobId</jobid>
        </urlParams>
    </widget>
</xml>
```

URL Parameter

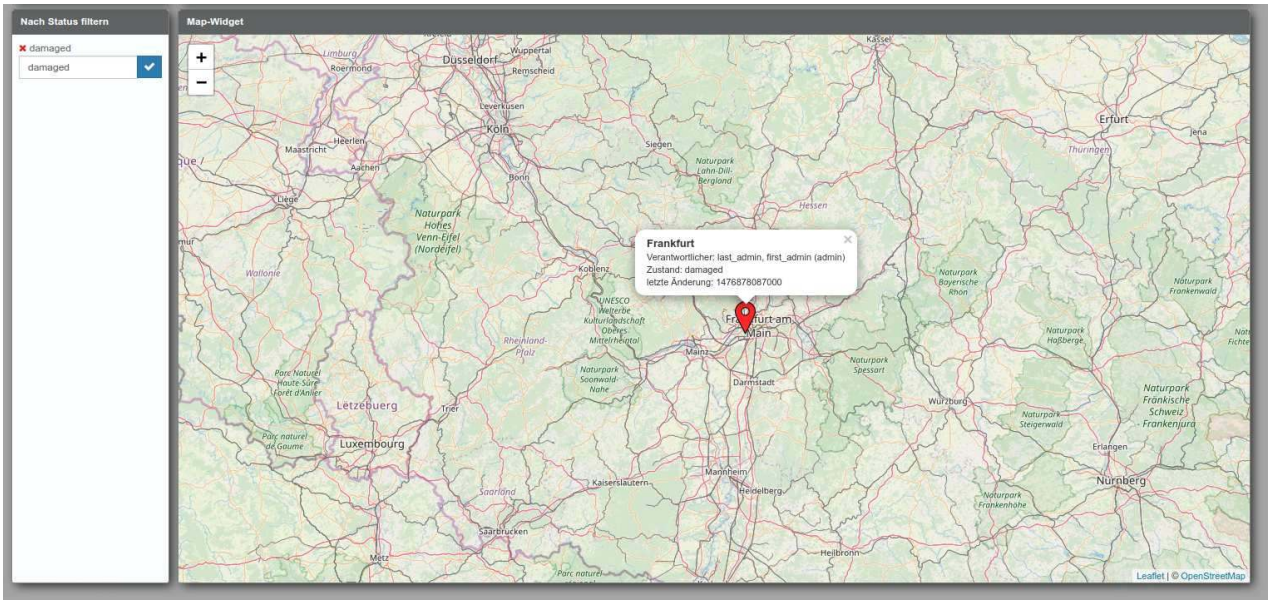
Im Element <urlParams> können die im JobResult-Widget verwendeten URL Parameter eingestellt werden.

5.5.13. Kartenwidget

Das Kartenwidget (mapWidget) dient der Visualisierung von geographischen Daten.

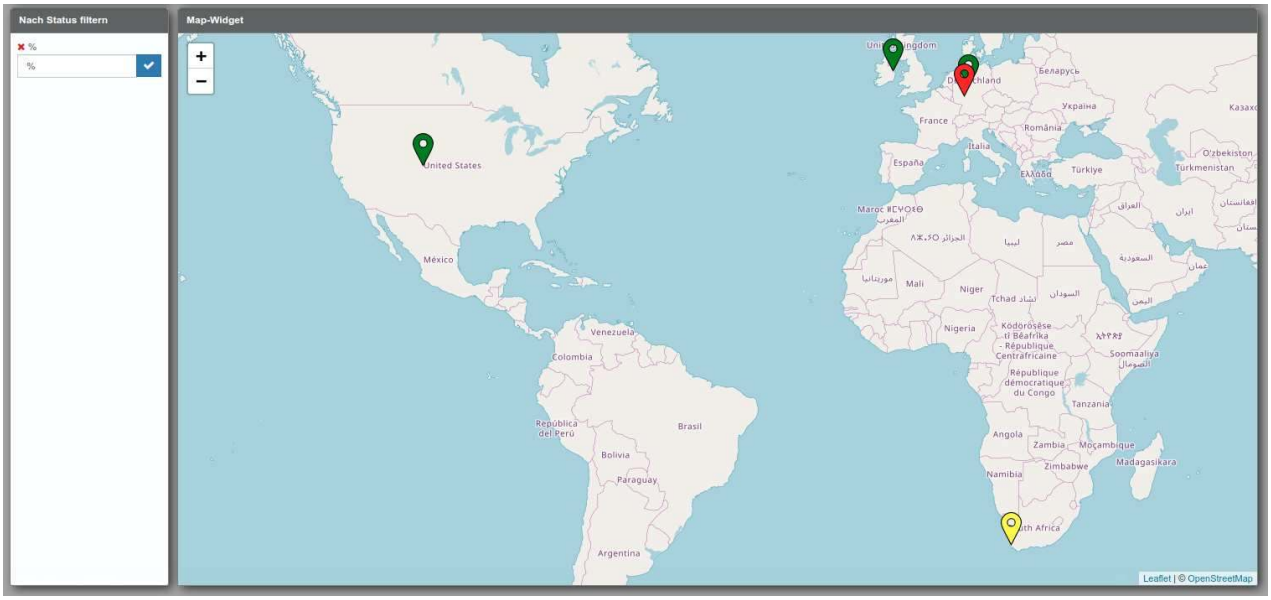
Damit können z.B. Maschinenzustände oder Warenbestände an verschiedenen Standorten visualisiert werden.Hierfür werden die Daten aus der per DataID definierten Query bezogen und als Koordinaten interpretiert, die ZWINGEND die Attribute **long** und **lat** (für longitude und latitude) enthalten müssen. Für jedes Element dieses Datensatzes wird ein Marker auf der Karte erstellt. Optional werden die Marker über das tag <color> im tag <markeroptions> eingefärbt. Des weiteren wird optional ein Popup eingebunden, welches sich bei einem Klick auf den Marker öffnet. Im tag <popup> können Titel und Label definiert werden. Das tag <field> referenziert auf eine Spalte aus der DataID. Die möglichen Typen im Element <type> richten sich nach den Spaltentypen (siehe: Spalten-Typen).

Beispiel: Popup zu einem Standort mit den Attributen popupTitle und popupBody



Über ein Einzelselektions-Widget auf der linken Seite werden Maschinen mit dem Status "damaged" angezeigt

Beispiel: Visualisierung verschiedener Standorte und Status



Datenstruktur:

Attribute	Type	Notwendig
long	float	mandatory
lat	float	mandatory

Beispiel:

	id	long	lat	status	name	status_message
1	1	22	23	1	SDF2345	ok
2	2	24	25	4	SA23456	critical

YUNA XML * **Beispiel für ein Map Widget**

```
<xml>
  <widget name="MapDemo_map">
    <widgettype>mapWidget</widgettype>
    <caption>
      <show>true</show>
      <label>Map-Widget</label>
    </caption>
    <triggerParams>
        <optional>
            <list>status</list>
        </optional>
    </triggerParams>
    <dataID>qy_MapDemo_map</dataID>
    <loadOnInit>true</loadOnInit>
    <mapOptions>
      <markerOptions>
```

```
      <!-- popup default definitions-->
        <popup>
          <title>Name</title>
          <body>
            <list>
              <label>Verantwortlicher</label>
              <field>User</field>
              <type>user</type>
            </list>
            <list>
              <label>Zustand</label>
              <field>Status</field>
            </list>
            <list>
              <label>letzte Änderung</label>
              <field>ChangedAt</field>
              <type>genDate</type>
              <format>short</format>
            </list>
          </body>
        </popup>
        <color>red</color>
        <!-- popup definitions for different conditions-->
        <conditions>
          <list>
            <value>
              <list>ok</list>
            </value>
            <field>Status</field>
            <color>green</color>
            <popup>
              <title>Name</title>
            </popup>
          </list>
          <list>
            <value>
              <list>unavailable</list>
            </value>
            <field>Status</field>
            <color>yellow</color>
          </list>
        </conditions>
      </markerOptions>
      <!-- enables or disables the fly-to-marker animation  -->
      <fitMarkers>true</fitMarkers>
    </mapOptions>
  </widget>
</xml>
```

Standard Einstellungen für Popups:

Innerhalb des <MarkerOptions> Elements können Standard-Einstellungen für Marker und Popup's vorgenommen werden.

<popup> Element

Hier kann das Popup gestaltet werden, dass beim Klicken auf einen Marker erscheint.

<color> Element

Hier wird die Farbe des Markers definiert.

<conditions>

Hier können die Farbe eines Markers und die Gestaltung des Popup's für unterschiedliche Zustände festgelegt werden.

Die unter <conditions> eingetragenen Definitionen für Marker und Popup's für einen Zustand werden anstelle der Standard-Einstellungen angewendet.

Beispiel für die Definition der Darstellung unterschiedlicher Zustände:

```
      <conditions>
        <list>
          <value>
            <list>ok</list>
          </value>
          <field>Status</field>
          <color>green</color>
          <popup>
            <title>Name</title>
          </popup>
        </list>
        <list>
          <value>
            <list>unavailable</list>
          </value>
          <field>Status</field>
          <color>yellow</color>
        </list>
      </conditions>
```

Beispiel: Map Widget Data-ID

```
<xml>
    <data name="qy_MapDemo_map" roles="System_Admin, AdHoc_Full_Issue">
        <!-- Setting the dataID -->
        <friendlyName>qy_MapDemo_map</friendlyName>
        <!-- StoredProcedure/QueryBuilder -->
        <type>QueryBuilder</type>
```

```
        <!-- Use filter on this query -->
        <filter>false</filter>
        <!-- Optional Path: Database Scheme Table -->
        <path/>
        <!-- Data-Query built with the QueryBilder (also see "QueryBuilder" in the Content Developer Guide) -->
        <query>
            <![CDATA[
                <QueryBuilder>
                    <select>
                        <table>YUNA-DataDB</table>
                        <table>data</table>
                        <table>mapData</table>
                        <fields>
                            <field>
                                    <name>Longitude</name>
                                    <as>long</as>
                            </field>
                            <field>
                                    <name>Latitude</name>
                                    <as>lat</as>
                            </field>
                            <field>
                                    <name>Status</name>
                            </field>
                            <field>
                                    <name>ChangedAt</name>
                            </field>
                            <field>
                                    <name>Name</name>
                            </field>
                            <field>
                                    <name>User</name>
                            </field>
                        </fields>
                        <where>
                            <like>
                                <field>
                                    <ID>status</ID>
                                    <name>Status</name>
                                </field>
                            </like>
                        </where>
                        <orderby/>
                        <limit>0</limit>
                        <limitoffset>0</limitoffset>
                    </select>
                </QueryBuilder>
            ]]>
        </query>
        <meta>
            <fields>
                <field>
                    <name>User</name>
                    <type>
                        <name>User</name>
                    </type>
                </field>
                <field>
                    <name>ChangedAt</name>
                    <type>
                        <name>ZonedTimestamp</name>
```

```
                        <params>
                            <param>
                                <name>timezone</name>
                                <value>UTC</value>
                            </param>
                            <param>
                                <name>absolute</name>
                                <value>true</value>
                            </param>
                        </params>
                    </type>
                </field>
            </fields>
        </meta>
    </data>
</xml>
```

Kachelserver konfigurieren (Beispiel)

Der Kachelserver lässt sich sowohl global in der PortalDB als auch lokal in der Widget-Definition konfigurieren.

Globale Konfiguration in der PortalDB

Die Konfiguration kann im Config Store in der PortalDB vorgenommen werden.

Key	Value	Bei installation auf einer Subdomain
map.tileLayer.url	https://Kachelserveradresse/{z}/{x}/{y}.png	https://{s}.Kachelserveradresse/{z}/{x}/{y}.png

Weiterführende Informationen zur Konfiguration erfahren Sie unter: https://leafletjs.com/reference-1.5.0.html.

Kachelserveradresse

Kachelserveradresse entspricht der Adresse Ihres Kachelservers

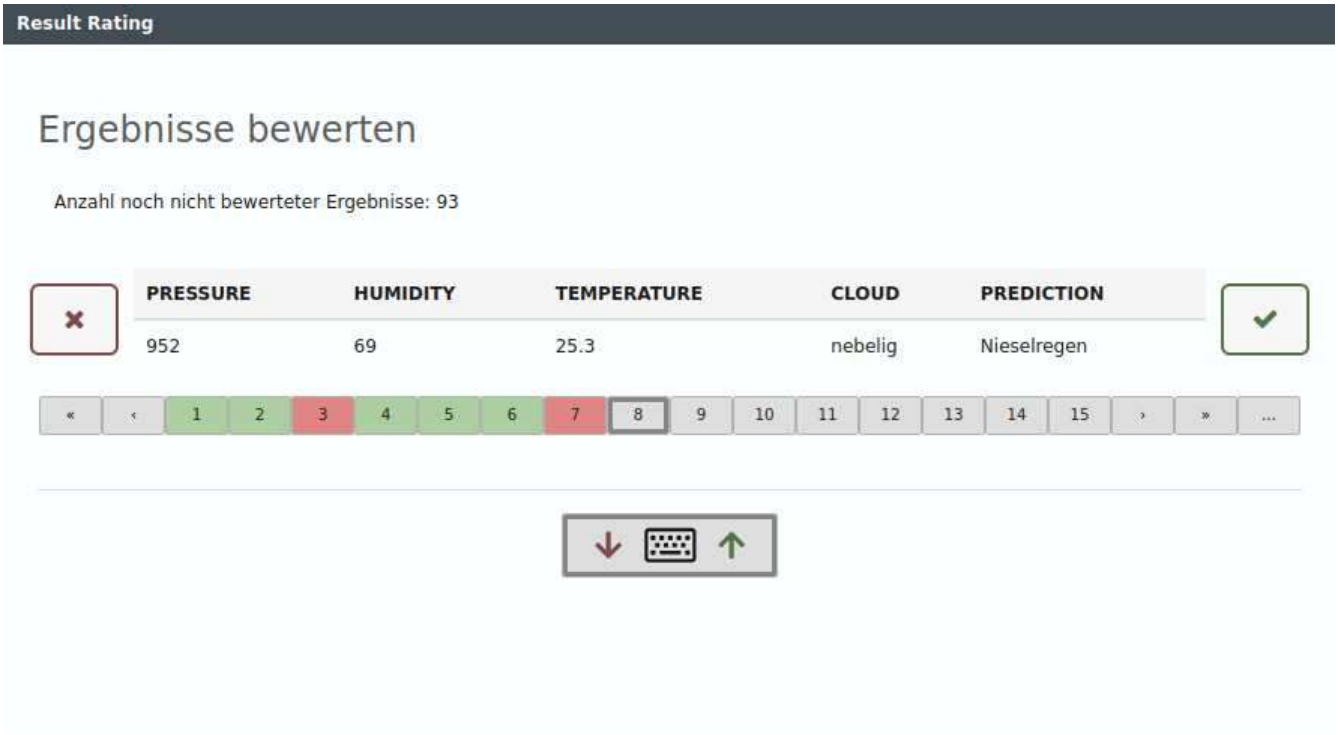
Lokale Konfiguration in der Widget-Definition

In der Widget Definition kann im tag <tileLayer> innerhalb des tags <mapOptions> der Kachelserver konfiguriert werden.

Beispiel:

```
<mapOptions>
    <tileLayer>https://{s}.Kachelserveradresse/{z}/{x}/{y}.png</tileLayer>
</mapOptions>
```

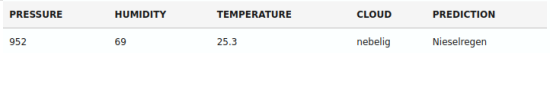
5.5.14. Result-Rating-Widget (resultrating)



Mit dem Result Rating können Fachanwender Ergebnisse von Jobausführungen manuell bewerten. Die Bewertung kann wiederum von Data Scientisten für die Verbesserung der Algorithmen in den jeweiligen Skripten genutzt werden.

Außerdem lassen sich durch das Result Rating wertvolle Erkenntnisse für den Aufbau von Trainingsdatensätzen für ein Machine-Learning Verfahren ableiten.

Aufbau

Element	Bild	Beschreibung
Ergebnis		Das ausgewählte Ergebnis, das vom Anwender bewertet werden kann.
Bewertungs-Leiste		Nummerierte Auflistung aller Ergebnisse, die bewertet werden können. Das aktuell ausgewählte *Ergebnis* wird darüber dargestellt. Bereits bewertete Ergebnisse werden entweder grün oder rot gefärbt, je nachdem ob es positiv oder negativ ist.
Tastenkürzel aktivieren		Mit dem Tastatur-Button können die Tastenkürzel aktiviert werden.Mit den Pfeiltasten links und rechts kann entlang der *Ergebnis-Leiste* zwischen den Ergebnissen hin und her geschaltet werden.Mit den Pfeiltasten hoch und runter kann das Ergebnis positiv oder negativ bewertet werden.

Element	Bild	Beschreibung
Bewertungs-Schalter		Mit dem Haken-Button kann das Ergebnis als positiv bewertet werden.Mit dem X-Button kann das Ergebnis als negativ bewertet werden. Sobald ein Ergebnis bewertet wurde, wird automatisch das nächste unbewertete Ergebnis ausgewählt.
		Die Bewertung eines Ergebnisses kann durch ein erneutes Klicken auf den Haken- bzw. X-Button rückgängig gemacht und somit in den Status "unbewertet" zurückgesetzt werden.
Anzahl unbewertete Ergebnisse		Hier wird angezeigt, wie viele unbewertete Ergebnisse noch vorliegen. Gibt es kein unbewertetes Ergebnis, wird dies im oberen Bereich des Widgets dargestellt.

Konfiguration des Result-Rating-Widgets (widgettype: resultrating)

YUNAML-Tag	Beschreibung	Notwendig	Beispiel
inputs	Konfiguration der InputChannel.		
inputs > **data**	Konfiguration des InputChannels für den Dateneingang.	✓	`<inputs> <data> DataInputChannel</data> </inputs>`
inputs > **selection**	Konfiguration des InputChannels für die aktuelle Selektion.	✗	`<inputs> <selection> SelectionInputChannel </selection> </inputs>`
outputs	Konfiguration der OutputChannel.		
outputs > **selected**	Konfiguration des OutputChannels für die aktuelle Selektion.	✗	`<outputs> <selection> SelectionOutputChannel </selection> </outputs>`

YUNAML Minimal-Beispiel für ein Result Rating Widget

```
<xml>
```

```
    <view name="result-rating-minimal-example" roles="System_Admin">
        <caption>Minimal Result-Rating Example</caption>
        <description>Minimal Result-Rating Example</description>
        <io-provider>
            <type>DataId</type>
            <config>
                <dataId>qy_dataToBeAssessed</dataId>
                <output>dataIdChannel</output>
            </config>
        </io-provider>
        <io-provider>
            <type>ResultRating</type>
            <config>
                <input>dataIdChannel</input>
                <output>ratedDataChannel</output>
                <rowIdentifierTemplate>resultId:{{ID}}</rowIdentifierTemplate>
                <queryIdentifier>minimalExampleQueryIdentifier</queryIdentifier>
            </config>
        </io-provider>
        <widget>
            <widgettype>resultrating</widgettype>
            <inputs>
                <data>ratedDataChannel</data>
            </inputs>
            <position>
                <x>0</x>
                <y>0</y>
            </position>
            <size>
                <x>6</x>
                <y>4</y>
            </size>
        </widget>
    </view>
</xml>
```

YUNAML Beispiel für ein Result-Rating-Widget mit verknüpftem Tabellen-Widget

```
<xml>
    <view name="result-rating-with-table-widget-example" roles="System_Admin">
        <caption>Result-Rating with Table-Widget Example</caption>
        <description>Result-Rating with Table-Widget Example</description>
        <io-provider>
            <type>DataId</type>
            <config>
                <dataId>qy_dataToBeAssessed</dataId>
                <output>dataIdChannel</output>
            </config>
        </io-provider>
```

```
        <io-provider>
            <type>ResultRating</type>
            <config>
                <input>dataIdChannel</input>
                <output>ratedDataChannel</output>
                <rowIdentifierTemplate>resultId:{{ID}}</rowIdentifierTemplate>
                <queryIdentifier>minimalExampleQueryIdentifier</queryIdentifier>
            </config>
        </io-provider>
        <widget>
            <widgettype>tabledirective</widgettype>
            <inputs>
                <data>ratedDataChannel</data>
                <selection>selectedRowChannel2</selection>
            </inputs>
            <outputs>
                <current>sortedAndFilteredDataChannel</current>
                <selected>selectedRowChannel1</selected>
            </outputs>
            <generalOptions>
                <addColumns>true</addColumns>
                <selectable>single</selectable>
            </generalOptions>
            <position>
                <x>0</x>
                <y>0</y>
            </position>
            <size>
                <x>11</x>
                <y>6</y>
            </size>
        </widget>
        <widget>
            <widgettype>resultrating</widgettype>
            <inputs>
                <data>sortedAndFilteredDataChannel</data>
                <selection>selectedRowChannel1</selection>
            </inputs>
            <outputs>
                <selected>selectedRowChannel2</selected>
            </outputs>
            <position>
                <x>0</x>
                <y>0</y>
            </position>
            <size>
                <x>6</x>
                <y>4</y>
            </size>
        </widget>
```

```
    </view>
</xml>
```

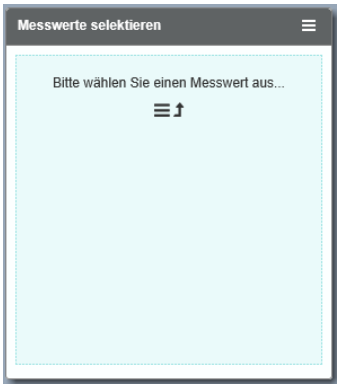
5.5.15. Sensorliste (sensorlistDirective)

Was ist die Sensorliste?

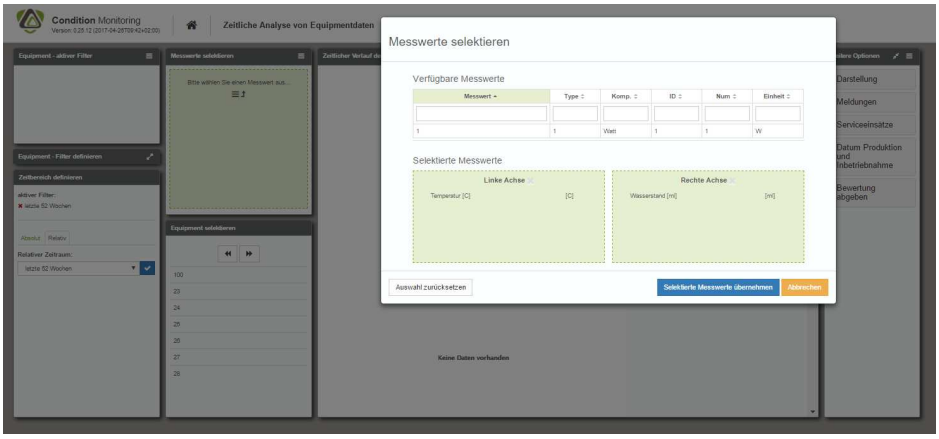
Die Sensorliste ist ein Widget-Typ über den Benutzer die Messwerte auswählen können, die im Stockchart dargestellt werden sollen. In der Sensorliste lassen sich vorhandene Messwerte der X- oder Y-Achse des Stockcharts zuweisen, wieder entfernen oder austauschen.

Nutzung der Sensorliste

Im Ausgangszustand ist die Sensorlist lediglich ein farblich hinterlegtes Quadrat mit Titel in der kopfzeile und dem Hinweis, der Benutzer solle bitte Messwerte auswählen.

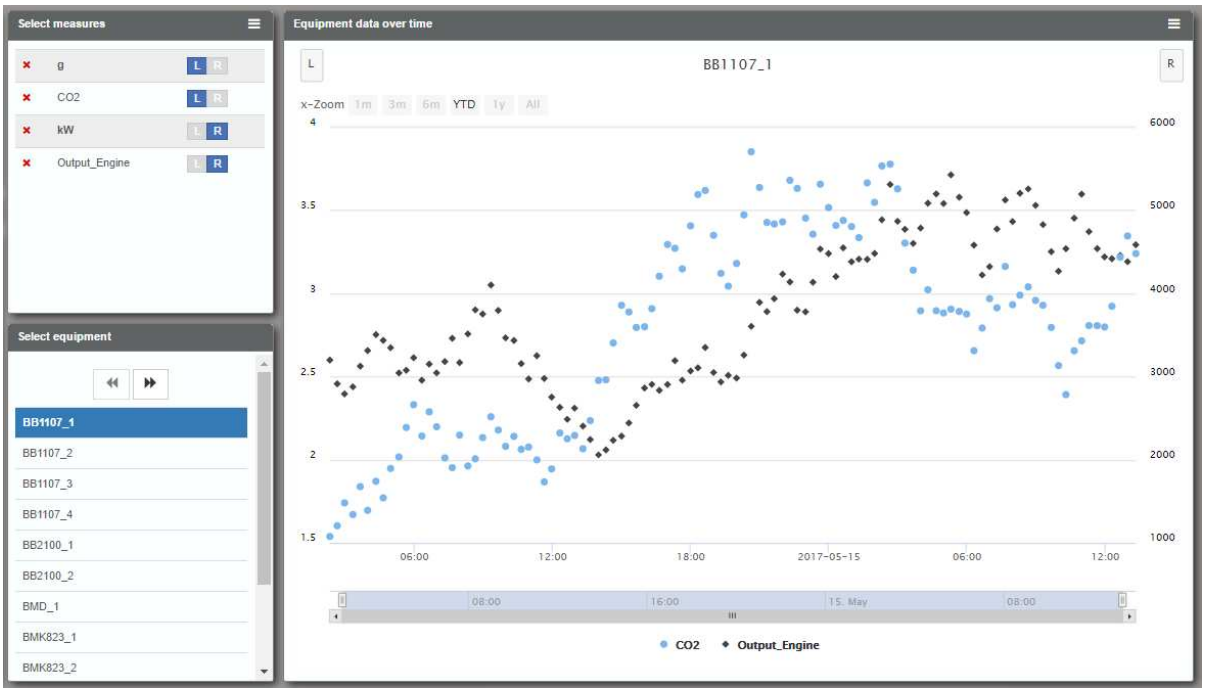


Durch einen Klick auf das Widget öffnet sich die Oberfläche zur Auswahl der Messwerte. Dort können alle im Portal angelegten Sensor-Messwerte durchsucht, sortiert, ausgewählt und den Diagrammachsen zugeteilt werden. Auch ein verschieben zwischen den Achsen ist durch einen Klick auf ">>" bzw. "<<" direkt neben dem jeweiligen Messwert möglich.



Nachdem die Messwerte ausgewählt und den Achsen zugeordnet wurden, werden die ausgewählten Messwerte inkl. Umschalter für die Achsenbelegung im ursprünglichen Widgetfeld in der Portal View angezeigt. Durch die Auswahl eines Equipments im darunterliegenden

Einzelselektion-Widget werden die Datenpunkte geladen und im Stockchart dargestellt.



Anlegen einer Sensorliste

XML

```
<xml>
  <widget name="template_widget_Sensorlist">
    <!-- Position from left top  -->
    <position>
        <!-- y -->
        <y>0</y>
        <!-- x -->
        <x>0</x>
    </position>
    <!-- Size of the Widget -->
    <size>
      <x>3</x>
      <y>3.4</y>
    </size>
    <!-- Caption -->
    <caption>
      <show>true</show>
      <label>Sensorlist-Template</label>
    </caption>
    <!-- Name of the WidgetType -->
    <widgettype>sensorlistdirective</widgettype>
    <!-- Query to execute -->
    <dataID>qy_NameOfDataID</dataID>
    <!-- URL-Paramters to listen on for triggering the widget -->
    <triggerParams>
      <list>filter2</list>
```

```
    </triggerParams>
    <!-- URL-Parameter to be set by the widget -->
    <urlParam>sensor</urlParam>
  </widget>
</xml>
```

Definierbare Parameter und Optionen

Parameter	Ausprägung	Bedeutung
Allgemeine Optionen		
widgettype	sensorlistdirective	Definiert den Widget-Typ
position	Zahlenwerte für die x- und y-Achse	Definiert die Position des Widgets im Grid
size	Zahlenwerte für die x- und y-Achse	Definiert die Größe des Widgets
dataID		Definiert die im Widget anzuzeigende Datenquelle
triggerparams		Definiert die Parameter, die für die Anzeige des Bilds im Image Viewer zuständig sind
urlParam		Definiert die Parameter, die durch die Auswahl eines Messwertes aus der Liste verfügbarer Messwerte an die URL angehängt werden. Dies ermöglicht die Weitergabe von Links zu Stockcharts, die immer die glichen Informationen anzeigen.
mandatory	true, false	Ist der Wert "true", wird das Widget hervorgehoben, wenn keine Auswahl getroffen wurde.
Caption-Optionen		
show	true, false	Definiert, ob die Caption angezeigt wird oder nicht.
	label	Freitext

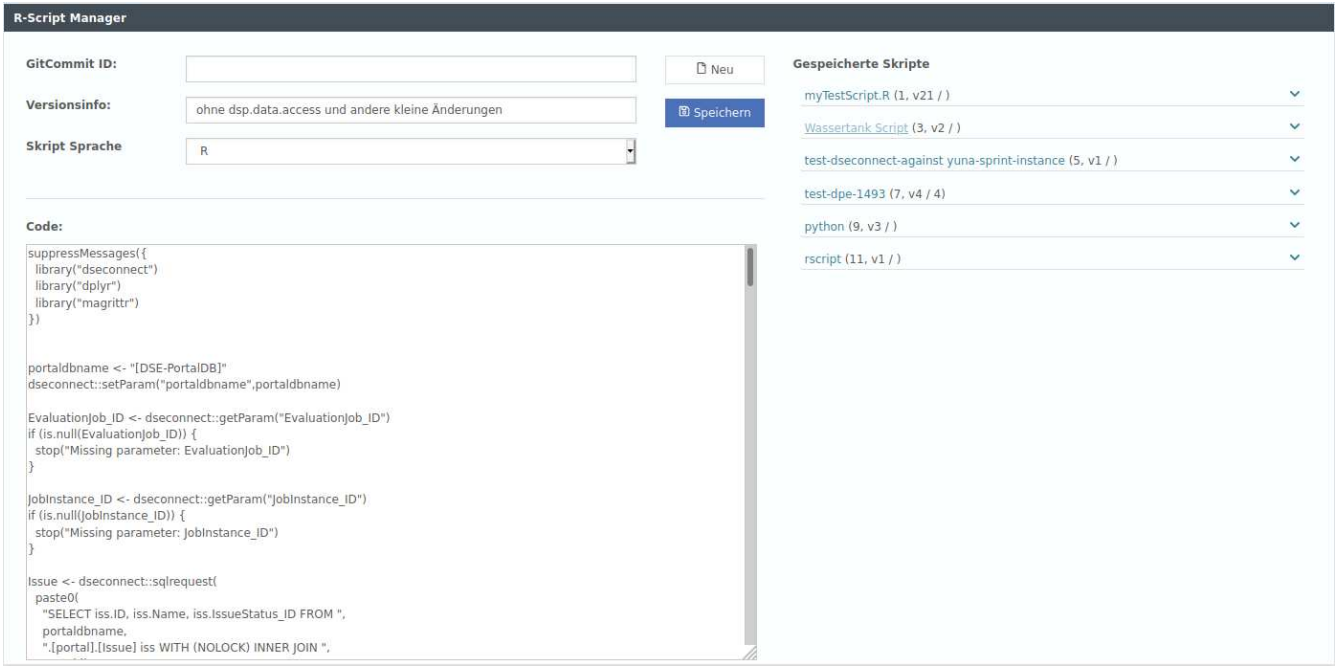
5.5.16. Skriptmanager (scriptdirective)

Über den Skriptmanager können Analyseskripte erstellt, gespeichert und geladen werden. Mit Analyseskripten in YUNA können Berechnungen von Daten durchführt werden. Anschließend können die Ergebnisse dargestellt werden. Anschließend können über den Job-Manager zur Überprüfung von Sachverhalten Jobs erstellen, die auf die gespeicherten Skripte zurückgreifen.

Über die Sprachauswahl "Skript Sprache" kann die Sprache ausgewählt werden, in der das Skript bei der Jobausführung interpretiert werden soll.

Um R-Skripte und/oder Python-Skripte ausführen zu können, muss ein Agent für die

jeweilige Sprache zu Verfügung stehen.



XML

```
<xml>
  <widget name="template_widget_Scriptmanager">
    <!-- Position from left top  -->
    <position>
        <x>0</x>
        <y>0</y>
    </position>
    <!-- Size of the Widget -->
    <size>
      <x>14</x>
      <y>7</y>
    </size>
    <!-- Caption -->
    <caption>
      <show>true</show>
      <label>Script-Editor-Template</label>
    </caption>
    <!-- Name of the WidgetType -->
    <widgettype>scriptdirective</widgettype>
  </widget>
</xml>
```

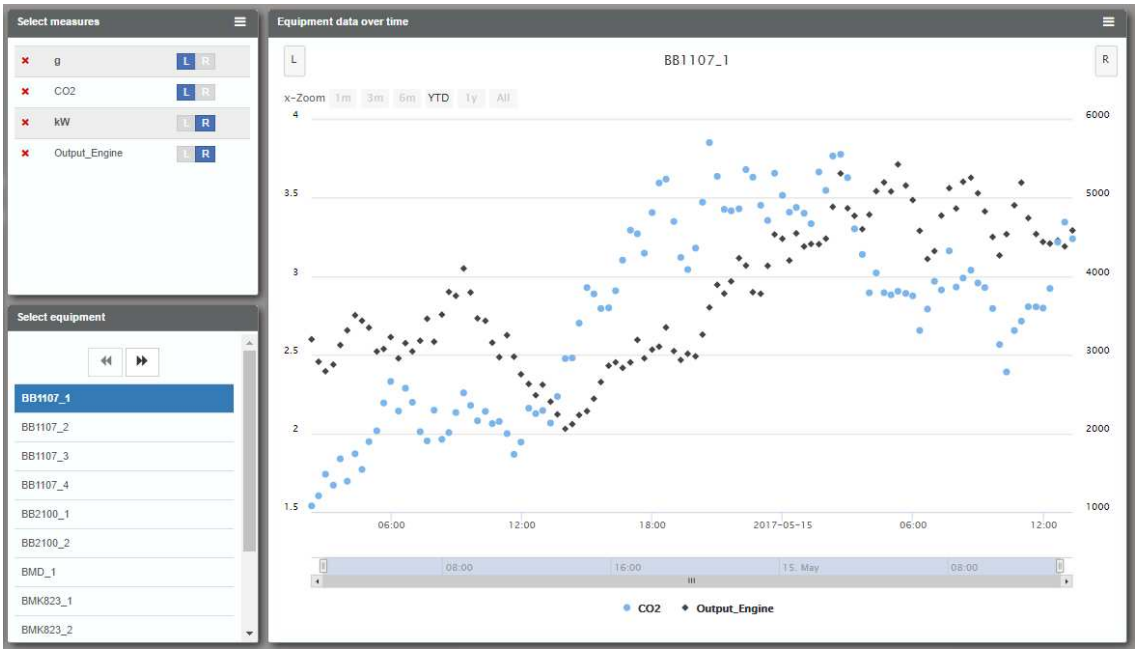
JSON

```
{
    "position": {
        "position": [0, 0]
    },
```

```
    "size": {
        "x": 14,
        "y": 7
    },
    "caption": {
        "show": true,
        "label": "Script-Editor"
    },
    "widgetname": "scriptdirective",
    "triggerParams": []
}
```

5.5.17. Stockchart (stockchartDirective)

Ein Stockchart ist ein Widget-Typ aus dem Bereich der Diagramme, der Informationen zu Equipments über einen Zeitverlauf anzeigen kann. Zunächst werden über eine Sensorliste verfügbare Sensoren bzw. Messwerte ausgewählt, die schließlich den Rahmen für das Stockchart-Widget bilden (Definition der X- und Y-Achse).



Den zu betrachtenden Zeitbereich im Stockchart-Widget auswählen:

- Über die Buttons im oberen Bereich können feste Zoomintervalle für die X-Achse definiert werden
- Durch ein Ziehen der Maus bei gleichzeitig gedrückter linker Maustaste über den Diagrammbereich.

Um ein Stockchart anzulegen und zu gestalten hat ein Dashboard Developer die folgenden Optionen:

Option	Inhalt	Bedeutung
Widgettype	developmentstockchart	Definiert das Widget vom Typ DevelopmentStockchart

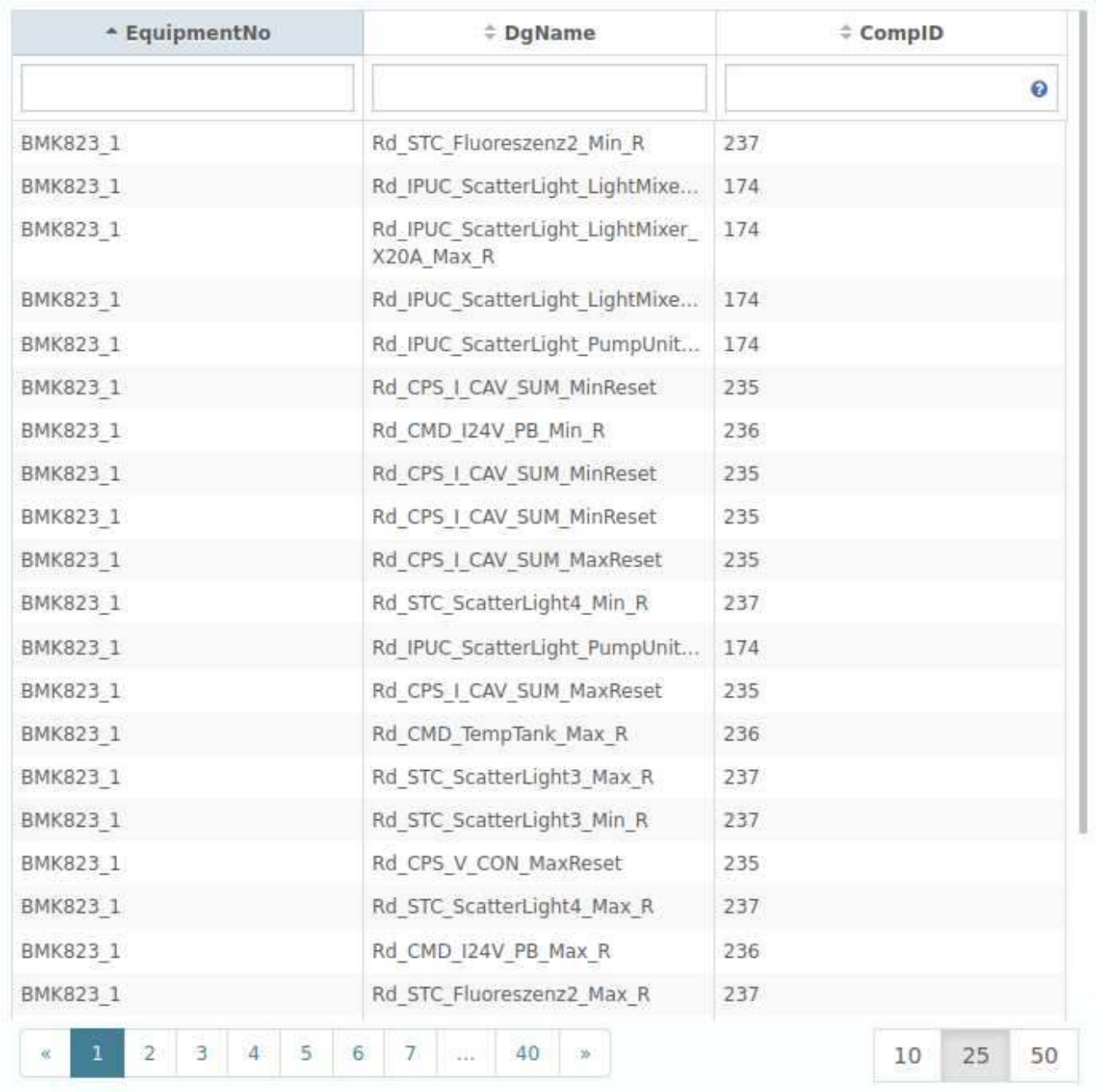
Option	Inhalt	Bedeutung
dataID	Name der Data_ID	Definiert die Data_ID, deren Daten vom Stockchart abgefragt bzw. dargestellt werden sollen
position	X & Y-Wert der Position im Grid	X & Y-Wert der Position im Grid
size	X & Y-Werte für die Größe des Widgets	X & Y-Werte für die Größe des Widgets
Caption		Über die Caption lassen sich zusätzliche Informationen und Funktionen des Widgets steuern (z.B. Titel)
triggerParams	Namen der TriggerParamater des Stockcharts	In den *triggerparams* kann definiert werden, auf welche URL-Parameter das Widget reagieren soll.Wenn sich der Wert eines hier angegebenen URL-Parameter verändert, wird das Tabellen-Widget neu geladen. Sind nicht alle URL-Parameter vorhanden, wird in der Titelleiste ein Ausrufezeichen angezeigt, an dem ein Tooltip dargestellt wird.Der Text im Tooltip ist über *paramsIncompleteInfo* und *paramsIncompleteInfoTkey* frei definierbar. Weitere Informationen: Triggerparameter.
paramsIncompleteInfo	Standardübersetzung	Der Text, der hierbei hinterlegt wird, wird als der Standardwert verwendet.Dieser kommt zum Einsatz, wenn die Übersetzung über den Translation-Key fehlgeschlagen ist. Weitere Informationen: Übersetzbare Inhalte.
paramsIncompleteInfo Tkey	Translation-Key	Hier wird der Translation-Key eingetragen.Über diesen wird aus der Aktuell gewählten Sprache die Übersetzung für den Tooltip angezeigt. Weitere Informationen: Übersetzbare Inhalte.

YUNA ML Anlegen eines Stockchart-Widgets

```
<xml>
  <widget name="template_widget_Stockchart">
    <position>
        <y>0</y>
        <x>0</x>
    </position>
    <size>
      <x>10</x>
      <y>7</y>
    </size>
```

```
    <caption>
      <show>true</show>
      <label>Stockchart-Template</label>
    </caption>
    <!-- Name of the WidgetType -->
    <widgettype>developmentstockchart</widgettype>
    <dataID>qy_NameOfDataID</dataID>
    <triggerParams>
        <mandatory>
            <list>sensor</list>
            <list>equi</list>
        </mandatory>
        <optional>
        </optional>
    </triggerParams>
  </widget>
</xml>
```

5.5.18. Tabellen-Widget (tabledirective)



EquipmentNo	DgName	CompID
BMK823_1	Rd_STC_Fluoreszenz2_Min_R	237
BMK823_1	Rd_IPUC_ScatterLight_LightMixe...	174
BMK823_1	Rd_IPUC_ScatterLight_LightMixer_X20A_Max_R	174
BMK823_1	Rd_IPUC_ScatterLight_LightMixe...	174
BMK823_1	Rd_IPUC_ScatterLight_PumpUnit...	174
BMK823_1	Rd_CPS_I_CAV_SUM_MinReset	235
BMK823_1	Rd_CMD_I24V_PB_Min_R	236
BMK823_1	Rd_CPS_I_CAV_SUM_MinReset	235
BMK823_1	Rd_CPS_I_CAV_SUM_MinReset	235
BMK823_1	Rd_CPS_I_CAV_SUM_MaxReset	235
BMK823_1	Rd_STC_ScatterLight4_Min_R	237
BMK823_1	Rd_IPUC_ScatterLight_PumpUnit...	174
BMK823_1	Rd_CPS_I_CAV_SUM_MaxReset	235
BMK823_1	Rd_CMD_TempTank_Max_R	236
BMK823_1	Rd_STC_ScatterLight3_Max_R	237
BMK823_1	Rd_STC_ScatterLight3_Min_R	237
BMK823_1	Rd_CPS_V_CON_MaxReset	235
BMK823_1	Rd_STC_ScatterLight4_Max_R	237
BMK823_1	Rd_CMD_I24V_PB_Max_R	236
BMK823_1	Rd_STC_Fluoreszenz2_Max_R	237

In dem Tabellen-Widget können Datensätze, zum Beispiel aus einer Datenbanktabelle, in YUNA dargestellt werden.

YUNAML-Tags

YUNAML-Tag	Beschreibung	Beispiel
dataID	Mithilfe einer *dataID* kann eine Datenbankabfrage ausgeführt werden.Das daraus resultierende Ergebnis wird dann in der Tabelle dargestellt. Weitere Informationen: Data ID - Definition der Daten.	`<dataID>qy_my_data_id</dataID>`

YUNAML-Tag	Beschreibung	Beispiel

inputs	Hier kann ein input channel definiert werden.Über einen input channel können Daten and das Tabellen-Widget übergeben und dargestellt werden.
	
	Weitere Informationen: Datasource.
		`<inputs>`
		`  <data>inputChannel</data>`
		`</inputs>`

outputs	Hier kann ein output channel definiert werden.Über einen output channel können Daten vom Tabellen-Widget an andere input-channel übergeben werden.
	
	Weitere Informationen: Datasource.
		`<outputs>`
		`  <current>`
		`  dataOutputChannel`
		`  </current>`
		`  <selected>`
		`  selectedDataOutputChannel`
		`  </selected>`
		`</outputs>`

generalOptions	Mithilfe der *generalOptions* ist es möglich grundlegende Eigenschaften der Tabelle zu Konfigurieren.
	
	Weitere Informationen: Grundlegende Eigenschaften.
		`<generalOptions>`
		`  <addColumns>true</addColumns>`
		`</generalOptions>`

cols	Mithilfe von *cols* können eigene Tabellenspalten definiert und angepasst werden.
	
	Weitere Informationen: Erweiterte Eigenschaften.
		`<cols>`
		`  <list>`
		`  <field>`
		`  DatabaseField`
		`  </field>`
		`  <title>`
		`  MyFieldName`
		`  </title>`
		`  <sortable>`
		`  DatabaseField`
		`  </sortable>`
		`  <filter>`
		`  <DatabaseField>`
		`  text`
		`  </DatabaseField>`
		`  </filter>`
		`  <show>true</show>`
		`  <width>100</width>`
		`  </list>`
		`</cols>`

YUNAML-Tag	Beschreibung	Beispiel

triggerParams	In den *triggerparams* kann definiert werden, auf welche URL-Parameter das Widget reagieren soll.Wenn sich der Wert eines hier angegebenen URL-Parameter verändert, wird das Tabellen-Widget neu geladen.
	
	Sind nicht alle URL-Parameter vorhanden, wird in der Titelleiste ein Ausrufezeichen angezeigt, an dem ein Tooltip dargestellt wird.Der Text im Tooltip ist über *paramsIncompleteInfo* und *paramsIncompleteInfoTkey* frei definierbar.
	
	Weitere Informationen: Triggerparameter.
		`<triggerParams>`
		`<mandatory>`
		`<list>`
		`UrlParameter1`
		`</list>`
		`</mandatory>`
		`</triggerParams>`

paramsIncompleteInfo	Der Text, der hierbei hinterlegt wird, wird als der Standardwert verwendet.Dieser kommt zum Einsatz, wenn die Übersetzung über den Translation-Key fehlgeschlagen ist.
	
	Weitere Informationen: Übersetzbare Inhalte.
		`<paramsIncompleteInfo>`
		`default translation`
		`</paramsIncompleteInfo>`

paramsIncompleteInfoTkey	Hier wird der Translation-Key eingetragen.Über diesen wird aus der Aktuell gewählten Sprache die Übersetzung für den Tooltip angezeigt.
	
	Weitere Informationen: Übersetzbare Inhalte.
		`<paramsIncompleteInfoTkey>`
		`my.translation.key`
		`</paramsIncompleteInfoTkey>`

sorting	Mithilfe von *sorting* wird festgelegt, über welches Feld Initial sortiert werden soll.	`<sorting>DatabaseField</sorting>`
sortingorder	**Mögliche Werte:** asc, desc.**Default:** asc Mithilfe von *sortingorder* wird festgelegt, ob das in *sorting* definierte Feld aufsteigend oder absteigend sortiert werden soll.	`<sortingorder>desc</sortingorder>`

countsOptions	Über *countsOptions* kann die Anzahl der Zeilen pro Seite der Tabelle definiert werden.Es können mehrere Werte eingetragen werden, wodurch im Tabellenwidget Buttons dargestellt werden um zwischen den verschiedenen Werten umzuschalten.	`<countsOptions>`
		`<list>15</list>`
		`<list>25</list>`
		`<list>50</list>`
		`</countsOptions>`

analytics	Mit dem YUNAML-Tag *analytics* können einer Tabelle analytische Funktionen hinzugefügt werden.
	
	Ist eine analytische Funktion aktiviert, kann sie über das Menü-Symbol erreicht werden.
	
	Weitere Informationen: Erweiterte Eigenschaften.
		`<analytics>`
		`<chart>true</chart>`
		`<export>true</export>`
		`<summary>true</summary>`
		`<search>true</search>`
		`</analytics>`

Darstellung von Daten

Damit das Tabellen-Widget Daten darstellen kann, müssen diese entweder über die d*ataID*, oder den *inputChannel* übergeben werden.

Anlegen einer Tabelle

```
<xml>
    <widget>
        <position>
            <x>0</x>
            <y>0</y>
        </position>
        <size>
            <x>8</x>
            <y>6</y>
        </size>
        <widgettype>tabledirective</widgettype>
        <dataID>qy_my_data_id</dataID>
        <generalOptions>
            <addColumns>true</addColumns>
        </generalOptions>
    </widget>
</xml>
```

Grundlegende Eigenschaften (generalOptions)

In den *generalOptions* können grundlegende Eigenschaften zur Darstellung der Tabelle festgelegt werden.Diese beziehen sich nur auf Spalten, die nicht separat mithilfe von *cols* definiert sind.

Die Angabe der *generalOptions* ist nicht zwingend erforderlich.

YUNAML-Tags für generalOptions

YUNAML-Tag	Mögliche Werte	default	Beschreibung	Beispiel
generalOptions > **addColumns**	true, false	false	Fügt alle Spalten aus der Datenherkunft (Data_ID) an die Tabelle an, sofern sie nicht über *cols* definiert sind.	`<addColumns>true</addColumns>`
generalOptions > **skipColumns**	YUNAML Liste		Verhindert die Anzeige von Spalten, die durch *addColumns* hinzugefügt werden.	`<skipColumns><list>DatabaseField</list></skipColumns>`

YUNAML-Tag	Mögliche Werte	default	Beschreibung	Beispiel
generalOptions > **convertColumns**	true, false	false	Überprüft die durch addColumns hinzugefügten Spalten auf numerischen Inhalt und wandelt diesen gegebenenfalls um. Dadurch werden diese Spalten numerisch sortierbar. Durch R erzeugte Werte Inf bzw. -Inf werden zu Infinity bzw. -Inifinity; NaN sowie NA zu NaN.	`<convertColumns>false</convertColumns>`
generalOptions > **sortable**	true, false	true	Erlaubt die Sortierbarkeit von Spalten einzustellen.	`<sortable>true</sortable>`

<table>
<tr><th>YUNAML-Tag</th><th>Mögliche Werte</th><th>default</th><th>Beschreibung</th><th>Beispiel</th></tr>
<tr>
<td>generalOptions > selectable</td>
<td>true, false, single</td>
<td>false</td>
<td>
<table>
<tr><th>Wert</th><th>Eläuterung</th></tr>
<tr><td>true</td><td>Es können 1-n Spalten ausgewählt werden.

Ausgewählte Spalten können über den output channel selected für andere input channel zur verfügung gestellt werden.</td></tr>
<tr><td>single</td><td>Es kann eine Spalte ausgewählt werden.

Die Ausgewählte Spalte kann über den output channel selected für andere input channel zur verfügung gestellt werden.</td></tr>
<tr><td>false</td><td>Es können keine Spalten ausgewählt werden.</td></tr>
</table>
</td>
<td><pre><selectable>single</selectable></pre></td>
</tr>
<tr>
<td>generalOptions > style</td>
<td>CSS als YUNAML</td>
<td></td>
<td>Definiert css styles für alle Spaltenüberschriften.

Kann in cols für einzelne Spalten überschrieben werden.</td>
<td><pre><style>
    <color>red</color>
    <background-color>
blue</background-color>
</style></pre></td>
</tr>
<tr>
<td>generalOptions > width</td>
<td>Angabe in Pixel</td>
<td></td>
<td>Definiert die Breite von Tabellenspalten.

Die tatsächliche Darstellung hängt vom verfügbaren Platz ab.</td>
<td><pre><width>120</width></pre></td>
</tr>
</table>

<table>
<tr><th>YUNAML-Tag</th><th>Mögliche Werte</th><th>default</th><th>Beschreibung</th><th>Beispiel</th></tr>
<tr>
<td>generalOptions > hideTableFooter</td>
<td>false, true</td>
<td>false</td>
<td>
<table>
<tr><th>Wert</th><th>Erläuterung</th></tr>
<tr><td>true</td><td>Sowohl die Seitenzahlen als auch die Anzahl der Zeilen pro Seite werden dargestellt</td></tr>
<tr><td>false</td><td>Die Anzahl der Zeilen pro Seite werden ausgeblendet.Die Seitenzahlen werden weiterhin angezeigt.</td></tr>
</table>
</td>
<td><pre><code><hideTableFooter>false</hideTableFooter></code></pre></td>
</tr>
</table>

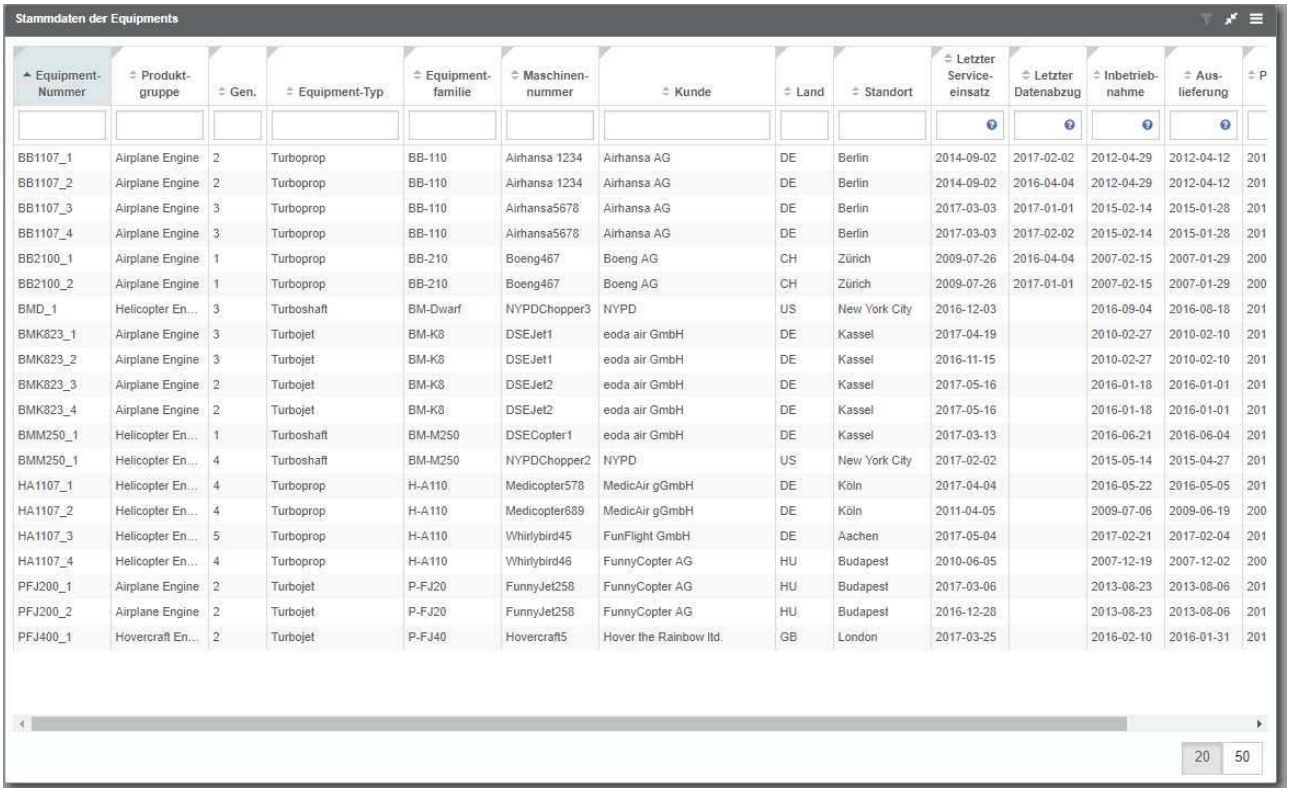
YUNAML

```
<generalOptions>
    <!-- adds all columns from the DataID to a table that are not defined by cols-->
    <addColumns>true</addColumns>
    <!-- hides specific columns from the DataID -->
    <skipColumns>
        <list>Job_ID</list>
        <list>Issue_ID</list>
    </skipColumns>
    <!-- converts added column-content to numerical values -->
    <convertColumns>true</convertColumns>
    <!-- enables/disables if columns are sortable -->
    <sortable>true</sortable>
    <!-- enables/disables if columns are filterable -->
    <filter>true</filter>
    <!-- options for formatting table headers -->
    <style>
        <color>red</color>
        <background-color>blue</background-color>
    </style>
</generalOptions>
```

Filterung und Sortierung in Tabellen

Tabellenspalten können von Usern des Portals durchsucht und so gefiltert werden. Hierzu stehen Nutzern unterschiedliche Möglichkeiten bereit, die sich auf die Inhalte der Tabelle auswirken.

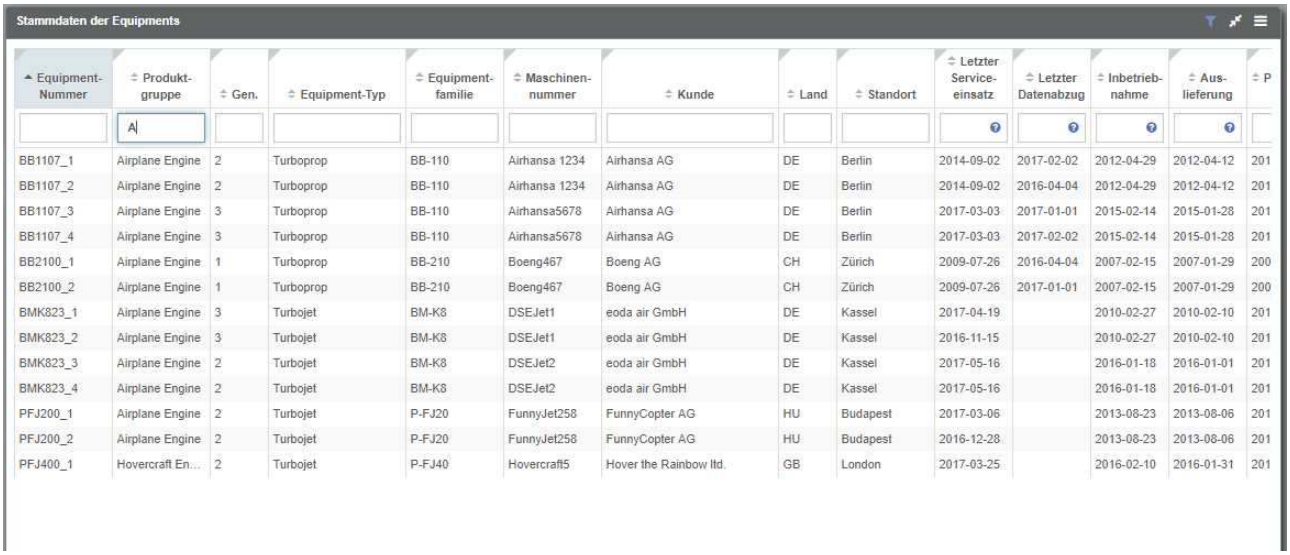
1. Im **Ursprungszustand** wird eine Tabelle ungefiltert und unsortiert dargestellt.



2. Es besteht die Möglichkeit, Tabelleninhalte für User filterbar zu machen. Möglich wird dies über die Definition des Parameters

```
<filter>true</filter>
```

Im folgenden Bild wird in der Spalte "Produktgruppe" der Inhalt der Tabelle gefiltert. Es werden nur noch die Inhalte der Tabelle angezeigt, bei denen in der Produktgruppe mindestens ein "a" enthalten ist. In diesem Beispiel bewirkt es, dass nur noch Datensätze der Produktgruppen "Airplane Engine" und "Hovercraft Engine" angezeigt werden nicht mehr aber Datensätze der Produktgruppe "Helicopter Engine".



Bei der Filterung von Tabellenspalten wird nicht zwischen Groß- und Kleinschreibung unterschieden.

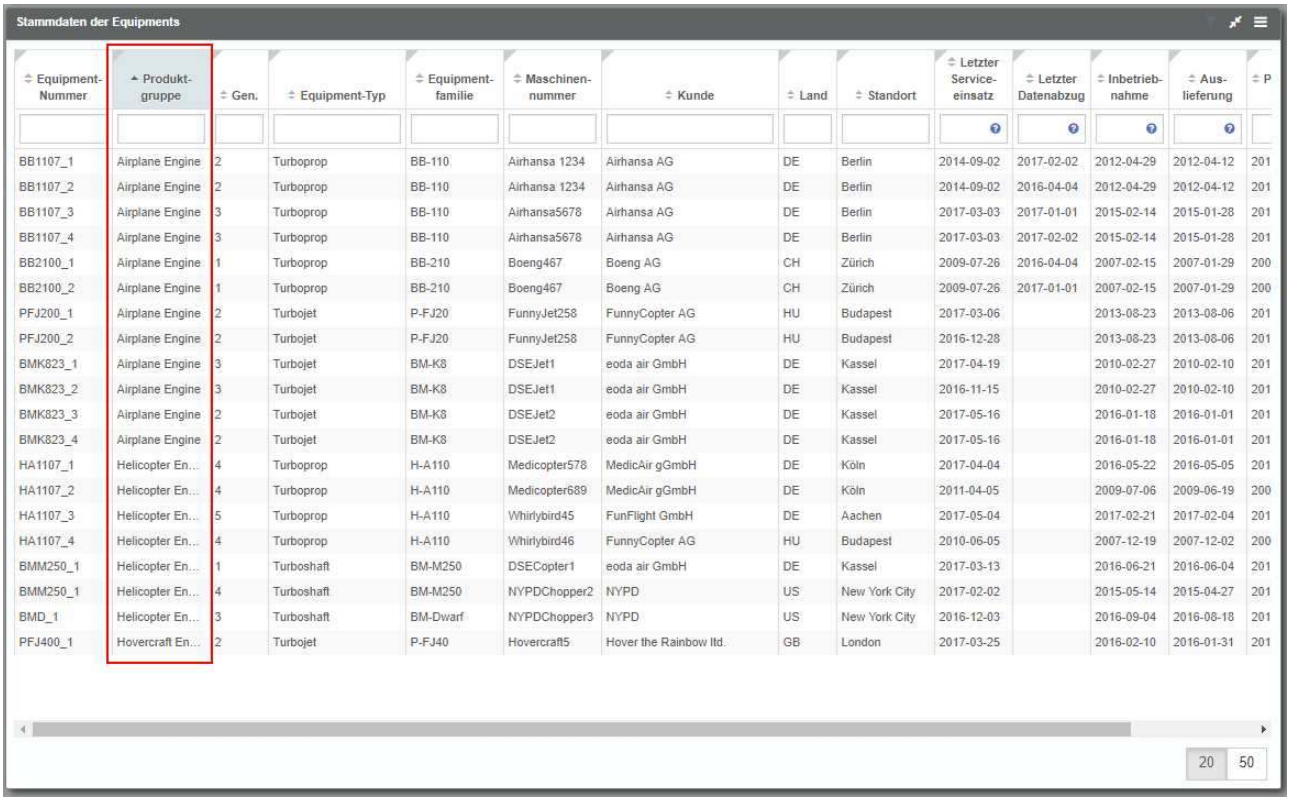
Eintrag	Bedeutung	Beispieleingabe	Beispielergebnis
a-z; A-Z; 0-9	Filtert die Tabelleninhalte und zeigt nur noch die Inhalte an, die den Sucheintrag enthalten.	Airplane	Zeigt nur noch Einträge an, bei denen in der gefilterten Spalte der Sucheintrag Airplane enthalten ist.
!a	Filtert die Tabelleninhalte und zeigt nur noch die Inhalte an, die **NICHT** den Sucheintrag hinter dem "!" enthalten	!Airplane	Zeigt nur noch Einträge an, bei denen in der gefilterten Spalte der Sucheintrag Airplane **NICHT** enthalten ist.
yyyy-mm-dd	Filterung nach Datum	2017-03-25	Zeigt nur Datensätze, die das gesuchte Datum in der Suchspalte enthalten

Die vollständige Liste von Operatoren sowie die zugehörige erlaubte Syntax zum Durchsuchen von Tabellenspalten ist abhängig vom Typ der Daten in der gewünschten Spalte. Für User kann eine Dokumentation zu den für die jeweilige Spalte relevanten Operatoren und Syntax verfügbar gemacht werden, die diese durch einen Klick im Suchfeld oberhalb einer Tabellenspalte auf das  -Icon ansehen können.

3. Tabelleninhalte können **sortiert** werden. Hierzu muss das Sortier-Symbol in der Kopfzeile der gewünschten Tabellenspalte angeklickt werden. Es steht Nutzern frei, ob sie Tabellenspalten aufsteigend oder absteigend sortieren möchten.

   Auch dieses Feature kann in der Dashboard-Definition aktiviert oder deaktiviert werden über den Operator

```
<sortable>true</sortable>
```



Equipment-Nummer	Produkt-gruppe	Gen.	Equipment-Typ	Equipment-familie	Maschinen-nummer	Kunde	Land	Standort	Letzter Service-einsatz	Letzter Datenabzug	Inbetrieb-nahme	Aus-lieferung	P
BB1107_1	Airplane Engine	2	Turboprop	BB-110	Airhansa 1234	Airhansa AG	DE	Berlin	2014-09-02	2017-02-02	2012-04-29	2012-04-12	201
BB1107_2	Airplane Engine	2	Turboprop	BB-110	Airhansa 1234	Airhansa AG	DE	Berlin	2014-09-02	2016-04-04	2012-04-29	2012-04-12	201
BB1107_3	Airplane Engine	3	Turboprop	BB-110	Airhansa5678	Airhansa AG	DE	Berlin	2017-03-03	2017-01-01	2015-02-14	2015-01-28	201
BB1107_4	Airplane Engine	3	Turboprop	BB-110	Airhansa5678	Airhansa AG	DE	Berlin	2017-03-03	2017-02-02	2015-02-14	2015-01-28	201
BB2100_1	Airplane Engine	1	Turboprop	BB-210	Boeng467	Boeng AG	CH	Zürich	2009-07-26	2016-04-04	2007-02-15	2007-01-29	200
BB2100_2	Airplane Engine	1	Turboprop	BB-210	Boeng467	Boeng AG	CH	Zürich	2009-07-26	2017-01-01	2007-02-15	2007-01-29	200
PFJ200_1	Airplane Engine	2	Turbojet	P-FJ20	FunnyJet258	FunnyCopter AG	HU	Budapest	2017-03-06		2013-08-23	2013-08-06	201
PFJ200_2	Airplane Engine	2	Turbojet	P-FJ20	FunnyJet258	FunnyCopter AG	HU	Budapest	2016-12-28		2013-08-23	2013-08-06	201
BMK823_1	Airplane Engine	3	Turbojet	BM-K8	DSEJet1	eoda air GmbH	DE	Kassel	2017-04-19		2010-02-27	2010-02-10	201
BMK823_2	Airplane Engine	3	Turbojet	BM-K8	DSEJet1	eoda air GmbH	DE	Kassel	2016-11-15		2010-02-27	2010-02-10	201
BMK823_3	Airplane Engine	2	Turbojet	BM-K8	DSEJet2	eoda air GmbH	DE	Kassel	2017-05-16		2016-01-18	2016-01-01	201
BMK823_4	Airplane Engine	2	Turbojet	BM-K8	DSEJet2	eoda air GmbH	DE	Kassel	2017-05-16		2016-01-18	2016-01-01	201
HA1107_1	Helicopter En...	4	Turboprop	H-A110	Medicopter578	MedicAir gGmbH	DE	Köln	2017-04-04		2016-05-22	2016-05-05	201
HA1107_2	Helicopter En...	4	Turboprop	H-A110	Medicopter689	MedicAir gGmbH	DE	Köln	2011-04-05		2009-07-06	2009-06-19	200
HA1107_3	Helicopter En...	5	Turboprop	H-A110	Whirlybird45	FunFlight GmbH	DE	Aachen	2017-05-04		2017-02-21	2017-02-04	201
HA1107_4	Helicopter En...	4	Turboprop	H-A110	Whirlybird46	FunnyCopter AG	HU	Budapest	2010-06-05		2007-12-19	2007-12-02	200
BMM250_1	Helicopter En...	1	Turboshaft	BM-M250	DSECopter1	eoda air GmbH	DE	Kassel	2017-03-13		2016-06-21	2016-06-04	201
BMM250_1	Helicopter En...	4	Turboshaft	BM-M250	NYPDChopper2	NYPD	US	New York City	2017-02-02		2015-05-14	2015-04-27	201
BMD_1	Helicopter En...	3	Turboshaft	BM-Dwarf	NYPDChopper3	NYPD	US	New York City	2016-12-03		2016-09-04	2016-08-18	201
PFJ400_1	Hovercraft En...	2	Turbojet	P-FJ40	Hovercraft5	Hover the Rainbow ltd.	GB	London	2017-03-25		2016-02-10	2016-01-31	201

20 50

Reload Counter für Tabellen

Definition des Reload Counters für den Table-Widget Memory Leak Workaround

Hintergrund des Workarounds

Aufgrund von Memory-Leaks in einer für das Tabellen-Widget verwendeten Bibliothek (ngTable) wurde ein Workaround eingebaut, der dafür sorgt, dass das Portal nach einer gewissen Anzahl von Aufrufen von Portal-Elementen, in denen ngTable verwendet wird (Table-Widget und Single Choice Directive), beim nächsten Aufruf neu geladen wird. So kann vermieden werden, dass ein häufiger Aufruf von Sichten mit Tabellen und Einzelselektionen den Speicherbedarf des Portals im Browser in die Höhe treibt und so das System verlangsamt.

Definition des Counters:

Die Einstellung des Counters zur erfolgt direkt über die Datenbank:

DB-Tabelle	Parametername	Eigenschaft	Mögliche Werte	Default-Wert
portal.ConfigStore	tableReloadCount	Definiert die Anzahl von ngTable-Abfragen, bevor ein Reload der Portal-View erzwungen wird.	alle positiven ganzzahligen Werte ab 1	10

Wirkweise des Workarounds:

Wird in der Datenbank der Parameter TableReloadCount beispielsweise auf den Wert 5 gesetzt, so "duldet" das Portal 5 Abfragen von Portalelementen, die ngTable beinhalten (z.B. 5 Table-Widgets in der Portal-View). Sobald die 6. Abfrage gestartet wird, wird automatisch die Portal-View neu geladen.

Standardmäßig - also wenn der Parameter TableReloadCount nicht anders definiert wird - steht der Parameter auf 10. D.h., bei der 11. Abfrage an ngTable-Elemente würde die PortalView neu geladen.

Spalten-Typen

Der Typ einer Spalte entscheidet, wie die Daten verarbeitet und anschließend in der Tabelle dargestellt werden. Auf den folgenden Seiten gehen wir auf die einzelnen Spalten-Typen ein, um ihre Funktion und Konfigurationsmöglichkeiten aufzuzeigen.

Text (default)

Der Typ "default" wird verwendet um reinen Text anzeigen zu lassen. Um eine Spalte als Text bzw. default zu definieren muss die Eigenschaft "type" weggelassen oder auf null gesetzt werden.

XML

```
<cols>
    <list>
        <field>Text_Field</field>
        <title>Textspalte</title>
        <show>true</show>
        <width>120</width>
    </list>
</cols>
```

JSON

```
"cols":
[{
    "field": "Text_Field",
    "title": "Textspalte",
    "show": true,
    "width": 120
}]
```

Truncation

Es ist zu empfehlen, der Spalte eine feste Breite zu vergeben. In diesem Fall wird zu langer Text, der die Spaltenbreite überschreitet, abgeschnitten und mit Rüberfahren des Mauszeigers über eine bestimmte Zelle in der Tabelle kann man sich den Text in voller länge anzeigen lassen. (vgl. "Truncation" bei den Informationen zur bedingten Formatierung von Zellen)

Zahl (number)

Der Spaltentyp "number" wird verwendet, um die Werte der Spalte als Zahl behandeln zu lassen.

XML

```
<cols>
    <list>
        <field>Numberfield</field>
        <title>Zahlenspalte</title>
        <type>number</type>
        <show>true</show>
    </list>
</cols>
```

Zahlenspalten können weiter konfiguriert werden um bestimmte Formatierungen anzuwenden. Dafür wird die Eigenschaft "options" verwendet. Es können alle Konfigurationsmöglichkeiten verwendet werden, die auch im "options"-Parameter von Intl.Numberformat verfügbar sind.

Beispiel: Darstellung von Währungen

XML

```
<cols>
    <list>
        <field>Numberfield</field>
        <title>Zahlenspalte</title>
        <type>number</type>
        <show>true</show>
        <options>
            <style>currency</style>
            <currency>EUR</currency>
        </options>
    </list>
</cols>
```

Success- & Error-Icons (flag)

Eine Tabellenspalte des Typs flag wird als Icon dargestellt.

Beispiel

Die Eigenschaft "field" erwartet einen Boolchen Ausdruck:

Wert	Symbol	Bedeutung
true		success
false		error
null/undefined		es liegt noch kein Ergebnis vor

XML

```
<cols>
    <list>
        <field>Active</field>
        <title>Aktiv</title>
        <sortable>Active</sortable>
        <filter>
            <Active>text</Active>
        </filter>
        <show>true</show>
        <type>flag</type>
    </list>
</cols>
```

JSON

```
"cols": [{
    "field": "Active",
    "title": "Aktiv",
    "sortable": "Active",
    "filter": {
        "Active": "text"
    },
    "show": true,
    "type": "flag"
}]
```


Wird aus der Datenbank NULL geliefert, so wird dies als false interpretiert.

D.h., wenn kein Wert in der Datenbank vorliegt, so wird derzeit ein error-Icon angezeigt.

Bild (image)

Eine Tabellen-Spalte des Typs "image" erwartet aus als Rückgabe der Datenbank den Dateinamen einer *.png-Datei, die sich im /images Ordner auf dem Server befindet. Dieses Bild wird dann direkt in der Tabelle im vollen Format dargestellt.

XML

```
<cols>
    <list>
        <field>image_name</field>
        <title>Bilder</title>
        <type>image</type>
        <show>true</show>
    </list>
</cols>
```

JSON

```
"cols":
[{
  "field": "Image_Name",
  "title": "Bilder",
  "show": true,
  "type": "image"
}]
```

Datum (genDate)

Einführung und vordefinierte Datumsformate

genDate ermöglicht es, eine Spalte als Datum ausgeben zu lassen. Hierzu erwartet das Frontend ein Datums-Objekt aus der Datenbank. Dieses wird standardmäßig in folgendem Format ausgegeben:

YYYY-MM-DD HH:mm:ss

Der jeweiligen Spalte kann optional die Eigenschaft "format" mitgegeben werden um ein anderes Format zur Anzeige im Frontend zu bestimmen:

XML

```
<cols>
    <list>
        <field>Date_Field</field>
        <title>datumsspalte</title>
        <type>genDate</type>
        <format>short</format>
        <show>true</show>
    </list>
</cols>
```

JSON

```
"cols":
[{
    "field": "Date_Field",
    "title": "Datumspalte",
    "type": "genDate",
    "format": "short"
    "show": true,
}]
```

Nutzbare Datums-Formate

Dem optionalen Feld "format" können folgende vordefinierte Werte vergeben werden:

Angabe	Format	Beispiel
"short"	YYYY-MM-DD	2016-06-05
"long"	YYYY-MM-DD HH:mm:ss.sss	Englisch:2016-06-05 13:22:24.000 Deutsch:2016-06-05 13:22:24,000
keine Angabe (default)	YYYY-MM-DD HH:mm:ss	2016-06-05 13:22:24

Standortabhängige Datumsformatierung

In der Formateinstellung "long" wird abhängig von der Lokalisierung Ihres Browsers das entsprechend lokalisierte Format angewandt. Im Falle einer deutschen Lokalisierung wird eine deutsche Formatierung gewählt. In allen anderen Fällen, die englische.

Darüberhinaus besteht hier die Möglichkeit ein individuell zusammengestelltes Datums-Format anzugeben. Die Möglichkeiten für diese Funktionalität wird im Folgenden beschrieben.

Das jeweils konfigurierte Datumsformat wird beim Datenexport analog für die jeweilige Spalte berücksichtigt.

Jahr, Monat und Tag

Format	Beispiel	Beschreibung
YYYY	2014	4 stelliges Jahr
YY	14	2 stelliges Jahr
Y	-25	Jahr mit jeglicher Anzahl an Ziffern und Vorzeichen
Q	1..4	Quartal des Jahres
M/MM	1..12	Monat
MMM/MMMM	Jan..Dezember	lokaler Monat-Name
D/DD	1..31	Tag des Monats
DDD/DDDD	1..365	Tag des Jahres
X	1410715640.579	Unix Zeitstempel
x	1410715640579	Unix Zeitstempel in ms

YYYY unterstützt 2 stellige Jahre und konvertiert diese in ein Jahr nahe 2000 (genau wie YY).

Kalenderwoche, Woche und Wochen

Die klein-buchstabigen Tokens nutzen die lokalen (länderspezifischen) Wochen-start-Tage, wohingegen die groß-buchstabigen diejenigen der ISO-Norm nutzen.

Format	Beispiel	Beschreibung
w/ww	1..53	lokale Kalenderwoche
e	0..6	lokaler Wochentag

Format	Beispiel	Beschreibung
ddd/dddd	Mon…Sonntag	lokaler Tag-Name
W/WW	1..53	ISO Kalenderwoche
E	1..7	ISO Wochentag

Stunden, Minuten, Sekunden, Millisekunden und Zeitverschiebungs-Tokens

Format	Beispiel	Beschreibung
H/HH	0..23	Stunden (24h → 0..23)
k/kk	1..24	Stunden (24h → 1..24)
h/hh	1..12	Stunden (12h)
a/A	am pm	Vor- oder Nachmittag
m/mm	0..59	Minuten
s/ss	0..59	Sekunden
S/SS/SSS	0..999	Millisekunden
Z/ZZ	+12:00	Zeitverschiebung zu UTC als +-HH:mm, +-HHmm, oder Z

Links (genLink)

In einer Tabelle dargestellte Links können individuell Konfiguriert werden um Ihre Funktion und Darstellung zu beeinflussen. Hierzu können jeder Spalte einer Tabelle in der JSON-Datei Eigenschaften festgelegt werden.

- Links können intern oder extern sein
- Dem Link kann eine beliebige URL zugewiesen werden
- Der Link kann aus mehreren statischen und/oder mehreren dynamischen Teilen bestehen
- Die statischen und dynamischen Teile können einen beliebigen Inhalt haben
- Die Darstellung kann als fester Text, Icon oder Zelleninhalt konfiguriert werden

Sobald eine Spalte dem Typ „genLink" zugeordnet ist, werden die Inhalte dieser Spalte als Link dargestellt.

Eine Spalte mit dem Typ Link definieren

XML

```
<cols>
    <list>
        <field>Link_Field</field>
        <title>Spalte mit Links</title>
        <type>genLink</type>
        <link>
            <url>name_of_view</url>
            <label>EquipmentNo</label>
            <icon>fa-home</icon>
            <result>true</result>
```

```
            <appendAllUrlParams>false</appendAllUrlParams>
            <linkparams>
                <list>
                    <searchfield>issueid</searchfield>
                    <search>
                        <list>Issue_ID</list>
                        <list>Konstante</list>
                    </search>
                </list>
                <list>
                    <searchfield>equi</searchfield>
                    <search>
                        <list>EquipmentNo</list>
                    </search>
                </list>
            </linkparams>
            <extern>true</extern>
        </link>
        <show>true</show>
    </list>
</cols>
```

JSON

```
"type": "genLink",
"link": {
    "url": "name_of_view",
    "label": "EquipmentNo",
    "icon": "fa-home",
    "result": "true",
    "appendAllUrlParams": "false",
    "linkparams": [
        {"searchfield": "issueid", "search": ["Issue_ID", "Konstante"]},
        {"searchfield": "equi", "search": ["EquipmentNo"]},
        ...
    ],
    "extern": true
},
"show": true
```

Feld	Typ / Werte	Beschreibung	Default
url	String	Hier wird der Pfad eingetragen, zu welchem verlinkt werden soll. Hier kann eine statische URL oder ein Verweis auf eine Tabellenspalte angegeben werden.	

Feld	Typ / Werte	Beschreibung	Default
label	String	Dient der individuellen Darstellung des Links.Die Darstellung kann eine feste Zeichenkette oder ein Verweis auf ein Tabellenfeld sein. Standardmäßig wird der Inhalt als HTML-Interpretiert. Dementsprechend kann HTML zur Formatierung verwendet werden.Um die Interpretation als HTMl zu vermeiden und stattdessen den uninterpretierten Text anzuzeigen, kann dem Label ein "\" vorangestellt werden.	Der Wert des Aktuellen Tabellenfeldes
icon	String	Mithilfe des Feld *icon* kann ein Symbol an dem Link angezeigt werden.Die Symbole werden von **Font-Awesome** genutzt. Für die Verwendung muss lediglich der Name der **CSS Klasse** des Symbols eingetragen werden. Siehe: Dokumentation der Icon von Font Awesome Derzeit sind nur die Symbole von Font-Awesome aus Version 4.7 verfügbar. Ggf. sind einige der mit Version 5 neu hinzugefügten Symbole noch nicht nutzbar.	
result	true, false	Mit dem Feld *result* kann definiert werden, dass der Link als Button dargestellt werden soll.	false
linkparams	Array	Jedes Objekt im Array steht für einen Link-Parameter.Dieser besteht aus *searchfield* und *search* und kann individuell konfiguriert werden.	
linkparams → searchfield	String	Name des Parameters. Mit diesem Namen wird der Parameter als URL-Parameter in die URL geschrieben.	
linkparams → search	Array von Strings	In der Liste können die verschiedenen Quellen oder auch konstante Werte eingefügt werden.Diese werden konkateniert als Wert an den URL-Parameter geschrieben. Mit einem Präfix kann die Verarbeitung des Wertes bestimmt werden. Siehe: Präfix.	
linkparams → separator	String	Hiermit kann ein eigener Parameter konfiguriert werden, mit dem dieser URL-Parameter von den anderen getrennt wird.	&
linkparams → token	String	Hiermit kann ein eigener Parameter konfiguriert werden, mit dem der Name dieses URL-Parameters von seinem zugewiesenen Wert getrennt wird.	=

<table>
<tr><th>Feld</th><th>Typ / Werte</th><th>Beschreibung</th><th>Default</th></tr>
<tr><td>linkparams → noseparator</td><td>true, false</td><td><table><tr><th>Wert</th><th>Erläuterung</th></tr><tr><td>true</td><td>Das Trennzeichen wird unterdrückt. Dadurch wird dieser URL-Parameter ohne Trennzeichen an die URL übergeben</td></tr><tr><td>false</td><td>Das Trennzeichen wird zur Trennung des URL-Parameters von anderen URL-Parametern verwendet.</td></tr></table></td><td>false</td></tr>
<tr><td>linkparams → notoken</td><td>true, false</td><td><table><tr><th>Wert</th><th>Erläuterung</th></tr><tr><td>true</td><td>Der Token wird unterdrückt. Dadurch wird der Name dieses URL-Parameters von seinem zugewiesenen Wert nicht durch ein token getrennt.</td></tr><tr><td>false</td><td>Der Token wird zur Trennung des Namen dieses URL-Parameters von seinem zugewiesenen Wert verwendet.</td></tr></table></td><td>false</td></tr>
<tr><td>appendAllUrlParams</td><td>true, false</td><td><table><tr><th>Wert</th><th>Erläuterung</th></tr><tr><td>true</td><td>Alle aktuell in der URL enthalten URL-Parameter werden an die verlinkte View weitergegeben. Parameter, die in linkparams angegeben sind werden hinzugefügt. Falls ein Parameter in linkparams und in der aktuellen URL vorhanden ist, wird der Wert aus den linkparams übernommen.

Die Konfigurationen durch die Felder Separator, token, noseparator und notoken werden ignoriert.</td></tr><tr><td>false</td><td>Die in linkparams konfigurierten Parameter, werden an die URL übergeben.In dem Fall, dass der Link auf das aktuelle Dashboard verweist, werden alle zu dem Zeitpunkt gesetzten URL-Parameter überschrieben.</td></tr></table></td><td>false</td></tr>
</table>

<table>
<tr><th>Feld</th><th>Typ / Werte</th><th>Beschreibung</th><th>Default</th></tr>
<tr><td>extern</td><td>true, false</td><td><table><tr><th>Wert</th><th>Erläuterung</th></tr><tr><td>true</td><td>Der Link wird in einem neuem Tab geöffnet. Dabei wird kein TabExchange durchgeführt. Mit dieser Einstellung muss die angegebene URL ein absoluter Pfad sein. Hierbei muss der URL der Präfixe "http://" für Webseiten oder "\\" für Netzwerkfreigaben vorangestellt sein.</td></tr><tr><td>false</td><td>Der Link wird standardmäßig im aktuellen Tab geöffnet. Durch einen Klick auf das Mausrad oder über das Kontextmenü → "Link in neuem Tab öffnen", kann der Link dennoch in einem neuen Tab geöffnet werden. Dabei wird ein TabExchange durchgeführt. Mit dieser Einstellung wird die angegebene URL den Namen einer view beinhalten</td></tr></table></td><td>false</td></tr>
<tr><td>openInNewTab</td><td>true, false</td><td><table><tr><th>Wert</th><th>Erläuterung</th></tr><tr><td>true</td><td>Der Link wird in einem neuen Tab geöffnet. Überschreibt den durch die Option "extern" gesetzten Standard.</td></tr><tr><td>false</td><td>Der Link wird im aktuellen Tab geöffnet. Überschreibt den durch die Option "extern" gesetzten Standard.</td></tr></table></td><td></td></tr>
</table>

TabExchange

Bei einem TabExchange wird der aktive Filter an die verlinkte view übergeben

Die URL setzt sich wie folgt zusammen:

1. Der Verweis wird aus dem Feld *url* entnommen.
2. Falls vorhanden, werden die Parameterfelder und Suchparameter der Liste *linkparams* entnommen und mit einem '**?**' von der URL getrennt in der angegebenen Reihenfolge angehängt.
3. Sind mehrere Link-Parameter angegeben, werden diese mit einem '**&**' verbunden.

Beispiel: Link mit mehreren Parameterfelder und einer Konstanten

XML

```
<cols>
    <list>
        <field>Link_Field</field>
        <title>Spalte mit Links</title>
        <type>genLink</type>
        <link>
```

```
            <url>dsp_Issue_singleResult</url>
            <linkparams>
                <list>
                    <searchfield>issueid</searchfield>
                    <search>
                        <list>Issue_ID</list>
                        <list>*02</list>
                    </search>
                </list>
                <list>
                    <searchfield>equi</searchfield>
                    <search>
                        <list>EquipmentNo</list>
                    </search>
                </list>
            </linkparams>
            <label>EquipmentNo</label>
        </link>
        <show>true</show>
    </list>
</cols>
```

JSON

```
"link": {
    "url": dsp_Issue_SingleResult",
    "linkparams": [
        {searchfield: "issueid", "search": ["Issue_ID", "*02"]},
        {searchfield: "equi", "search": ["EquipmentNo"]}
    ],
    "label": "EquipmentNo"
}
```

ergibt folgende URL:
http://srvip/datascienceportal/#/dsp_Issue_SingleResult?[http://srvIP/datascienceportal/#/dsp_Issue_SingleResult?]issueid=9999&equi=equipmentno

Hierbei können beliebig viele Linkparameter, sowie je beliebig viele Suchparameter angegeben werden. Falls mehrere Suchparameter angegeben werden, werden diese ohne Trennzeichen aneinander gehängt (z.B. um Konstanten einzubauen).

Parameter und Individuelle Konfiguration

Einem Link können 0-n Parameter angehängt werden. Jedes Parameterfeld kann im „*linkparams*"-Bereich individuell konfiguriert werden.In „*searchfield*" muss ein Identifikator für das Parameterfeld festgelegt werden, welchem unter „*search*" ein oder mehrere Werte angehängt werden (siehe oberes Beispiel).

In beiden Feldern können Verweise oder konstante Texte verwendet werden. Um zu definieren wie sich das Parameterfeld verhalten soll, gibt es folgende Präfixe:

Präfix	Bedeutung	Beispiel
*	Parameter ist eine konstante ZeichenfolgeSiehe: Konstante	*MeinKonstanterWert
?	Parameter wird von dem angegebenen URL-Parameter entnommenSiehe: URL-Parameter	?MeinUrlParameter
\	Parameter wird nicht kodiertSiehe: Kodierung	\
ohne	Parameter wird aus dem angegebenen Tabellenfeld entnommenSiehe: Tabellenfeld	MeinTabellenFeld

Aktueller Wert eines Tabellenfeldes übernehmen

Wenn der Präfix entfällt wird der Wert aus dem angegebenen Tabellenfeld der aktuellen Zeile entnommen und als Wert für den URL-Parameter an den Link übergeben.

Konstanten Wert an Link übergeben

Mit dem Zeichen '*' kann definiert werden, dass es sich bei dem Wert um eine Konstante handelt.Eine Konstante ist eine feste Zeichenfolge. Diese wird so wie angegeben als Wert für den URL-Parameter an den Link übergeben.

Aktuellen URL-Parameter an Link übergeben

Ein Search-Value kann mit dem Value eines beliebigen, aktuellen URL-Parameters befüllt werden. Hierzu muss im Feld „search“ der Bezeichner des URL-Parameters mit einem vorhergehendem „?“-Zeichen eingetragen werden.

URL mit der die aktuelle View aufgerufen wurde:

.../#/cmp_Issue_Manager_Rating?equi=equipmentno**Link mit URL-Parameter**

XML

```
<link>
    <url>dsp_Issue_singleResult</url>
    <linkparams>
        <list>
            <searchfield>equino</searchfield>
            <search>
                <list>?equi</list>
            </search>
        </list>
    </linkparams>
</link>
```

JSON

```
"link":{
    "url": "dsp_Issue_SingleResult",
    "linkparams": [{
        "searchfield": "equino",
```

```
        "search": ["?equi"]
    }]
}
```

Ergibt folgenden Link:

.../#/dsp_Issue_SingleResult?equino=equipmentno

Beim übergeben eines URL-Parameters sollte darauf geachtet werden, dass der Bezeichner des Parameters exakt wie in der URL im „search“-Feld angegeben wird. Sollte der Parameter nicht in der aktuellen URL vorhanden sein oder keinen Wert besitzen, können Fehler auftreten.

Kodieren von Parametern

Standardmäßig werden übergebene Parameter kodiert, sodass gültige URLs ohne fehlerhafte Sonderzeichen generiert werden. In besonderen Fällen, zum Beispiel wenn sie bereits korrekt kodierte Inhalte referenzieren, führt dies zu einer doppelten Kodierung, was zu Fehlern in der URL führt. Um dies zu vermeiden kann dem jeweiligen YUNAML Element ein Backslash ("\") vorangestellt werden, um die automatische Kodierung zu deaktivieren.Beachten Sie, dass durch YUNA in diesem Fall keine Kodierung vorgenommen wird und der Dashboard-Entwickler dafür verantwortlich ist, die Werte korrekt zu kodieren.

Beispiel für einen mailto-genLink mit Zeilenumbruch

```
<link>
    <url>mailto:markus.mustermann@example.com</url>
    <linkparams>
        <list>
            <search>
                <list>*subject=Email-Titel</list>
                <list>*body=Erste Zeile des Email-Texts</list>
                <list>\
</list>
                <list>*Zweite Zeile des Email-Texts</list>
            </search>
            <notoken>true</notoken>
        </list>
    </linkparams>
    <extern>true</extern>
</link>
```

Trennzeichen

Jedem Parameterfeld können selbst definierte Trennzeichen vergeben werden. Hierzu werden dem Objekt des Parameters die Felder „separator" und „token" mitgegeben. In „separator" kann definiert werden, welches Trennzeichen verwendet werden soll um das Parameterfeld vom restlichen Link zu trennen. Mit „token" bestimmt man das Trennzeichen, welches den Identifikator vom Value trennt.KodiertBeispiel für einen Link mit einer Konstanten und definiertem Trennzeichen:

Beispiel: Link mit einer Konstante und definierte Trennzeichen

XML

```
<link>
    <url>dsp_Issue_singleResult</url>
    <linkparams>
        <list>
            <searchfield>EquipmentNo</searchfield>
            <search>
                <list>EquipmentNo</list>
            </search>
            <separator>%</separator>
            <token>$</token>
        </list>
    </linkparams>
    <label>EquipmentNo</label>
</link>
```

JSON

```
"link":{
    "url": "dsp_Issue_SingleResult",
    "linkparams": [{
        "searchfield": "*EquipmentNo",
        "search": ["EquipmentNo"]
        "separator": "%",
        "token": "$"
        //"notoken": true,
        //"noseparator": true
 }]
}
```

ergibt folgende URL:

.../#/dsp_Issue_SingleResult%EquipmentNo$equipmentno

hier wird '**%**' anstelle des '**?**' und '**$**'„?" zum Trennung des ersten Parameterfeldes zum Link. „&" für jedes weitere angehängte Parameterfeld und. „=" zwischen Identifikator und Value eines Parameterfeldes.

Die Benutzung der Trennzeichen kann mit den Parametern "notoken" und "noseparator" unterdrückt werden, indem man diese auf den Wert "true" setzt.

Konditions-Status (highlight)

Der Spalten-Typ "highlight" bietet die Möglichkeit, einen Wert gefolgt von einem farbigem Punkt zu Darstellung eines Status anzuzeigen.

Wert < 0.8 wird gefolgt von ● (success)Bei den momentanen Voreinstellungen ergeben sich für folgende Werte die jeweilige Ausgabe:

Wert >= 0.8 && Wert < 1 wird gefolgt von 🟠 (warning)

Wert == 1 wird gefolgt von 🔴 (error)

XML

```
<cols>
    <list>
        <field>Analysis</field>
        <title>Aktiv</title>
        <type>highlight</type>
        <show>true</show>
    </list>
</cols>
```

JSON

```
"cols":
[{
  "field": "Analysis",
  "title": "Aktiv",
  "show": true,
  "type": "highlight"
}]
```

Benutzer (user)

Der Spaltentyp 'user' wird verwendet, um die Darstellung von Benutzernamen im Portal einheitlich und eindeutig zu gestalten.

Sollen im Portal, z.B. einer Tabellenspalte oder einem Menü, Benutzer angezeigt werden, sollte für diese der Typ 'user' verwendet werden. Dieser formatiert auf Basis der bereitgestellten Benutzer-IDs (Tabellenspalte 'ID' in der Usertabelle) oder vorformatierten User-Namen (Query mit Metadatentyp 'user') den dargestellten Inhalt in der Form ‘GmbH,eoda ({username})’ (z.B. "Mustermann, Max (MusterMa)"

XML

```
<cols>
    <list>
        <field>ID</field>
        <title>Benutzer</title>
        <type>user</type>
        <show>true</show>
    </list>
</cols>
```

JSON

```
"cols": [{
```



```
    ...
    "type": "user"
}]
```

Favorit (favorite)

Der Spaltentyp 'favorite' wird verwendet, um Tabellenzeilen als Favoriten kennzeichnen zu können.

Damit dieser Spaltentyp korrekt funktioniert, müssen über einen Result-Rating-Provider die Favoriteninformationen an die Tabellendaten angehängt werden.

XML

```
<cols>
    <list>
        <field>favorite</field>
        <title>Favorit</title>
        <type>favorite</type>
        <show>true</show>
    </list>
</cols>
```

Analytische Funktionen

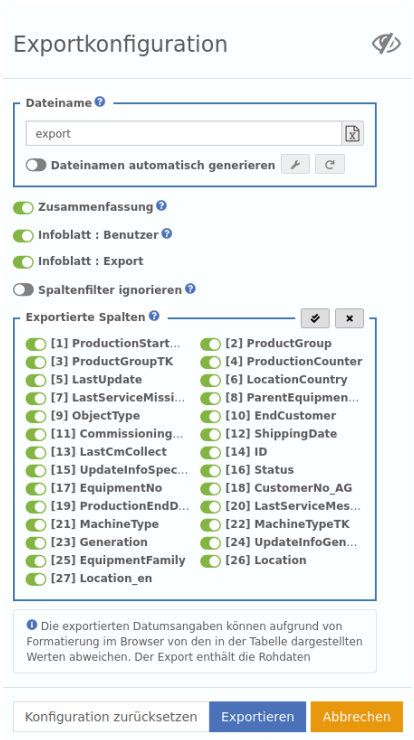
Beispiel

```
<analytics>
  <export>true</export>
  <chart>true</chart>
  <summary>true</summary>
  <search>always</search>
</analytics>
```

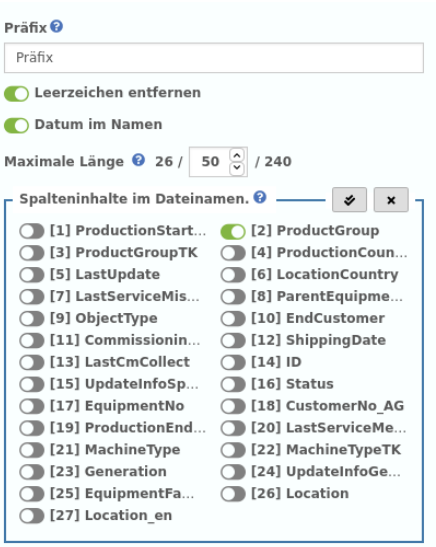
Funktion	Beschreibung
Export	Export der Tabelle nach XLSX
Chart	Diagramm auf Basis der Tabellendaten
Summary	Zusammenfassung pro Tabellenspalte
Search	Suchfunktion über die Komplette Tabelle

Export

YUNAML-Tag	Default	Beschreibung	Beispiel

analytics > **export**	true	Mit dem Export können die Inhalte ausgewählter Spalten über das Menü-Symbol des Tabellenwidget als Rohdaten exportiert werden.Wie die nachfolgende Abbildung zeigt, stehen beim Export verschiedene Konfigurationsmöglichkeiten zur Verfügung. Die Standardeinstellungen in diesem Dialog können durch eine erweiterte Definition in YUNAML (siehe Beispiel - Erweiterte Definition) einfach überschrieben und so im jeweiligen Kontext sinnvoll vorbelegt werden.
		
		Neben der Konfiguration der zu exportierenden Spalten ist es auch möglich den Dateinamen automatisch generieren zu lassen. Auch die dafür zur Verfügung stehenden Optionen, wie beispielsweise ein fester Präfix oder ein angehangener Zeitstempel, sind konfigurierbar.Die Optionen sind nachfolgend noch detailliert beschrieben (siehe **fileNameGenerator** Element).
		
		Der Export hat eine **einfache** und **erweiterte** Definition.
		In der einfachen Definition muss lediglich *true* oder *false* angegeben werden.
		In der erweiterten Definition ist es möglich die Optionen des Exports zu setzen und zu überschreiben.
		
		 
			Einfache Definition
			
			`<export>false</export>`
			
			Erweiterte Definition
			
			`<export>`
			`  <settingsId>export_dpe-1737</settingsId>`
			`  <fileName>dpe-1737-example</fileName>`
			`  <fileNameGenerator>`
			`  <active>true</active>`
			`  <removeWhitespace>true</removeWhitespace>`
			`  <appendTimestamp>true</appendTimestamp>`
			`  <maxLength>50</maxLength>`
			`  <prefix>export-dpe-1737</prefix>`
			`  <includeColumns>`
			`  <list>ProductGroup</list>`
			`  </includeColumns>`
			`  </fileNameGenerator>`
			`  <summary>true</summary>`
			`  <userInfo>true</userInfo>`
			`  <exportInfo>false</exportInfo>`
			`  <ignoreFilter>false</ignoreFilter>`
			`  <includeColumns>`
			`  <list>ProductGroup</list>`
			`  <list>Generation</list>`
			`  </includeColumns>`
			`  <excludeColumns>`
			`  <list>ProductionCounter</list>`
			`  </excludeColumns>`
			`</export>`

analytics > export > **settingsId**		Mithilfe einer *settingsId* werden die Einstellungen, die ein Benutzer über die Oberfläche am Export vornimmt, unter dieser *settingsId* gespeichert. Dadurch wird das Export-Fenster wieder mit den selben Einstellungen geöffnet, die der Benutzer zuvor vorgenommen hat.Die *settingsId* kann auch in anderen Widgets verwendet werden, wodurch der Export dieses Widgets auch mit diesen Einstellungen geöffnet wird.	`<settingsId>export_dpe-1737</settingsId>`
analytics > export > **fileName**	export	Der Name mit dem die Datei standardmäßig exportiert wird.	`<fileName>dpe-1737-example</fileName>`

analytics > export > **fileNameGenerator**		Stellt für diverse Einstellungen **Standardwerte** für den Dateinamensgenerator fest
		 
			`<fileNameGenerator>`
			`<active>true</active>`
			`<prefix>export-dpe-1737</prefix>`
			`<removeWhitespace>true</removeWhitespace>`
			`<appendTimestamp>true</appendTimestamp>`
			`<maxLength>50</maxLength>`
			`<includeColumns>`
			`<list>ProductGroup</list>`
			`</includeColumns>`
			`</fileNameGenerator>`

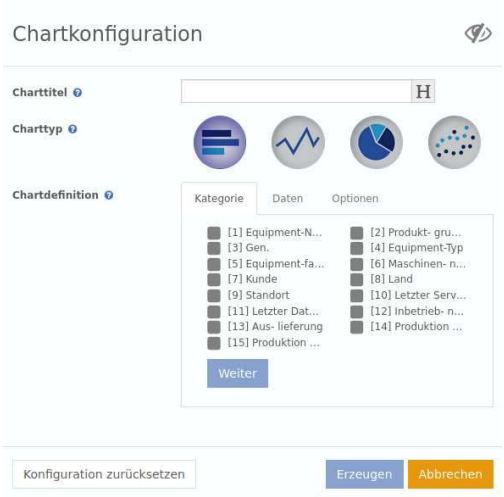
... > fileNameGenerator > **active**	false	Legt fest ob der Dateinamensgenerator aktiviert ist.	
... > fileNameGenerator > **prefix**		Definiert den Präfix des Dateinamensgenerator.	
... > fileNameGenerator > **removeWhitespace**	true	Leerzeichen werden aus dem generierten Dateinamen entfernt.	

... > fileNameGenerator > **appendTimestamp**	false	Ein Datum mit dem Format **YYYY-MM-DD_HH-mm-ss** wird an das Ende des Exportnamen geschrieben.	
... > fileNameGenerator > **maxLength**	240	Definiert die maximale Zeichenlänge des Dateinamens.Wird diese Länge überschritten wird der Dateiname abgeschnitten. ⚠️ Besonderheiten *appendTimestamp*: • Das Datum wird immer vollständig angehängt *includeColumns:* • Überschreitet der erste Wert / das erste Tupel die verfügbare Zeichenlänge, so wird nur ein Teil davon für den Dateinamen verwendet.Weitere Werte / Tupel werden nur für den Dateinamen verwendet, wenn sie die maximale Länge **nicht** überschreiten.	
... > fileNameGenerator > **includeColumns**		Definiert die standardmäßig ausgewählten Spalten für den Dateinamensgenerator. Werte aus der definierten Spalte werden in den Dateinamen geschrieben.Sind mehrere Spalten definiert, werden die entsprechenden Tupel in den Dateinamen geschrieben.	
analytics > export > **summary**	true	Ein Tabellenblatt mit einer Zusammenfassung aller Spalten wird im Export erzeugt.	`<summary>true</summary>`
analytics > export > **userInfo**	true	Ein Tabellenblatt mit dem Namen **Information** wird mit Informationen zu dem aktuell angemeldeten Nutzer im Export erzeugt.	`<userInfo>true</userInfo>`
analytics > export > **exportInfo**	true	Ein Tabellenblatt mit dem Namen **Information** wird mit Informationen zum Export erzeugt.	`<exportInfo>false</exportInfo>`
analytics > export > **ignoreFilter**	false	Die Spaltenfilter werden beim Export ignoriert.	`<ignoreFilter>false</ignoreFilter>`

analytics > export > **includeColumns**		Legt fest welche Spalten standardmäßig exportiert werden.
		
		Standardmäßig werden alle Spalten exportiert.
			`<includeColumns>`
			`  <list>ProductGroup</list>`
			`  <list>Generation</list>`
			`</includeColumns>`

analytics > export > **excludeColumns**		Legt fest welche Spalten **nicht** exportiert werden. Diese Einstellung kann vom Benutzer **nicht** geändert werden.	`<excludeColumns>`
			`  <list>`
			`ProductionCounter</list>`
			`</excludeColumns>`

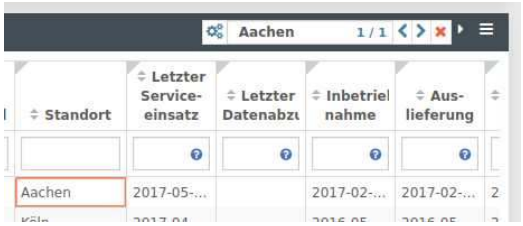
Chart

YUNAML-Tag	Default	Beschreibung	Beispiel
analytics > **chart**	false	Ist die Funktion *chart* aktiviert können die Daten aus der Tabelle in einem Diagramm dargestellt werden.Wird ein Diagramm erstellt, öffnet sich zunächst ein Fenster in dem das Diagramm konfiguriert werden muss. 	`<chart>true</chart>`

Summary

YUNAML-Tag	Default	Beschreibung	Beispiel
analytics > **summary**	false	Ist die Funktion *Summary* aktiviert wird eine Zusammenfassung einer Tabellenspalte zur Verfügung gestellt.Zu finden ist diese Zusammenfassung als graues Dreieck in der oberen linken Ecke einer Spalte. 	`<summary>true</summary>`

Search

YUNAML-Tag	mögliche Werte	Default	Beschreibung	Beispiel
analytics > **search**	true, false, always	true	Ist die Funktion *search* aktiviert kann über das Menü-Symbol eine Suchleiste zum Tabellenwidget hinzugefügt werden. Über dieses Suchfeld kann die gesamte Tabelle durchsucht werden. Werden Ergebnisse gefunden, wird die entsprechende Stelle markiert.  Mit dem Wert "always" wird die Suche dauerhaft angezeigt.	`<search>always</search>`

Erweiterte Eigenschaften

Individuelle Spalten

Mithilfe des YUNAML-Tags *cols* können individell konfigurierte Spalten erzeugt werden.

Um die Spalten individuell konfigurieren zu können, muss im YUNAML-Code der Tabelle zunächst ein Objekt *cols* erzeugt werden. Dieses Objekt enthält arrays, die für die einzelnen Spalten stehen. Innerhalb der arrays können Eigenschaften definiert werden, die sich anschließend als Eigenschaften der Tabellenspalten äußern (Spaltenbreite, Spaltentitel, Filter,...).

YUNAML-Tags für cols

<table>
<tr><th>Feld</th><th>Mögliche Werte</th><th>Beschreibung</th><th>Beispiel</th></tr>
<tr><td>cols > list > field</td><td>Spaltenname</td><td>Verweis auf die darzustellende Spalte aus der Tabelle oder der Data ID.</td><td><pre><field>DatabaseField</field></pre></td></tr>
<tr><td>cols > list > title</td><td>Individueller Titel</td><td>Anzeigename für die Spalte.<table><tr><th>Spezielle Zeichen</th><th>Erläuterung</th></tr><tr><td>\n</td><td>Fügt einen Zeilenumbruch an der Stelle des Textes ein</td></tr><tr><td>\+</td><td>formatiert den nachfolgenden Text als Subtitle (kleinere Schriftart)</td></tr><tr><td>\n\+</td><td>Fügt einen Zeilenumbruch ein und definiert den nachfolgenden Text als Subtitle</td></tr></table></td><td><pre><title>MyFieldName</title></pre></td></tr>
<tr><td>cols > list > show</td><td>true, false</td><td>Standardmäßig true

Spalte kann ein- oder ausgeblendet werden.</td><td><pre><show>true</show></pre></td></tr>
<tr><td>cols > list > sortable</td><td>Spaltenname</td><td>Name der Spalte auf dessen Basis diese Spalte sortiert werden kann.</td><td><pre><sortable>DatabaseField</sortable></pre></td></tr>
<tr><td>cols > list > copy</td><td>Boolean / YUNAML</td><td>Ein Button zum Kopieren der Daten wird an der Spalte dargestellt.Wird dieser betätigt, werden sämtliche Daten aus dieser Spalte in der Zwischenablage abgelegt.

Ist copy auf true bzw. der Parameter unique auf true gesetzt werden doppelte Werte nicht mehrfach in der Zwischenablage gespeichert.

 In manchen Browsern steht die Kopierfunktion nur unter HTTPS zur Verfügung. Steht die Kopierfunktion nicht zur Verfügung, wird das Icon nicht angezeigt.</td><td>Einfache Definition<pre><copy>true</copy></pre>Erweiterte Definition<pre><copy>
    <unique>
true</unique>
</copy></pre></td></tr>
</table>

Feld	Mögliche Werte	Beschreibung	Beispiel
cols > list > **filter**	<Spaltenname>typ</Spaltenname> **Mögliche Typen:**number,text,select-single-custom,select-multiple-custom,number-custom,date-custom	Mithilfe von dem YUNAML-Tag *filter* kann für eine Spalte ein Filter definiert werden.Dafür muss, analog zu YUNAML-Tags, der Name der Spalte innerhalb von spitzen Klammern stehen. Innerhalb dieses Tags muss dann nurnoch der Typ eingetragen werden. Weitere Informationen: FilterTypen.	`<filter> <MyFieldName> text</MyFieldName> </filter>`
cols > list > **type**	msgicon,image,genLink,genDate,flag,highlight,null	Der Typ der Spalte bestimmt in wie der Inhalt dargestellt werden soll.In einem anderen Artikel gehen wir auf die verschiedenen Spalten-Typen detailliert ein.	`<type>genLink</type>`
cols > list > **link**	YUNAML	Ist der *type* **genLink** kann mit dem YUNAML-Tag *link* die Funktion und Darstellung des Links angepasst werden. Weitere Informationen: Links.	`<link> <url>mailto:foo@bar.de </url> <extern> false</extern> <label> LinkLabel</label> </link>`

Feld	Mögliche Werte	Beschreibung	Beispiel
cols > list > **width**	Integer	Legt die Breite der Spalte in Pixel fest. *Sonderfälle bei der Definition der Spaltenbreite* Ist die Summe aller Spaltenbreiten **kleiner** als die Breite der Tabelle, werden die Spaltenbreiten hoch skaliert bis die ganze Tabelle ausgefüllt ist. Ist die Summe der definierten Spaltenbreiten **größer** als die Breite der Tabelle, wird eine horizontale Scrollbar angezeigt. Wurde nur für ein Teil der Spalten eine Breite definiert, so werden zunächst die Spalten mit definierter Breite dargestellt. Der restliche Platz wird gleichmäßig auf alle Spalten aufgeteilt.Übersteigt auch hier die Summe der Spaltenbreiten die Breite der Tabelle, so wird die Spaltenbreite der Spalten deren breite nicht definiert wurde auf die benötigte breite der Kopfzeile gesetzt.	`<width>300</width>`

cols > list > **style**	CSS in form von YUNA-ML	Formatiert die Spaltenüberschrift.Überschreibt die bei *generalOptions* gesetzten Eigenschaften.	`<style>`
			`<color>`
			`green</color>`
			`</style>`

Feld	Mögliche Werte	Beschreibung	Beispiel
cols > list > **cell**	YUNAML	Bietet die Möglichkeit einzelne Zellen zu formatieren. Weiter Informationen: Formatierung einzelner Zellen.	<cell> <truncate> true</truncate> <conditions> <list> <value> <list>1</list> <list>2</list> </value> <field>Category_ID</fi eld <color>Tomato</color> </list> </conditions> </cell>

YUNAML

```
<widget name="tbl_InstBasis_Overview">
    <widgettype>tabledirective</widgettype>
    <position>
        <x>5.5</x>
        <y>3</y>
    </position>
    <size>
        <x>13</x>
        <y>5</y>
    </size>
    <dataID>qy_InstBasis_Overview</dataID>
    <triggerParams>
        <mandatory>
            <list>filter2</list>
        </mandatory>
    </triggerParams>
    <sorting>EquipmentNo</sorting>
    <sortingorder>asc</sortingorder>
    <analytics>
        <summary>true</summary>
        <chart>true</chart>
        <export>true</export>
        <search>true</search>
    </analytics>
    <cols>
        <list>
            <field>EquipmentNo</field>
```

```
            <title>Item</title>
            <sortable>EquipmentNo</sortable>
            <filter>
                <EquipmentNo>text</EquipmentNo>
            </filter>
            <show>true</show>
            <width>100</width>
            <style></style>
        </list>
        <list>
            <field>ProductGroup</field>
            <title><![CDATA[Produkt-<br>gruppe]]></title>
            <sortable>ProductGroup</sortable>
            <copy>true</copy>
            <filter>
                <ProductGroup>text</ProductGroup>
            </filter>
            <show>true</show>
            <search>ProductGroup</search>
            <width>100</width>
            <style></style>
        </list>
        <list>
            <field>OperatingHoursIemOn</field>
            <title>ItemEin\n[h]</title>
            <sortable>OperatingHoursItemOn</sortable>
            <type>number</type>
            <show>true</show>
            <width>90</width>
            <style></style>
            <filter>
                <OperatingHoursItemOn>number-custom</OperatingHoursItemOn>
            </filter>
        </list>
        <!-- Weitere Listenelemente für neue Spalten hinzufügen -->
    </cols>
</widget>
```

Filtertypen

Typ	Beschreibung	Beispiel
text	Das Feld zum filtern der Tabelle lässt die Eingabe diverser Zeichen zu.	`<MyFieldName>text</MyFieldName>`
number	Das Feld zum filtern der Tabelle lässt nur Zahlen zu. Das Dezimaltrennzeichen kann je nach Browser variieren.	`<MyFieldName>number</MyFieldName>`

Typ	Beschreibung	Beispiel
select-single-custom	Es ist möglich **einen** Wert zum filtern der Tabelle aus einer vordefinierten Dropdown-Liste auszuwählen.  Weitere Informationen: Definition der Dropdownliste.	`<MyFieldName>select-single-custom</MyFieldName>`
select-multiple-custom	Es ist möglich **mehrere** Werte zum filtern der Tabelle aus einer vordefinierten Dropdown-Liste auszuwählen.  Weitere Informationen: Definition der Dropdownliste.	`<MyFieldName>select-multiple-custom</MyFieldName>`
date-custom	Das Feld zum filtern der Tabelle folgt einer gewissen Syntax, wodurch es möglich ist ein Datumsfeld spezifischer zu filtern. Die Syntax dazu kann über das ? Symbol in der entsprechenden Spalte abgerufen werden. 	`<MyFieldName>date-custom</MyFieldName>`

Typ	Beschreibung	Beispiel
number-custom	Das Feld zum filtern der Tabelle folgt einer gewissen Syntax, wodurch es möglich ist ein numerisches Feld spezifischer zu filtern. Die Syntax dazu kann über das ? Symbol in der entsprechenden Spalte abgerufen werden. 	`<MyFieldName>number-custom</MyFieldName>`

Definition der Dropdown-Liste

YUNAML-TAG	Mögliche Werte	Beschreibung
filterDisplay	icon, lbl, both	**Wert** / **Beschreibung** icon: In der Dropdown-Liste erscheint nur das im YUNAML-Tag *filterData* definierte Symbol.  lbl: In der Dropdown-Liste erscheint nur das im YUNAML-Tag *filterData* defnierte label.  both: In der Dropdown-Liste erscheint sowohl das im YUNAML-Tag *filterData* definierte icon, als auch das label. 
filterData > **list**		Mit dem YUNAML-Tag *list* werden die einzelnen Einträge der Dropdown-Liste konfiguriert.
filterData > list > **value**		Mit *value* kann man den Wert für den Eintrag in der Dropdown-Liste festlegen.Dadurch ist es möglich fest definierte Werte zum Filtern einer Spalte zu verwenden. Dieser Wert wird nicht in der Dropdown-Liste angezeigt.
filterData > list > **label**		Mit *label* kann man die Beschriftung des Eintrags in der Dropdown-Liste festlegen.

YUNAML-TAG	Mögliche Werte	Beschreibung
filterData > list > **icon_type**	fa	Die Symbole von FontAwesome können verwendet werden.
filterData > list > **icon_color**		Es ist möglich eine eigene Farbe für das Symbol zu definieren. Erlaubt sind alle Farben oder Farbdefinitionen, die auch in CSS zugelassen sind. Zum Beispiel **blue** oder **#426bf4**
filterData > list > **icon**		Die Symbole von FontAwesome können hier referenziert werden.Dafür muss der Name des FontAwesome-Symbols angegeben werden.Wenn zum Beispiel die Klasse des Symbols **fas fa-cloud** ist, muss im YUNAML-Tag *icon* **fa-cloud** angegeben werden. Hier gehts zu FontAwesome.

Formatierung einzelner Zellen

Über das Feld *cell* kann auf die Formatierung einzelner Zellen Einfluss genommen werden.

Felder zur Zellenformatierung in *cell*

Feld	Mögliche Werte	Default	Beschreibung	Beispiel
truncate	true, false	true	**Wert** / **Beschreibung** true: Inhalt, der für die Zelle zu lang ist, wird gekürzt und mit "..." dargestellt. Befindet sich der Mauszeiger über der Zelle, wird der Inhalt vollständig dargestellt. false: Der Inhalt der Spalte wird immer vollständig angezeigt.	`<truncate>true</truncate>`

Feld	Mögliche Werte	Default	Beschreibung	Beispiel
conditions	YUNA-ML		Es ist möglich, abhängig von einer Bedingung, die Darstellung einer Zelle zu überschreiben.Trifft eine Bedingung zu kann zum Beispiel die Zelle grün gefärbt werden. Weitere Informationen: Bedingungsabhängige Formatierung.	`<conditions><list><value><list>Störung</list><list>Nicht vorhanden</list></value><field>Category_ID</field><color>Tomato</color></list></conditions>`
width	Integer		Legt die Breite der Spalte in Pixel fest. *Sonderfälle bei der Definition der Spaltenbreite* Ist die Summe aller Spaltenbreiten **kleiner** als die Breite der Tabelle, werden die Spaltenbreiten hoch skaliert bis die ganze Tabelle ausgefüllt ist. Ist die Summe der definierten Spaltenbreiten **größer** als die Breite der Tabelle, wird eine horizontale Scrollbar angezeigt. Wurde nur für ein Teil der Spalten eine Breite definiert, so werden zunächst die Spalten mit definierter Breite dargestellt. Der restliche Platz wird gleichmäßig auf alle Spalten aufgeteilt.Übersteigt die Summe der Spaltenbreiten die Breite der Tabelle, so wird die Spaltenbreite der Spalten deren breite nicht definiert wurde auf die benötigte breite der Kopfzeile gesetzt.	`<width>300</width>`

Bedingungsabhängige Formatierung

Es können mehrere bedingungsabhängige Formatierungen für eine Zelle konfiguriert werden.Dabei wird jeder *conditions* block nacheinander geprüft. Die Formatierung der letzten zutreffende Bedingung wird angewendet.Sind mehrere Bedingungen in einem Listenelement von *conditions* definiert, werden diese Bedingungen mit **oder** verknüpft.

Feld	Mögliche Werte	default	Beschreibung	Beispiel

conditions > list > **value**	Zahl,Text,Datum,
	
	min,max,null,notNull
			Wert mit denen der Zellinhalt unter Verwendung des Operators verglichen wird.
			
			Es ist auch möglich mehrere Werte für *value* zu definieren.Dafür muss innerhalb des *value* -tags ein *list*-tag für jeden Wert definiert werden.
			
			Siehe Beispiel für mehrere Werte.
				Ein Wert
				`<value>1</value>`
				
				Mehrere Werte
				`<value><list>1</list><list>2</list></value>`

...> value > **min** ...> value > **max**	Zahl oder Datum		Minimalwert für Bereichsvergleiche. Maximalwert für Bereichsvergleiche.	`<value><min>1</min><max>8</max></value>`
... > value > **null**	true | false	false	Wenn **true** wird die Formatierung angewendet wenn der Zellinhalt null ist.	`<null>true</null>`
... > value > **notNull**	true | false	false	Wenn **true** wird die Formatierung angewendet wenn der Zellinhalt nicht null ist.	`<notNull>true</notNull>`

<table>
<tr><th>Feld</th><th>Mögliche Werte</th><th>default</th><th>Beschreibung</th><th>Beispiel</th></tr>
<tr>
<td>conditions > list > operator</td>
<td>
<table>
<tr><th>Operator</th><th>Kürzel</th><th>Beschreibung</th></tr>
<tr><td>==</td><td>eq</td><td>gleich</td></tr>
<tr><td>!=</td><td>neq</td><td>ungleich</td></tr>
<tr><td>>=</td><td>gte</td><td>größer als oder gleich</td></tr>
<tr><td>></td><td>gt</td><td>größer als</td></tr>
<tr><td>⇐</td><td>lte</td><td>kleiner als oder gleich</td></tr>
<tr><td><</td><td>lt</td><td>kleiner als</td></tr>
</table>
</td>
<td>==</td>
<td>Vergleichsoperator für den Vergleich der Werte in value und der Werte in der Zelle.

Falls die Nutzung der Symbole zu Problemen führt, kann stattdessen das Kürzel verwendet werden.</td>
<td><operator>!=</operator></td>
</tr>
<tr>
<td>conditions > list > field</td>
<td>Namen von Feldern der Tabelle</td>
<td>self</td>
<td>Mit field kann definiert werden, dass der Wert für die Bedingung aus einer anderen Splate genommen wird.</td>
<td><field>MyColumn3</field></td>
</tr>
<tr>
<td>conditions > list > color</td>
<td>CSS-Farbangabe</td>
<td>lightblue</td>
<td>Farbe des Zellhintergrunds, wenn Bedingung erfüllt ist.</td>
<td><color>#ffff68f</color></td>
</tr>
<tr>
<td>conditions > list > icon</td>
<td>FontAwesome-Icons</td>
<td></td>
<td>Das Icon, das statt dem Zellinhalt angezeigt wird, wenn die Bedingung zutrifft.

Dafür muss der Name des FontAwesome-Symbols angegeben werden.Wenn zum Beispiel die Klasse des Symbols fas fa-cloud ist, muss im YUNAML-Tag icon fa-cloud angegeben werden.

Hier gehts zu FontAwesome.</td>
<td><icon>fa-cloud</icon></td>
</tr>
</table>

Beispiel für die Nutzung der Möglichkeiten für cell

Im folgenden Beispiel wird eine Zelle in der Spalte *CategoryName* in der Farbe Tomato eingefärbt, wenn der Wert in der Spalte *Category_ID* 1 oder 2 ist.

XML-Code der cell-Definition einer Tabellenzelle

```
<cols>
```

```
    <list>
        <field>CategoryName</field>
        <title>Kategorie</title>
        <show>true</show>
        <cell>
            <truncate>true</truncate>
            <conditions>
                <list>
                    <value>
                        <list>1</list>
                        <list>2</list>
                    </value>
                    <field>Category_ID</field>
                    <color>Tomato</color><!-- Red -->
                </list>
                <list>
                    <value>
                        <list>3</list>
                        <list>4</list>
                        <list>5</list>
                        <list>6</list>
                    </value>
                    <field>Category_ID</field>
                    <color>#fff68f</color><!-- Yellow -->
                </list>
                <list>
                    <value>
                        <list>7</list>
                        <list>8</list>
                    </value>
                    <field>Category_ID</field>
                    <color>rgb(255,92,66)</color>
                </list>
            </conditions>
        </cell>
        <width>300</width>
    </list>
</cols>
```

Filter an Query anhängen

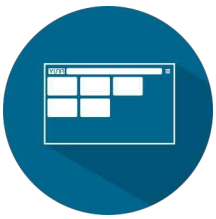
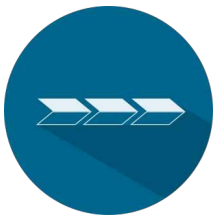
Falls die Spalten, auf die der Filter wirkt, nicht vorhanden sind, wird die Basistabelle des Filters (z.B. die Installierte Basis) über die definierten Felder mit der Tabelle, auf der gefiltert werden soll, gejoined:

	ID	Query	Filter	QueryTablePath
1	10001	<QueryBuilder> <select> <table>CM-DataDB<...	1	CM-DataDB data DeviceMessage
2	10002	<QueryBuilder> <select> <table>CM-Data...	0	NULL
3	10003	<QueryBuilder> <select> <table>CM-DataDB<...	0	NULL
4	10004	<QueryBuilder> <select> <table>CM-DataDB</...	0	NULL
5	10005	<QueryBuilder> <select> <table>CM-DataDB<...	0	NULL
6	10006	<QueryBuilder> <select> <table>CM-DataDB<...	1	CM-DataDB tls170-ImportData vwInstBasis
7	10007	<QueryBuilder> <select> <table>CM-PortalDB...	0	NULL

1. **Eintrag in [PortalDB].[sp].[Query]** mit
   a. Filter = 1
   b. QueryTablePath = Tabelle auf die der angehangene Filter wirken soll
2. **Eintrag in [PortalDB].[sp].[FilterAssoc]**
   a. Query_ID = ID der Query, an die ein Filter angehangen werden soll
   b. Filter_ID = ID des Filters, der an die Query angehangen werden soll
   c. FilterField = Feld des Filters für den JOIN
   d. QueryResultField = Feld des Querys für den JOIN

5.6. Views

In den folgenden Abschnitten stellen wir Ihnen die Views vor, die Sie im YUNA Portal verwenden können.

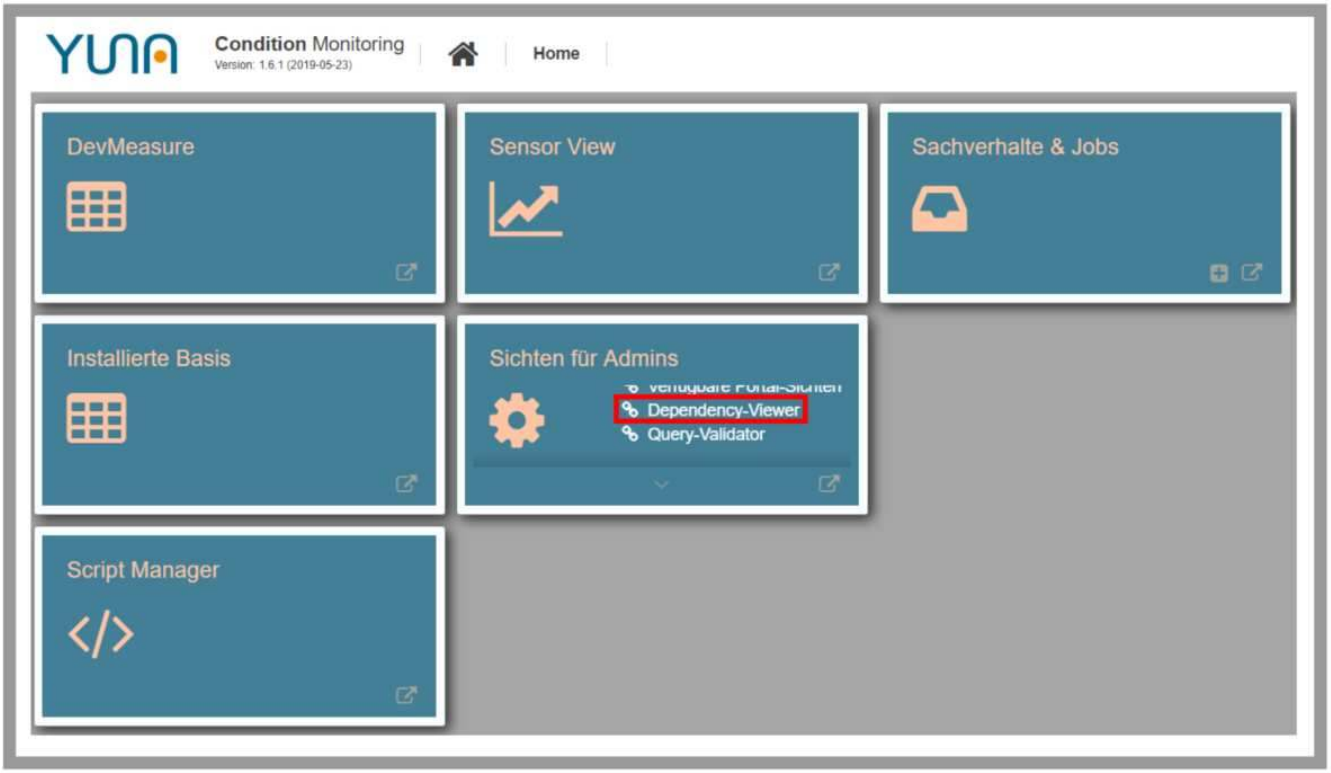
					
Landing Page	Portal Views	Issue Manager	Systemübersicht	Installierte Basis	Themes Manager
					
Filterverwaltung	Dependency Viewer	Job Manager	Query Validator		

5.6.1. Dependency Viewer (dependencyDirective)

Der Dependency Viewer ist eine View, die Systemadministratoren des YUNA Portals zur Verfügung gestellt werden kann. In ihr können sich Admins und/oder Dashboard Developer über Abhängigkeiten zwischen Widgets und Views informieren.

Zu Dependency Viewer navigieren

 Klicken sie im Dashboard auf Dependency-Viewer



Dependency Viewer - Übersicht


```
<xml>
  <widget name="template_widget_Dependency">
    <!-- Position from left top  -->
    <position>
        <!-- y -->
        <y>0</y>
        <!-- x -->
        <x>0</x>
    </position>
```

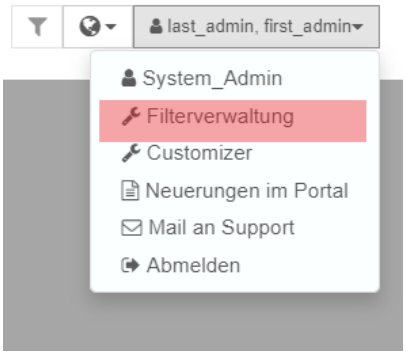
```
    <!-- Size of the Widget -->
    <size>
      <x>15</x>
      <y>6.5</y>
    </size>
    <!-- Caption: title over the widget and additional features (Contextmenue..) -->
    <caption>
        <!-- Display caption -->
        <show>true</show>
        <!-- Title of the Widget -->
        <label>Dependency-Viewer-Template</label>
    </caption>
    <!-- Name of the WidgetType -->
    <widgettype>dependencydirective</widgettype>
    <!-- No URL-Paramters needed -->
    <triggerParams/>
    <!-- Query to execute -->
    <dataID>qy_NameOfDataID</dataID>
  </widget>
</xml>
```

5.6.2. Filterverwaltung

In der Filterverwaltung des YUNA Portals können Benutzer eigene (Primär-)Filter erstellen und unter eigens gewählten Namen und einer Beschreibung speichern. Zudem kann ein Benutzer in der Filterverwaltung auswählen, ob er den erstellten Filter global speichern und somit allen angemeldeten Benutzern des YUNA Portals zur Verfügung stellen möchte, oder ob er den Filter als eigenen Filter speichern möchte, der anschließend nur für seinen eigenen Benutzeraccount sichtbar ist.

Weiterhin können Benutzer des Portals bestehende Filter suchen und auswählen. Hierbei gibt es die Möglichkeit zur Auswahl von globalen Filtern, die allen Nutzern des YUNA Portals zur Verfügung gestellt werden und lokalen Filtern, die nur vom Benutzer selbst sicht- und nutzbar sind.

Über das Dropdown Menü können Sie die **Filterverwaltung** öffnen:



Filterverwaltung

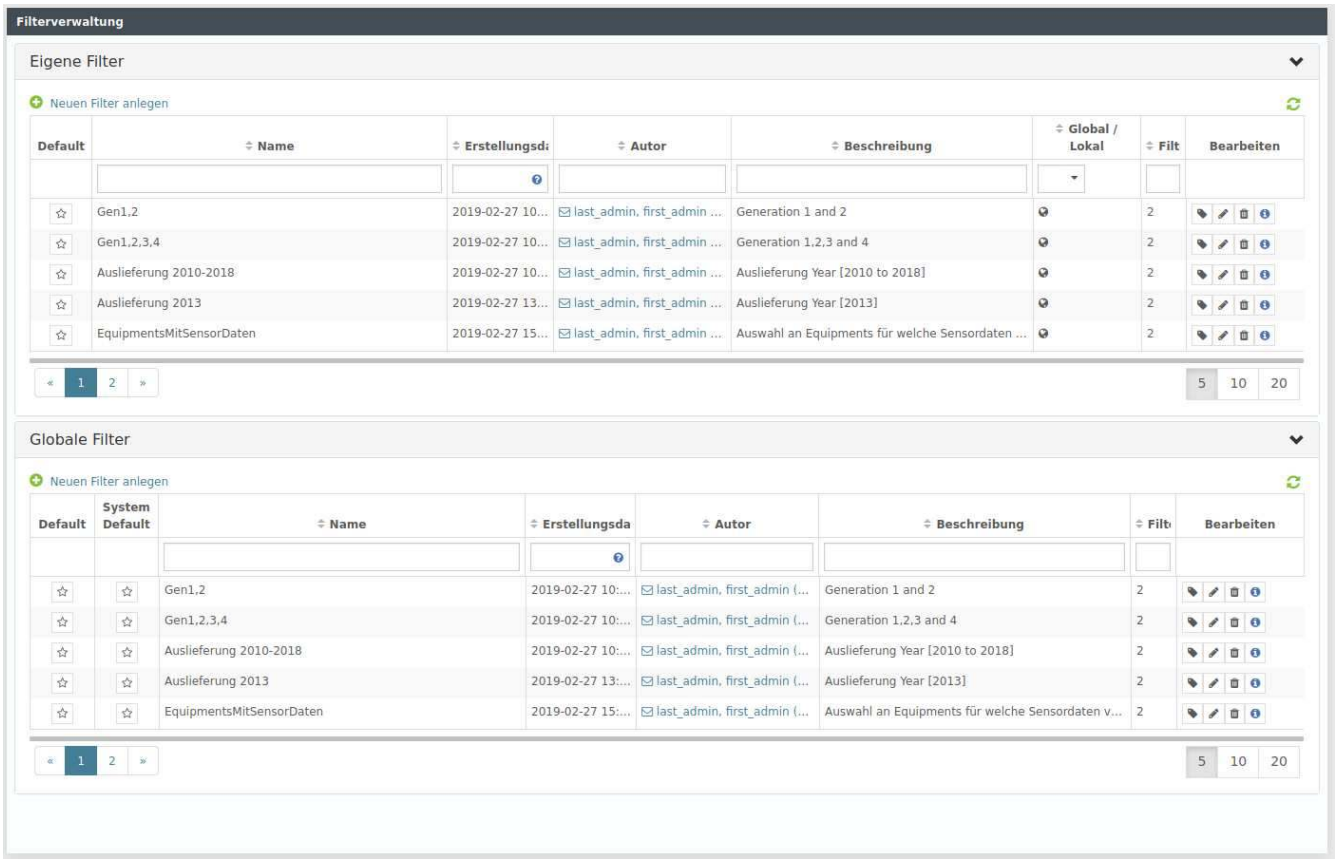
Filter werden in der Filterverwaltung aufgelistet und können bearbeitet, gelöscht oder als Standard gesetzt werden.

Zudem können die gelisteten Filter als Standardfilter für die eigene Ansicht der Daten im Portal gesetzt werden.

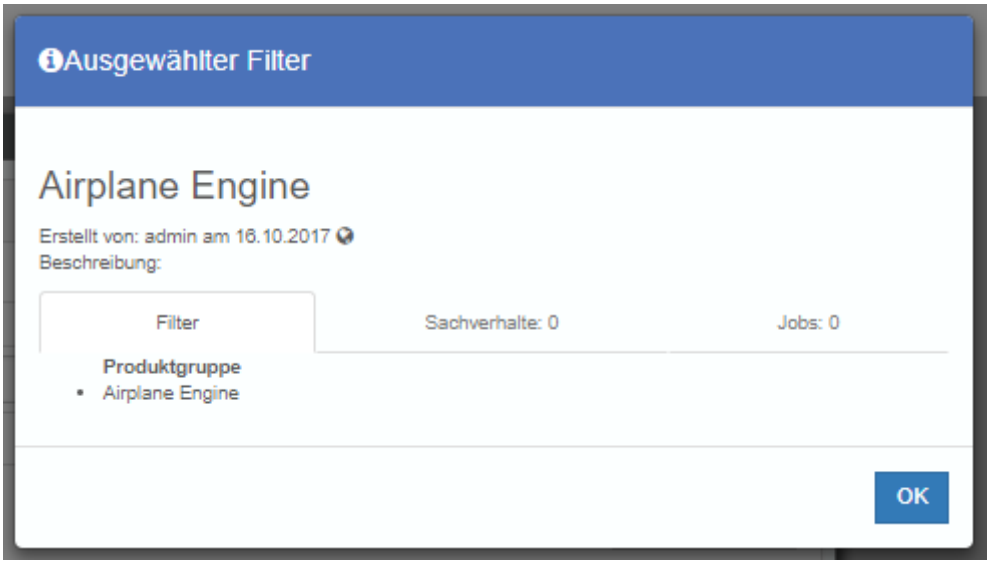
<table>
<tr><th>Feld</th><th>Bedeutung</th></tr>
<tr><td>Default</td><td>Wenn gesetzt, wird dieser Filter immer geladen. Dadurch muss er nicht mehr explizit ausgewählt werden.</td></tr>
<tr><td>System Default</td><td>Wenn gesetzt, wird dieser Filter für jeden Benutzer immer geladen.</td></tr>
<tr><td>Name</td><td>Der Name des Filters.</td></tr>
<tr><td>Erstellungsdatum</td><td>Das Datum, an dem der Filter erstellt wurde.</td></tr>
<tr><td>Autor</td><td>Der Benutzer, der den Filter angelegt hat.</td></tr>
<tr><td>Beschreibung</td><td>Eine ausführliche Beschreibung des Filters</td></tr>
<tr><td>Global/Lokal</td><td>Eine symbolische Darstellung, ob ein Filter global oder lokal ist</td></tr>
<tr><td>FilterID</td><td>Die ID des Filters.

Jedem Filter ist einer FilterID zugeordnet. Komponenten, wie z.B. der Issue-Workflow oder das Filterwidget, verwenden immer die Filter einer bestimmten FilterID. Die FilterID eines Filters kann nicht verändert werden, da über sie festgelegt ist, auf welche Datengrundlage dieser Filter angewendet wird.
In der URL vieler Dashboards taucht die FilterID als URL-Parameter auf und sieht beispielsweise folgendermaßen aus:...?
filter2=07e6a59b4319f9a68729289851f955eb03be13fa3f4b86920a093a9902da1301</td></tr>
<tr><td>Bearbeiten</td><td>
<table>
<tr><th>Symbol</th><th>Erklärung</th></tr>
<tr><td>  </td><td>Metadaten bearbeiten: Mit Betätigung des Buttons können Daten wie Name, Beschreibung oder Besitzer geändert. Siehe: Aktive Filter, Filter speichern unter</td></tr>
<tr><td>  </td><td>Filter bearbeiten: Mit Betätigung des Buttons kann bearbeitet werden, auf welche Daten der Filter angewandt wird</td></tr>
<tr><td>  </td><td>Filter löschen: Mit Betätigung des Buttons kann der Filter gelöscht werden.</td></tr>
<tr><td>  </td><td>Filter Info: Mit Betätigung des Buttons können weitere Informationen über den Filter angezeigt werden.</td></tr>
</table>
</td></tr>
</table>

Screenshot aus der Software. Die einzelnen Felder werden in der Tabelle oben beschrieben



Bei Auswahl eines Filters werden zusätzliche Informationen zum Filter angezeigt, sowie die Information ausgegeben, wie viele Sachverhalte und Jobs auf den Filter referenziert wurden.



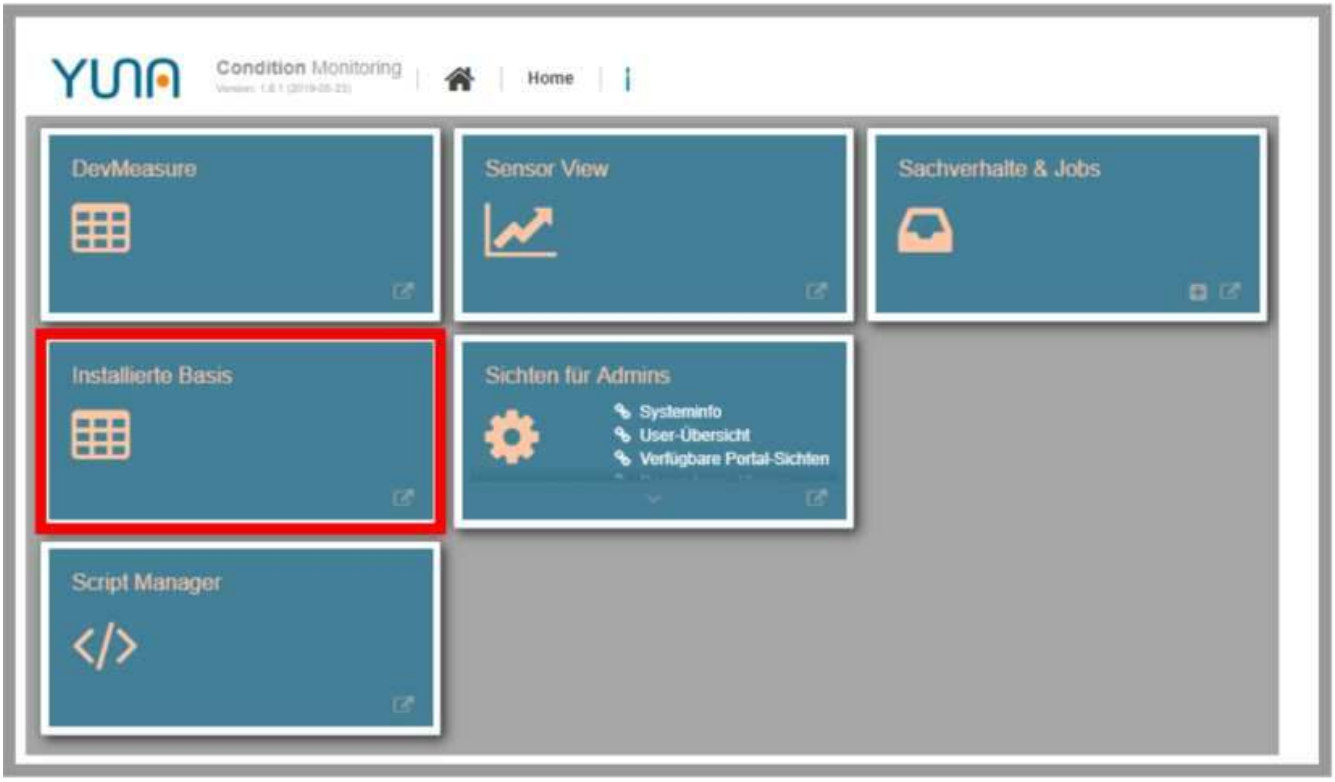
Zum Thema "Filter referenzieren und kopieren" finden sich weitere Infos im Dokumentationsteil zu <<filterfunktionen, Filterfunktionen.

5.6.3. Installierte Basis

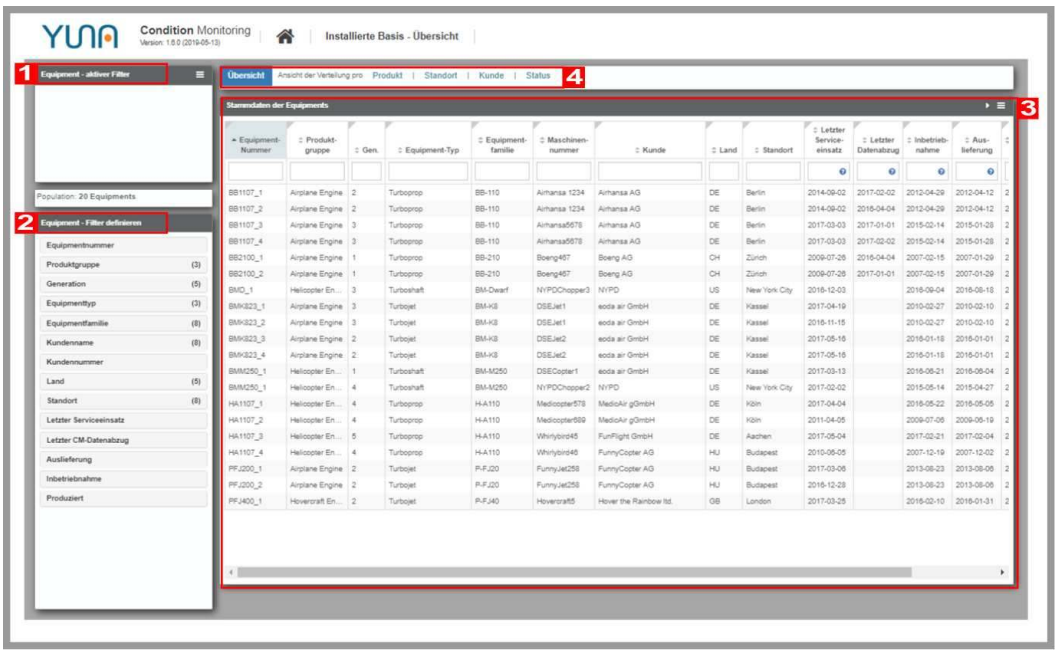
Die Installierte Basis ist eine Portal View des YUNA Portals, in der Informationen zu den Stammdaten im System hinterlegt sind, die in verschiedenen Widgets aufgeführt werden. Diese Informationen lassen sich in den Widgets filtern, durchsuchen, sortieren und grafisch darstellen.

Zu Installierte Basis navigieren

Klicken sie im Dashboard auf Installierte Basis



Installierte Basis - Übersicht



1 Aktiver Filter

2 Filter definieren

3 Tabellen-Widget

4 Html-Widget

Aktiver Filter

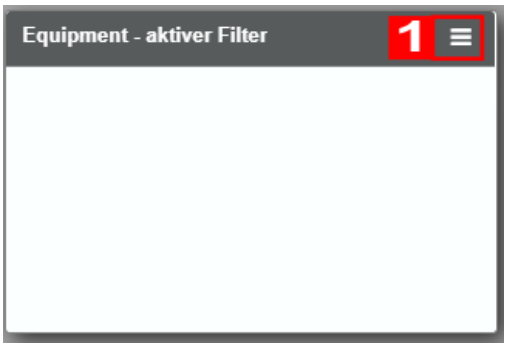
Hier kann ein aktiver Filter für die Stammdaten gespeichert, ausgewählt oder neu angelegt werden.

Ein aktiver Filter dient dazu, eine eingeschränkte Auswahl der Stammdaten im Tabellen Widget

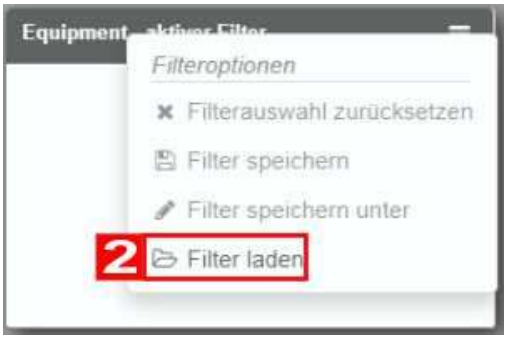
anhand definierter Kriterien darzustellen.

Filter laden

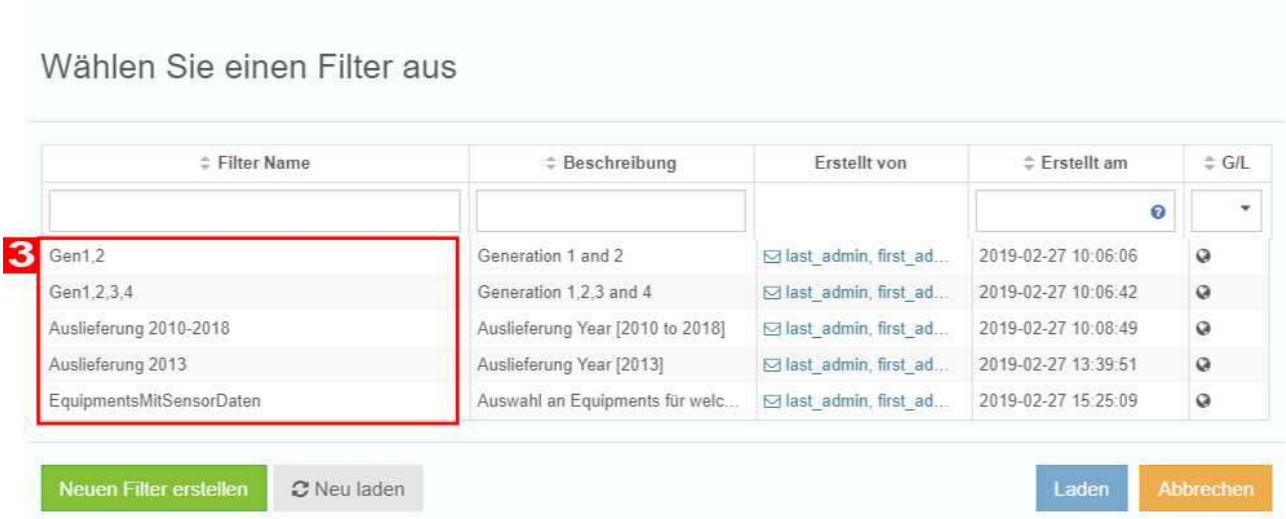
1. **Auf das Menüsymbol für aktive Filter klicken**



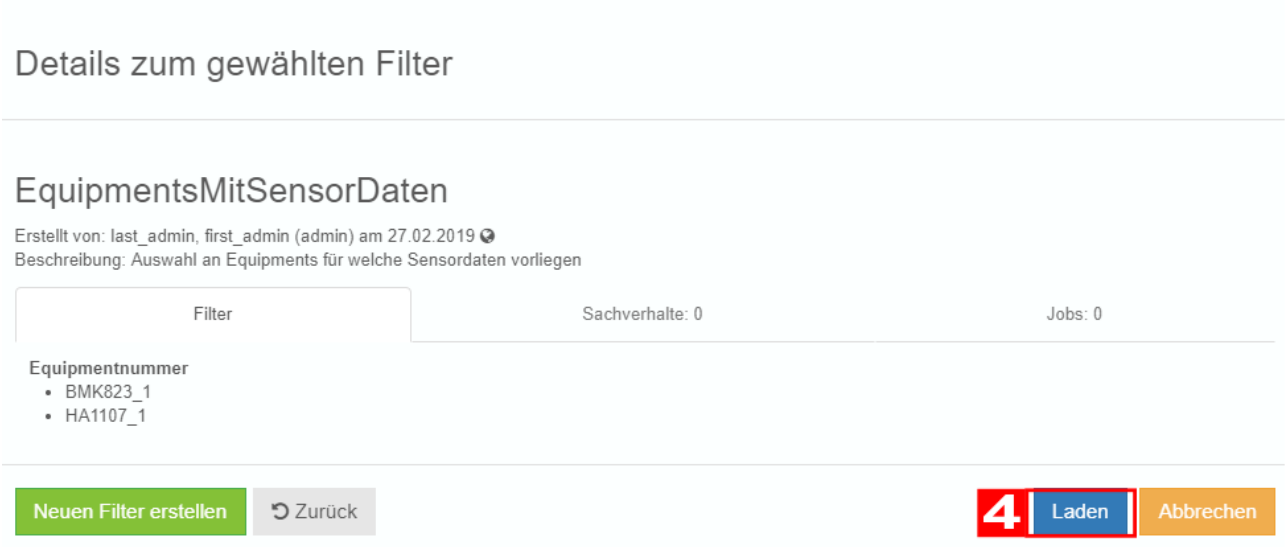
2. **Auf Filter laden klicken**



3. **Auf den zu ladenden Filter klicken**

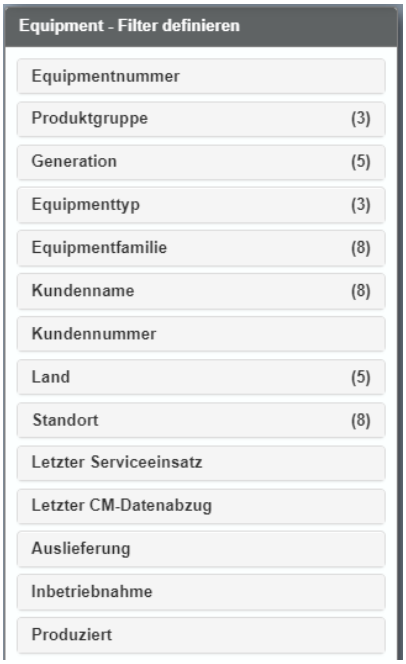


4. **Auf Laden klicken**



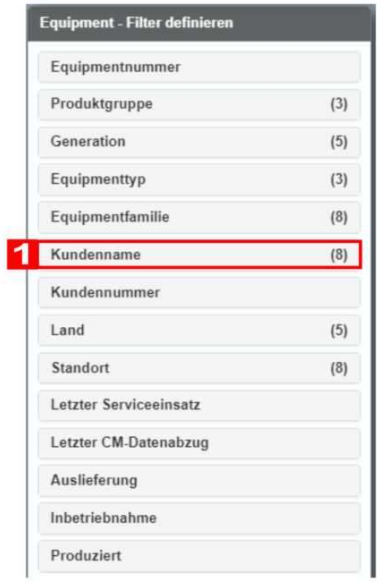
Filter definieren

Hier kann ein aktiver Filter definiert werden.

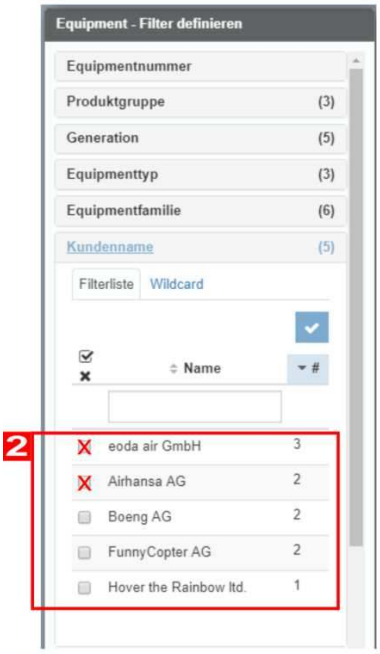


Beispiel: Filter für Kundennamen definieren

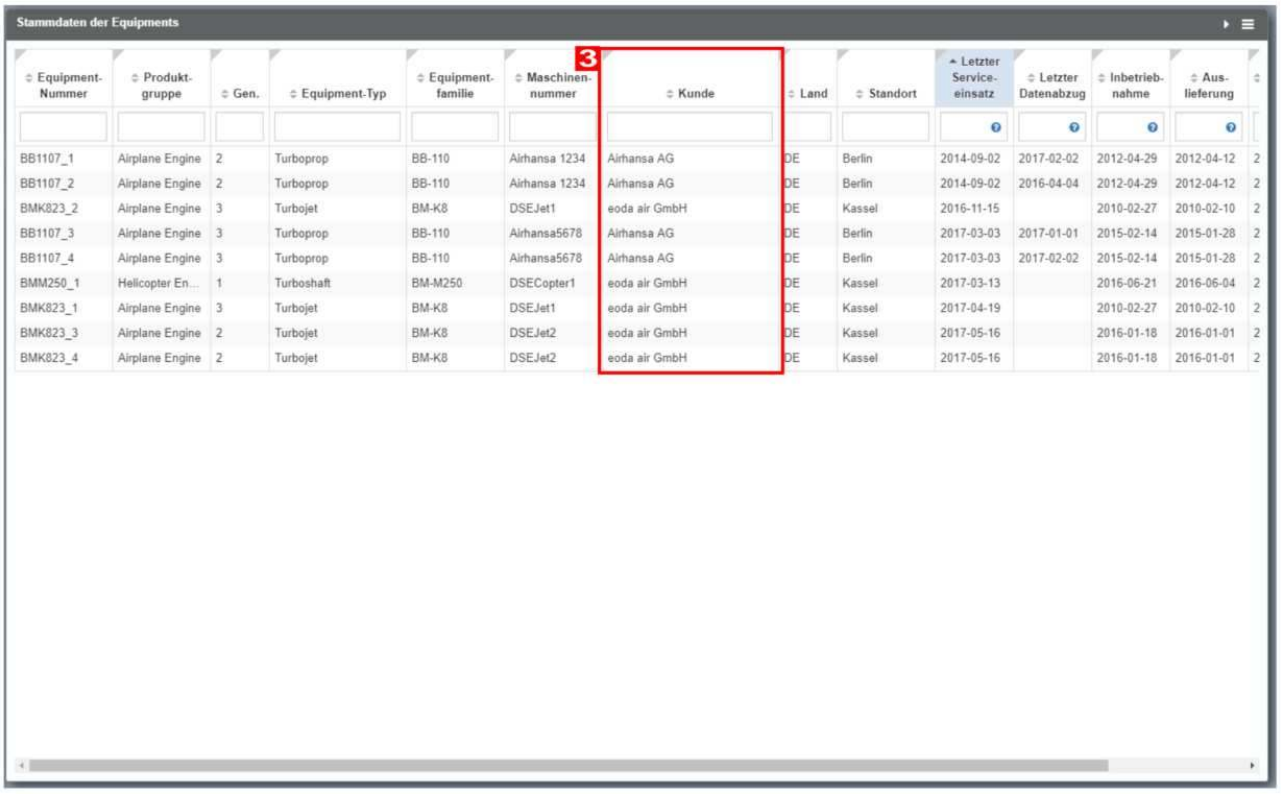
1. **Auf Kundenname klicken**



2. **Kunden auswählen nach denen gefiltert werden soll**



3. **Ergebnis : Im Tabellen Widget werden nur Daten zu den ausgewählten Kunden angezeigt**

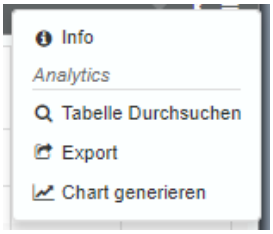


Equipment-Nummer	Produktgruppe	Gen.	Equipment-Typ	Equipmentfamilie	Maschinennummer	Kunde	Land	Standort	Letzter Serviceeinsatz	Letzter Datenabzug	Inbetriebnahme	Auslieferung
BB1107_1	Airplane Engine	2	Turboprop	BB-110	Airhansa 1234	Airhansa AG	DE	Berlin	2014-09-02	2017-02-02	2012-04-29	2012-04-12
BB1107_2	Airplane Engine	2	Turboprop	BB-110	Airhansa 1234	Airhansa AG	DE	Berlin	2014-09-02	2016-04-04	2012-04-29	2012-04-12
BMK823_2	Airplane Engine	3	Turbojet	BM-K8	DSEJet1	eoda air GmbH	DE	Kassel	2016-11-15		2010-02-27	2010-02-10
BB1107_3	Airplane Engine	3	Turboprop	BB-110	Airhansa5678	Airhansa AG	DE	Berlin	2017-03-03	2017-01-01	2015-02-14	2015-01-28
BB1107_4	Airplane Engine	3	Turboprop	BB-110	Airhansa5678	Airhansa AG	DE	Berlin	2017-03-03	2017-02-02	2015-02-14	2015-01-28
BMM250_1	Helicopter En...	1	Turboshaft	BM-M250	DSECopter1	eoda air GmbH	DE	Kassel	2017-03-13		2016-06-21	2016-06-04
BMK823_1	Airplane Engine	3	Turbojet	BM-K8	DSEJet1	eoda air GmbH	DE	Kassel	2017-04-19		2010-02-27	2010-02-10
BMK823_3	Airplane Engine	2	Turbojet	BM-K8	DSEJet2	eoda air GmbH	DE	Kassel	2017-05-16		2016-01-18	2016-01-01
BMK823_4	Airplane Engine	2	Turbojet	BM-K8	DSEJet2	eoda air GmbH	DE	Kassel	2017-05-16		2016-01-18	2016-01-01

Tabellen Widget

Im Tabellenwidget (größtes Widget im Zentrum der View) werden die Informationen zu den gefilterten Stammdaten angezeigt. Die Einträge können auch in der Tabelle durchsucht und sortiert werden. Weiterhin befinden sich in der Titelzeile des Tabellen-Widgets drei Symbole:

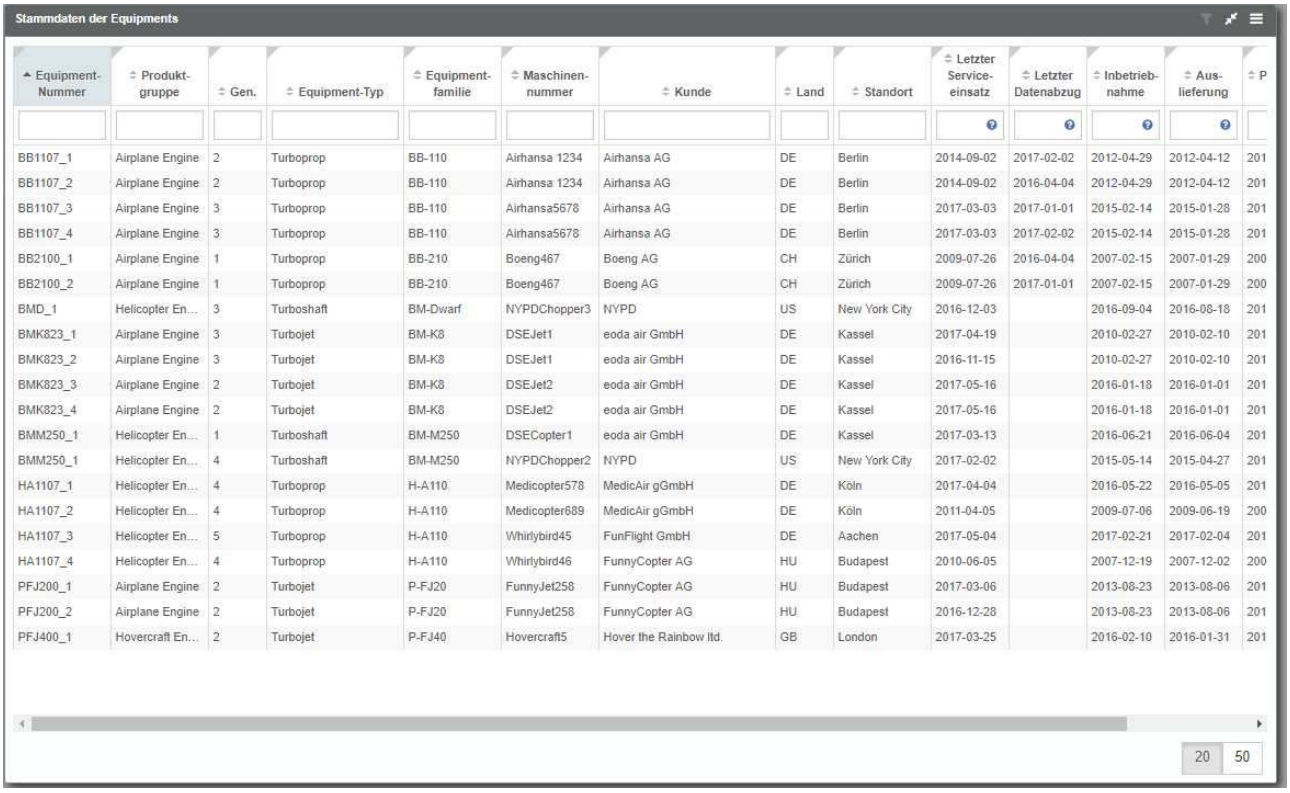
- Der Trichter dient zur Visualisierung einer aktiven Spaltenfilterung (haben Sie in einer Spalte der Tabelle durch die Eingabe in eins der Felder eine Spaltenfilterung aktiviert, so leuchtet der Trichter blau)
- Die zwei Pfeile dienen zum ein- und auszuklappen des Tabellen Widgets
- Die drei Balken öffnen das Kontextmenü des Tabellen-Widgets, in dem sich weitere Informationen und Funktionen zum Widget selbst befinden



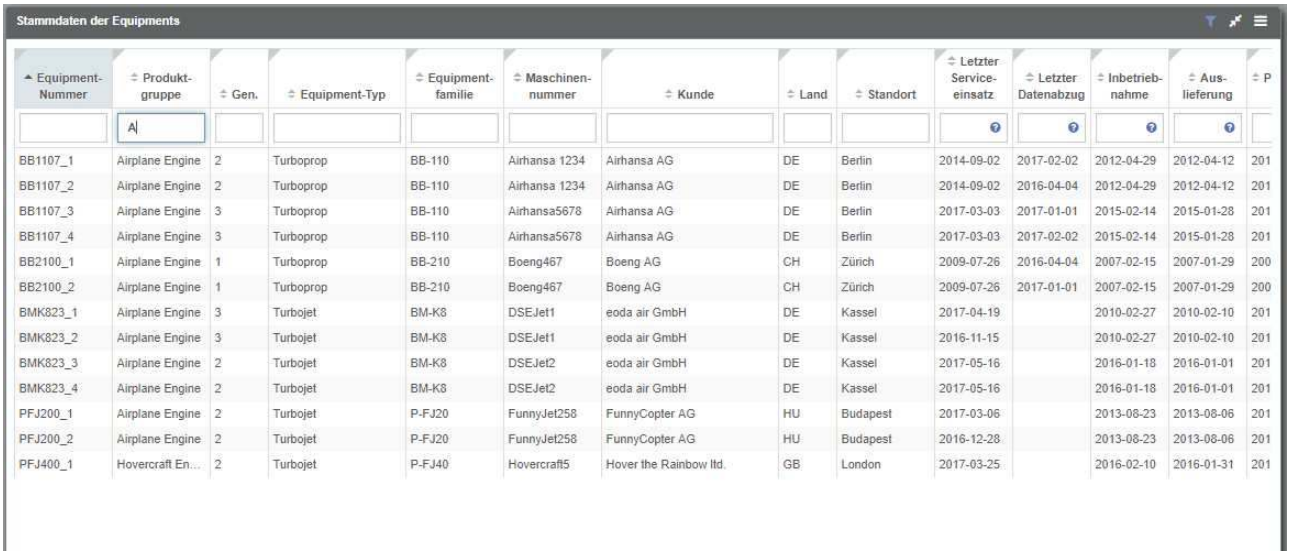
Filterung und Sortierung in Tabellen

Tabellenspalten können von Usern des Portals durchsucht und mithilfe von Spaltenfiltern werden. Hierzu stehen Nutzern unterschiedliche Möglichkeiten bereit, die sich auf die Inhalte der Tabelle auswirken.

1. Im **Ursprungszustand** wird eine Tabelle ungefiltert und unsortiert dargestellt.



2. Es gibt Möglichkeiten zur **Filterung der Tabelleninhalte**. Im folgenden Bild wird in der Spalte "Produktgruppe" der Inhalt der Tabelle gefiltert ("Spaltenfilter"). Es werden nur noch die Inhalte der Tabelle angezeigt, bei denen in der Produktgruppe mindestens ein "a" enthalten ist. In diesem Beispiel bewirkt es, dass nur noch Datensätze der Produktgruppen "Airplane Engine" und "Hovercraft Engine" angezeigt werden nicht mehr aber Datensätze der Produktgruppe "Helicopter Engine".

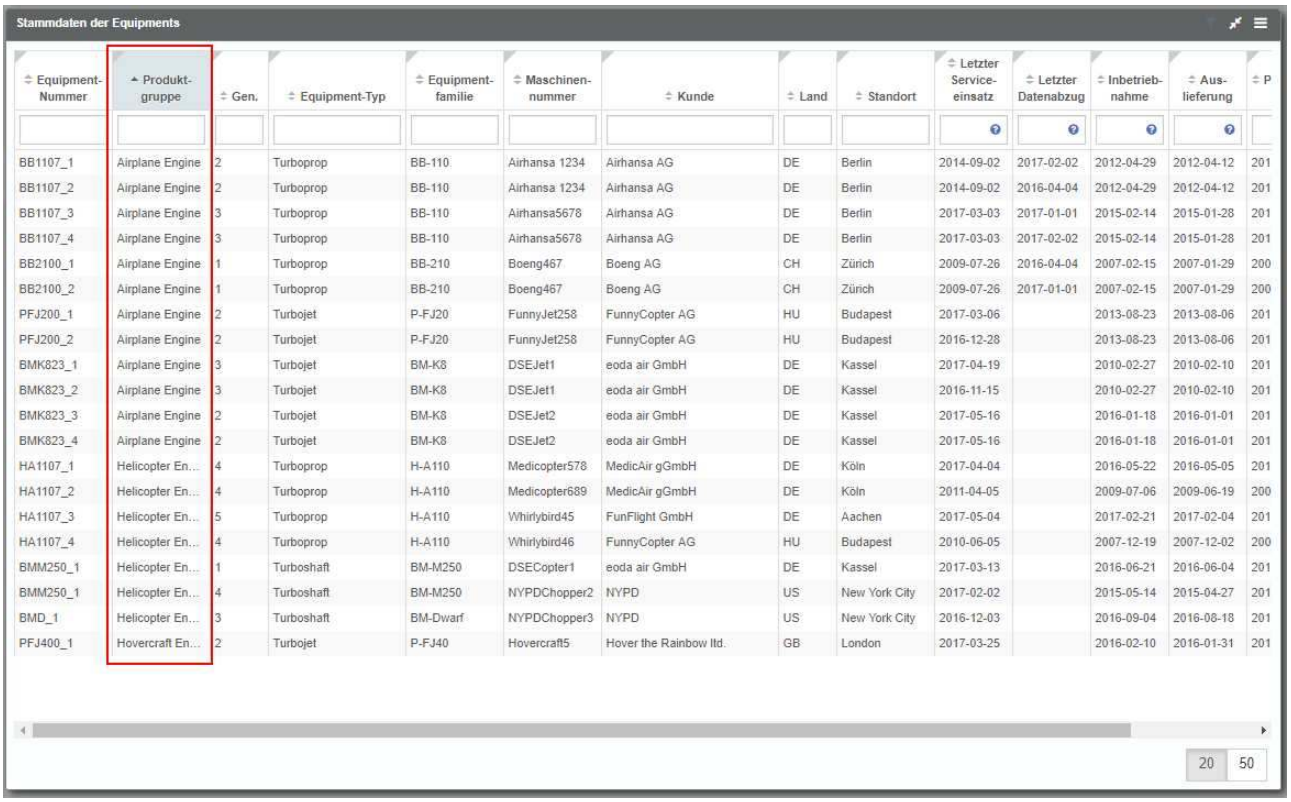


Bei der Filterung von Tabellenspalten wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Eintrag	Bedeutung	Beispieleingabe	Beispielergebnis
a-z; A-Z; 0-9	Filtert die Tabelleninhalte und zeigt nur noch die Inhalte an, die den Sucheintrag enthalten.	Airplane	Zeigt nur noch Einträge an, bei denen in der gefilterten Spalte der Sucheintrag Airplane enthalten ist.
!a	Filtert die Tabelleninhalte und zeigt nur noch die Inhalte an, die **NICHT** den Sucheintrag hinter dem "!" enthalten	!Airplane	Zeigt nur noch Einträge an, bei denen in der gefilterten Spalte der Sucheintrag Airplane **NICHT** enthalten ist.
yyyy-mm-dd	Filterung nach Datum	2017-03-25	Zeigt nur Datensätze, die das gesuchte Datum in der Suchspalte enthalten

Die vollständige Liste von Operatoren sowie die zugehörige erlaubte Syntax zum Durchsuchen von Tabellenspalten ist abhängig vom Typ der Daten in der gewünschten Spalte. Die für die jeweilige Spalte relevanten Operatoren und Syntax können durch einen Klick im Suchfeld oberhalb einer Tabellenspalte auf das  -Icon angesehen werden.

3. Auch können Tabelleninhalte **sortiert** werden. Hierzu muss das Sortier-Symbol in der Kopfzeile der gewünschten Tabellenspalte angeklickt werden. Es steht Nutzern frei, ob sie Tabellenspalten aufsteigend oder absteigend sortieren möchten.



Wird eine Tabelle anhand einer Spalte sortiert, so wirkt sich die Sortierung auch auf alle anderen Spalten der Tabelle aus. Auf diese Weise bleiben alle Einträge eines Datensatzes in der gleichen Zeile zusammen.

Beispiel: Aufsteigende Sortierung einer Tabelle nach aufsteigender alphabetischer Reihenfolge der Spalte Ort

Ausgangstabelle			Sortierte Tabelle		
ID	**KFZ-Kennzeichen**	**Ort**	**ID**	**KFZ-Kennzeichen**	**Ort**
1	KS	Kassel	2	B	Berlin
2	B	Berlin	1	KS	Kassel
3	OF	Offenbach	3	OF	Offenbach

Export-Funktion des Tabellen-Widgets

Im Kontextmenü des Tabellen-Widgets können die Tabelleninhalte bspw. exportiert werden. Für den Export stehen Konfigurationsmöglichkeiten bereit:

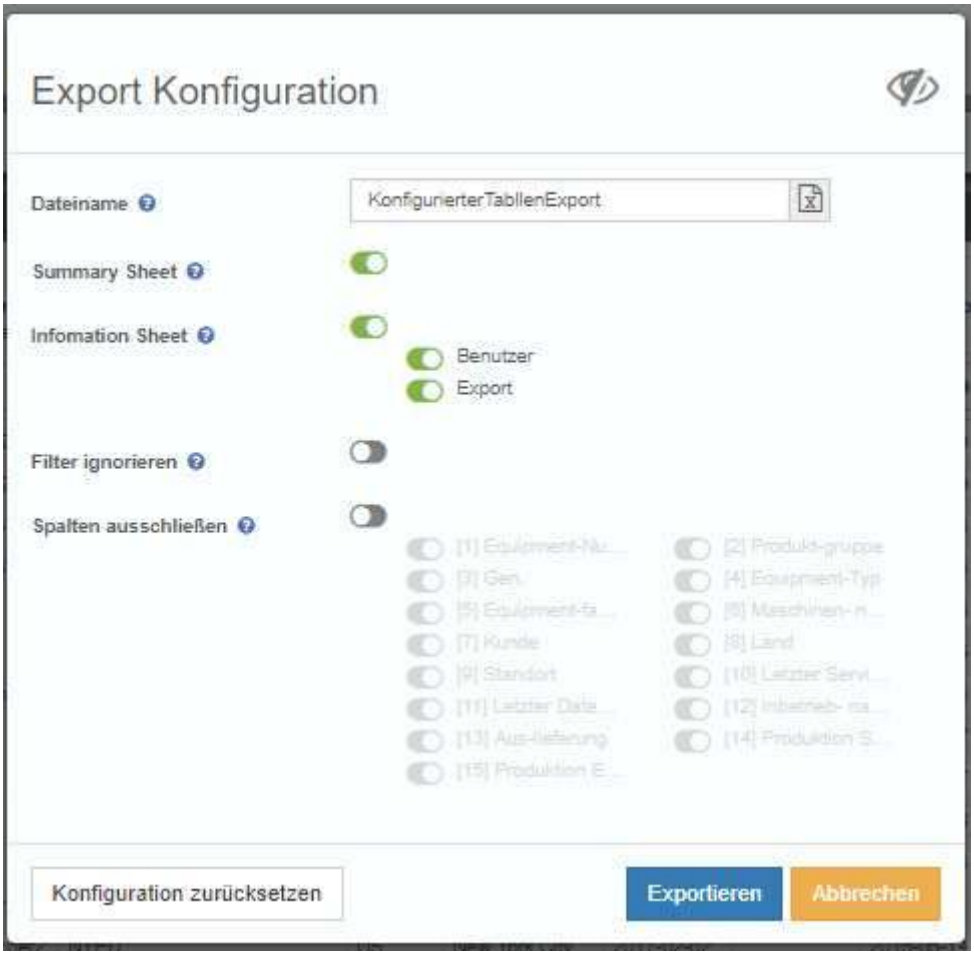
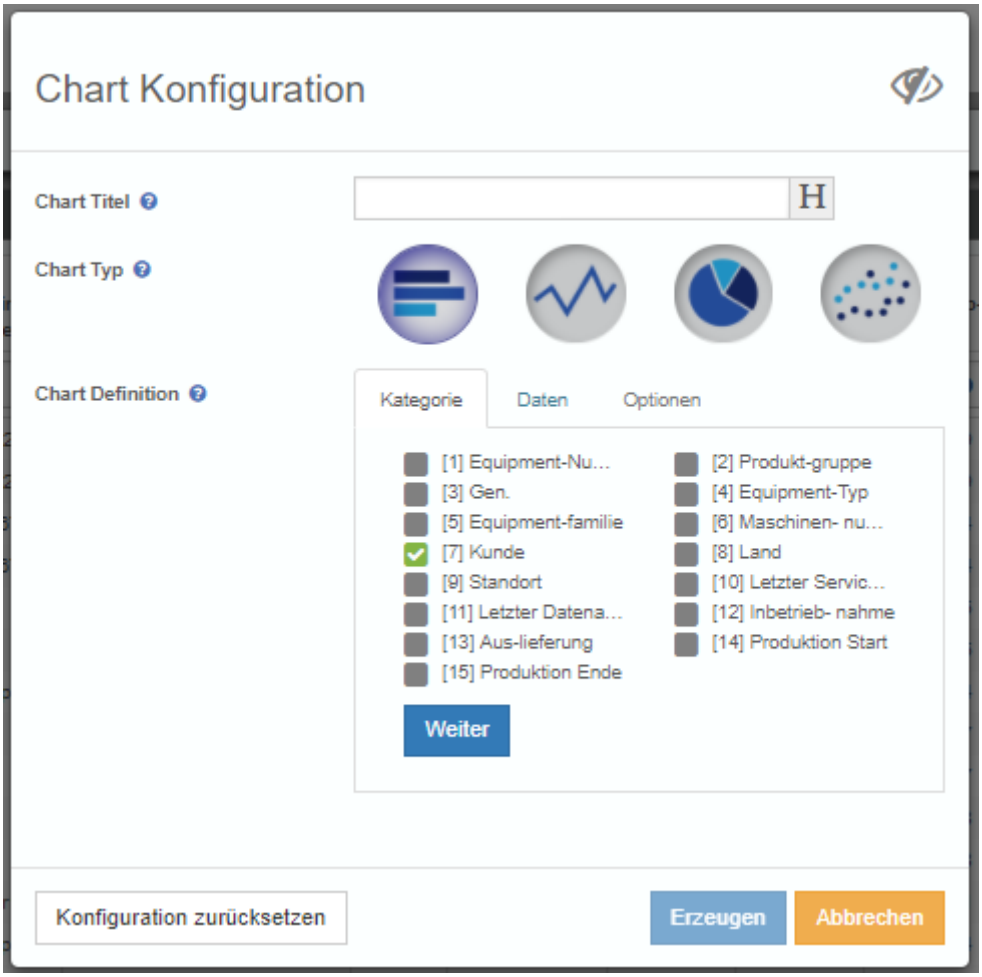


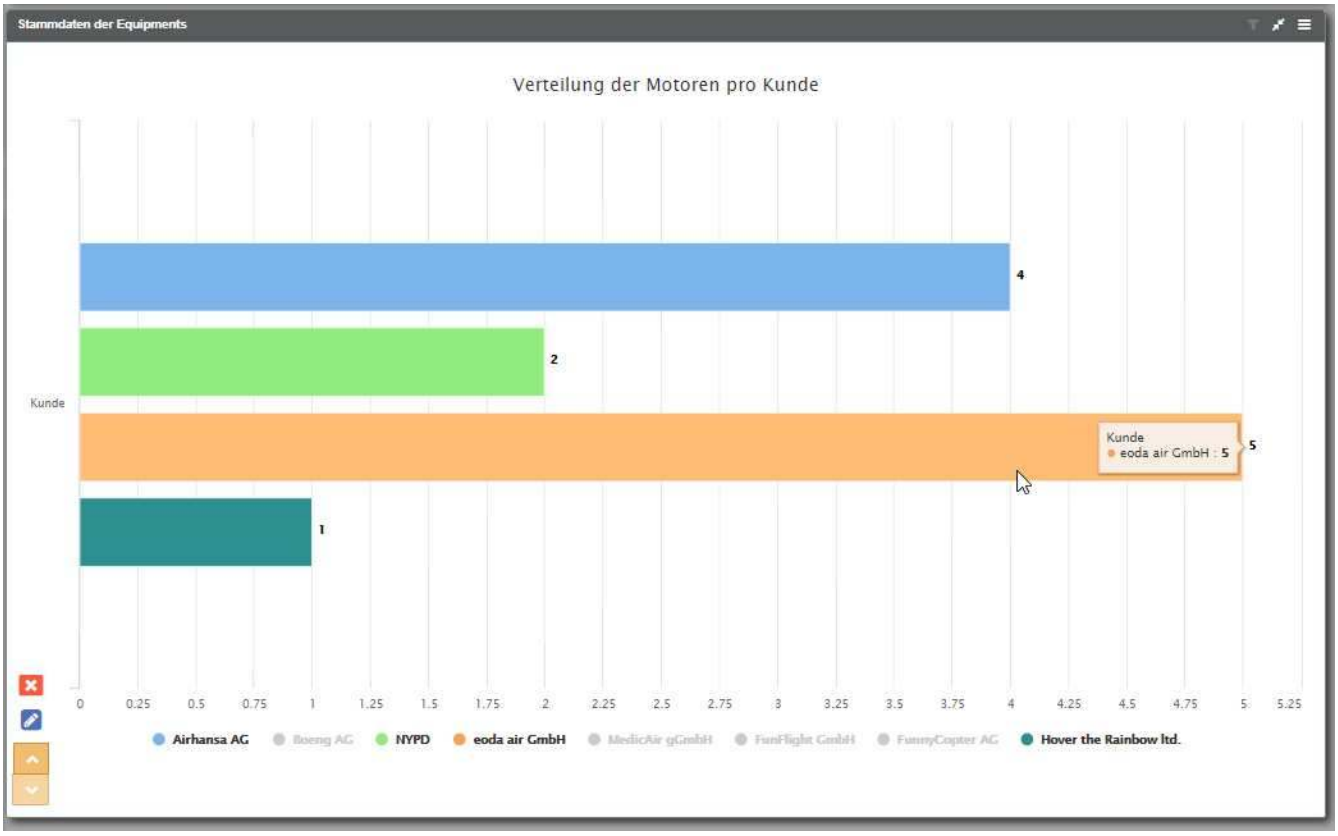
Chart aus Tabelle erzeugen

Eine weitere Funktion des Tabellen-Widgets ist die Möglichkeit, sich unterschiedliche Diagramme aus den Daten der Tabelle erzeugen zu lassen. Wählen Sie hierzu einfach die Funktion "Chart generieren" im Kontextmenü aus. Dies öffnet die Chartkonfiguration, in der Sie den Typ des Charts, die Datengrundlage, die Achsenbeschriftungen etc. festlegen können.

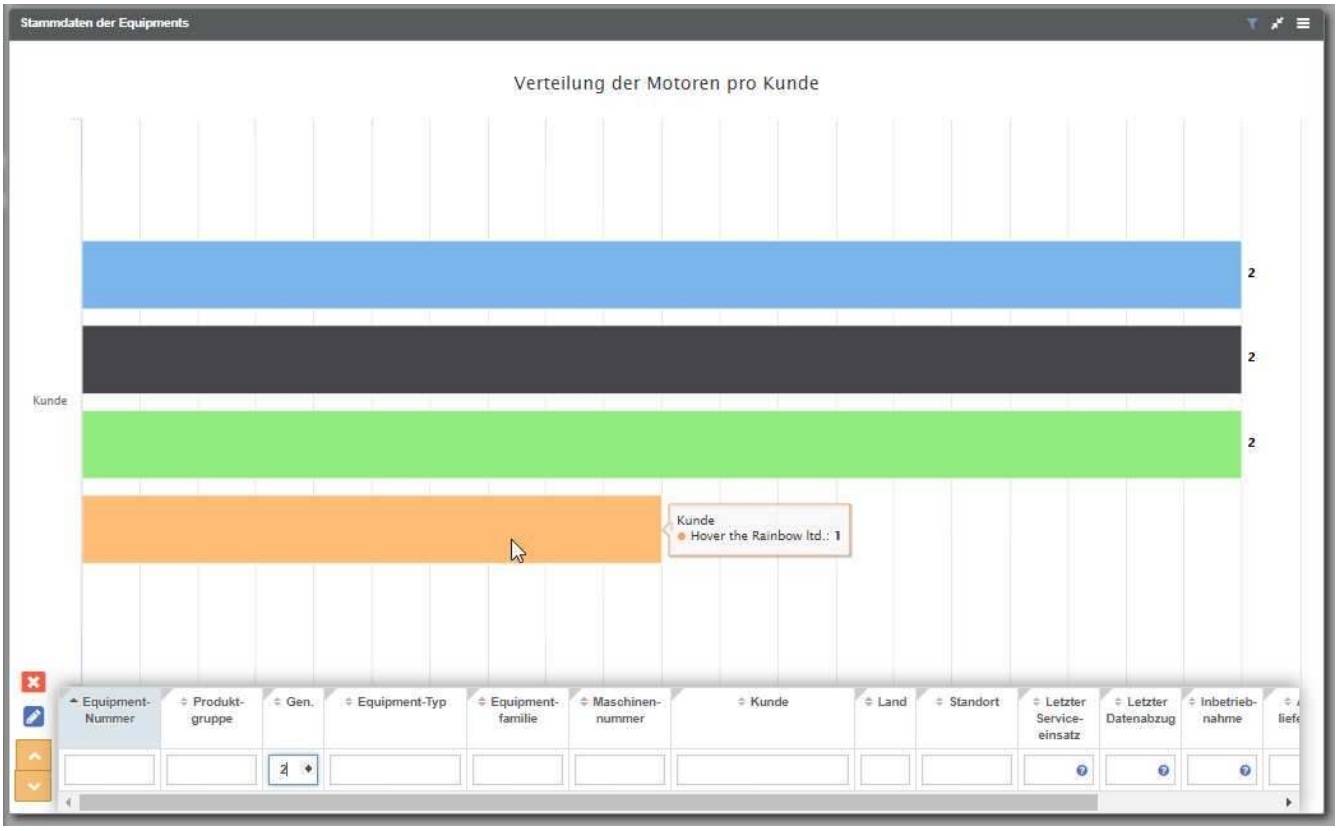


Nachdem Sie durch das Menü geführt wurden, wird ein Chart erzeugt. In diesem Fall ein Balkendiagramm, dass die Verteilung der Kunden nach Ländern sortiert anzeigt.

Weiterhin stehen Ihnen vielfältige Funktionen bereit, um das Chart auch nachträglich anzupassen und zu überarbeiten. So können Sie bspw. durch einen Klick auf die Legendeneinträge im unteren Teil des Diagramms Einträge ein- und ausblenden.

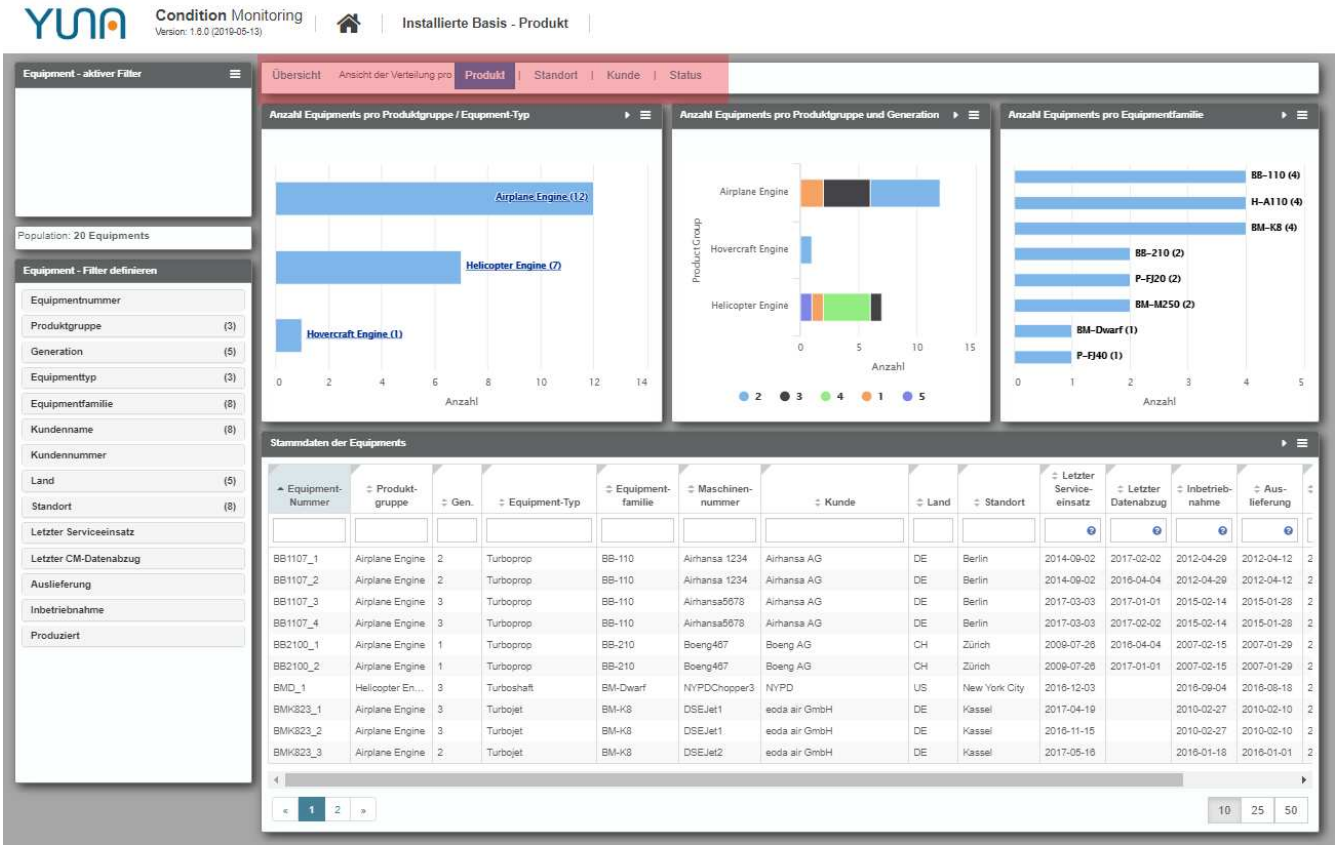


Außerdem können Sie durch einen Klick auf den oberen orangen Pfeil (unten links im Diagramm) die Kopfzeilen des Tabellen-Widgets einblenden, auf dem das Diagramm basiert. So haben Sie die Möglichkeit das Diagramm weiter zu filtern, indem Sie Einträge in die Eingabefelder vornehmen.



Html-Widget

Das html-Widget (Menüleiste) oberhalb der Tabelle gibt dem Benutzer zudem die Möglichkeit, sich die Daten zu den Equipments der Installierten Basis aggregiert nach Kategorien wie Produkten, Standorten, Kunden oder Stati anzuschauen. So werden nach Klick auf "Ansicht der Verteilung pro Produkt" zusätzlich zu den Informationen im Tabellen-Widget im unteren Bereich der View drei Chart-Widgets angezeigt. Diese zeigen die Verteilung der Equipments der Installierten Basis nach Produktgruppen, nach Anzahl der Items pro Produktgruppe und Generation sowie nach Anzahl der Items pro Produktfamilie.



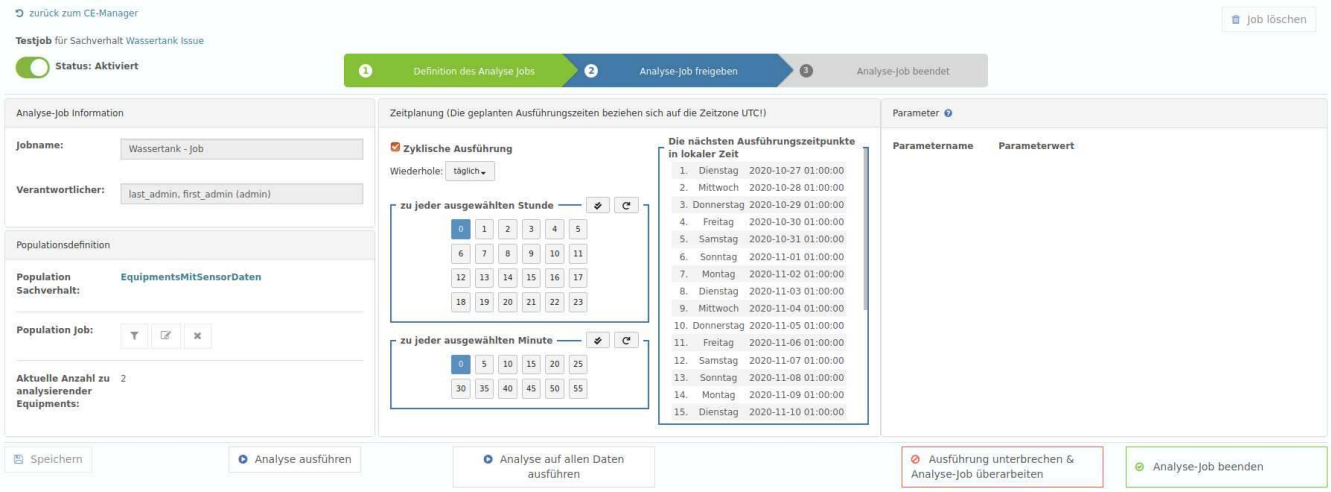
Übersicht über die typischerweise vorhandenen Funktionen in der View Installierte Basis:

- Filter
  - Filter laden
  - Daten selektieren und filtern
  - Datenselektion zurücksetzen
  - Selektierte Daten als neuen speichern
- Informationen
  - zu den einzelnen Widgets anzeigen lassen
- Tabelleninhalte
  - sortieren
  - durchsuchen
  - exportieren
    - konfigurieren
    - als *.xlsx

5.6.4. Jobmanager (ceJobDirective)

Was ist der Jobmanager?

Im YUNA Portal wird unter einem Analysejob die geplante Ausführung eines Analyseskripts verstanden. Diese Ausführung kann entweder manuell erfolgen oder automatisiert in frei definierbaren Zyklen.In diesem Fall wird von Cyclic Evaluation Jobs gesprochen (CE-Jobs). Diese Jobs können im Jobmanager angelegt und definiert werden.Des weiteren werden die Jobs im Jobformular durch die Phasen des Job-Workflows geführt.



Anlegen eines Jobmanagers

XML

```
<xml>
  <widget name="template_widget_Job">
    <!-- Position from left top  -->
    <position>
        <x>0</x>
        <y>0</y>
    </position>
    <!-- Size of the Widget -->
    <size>
      <x>14</x>
      <y>7.5</y>
    </size>
    <!-- Caption: title over the widget and additional features (Contextmenue) -->
    <caption>
        <!-- Display caption -->
      <show>true</show>
        <!-- Title of the Widget -->
      <label>Job-Editor-Template</label>
    </caption>
    <!-- Name of the WidgetType -->
    <widgettype>cejobdirective</widgettype>

    <urlParams>
```

```
        <jobid>myJobId</jobid>
        <issueid>myIssueId</issueid>
    </urlParams>

  </widget>
</xml>
```

JSON

```
{
    "position": {
        "position": [0, 0]
    },
    "size": {
        "x": 14,
        "y": 7.5
    },
    "caption": {
        "show": true,
        "label": "Job-Editor-Template"
    },
    "widgetname": "cejobdirective",
    "triggerParams": []
}
```

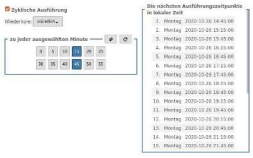
URL Parameter

Im Element <urlParams> können die vom Jobmanager verwendeten URL Parameter eingestellt werden.

Zyklische Ausführung

Unter der Zyklischen Ausführung versteht man eine Zeitplanung, in welcher Jobs regelmäßig ausgeführt werden.Die Zeitplanung erfolgt durch eine Quarz kompatible Cron-Expression.

Diese kann wie folgt definiert werden.

Zyklus	Optionen	Mehrfach auswahl	Bedeutung	Bild	Beispiel
Stündlich	Minuten	Ja	Der Job wird einmal pro Stunde zu den gewählten Minuten ausgeführt		gewählte Minuten: 15,45 die Ausführungszeitpunkte sind:00:15, 00:45 .. 23:15, 23:45

Zyklus	Optionen	Mehrfach auswahl	Bedeutung	Bild	Beispiel
Täglich	Minuten Stunden	Ja Ja	Der Job wird einmal pro Tag zu den gewählten Stunden und Minuten ausgeführt		gewählte Minuten: 15, 45gewählte Stunden: 2 die Ausführungszeitpunkte sind: Mo 02:15, Mo 02:45 .. So 02:15, So 02:45
Wöchentlich	Minuten Stunden Wochentage	Ja Ja Ja	Der Job wird einmal pro Woche zu den gewählten Wochentagen, Stunden und Minuten ausgeführt		gewählte Minuten: 15, 45gewählte Stunden: 2gewählte Wochentage: Mo, Fr die Ausführungszeitpunkte sind: Mo 02:15, Mo 02:45, Fr 02:15, Fr 02:45

Zyklus	Optionen	Mehrfach auswahl	Bedeutung	Bild	Beispiel
Monatlich	Minuten	Ja	Der Job wird einmal Pro Monat zum gewählten vorkommen des Wochentags, zu den Stunden und Minuten ausgeführt.		gewählte Minuten: 15, 45gewählte Stunden: 2gewählter Wochentag: Migewähltes vorkommen: Dritte die Ausführungszeitpunkte sind:(ausgehend vom 01.01.2020) Mi 15.01.2020 02:15Mi 15.01.2020 02:45Mi 19.02.2020 02:15Mi 19.02.2020 02:45
	Stunden	Ja			
	Wochentage	Nein			
	Vorkommen	Nein			
Benutzerdefiniert	Individuell	Nein	Der Job wird der eingegebenen Cron-Expression entsprechend ausgeführt		**0 15,45 2 ? * * *** Die Ausführungszeitpunkte sind: Mo 02:15, Mo 02:45 .. So 02:15 So 02:45

5.6.5. Landing Page: Ihre Startseite

Nach erfolgreicher Anmeldung gelangt ein Benutzer auf seinen persönlichen Startbildschirm, die sogenannte Landing Page. Diese Landing Page ist eine Portal View und besteht - je nach Berechtigungen für den Account des Benutzers - aus bestimmten Kacheln. Diese Kacheln sind Objekte vom Typ Kachel-Widget.

Diese Kachel-Widgets sind Links und öffnen beim Anklicken eine weitere Portal View, in der 1 bis n Widgets zusammengefasst werden. So kann sich ein Benutzer beispielsweise nach Klick auf die Kachel "Installierte Basis" Informationen zu den, in YUNA hinterlegten Anlagen anzeigen lassen. Diese können über verschiedene Auswahlen gefiltert werden und sich in (interaktiven) Charts oder Tabellen aufbereiten lassen.

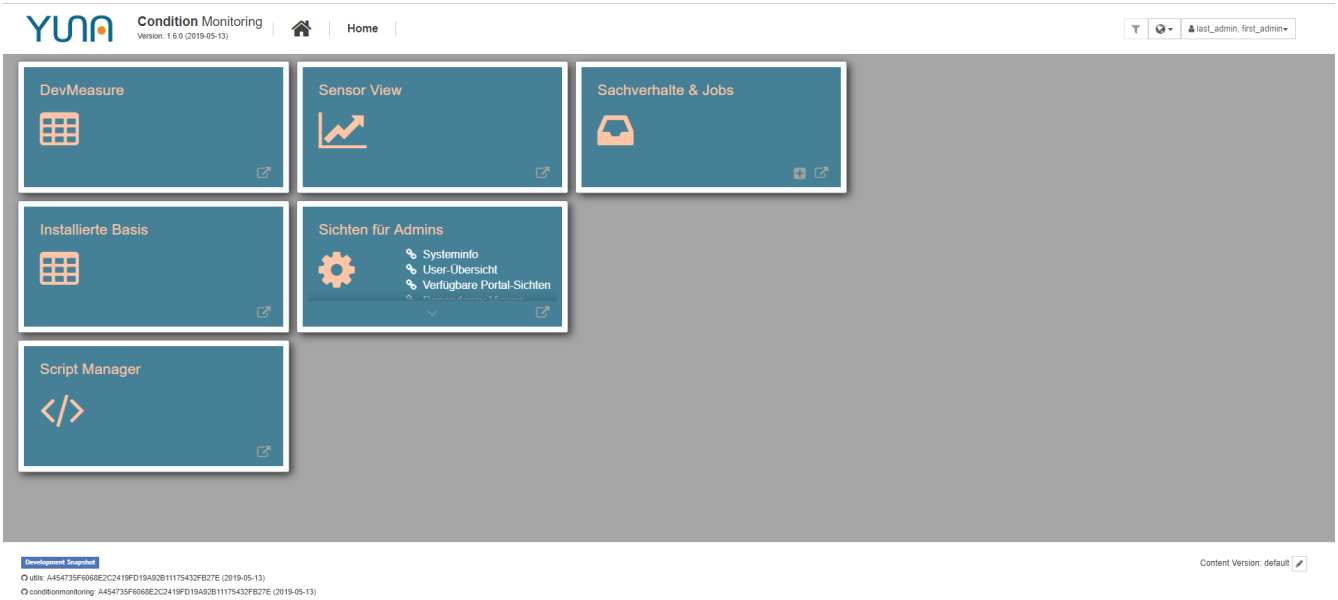
Nutzerindividuelle Inhalte

Für diese Dokumentation wird das Portal anhand von zwei Benutzerrollen vorgestellt, die

unterschiedliche Berechtigungen für das Portal besitzen: System_Admin und AdHoc_Full_Issue. Aufgrund der unterschiedlichen Berechtigungen der Rollen befinden sich auf der Landing Page für die Nutzerrollen folgende Kacheln:

System_Admin	AdHoc_Full_Issue
• Installierte Basis • Sensor View • Sachverhalte & Jobs • Script-Manager • Sichten für Admins	• Installierte Basis • Sensor View • Sachverhalte & Jobs • Script-Manager

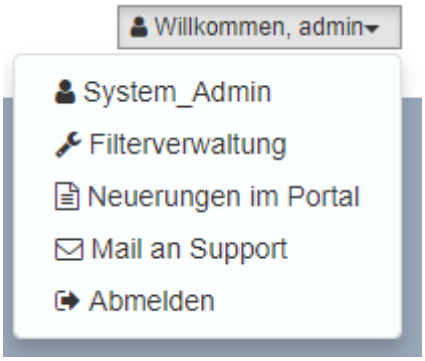
Ansicht für Benutzer der Rolle System_Admin:



Ansicht für Benutzer der Rolle AdHoc_Full_Issue:

Zudem erscheint anstatt der Eingabefelder für den Login im oberen rechten Bereich ein Dropdown-Menü. Neben der Angabe des Nutzernamens und der -rolle finden sich dort die Möglichkeiten zum

Aufruf der Filterverwaltung, zur Anzeige einer Übersicht der Neuerungen (Change Log), zur Kontaktierung des Supports via Email, sowie die Möglichkeit, sich aus dem YUNA Portal abzumelden. Benutzer können zudem, im Header über auftretende Störungen, Wartungen, Updates, etc. benachrichtigt werden.



XML

```xml
<widget>
     <position>
         <x>0</x>
         <y>0</y>
     </position>
     <size>
       <x>4</x>
       <y>2</y>
     </size>
     <widgettype>defaultlinkdirective</widgettype>
     <landingpage>
       <link>Iris</link>
       <linkName>Zur Iris Tabelle</linkName>
       <icon>fa fa-inbox</icon>
       <addnew>false</addnew>
       <newItemLink>cmp_Issue_Manager_Editor</newItemLink>
     </landingpage>
</widget>
```

JSON

```json
{
  "position": {
    "position": [0, 0]
  }
  "size": {
    "x": 4,
    "y": 2
  }
  "widgetname": "defaultlinkdirective",
  "landingpage": {
```

```
    "link": "Iris",
    "linkName": "Zur Iris Tabelle",
    "icon": "fa fa-inbox",
    "addnew": false,
    "newItemLink": "cmp_Issue_Manager_Editor"
  }
}
```

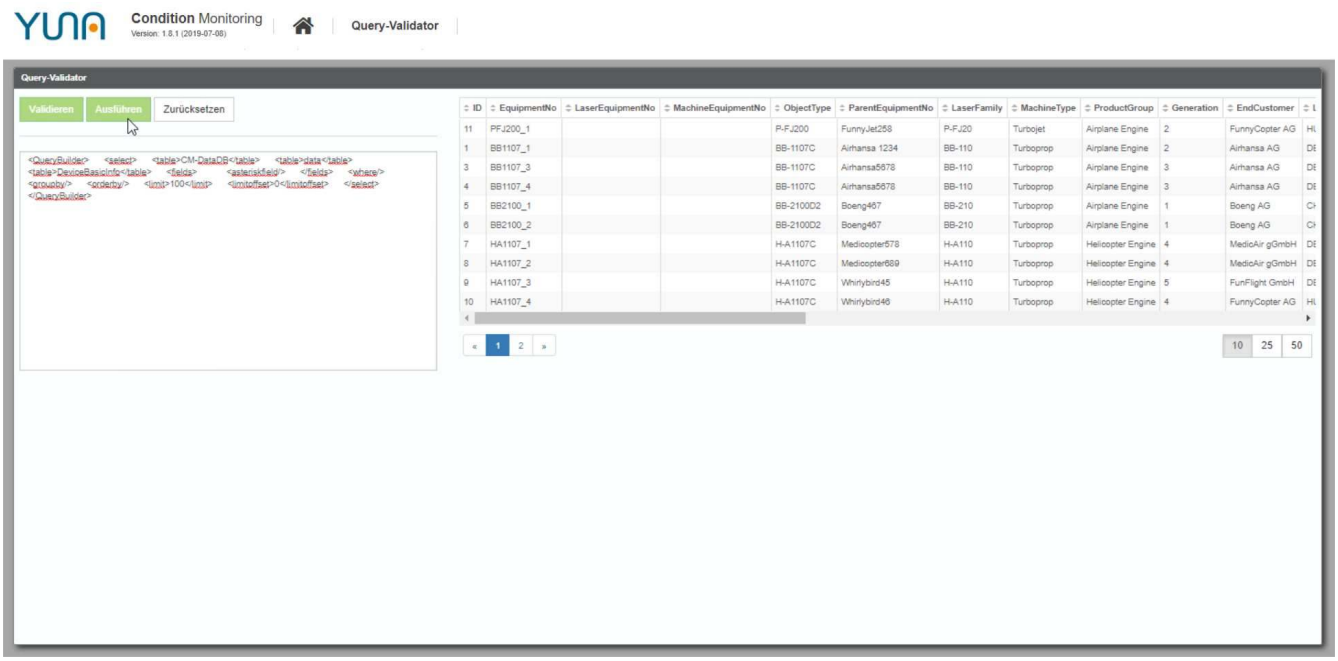
Verfügbare Optionen:

Feld	Mögliche Werte	default	Beschreibung
position	Zahlenwerte für x- und y-Koordinaten	0 / 0	Definiert die Position des Widgets im Grid
size	Zahlenwerte für x- und y-Koordinaten		Definiert die Größe des Widgets
widgetname	defaultlinkdirective		Definiert den Widgettyp
triggerParams	Namen der optionalen und verpflichtenden Parameter		Definiert die Parameter, auf die das Widget "hört"
link			Definiert das Ziel der Kachel (Link zu einer anderen Portal View)
linkName			Definiert den in der Kachel angezeigten Text, der als Link zu einer anderen Portal View fungiert.
icon	fontawesome icons		Definiert, ob ein Icon bzw. welches Icon in der Kachel angezeigt werden soll.
addnew	true false	true	Definiert, ob in der Kachel ein Button zur Erstellung eines neuen DSP-Elements enthält. Dieses Element kann z.B. eine neue Kachel auf der Landing Page oder ein neuer Sachverhalt sein.
newItemLink			Definiert den Zielort des neuen DSP-Elements, das sich durch den Button erzeugen lässt.

Über die Option "Link" können für die Landing Page / defaultLinkDirective nur Verlinkungen zu Views innerhalb des Portals definiert werden, die beim Klick auf die entsprechende Kachel geladen werden sollen. Der Typ "Link" für die Landing Page unterscheidet sich also vom Typ "GenLink", der für das Tabellen-Widget definiert werden kann.

5.6.6. Query Validator

Der Query Validator ist ein Hilfstool für Administratoren und Dashboard Developer, um die von ihnen erstellten Queries auf Korrektheit und späteres Aussehen zu testen, ohne die Query erst deployen zu müssen.



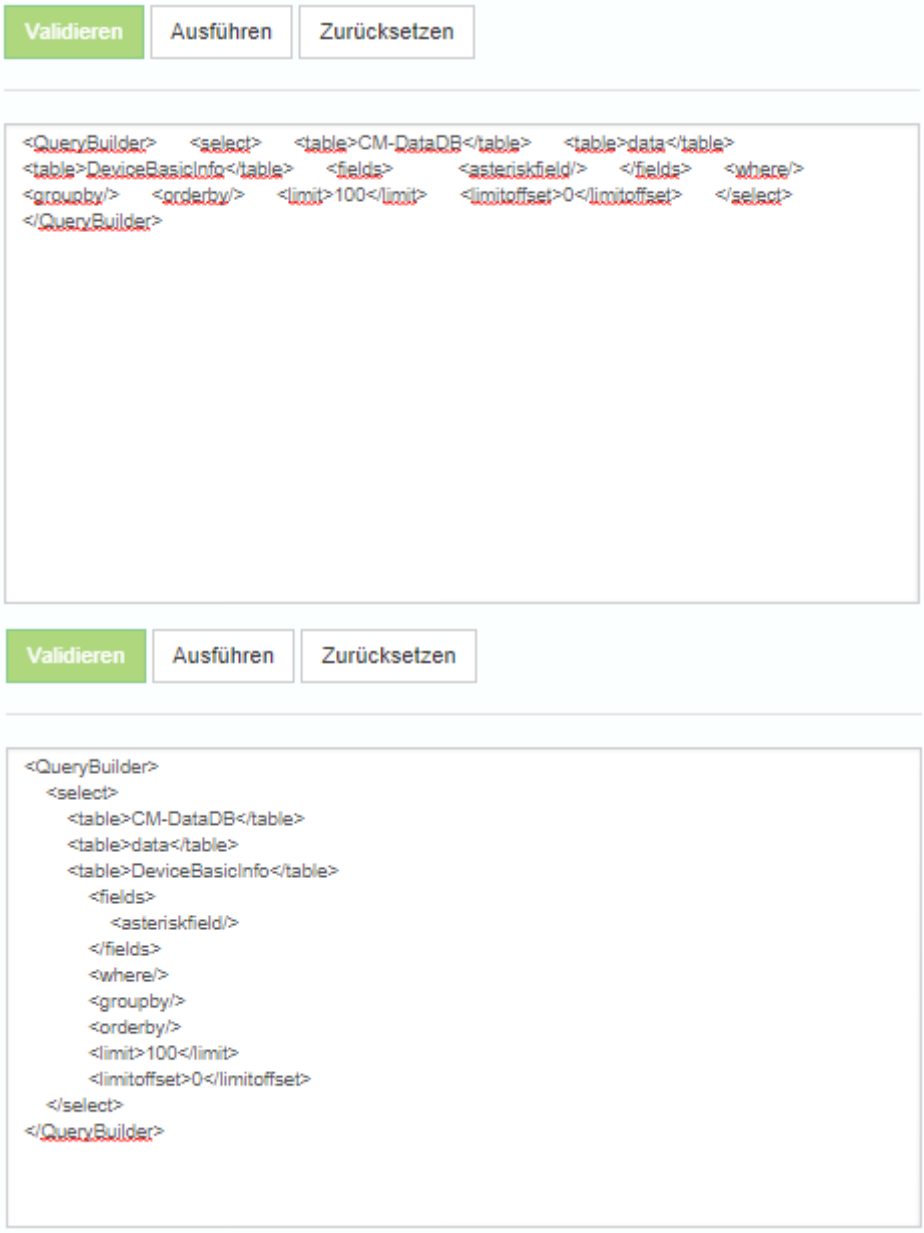
Nutzung des Query Validators

Schritt 1: Aufrufen des Query Validators

Zunächst muss der Query Validator aufgerufen werden. Nutzen Sie hierfür auf der Landing Page die Kachel "Sichten für Admins" und wählen dort den Query Validator aus. Je nach verfügbaren Views für Admins kann es sein, dass Sie zunächst den kleinen Pfeil am unteren Ende der Kachel anklicken müssen, um innerhalb der Kachel nach unten zu scrollen.

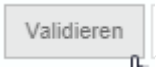
Schritt 2: Erzeugen / Einfügen der Query

Erstellen Sie die Query, mit der Sie die Datenbankabfrage durchführen möchten oder fügen Sie die bereits erstellte Query in das Eingabefeld ein. Es spielt keine Rolle, ob der Code am Stück oder mit Zeilenumbrüchen eingefügt wird.



Schritt 3: Query validieren

Nachdem die aus Ihrer Sicht vollständige Query eingefügt wurde können Sie die Query auf formale Korrektheit prüfen lassen, indem Sie auf den Button "validieren" klicken. Ist die Query formal korrekt, so färbt sich der button grün (

), andernfalls wird der Button orange oder dunkelgrau 

.

Schritt 4: Ergebnisvorschau anzeigen

Durch Anklicken des Buttons "Ausführen" wird eine als korrekt validierte Query ausgeführt und das Ergebnis der Datenabfrage direkt im Query Validator angezeigt. So können Queries und kleine Anpassungen daran direkt im Browser geprüft werden, ohne dass zuvor auf die Datenbank deployed werden muss.

5.6.7. Sachverhalte und Jobs

Analysearten im Kontext des YUNA Portals

Im YUNA Portal wird zwischen zwei Arten von Analysen unterschieden: Ad-Hoc-Analysen und Zyklischen Analysen.

- Unter Ad-hoc-Analysen wird verstanden, dass im Portal vorhandene Daten (Stammdaten, Sensordaten,...) in Tabellen und Diagrammen betrachtet, gefiltert und durchsucht werden können.
- Als Zyklische Analysen (CE - cyclic evaluations) werden komplexe Analysen von für Sie interessanten Daten oder Fragestellungen (so. Sachverhalten) durch die Nutzung von Analyseskripten bezeichnet, die manuell oder in bestimmten Zyklen ausgeführt werden. Diese Skripte können Sie entweder selbst erstellen, im Portal einpflegen und die Analysedurchläufe mit Analyse-Jobs selbst durchführen oder Data Scientists damit beauftragen. Für die Ausführung dieser Skripte können Jobs definiert werden.

Sachverhalte: Erster Überblick

Für die Ausführung von zyklischen Analysen ist die Nutzung von Sachverhalten und das Durchlaufen des Sachverhalts-Workflows nötig.

Ein Sachverhalt ist typischerweise ein Problem oder eine Fragestellung aus der realen Welt, für die ein Data Scientist ein statistisches Modell auf Basis vorhandener Daten erarbeiten soll. Ziel des Modells ist es, den Sachverhalt so gut zu modellieren, dass das Skript Vorhersagen zukünftiger Ereignisse erlaubt.In Form einer wiederkehrenden Analyse auf neue Daten in einen regelmäßigen Prozess (Cyclic Evaluation) angewendet, könnte diese Analyse zukünftige Ereignisse automatisch anzeigen. An der Klärung des Sachverhaltes sind verschiedene Rollen mit unterschiedlichen Funktionen und Berechtigungen beteiligt.

Die Begriffe Issue und Sachverhalt werden im Kontext des YUNA Portals synonym verwendet.

Jobs: Erster Überblick

Die manuelle oder zyklische Ausführung dieser Analyseskripte zur Beantwortung der Fragestellungen aus der realen Welt im Zuge der Sachverhalte wird über sogenannte Jobs gesteuert. Entlang des Sachverhalts-Workflows wird zwischen drei Job-Typen unterschieden, für die jeweils gezielt bestimmte Parameter, Ausführungszeiten, Verantwortlichkeiten und Filtermengen festgelegt werden können.

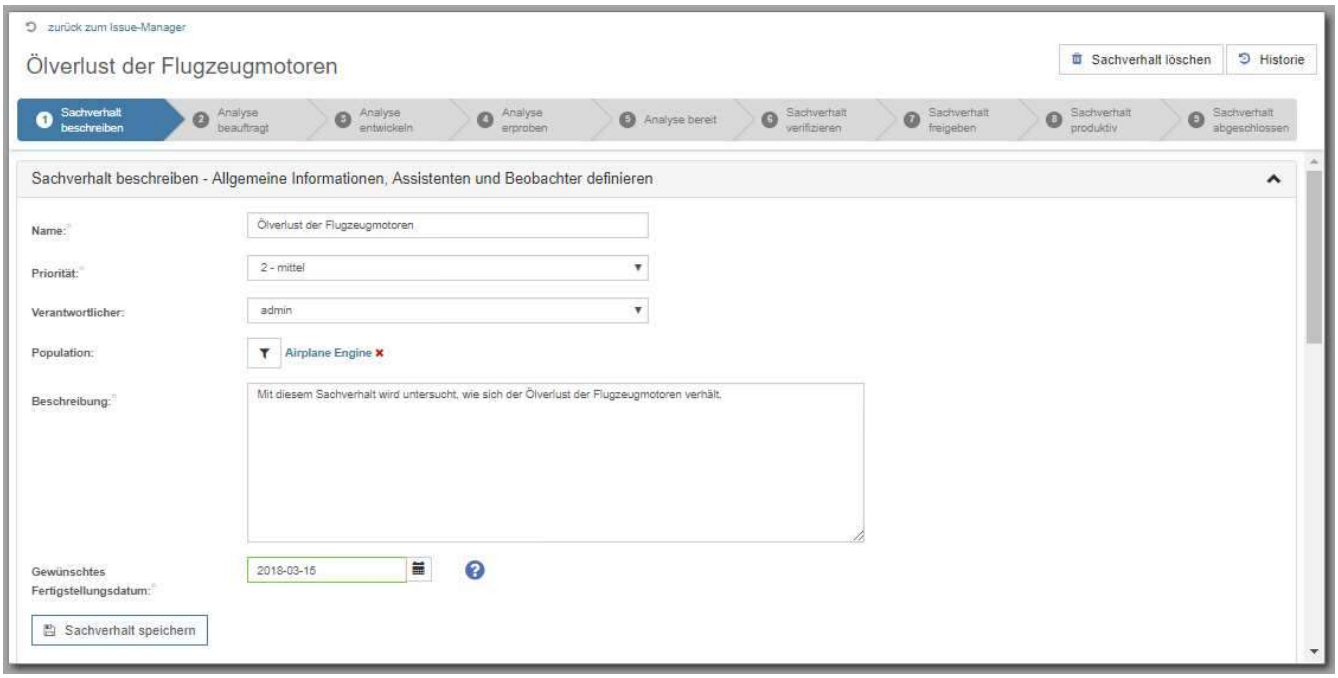
Auf den nächsten Seiten erhalten Sie weitere Informationen darüber, wie Sie Sachverhalte und Jobs anlegen und mit diesen interagieren können.

Sachverhaltsmanager

Sachverhalte anlegen, editieren und deren Status oder Ergebnisse anzeigen

Über den Issue Manager kann das Sachverhaltsformular gestartet werden. Es basiert auf den Phasen des Sachverhalts-Workflow und bildet den Rahmen für die Untersuchung von Sachverhalten. Im Sachverhaltsformular finden sich alle Optionen, die dem Sachverhaltsverantwortlichen oder den Assistenten zur Verfügung stehen, um die Untersuchung

des Sachverhalts durchführen zu können. Dabei werden Sie durch den Aufbau des Formulars Schritt für Schritt durch die Phasen geleitet. Zwischen den einzelnen Phasenübergängen kann ein Kommentar zum Grund des Statusübergangs eingegeben werden. Diese Informationen werden im Log des Sachverhalts hinterlegt. Dies dient zusammen mit der Speicherung der Informationen zum Datum und des Benutzers der Nachverfolgbarkeit bei der Untersuchung von Sachverhalten. So lässt sich jederzeit prüfen, wer wann was an welchem Sachverhalt geändert hat.



Die Phasen eines Sachverhalts

Sachverhaltsphase	Phasentitel	Inhalt / Funktionen	abgeschlossen wenn...
1	Sachverhalt beschreiben	**Inhalt:** • Hier werden die Informationen zu einem Sachverhalt gesammelt und beschrieben. Sie bilden die Basis einer Sachverhaltsuntersuchung. **Funktionen:** • Sachverhalt betiteln • Sachverhalt priorisieren • Verantwortlichen definieren • Filter für die zu untersuchende Anlagenpopulation auswählen oder definieren • Sachverhalt beschreiben (Freitext) • gewünschtes Fertigstellungsdatum festlegen • Sachverhalt speichern • Zum Sachverhalt gehörende Informationen (z.B. Links zu Dokumentationen oder zu Tasks in Systemen wie JIRA oder TFS) • Festlegung von Assistenten (dürfen den Sachverhalt im Nachhinein bearbeiten und haben die gleichen Rechte wie der Verantwortliche) • Beobachter festlegen (Einzelpersonen oder Gruppen, die den Sachverhalt lesend verfolgen dürfen)	• Sachverhalt ist betitelt • Sachverhalt wurde beschrieben • Gewünschtes Fertigstellungsdatum wurde definiert • Analyse gespeichert und beauftragt
2	Analyse beauftragt	**Inhalt:** • Die Untersuchung des angelegten Sachverhalts wird bestätigt und eine Analyse soll beauftragt werden. **Funktionen:** • Link zu einem Task (JIRA, TFS,...) hinterlegen • voraussichtliches Fertigstelungsdatum festlegen • Analyseauftrag ablehnen oder speichern und in Phase 3 eine Analyse entwickeln	• Task ist hinterlegt • vorauss. Fertigstellungsdatum ist gepflegt • Analyse ist gespeichert und die Analyse soll entwickelt werden

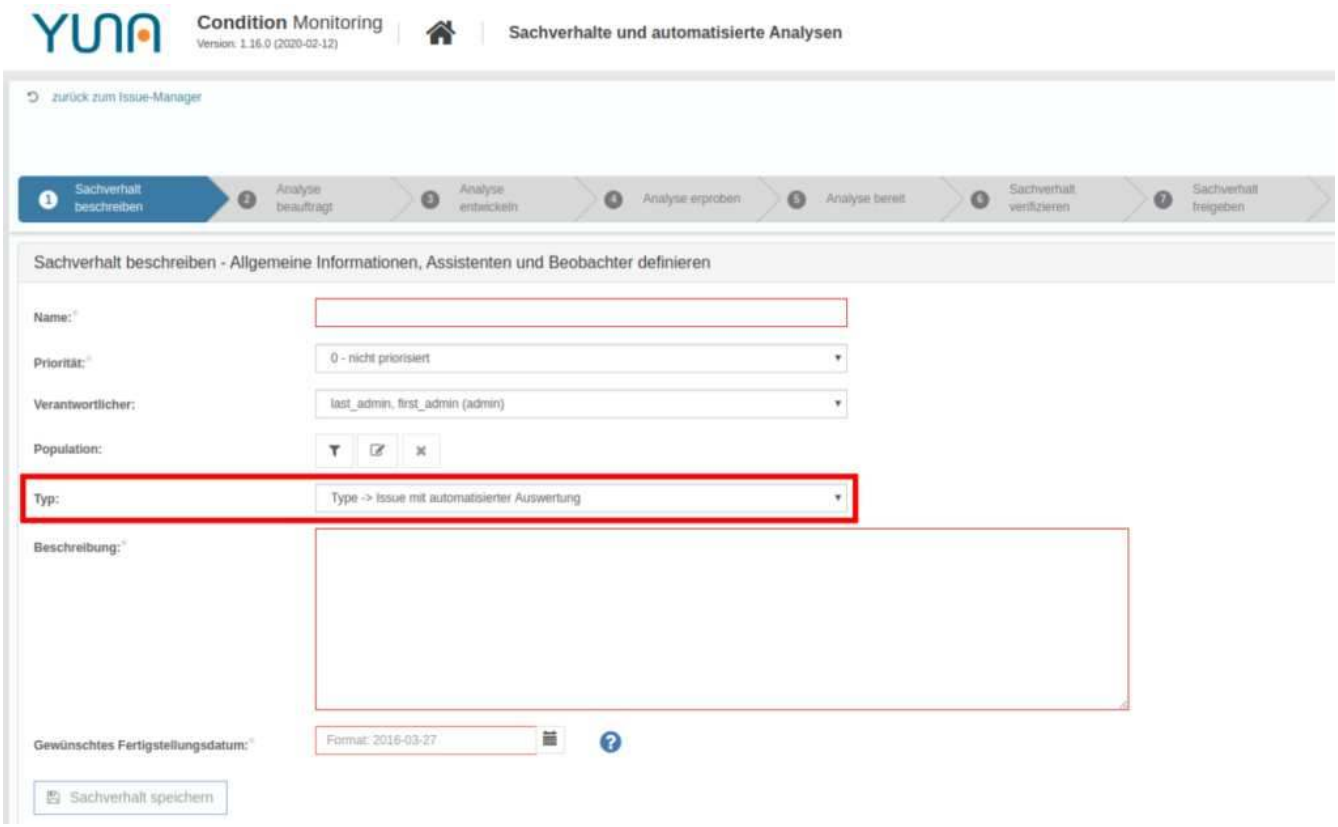
Sachverhaltsphase	Phasentitel	Inhalt / Funktionen	abgeschlossen wenn...
3	Analyse entwickeln	**Inhalt:** • Zur Untersuchung des Sachverhalts soll eine Analyse der zugrundeliegenden Daten erfolgen. Diese Analyse beruht auf einem Analyseskript (z.B. ein R-Skript). In dieser Phase wird das Skript ausgewählt, beschrieben, ggf. mit weiteren Parametern versehen und anschließend zur Erprobung freigegeben (Phase 4: Test der Funktionalität des Skripts auf den Daten) **Funktionen:** • Auswählen eines Analyseskripts (Auswahlmöglichkeiten zwischen bereits eingecheckten Skripten. Alternativ im Skript-Manager neues Skript erstellen und einchecken) • Beschreibung der Analyse und des genutzten Skripts • Festlegung eines Analyseverantwortlichen • Analyseinformationen speichern • Parameterdefinitionen zur Analyse hinzufügen und Beschreiben • Geheime Zugangsdaten für Analysejobs hinterlegen und verschlüsseln	• Analyseskript ist ausgewählt • Analyse ist beschrieben • Analyseverantwortlicher ist definiert • Informationen zur Analyse sind gespeichert und die Erprobung des Skripts wurde freigegeben
4	Analyse erproben	**Inhalt:** • In dieser Phase wird die Funktionalität des Analyseskripts auf den Daten geprüft. Hierzu werden Jobs definiert, die entweder manuell oder automatisch in frei definierbaren Zyklen das Skript ausführen. Abschließend wird bewertet, ob das Skript die gewünschten Ergebnisse liefert und zur weiteren Untersuchung des Sachverhalts genutzt werden kann,oder ob die Analyseinformationen, das Skript oder Jobs angepasst werden müssen. **Funktionen:** • Angelegte Jobs zum Sachverhalt, deren Status und die Ergebnisse anzeigen lassen • Neuen Job anlegen (im CE-Manager) • Ergebnisse zu Analysejobs anzeigen lassen • Erprobung abbrechen (Analyseinformationen, Skript oder job ändern) oder Speichern und Analyse zur Verifikation freigeben	• Job wurde ausgeführt, um Skript zu testen • Ergebnis ist zufriedenstellend und die Analyse wird freigegeben.

Sachverhaltsphase	Phasentitel	Inhalt / Funktionen	abgeschlossen wenn...
5	Analyse bereit	**Inhalt:** • In dieser Phase werden Anwender definiert, die für die Verifizierung des Sachverhalts zuständig sind. **Funktionen:** • Anwender hinzufügen (Einzelpersonen oder Gruppen) • Erprobung abbrechen oder abschließen und mit der Verifizierung des Sachverhalts beginnen.	• Anwender sind festgelegt • Erprobung wurde abgeschlossen und das Skript für die Verifikation des Sachveralts freigegeben.
6	Sachverhalt verifizieren	**Inhalt:** • Nachdem zuvor das Skript verifiziert wurde, wird in dieser Phase der Sachverhalt bzw. die Problemstellung verifiziert. **Funktionen:** • Angelegte Jobs zum Sachverhalt, deren Status und die Ergebnisse anzeigen lassen • Neuen Job anlegen (im CE-Manager) • Ergebnisse zu Analysejobs zur Sachverhaltsverifikation anzeigen lassen • Verifizierung des Sachverhalts abbrechen (Analyseinformationen, Skript oder job ändern, anschließend Analyse neu erproben) oder Speichern und Analyse für den Live-Betrieb freigeben	• Job wurde ausgeführt, um Sachverhalt zu verifizieren • Ergebnis ist zufriedenstellend und die Analyse wird in den Live-Betrieb überführt
7	Sachverhalt freigeben	**Inhalt:** • Nach einer Verifizierung der Ergebnisse kann entschieden werden, ob weitere Maßnahmen erforderlich sind oder nicht. Sind keine weiteren Maßnahmen erforderlich, so kann der Sachverhalt abgeschlossen werden. **Funktionen:** • Auswahl, ob weitere Maßnahmen erforderlich sind oder nicht • Informationen zu weiteren Maßnahmen definieren (optionale oder verpflichtende Maßnahme? Inhalt der Maßnahme? Link zu Tools wie JIRA oder TFS)	• Erforderlichkeit einer Maßnahme bestimmt wurde • Informationen zur weiteren Maßnahme definiert wurden

Sachverhaltsphase	Phasentitel	Inhalt / Funktionen	abgeschlossen wenn...
8	Sachverhalt produktiv	**Inhalt:** • Nachdem der Sachverhalt freigegeben wurde kann der Sachverhalt im Live-Betrieb untersucht werden. Hierzu wird ein Produktivjob angelegt, der das Skript in selbst definierbaren Intervallen oder manuell ausführt. Nach einer Beobachtung der Analyseergebnisse zum Sachverhalt über einen frei gewählten Zeitraum (Orientierung am geplanten Fertigstellungsdatum ist sinnvoll) kann entschieden werden, ob weitere Maßnahmen erforderlich sind oder nicht. Sind keine weiteren Maßnahmen erforderlich, so kann der Sachverhalt abgeschlossen werden. **Funktionen:** • Angelegte Jobs zum Sachverhalt, deren Status und die Ergebnisse anzeigen lassen • Neuen Job anlegen (im CE-Manager) • Ergebnisse zu Produktivjobs zum Live-Betrieb der Sachverhaltsuntersuchung anzeigen lassen • Untersuchung abbrechen, um Analyse zu verbessern oder Sachverhalt abschließen	• Job wurde ausgeführt, um den Sachverhalt im Live-Betrieb zu untersuchen • Ergebnis ist zufriedenstellend und der Sachverhalt wird abgeschlossen
9	Sachverhalt abgeschlossen	**Inhalt:** • Der Sachverhalt ist nun abgeschlossen. Für eine erneute Untersuchung bzw. Anpassung kann er jedoch erneut aufgenommen werden. **Funktionen:** • Sachverhalt wieder aufnehmen	---

Datenbankzugriffe aus verschiedenen Sachverhaltstypen

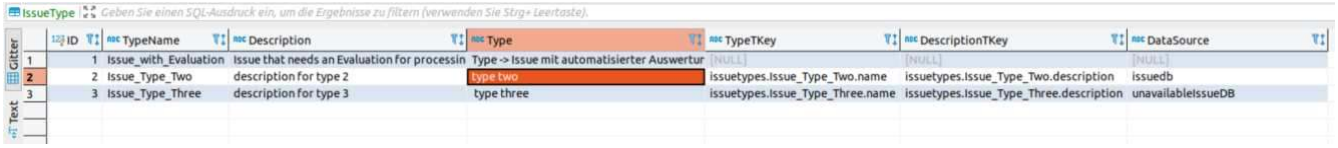
Für jeden Sachverhalt kann ein YUNA Systemadministrator im Sachverhaltsformular einen Sachverhaltstyp einstellen. Dem Sachverhalt kann in der Issuetypes Tabelle eine DataSource zugeordnet werden.



Issuetypes Tabelle

Die Grundlage für die auswählbaren Sachverhaltstypen bildet die Tabelle issuetypes in der PortalDB.

Wenn in der config.yaml kein Default eingestellt ist, dann wird der Sachverhaltstyp mit der ID 1 als Rückfallwert verwendet. Sollte ID 1 nicht vorhanden sein, startet das YUNA Portal nicht.



IssueType

ID	TypeName	Description	Type	TypeTKey	DescriptionTKey	DataSource
1	Issue_with_Evaluation	Issue that needs an Evaluation for processin	Type -> Issue mit automatisierter Auswertur	[NULL]	[NULL]	[NULL]
2	Issue_Type_Two	description for type 2	type two	issuetypes.Issue_Type_Two.name	issuetypes.Issue_Type_Two.description	issuedb
3	Issue_Type_Three	description for type 3	type three	issuetypes.Issue_Type_Three.name	issuetypes.Issue_Type_Three.description	unavailableIssueDB

Datenbanken für den Zugriff konfigurieren

Die Datenbank, auf die ein Sachverhaltstyp referenziert, kann über eine "ID" in der dse.conf.xml eingestellt werden.

Beispiel:

```
....
    <dbservice id="default">
        <url>jdbc:sqlserver://localhost:1433;applicationName=DSP-Default;encrypt=true;trustServerCertificate=true;</url>
        <driver>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver>
        <servername>localhost</servername>
        <user>SA</user>
        <name>DSE-PortalDB</name>
        <password>yourStrong(!)Password</password>
        <initialSize>20</initialSize>
        <maxTotal>100</maxTotal>
```

```
        <type>mssql</type>
    </dbservice>

    <dbservice id="spconnectserver">
        ...
    </dbservice>

    <dbservice id="defaultForIssues">
        ...
    </dbservice>
....
```

Default-Werte konfigurieren

Sachverhaltstyp

In der config.yaml kann der Default-Werte für den Typ des Sachverhalts (defaultTypeName) eingestellt werden. Wenn kein Default-Wert für den Sachverhaltstypen konfiguriert ist, dann wird der Sachverhaltstyp mit der ID 1 als Rückfallwert verwendet. Sollte ID 1 nicht vorhanden sein, startet das YUNA Portal nicht.

DataSource für Sachverhalte

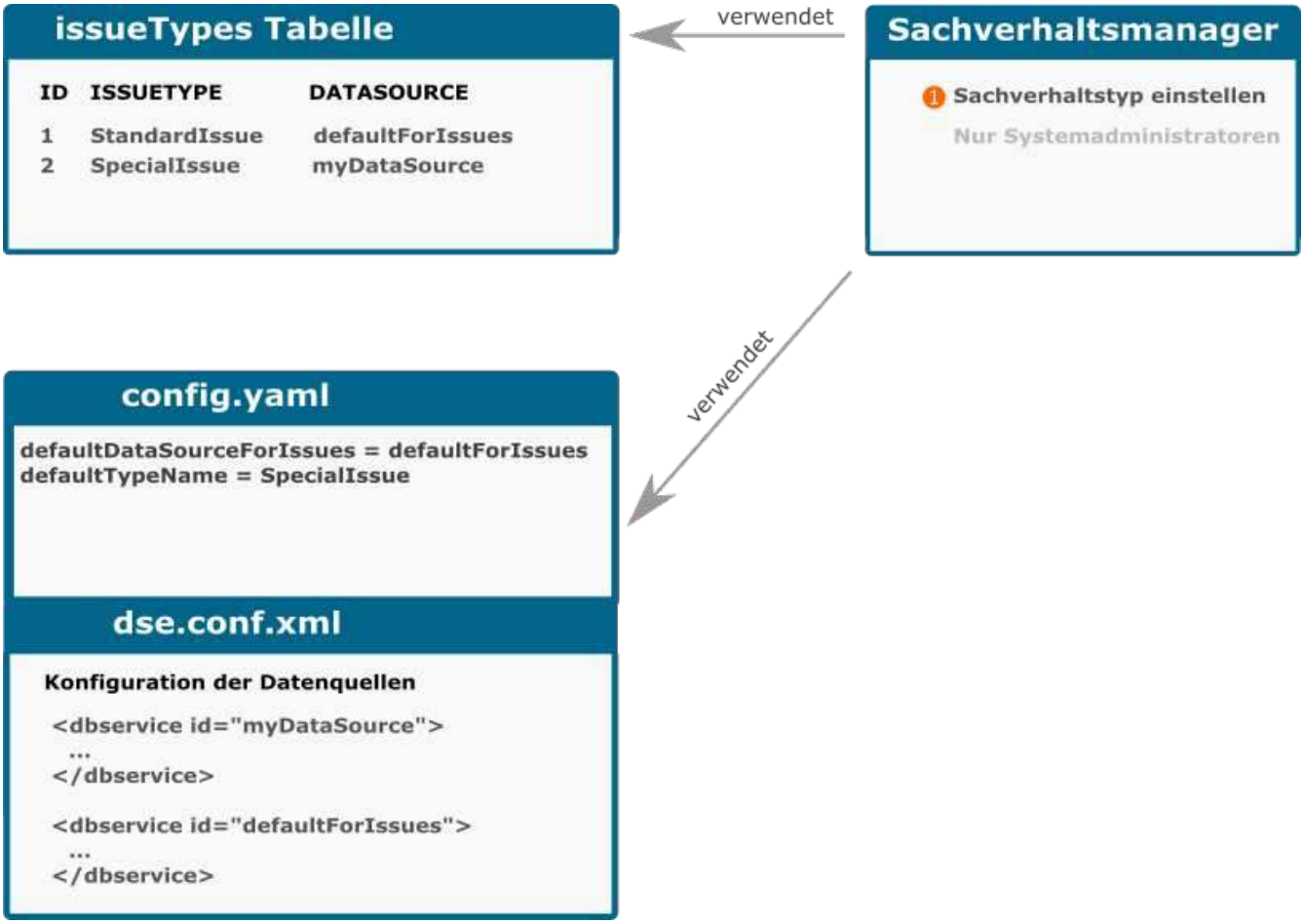
Ausserdem kann hier ein Default-Wert (defaultDataSourceForIssues) für die zu verwendende DataSource eingestellt werden, falls für einen verwendeten IssueType keine DataSource konfiguriert ist. Wenn in der config.yaml kein Default für "defaultDataSourceForIssues" eingestellt ist, dann wird als Rückfallwert der dbservice mit der id "spconnectserver" verwendet. Wenn dieser Eintrag nicht vorhanden ist, dann wird als Rückfallwert der dbservice mit der id "default"verwendet.

Beispiel:

```
de.eoda.dse.portal.issue.provider.IssueServiceProvider: {
  defaultDataSourceForIssues: "defaultForIssues",
  defaultTypeName: "Issue_with_Evaluation"
}
```

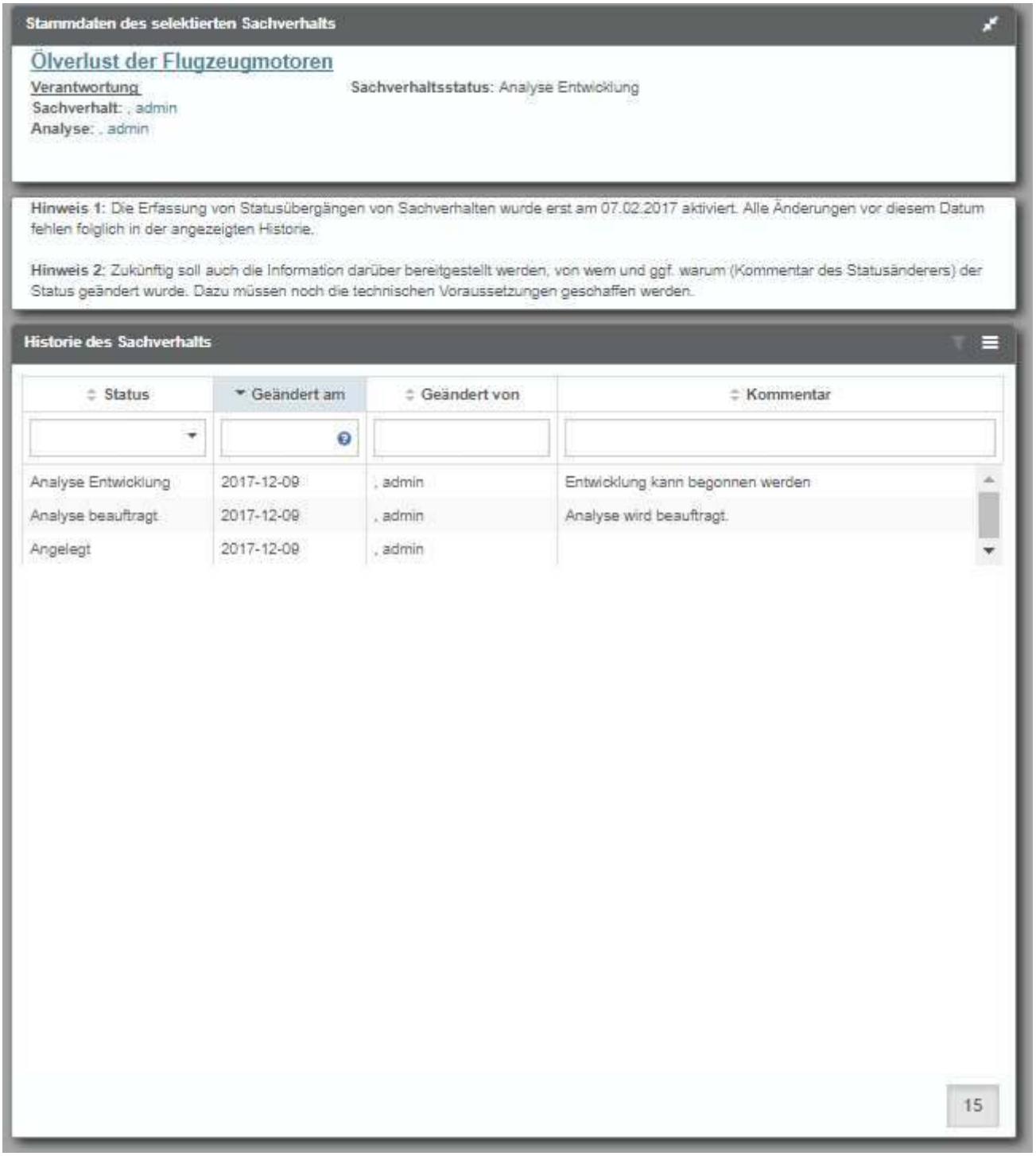
Im gezeigten Beispiel wird für den Sachverhaltstyp "StandardIssue" die DataSource "defaultForIssues" als Default definiert. Für den Sachverhaltstyp "SpecialIssue" wird die Datenquelle "myDataSource" eingetragen. Als Default für Sachverhaltstypen ist der Typ "SpecialIssue" eingestellt. D.h. wenn kein Sachverhaltstyp bei Sachverhalten angegeben ist, dann wird dieser Typ angenommen.

Sollten neue Sachverhaltstypen in der DB konfiguriert werden, für die keine Datasource eingetragen ist, dann greift der "defaultDataSourceForIssues" Eintrag aus der config.yaml und es wird die DataSource mit dem Namen "defaultForIssues" für die Skriptausführung verwendet.



Statushistorie

Um eine Nachverfolgbarkeit bei Änderungen an den Sachverhaltsanalysen gewähleisten zu können, existiert eine Statushistorie. In ihr wird aufgelistet, wer wann aus welchem Grund welchen Status geändert hat. Die Inhalte dieser Tabelle lassen sich sowohl durchsuchen als auch sortieren und exportieren.



Erklärungen zum Aufbau sowie zu den Eigenschaften und Funktionen von Jobs und dem Jobmanager finden sich in den nächsten Abschnitten.

Anlegen eines Sachverhaltsmanagers

XML

```
<xml>
  <widget name="template_widget_Issue">
    <!-- Position from left top  -->
    <position>
        <x>0</x>
```

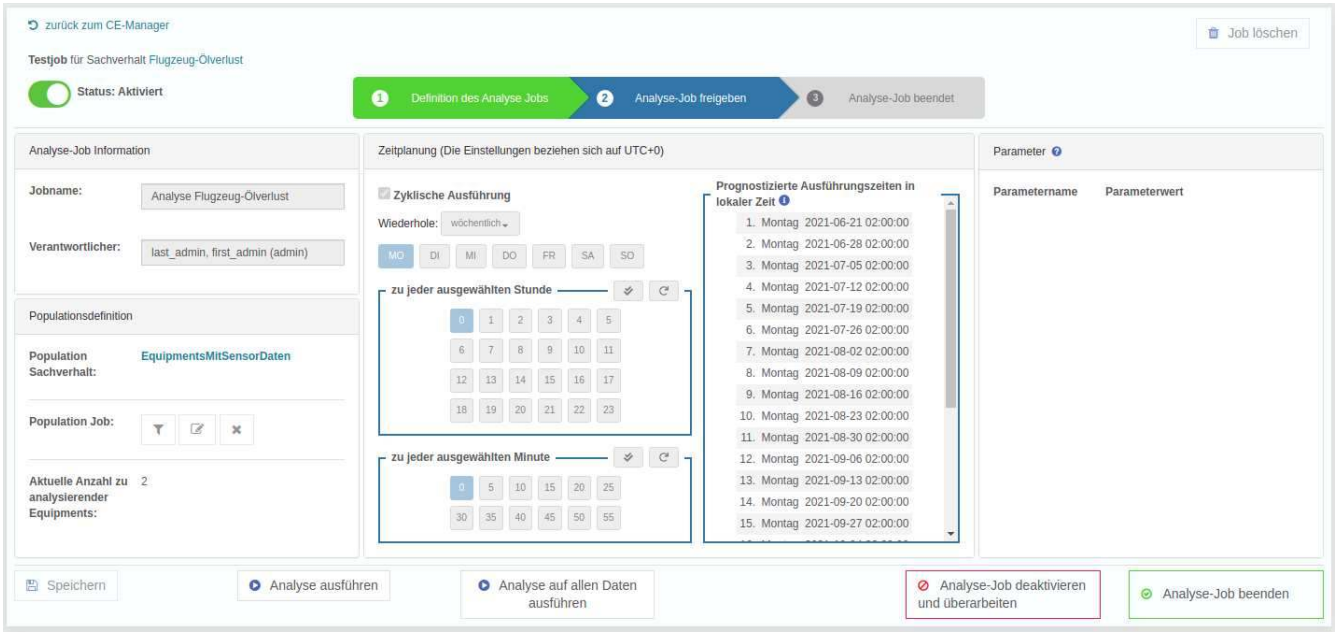
```
        <y>0</y>
    </position>
    <!-- Size of the Widget -->
    <size>
      <x>14</x>
      <y>7</y>
    </size>
    <!-- Name of the WidgetType -->
    <widgettype>issuedirective</widgettype>
  </widget>
</xml>
```

JSON

```
{
    "position": {
        "position": [0, 0]
    },
    "size": {
        "x": 14,
        "y": 7
    },
    "widgetname": "issuedirective",
    "triggerParams": []
}
```

Jobmanager

Im Jobmanager werden Jobs angelegt, Parameter und Filtermengen festgelegt sowie das Durchlaufen der Jobphasen gesteuert.



Die Phasen des Jobs

Phase 1: Definition des Analyse-Jobs

In dieser Phase werden die Rahmenbedingungen des Analysejobs definiert. So kann neben der Benennung des Jobs ein Zuständiger für den Analysejob ausgewählt werden. Zudem muss definiert werden, in welchem Zyklus der Job ausgeführt werden soll (stündlich, täglich, wöchentlich, monatlich oder manuell). Weiterhin kann ein Filter auf den Job angewand werden, um die mit diesem Job zu untersuchende Anlagenpopulation zu beschränken. Abschließend muss der Job gespeichert und freigegeben werden.

Phase 2: Analyse-Job freigeben

In dieser Phase wird der freigegebene Analyse-Job aktiviert oder deaktiviert und kann anschließend entweder ausgeführt, überarbeitet oder nach der Testphase abgeschlossen werden.

Phase 3: Analyse-Job beendet

Nachdem der Analyse-Job ausgeführt wurde, können im Sachverhaltsmanager die Ergebnisse des Jobs abgerufen werden. Dies öffnet eine neue View (z.B. die Result View), in der tabellarisch und grafisch die Ergebnisse der Analyse aufgeführt werden.

Durchlauf von Jobs am Wochenende

Wurden Jobs so definiert, dass sie am Wochenende laufen sollen, so werden sie derzeit in eine Warteschleife gelegt und am Montagmorgen gestartet. Dies liegt daran, dass erfahrungsgemäß keine Jobs am Wochenende laufen sollen, um am Wochenende einen Neustart oder eine Watung von Servern zu ermöglichen, ohne einen Abbruch von teils lange laufenden Jobs zu gefährden.

Ausführen von Jobs

Dem User stehen prinzipiell zwei Wege zur Verfügung, um die Analyse im Rahmen des Jobs auszuführen:

Button-Titel	Funktion	Kurzfassung
Analyse ausführen	Manuelles Starten des Jobs und Übergabe des Parameters "evaluateAllData" mit dem Wert FALSE an das Skript. Wird im Skript die Funktion skipEvaluation4Device verwendet, wird die Auswertung für ein Element nicht erneut durchgeführt, wenn seit dem letzten Auswertungsdurchlauf zu diesem Element keine Daten importiert wurden.	Keine Analyseausführung für Equipments ohne neue Daten seit letzem Durchlauf.

Button-Titel	Funktion	Kurzfassung
Analyse auf allen Daten ausführen	Manuelles Starten des Jobs und Übergabe des Parameters "evaluateAllData" mit dem Wert TRUE an das Skript. Wird im Skript die Funktion skipEvaluation4Device (Auswertung für ein Element wird nicht durchgeführt, wenn seit dem letzten Auswertungsdurchlauf zu diesem Element keine Daten importiert wurden) verwendet, liefert diese dann immer FALSE, auch wenn keine neuen Daten importiert wurden. Die Auswertung erfolgt dann für alle Elemente der Jobpopulation.	Analyse wird für alle Equipments ausgeführt, auch ohne neue Daten seit letztem Durchlauf.

Filter kopieren und referenzieren

Es ist möglich, die Population von Sachverhalten und Jobs durch Filter zu definieren. Hierfür kann entweder eine Population selbst erstellt und ein entsprechender Filter gespeichert werden. Es kann auch auf bereits bestehende Populationen, durch Auswahl eines gespeicherten Filters, zurückgegriffen werden. Bei der Auswahl einer bereits existenten Population kann vom Jobverantwortlichen zudem gewählt werden, ob der bestehende Filter kopiert oder ob auf ihn referenziert werden soll.

Im Fall einer Kopie wird im weiteren Verlauf des Job-Workflows für die Population die Bezeichnung "feste Definition" angezeigt. Dies bedeutet, dass die für den Job gewählte Population dauerhaft bestehen bleibt - auch, wenn anschließend jemand den Populationsfilter überarbeitet.

Wird die Option "Filter referenzieren" gewählt, so bleibt weiterhin die Populationsbezeichnung sichtbar und flexibel. Wird also nach der Wahl der Jobpopulation etwas am Filter geändert, so wird diese Änderung automatisch auch auf die Jobpopulation angewendet.

Weitere Details zur Umsetzung:

Population eines jobs

Die Population, über die eine automatisierte Auswertung zu einem Sachverhalt in Form eines Jobs erfolgt, ergibt sich immer aus der Schnittmenge der Population des Sachverhalts und der Population des Jobs.

Grundsätzlich

Für die Definition der gültigen Population eines Jobs gibt es folgende zwei Möglichkeiten:

1. Feste Definition der Population im job
2. Referenz auf einen gespeicherten Filter als Population des Jobs mit automatischem Update

Details zu Möglichkeit 1 -"Feste Definition der Population im Job"

- Es besteht die Möglichkeit einen gespeicherten Filter auszuwählen. Die Filterkriterien des gewählten Filters werden fest mit dem Sachverhalt verknüpft. Filterinfos wie der Name o.ä. werden nicht verknüpft.

- Die Definition der Population eines Jobs ist in den AdHoc-Analysen nicht über das Filterkonzept "Gespeicherte Filter laden" erreichbar.
- Eine Änderung der Population ist nur durch den Verantwortlichen des Jobs (nicht durch die Assistenten) im Job selbst möglich.
- Für eine Änderung der Population ist eine Deaktivierung und nach der Speicherung der Änderungen eine erneute Freigabe des Jobs nötig.

Details zu Möglichkeit 2 -"Referenz auf einen gespeicherten Filter als Population des Jobs mit automatischem Update"

- Es besteht die Möglichkeit einen gespeicherten Filter auszuwählen. Es wird eine Referenz auf den gewählten Filter mit dem Job verknüpft. Änderungen an dem Filter wirken sich automatisch auf den Job aus.
- Die entsprechenden Jobverantwortlichen werden über eine Änderung des refernzierten Filters per Mail informiert. Die Auswirkung erfolgt automatisch.
- Eine Änderung der Population auf Grund einer Änderung am referenzierten Filter hat keine Auswirkung auf den Status des Jobs.

Generell gilt:

- Ein gespeicherter Filter, auf den eine Referenz besteht, kann nicht gelöscht werden. Beim Versuch, einen solchen Filter zu löschen, wird der Anwender darauf hingewiesen, in welchen Sachverhalten / Jobs eine Referenz auf den entsprechenden Filter besteht und wer für diese verantwortlich ist. Wie in einem solchen Fall weiter verfahren wird, liegt in der Verantwortung der beteiligten Anwender.
- In der Filterverwaltung und im "Filter laden"-Dialog wird zu jedem Filter als weitere Information angezeigt, ob eine Referenz auf einen Filter besteht. Wenn ja, sieht der Nutzer in einer Detailsicht, in welchen Sachverhalten / Jobs eine Referenz auf den entsprechenden Filter definiert ist.

Löschen von Jobs

Die Möglichkeiten zum Löschen von jobs in Sachverhalten unterscheidet sich nach der Art der Jobs:

Jobtyp	Sachverhaltsphase	Bedeutung
Erprobungsjob	4 - Analyse erproben	In diesen Jobs wird die Funktionalität der Analyseskripte auf den Daten geprüft. Hierzu werden Jobs definiert, die entweder manuell oder automatisch in frei definierbaren Zyklen die Skripte ausführen. Abschließend wird bewertet, ob die Skripte die gewünschten Ergebnisse liefern und zur weiteren Untersuchung des Sachverhalts genutzt werden kann, oder ob die Analyseinformationen, die Skripte oder Jobs angepasst werden müssen.
Verifikationsjob	6 - Sachverhalt verifizieren	Nachdem zuvor die Skripte erprobt wurden, wird in dieser Sachverhaltsphase der Sachverhalt bzw. die Problemstellung anhand von Verifikationsjobs verifiziert.

Jobtyp	Sachverhaltsphase	Bedeutung
Produktivjob	7 - Sachverhalt freigeben	Nachdem Analyseskripte und Sachverhalte in den vorherigen Phasen getestet und verifiziert wurden, kann der Sachverhalt im Live-Betrieb untersucht werden. Hierzu werden erneut Analysejobs (=Verifikationsjobs) angelegt, die die Skripte in selbst definierbaren Intervallen oder manuell ausführen. Nach einer Beobachtung der Analyseergebnisse zum Sachverhalt über einen frei gewählten Zeitraum (Orientierung am geplanten Fertigstellungsdatum ist sinnvoll) kann entschieden werden, ob weitere Maßnahmen erforderlich sind oder nicht. Sind keine weiteren Maßnahmen erforderlich, so kann der Sachverhalt abgeschlossen werden. Sind hingegen weitere Maßnahmen erforderlich, so können Informationen zu den Maßnahmen hinterlegt werden (z.B. Tasks in JIRA oder TFS) und die Analyse steht erneut zur Verbesserung bereit.

Beim Löschen eines Jobs werden auch die erzeugten Ergebnisse des Jobs dauerhaft gelöscht

Ergebnisse von Jobs

Im Skriptlog können die Ergebnisse einzelner Ausführungen eines Jobs eingesehen werden.

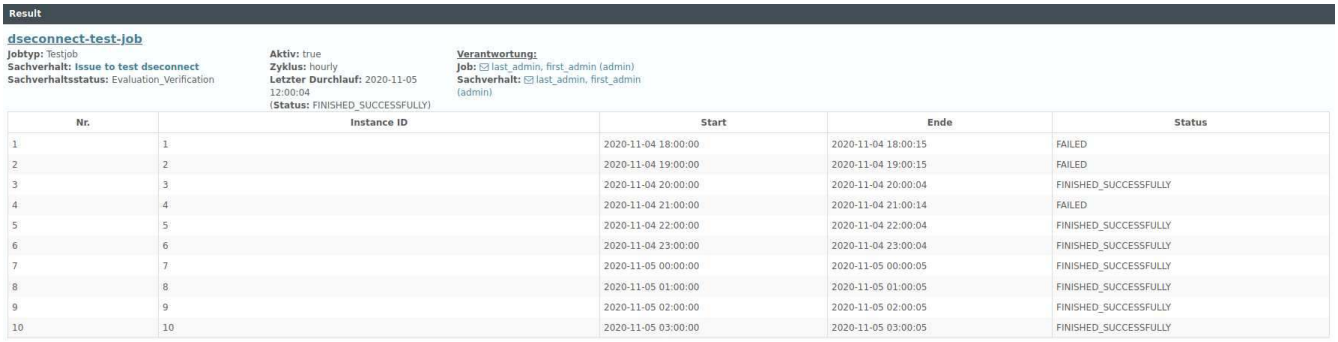
Wo finde ich die Ergebnisse eines Jobs

In der Übersicht aller Jobs findet sich eine Spalte ***Erg***. Falls mindestens ein Skriptlog Eintrag zu einem Job existiert ist in der Spalte ein Auflistung-Symbol zu sehen.

Mit klick auf das Symbol gelangt man in die Übersicht der Ausführungsergebnisse zu diesem Job.

Anschließend wir die Ergebnissicht des jeweiligen Jobs geöffnet. Um welche Sicht es sich dabei handelt, wird in Phase 3 "Analyse entwickeln" im Sachverhaltsmanager konfiguriert.

Standardmäßig wird hier auf die im folgenden Screenshot gezeigte Sicht zur Anzeige der verschiedenen Jobdurchläufe des jeweiligen Jobs verlinkt.



Jede Zeile ist verlinkt mit dem Skriptlog des jeweiligen Jobdurchlaufs. Es muss also nur auf die Zeile geklickt werden, deren Log man lesen möchte.Daraufhin öffnet sich ein Fenster mit dem Log der entsprechenden
Jobausführung.

Zur Verfügung stehende Parameter innerhalb einer Skriptsession

Parameter

Folgende Variablen stehen innerhalb der aktiven Skript-Session im Server-Mode zur Verfügung. Die Parameter können über die spconnect-Funktion 'getParam' abgerufen werden:

```
# Beispielabfrage für Issue ID
issueid <- spconnect::getParam('Issue_ID')
```

Parametername	Datentyp	Ausprägung (Beispiel)	Beschreibung
evaluateAllData	boolean	TRUE, FALSE	Skript wird auf allen Daten ausgeführt
manualExecution	boolean	TRUE, FALSE	Skript wird manuell ausgeführt
JobInstance_ID	character	'1'	Aktuelle Job Instanz ID
EvaluationJob_ID	numeric	2	Aktuelle Evaluation Job ID (aka Job ID)
Issue_ID	numeric	33	Aktueller Sachverhalt ID
EvaluationJobType	numeric	1,2,3	Aktueller Job Typ (1 = Erprobung, 2 = Verifizierung, 3 = Produktiv)

Parametername	Datentyp	Ausprägung (Beispiel)	Beschreibung
EvaluationJobStatus_ID	numeric	1,2,3	Aktueller Job Status ID (1= Definition, 2 = Analyse freigeben, 3 = Analyse beendet)
EvaluationInfo_ID	numeric	4	Aktueller Evaluation Info ID
scheduledTime	numeric	1553667436207	Startzeitpunkt des Jobs (Millisekunden seit 1970)
startTime	numeric	1553667436207	Startzeitpunkt des Jobs (Millisekunden seit 1970)
name 1	character	'param wert'	Inhalt des Parameters 'name 1'; dies kann ein beliebiger angelegter Parameter aus dem Issue-Formular sein, inklusive verschlüsselter Zugangsdaten

Skriptlog

Speicherung der Jobinstanzparameter und des Skriptlogs

In der YUNA Datenbank werden im Schema core in der Tabelle scriptlog alle relevanten Informationen zur Skriptausführung gesichert. Zusätzlich zum eigentlichen Skriptlog werden dort für jede Job-Instanz auch die Werte der Job-Parameter zum Zeitpunkt der Ausführung gesichert.

	id	jobinstance_id	session_id	level	log	number	source	timestamp	Qualifier
1	1	1	[NULL]	3		0	JobRunner	2019-10-31 12:26:55	4
2	2	1	[NULL]	3	param1=param1 Testing	1	JobRunner	2019-10-31 12:26:55	6
3	3	1	[NULL]	3	de.eoda.dse.core.action.job_id=1	2	JobRunner	2019-10-31 12:26:55	7
4	4	1	[NULL]	3	responsible_user=3	3	JobRunner	2019-10-31 12:26:55	8
5	5	1	[NULL]	3	issue_ID=1	4	JobRunner	2019-10-31 12:26:55	8
6	6	1	[NULL]	3	resourceVersion=1	5	JobRunner	2019-10-31 12:26:55	8
7	7	1	[NULL]	3	type=manuell	6	JobRunner	2019-10-31 12:26:55	8
8	8	1	[NULL]	3	resourceNodeId=2	7	JobRunner	2019-10-31 12:26:55	8
9	9	1	[NULL]	3	minute=0	8	JobRunner	2019-10-31 12:26:55	8
10	10	1	[NULL]	3	evaluationInfo_ID=1	9	JobRunner	2019-10-31 12:26:55	8
11	11	1	[NULL]	3	issueStatus=3	10	JobRunner	2019-10-31 12:26:55	8
12	12	1	[NULL]	3	filterInfo_ID=-1	11	JobRunner	2019-10-31 12:26:55	8
13	13	1	[NULL]	3	hour=0	12	JobRunner	2019-10-31 12:26:55	8
14	14	1	[NULL]	3	populationSubFilter=null	13	JobRunner	2019-10-31 12:26:55	8
15	15	1	[NULL]	3	jobResultLink=null	14	JobRunner	2019-10-31 12:26:55	8
16	16	1	[NULL]	3	evaluationJobStatus_ID=2	15	JobRunner	2019-10-31 12:26:55	8
17	17	1	[NULL]	3	day=0	16	JobRunner	2019-10-31 12:26:55	8
18	18	1	[NULL]	3	evaluationJobType_ID=1	17	JobRunner	2019-10-31 12:26:55	8
19	19	1	7f09cddb-7b00-4fdd-9512-edebcb2df4c9	1	¶R version 3.6.0 (2019-04-26) -- "Planting of	0	r-agent	2019-10-31 12:26:56	0
20	20	1	7f09cddb-7b00-4fdd-9512-edebcb2df4c9	1	> library(coreConnectivity)¶	1	r-agent	2019-10-31 12:26:56	0

Werte für Qualifier-Schlüssel

Anhand des Eintrags in der Spalte Qualifier kann man die Einträge in der Tabelle nach Art der Information filtern.

Wert	Parameter	Erläuterung
0	TEXT	Das eigentliche Skriptlog aus der Ausführungsumgebung des Agenten
1	RUNTIME_INFO	Generelle Laufzeitinfiormationen
2	PACKAGE_LOAD_INFO	Informationen über die in die Laufzeitumgebung des Agenten geladenen Pakete

Wert	Parameter	Erläuterung
3	PACKAGE_UNLOAD_INFO	Informationen über die aus der Laufzeitumgebung des Agenten entfernten Pakete
4	JOB_START	Information zum Jobstart
5	JOB_FINISH	Information zum Jobende
6	JOB_PARAMETER	Werte der konfigurierbaren Jobparameter
7	JOBINSTANCE_PARAMETER	Werte spezieller Jobinstanzparameter, z.B. das Flag zur manuellen Ausführung (manualExecution), oder Flag für einen gesetzten Filter (evaluateAllData-Flag)
8	JOB_FIELD	Werte aller intern gesetzten Jobparameter, z.B. der Ausführungstyp (type), die ID des zugehörigen Issue (issue_ID) oder der für die Population eingestellte Filter (populationSubFilter)
9	JOB_SCHEDULED	Information zum Zeitpunkt, wenn der Job eingeplant wurde
10	JOB_CANCELLED	Information zum Zeitpunkt, wenn der Job abgebrochen wurde
20	SCRIPT_SESSION_START	Eine Script-Session wurde gestartet
21	SCRIPT_SESSION_STOP	Eine Script-Session wurde beendet
22	NODE_CONTENT_ID	Informationen zur aktuell ausgeführten nodecontent_id

Werte für Level

Die Spalte Level git die Klassifikation des Log-Eintrages an.

Wert	Methode	Erläuterung
1	VERBOSE	Detailinformtaionen, u.a. das Konsolenlog aus der Ausführungsumgebung
2	DEBUG	Weiterführende Informationen zur Fehlerdiagnose
3	INFO	Standardinformationen zur Jobausführung.
4	WARN	Warnungen zu Problemen, die bei der Ausführung aufgetreten sind
5	ERROR	Informationen zu Fehlern bei der Ausführung
6	FATAL	Informationen zu schweren Fehler, die zu einem Systemabbruch führen

Source

In der Spalte Source wird die ausführende Instanz die den Logeintrag verursacht angegeben.

Wie kann ich die Werte der Job-Parameter für einzelne Jobinstanzen ermitteln?

Datenbankview

Um die im Skriptlog gespeicherten Parameterwerte einzelner Jobinstanzen leichter auslesen zu

können, wird im Schema portal die Datenbankview JobInstanceParameterView bereitsgestellt.

Anzeige aller Jobparameter durch Abfrage der View *Quelle erweitern*

```
SELECT * FROM [portal].[JobInstanceParameterView]
```

Diese zeigt für jede Jobinstanz die geloggten Jobparameter und ihre Werte sowie den Qualifier an.

	jobinstance_id	param	value	Qualifier
1	1	responsible_user	3	8
2	1	issue_ID	13	8
3	1	resourceVersion	1	8
4	1	description	null	8
5	1	type	hourly	8
6	1	resourceNodeId	6	8
7	1	minute	0	8
8	1	evaluationInfo_ID	4	8
9	1	issueStatus	3	8
10	1	filterInfo_ID	-1	8
11	1	hour	0	8
12	1	populationSubFilter	null	8
13	1	jobResultLink	null	8
14	1	evaluationJobStatus_ID	2	8
15	1	evaluationJobType_ID	1	8
16	1	day	0	8
17	2	responsible_user	3	8
18	2	issue_ID	13	8
19	2	resourceVersion	1	8
20	2	description	null	8
21	2	type	hourly	8

Rest Endpunkt

Unter der URL <***your-host*>*/backend/de.eoda.dse.core.job.rest/logs/parameter/{jobInstanceid}*** ist ein Rest Endpunkt erreichbar, über den die Skriptlogeinträge für alle Jobparameter zu den einzelnen Jobinstanzen abgefragt werden können.Dazu muss lediglich die **jobinstanceid** angegeben werden.

Die Jobparameter werden über die Qualifier 6, 7 und 8 (vgl. hierzu Qualifier Tabelle) in der Skrriptlog-Tabelle gefiltert und durch das Splitten am "=" Zeichen des Eintrags in der Spalte log in Key-Value-Paare zerlegt.

URI	Path Parameter	Beispiel Wert	Komplettes Beispiel
/backend/de.eoda.dse.core.job.rest/logs/parameter/{jobInstanceid}	jobinstanceid	15	/backend/de.eoda.dse.core.job.rest/logs/parameter/15

Als Ergebnis eines Aufrufs der Restschnittstelle erhält man ein JSON Liste mit allen Jobparametern zu der angegebenen Jobinstanz. Ein Jobparameter wird darin jeweils durch ein Objekt mit drei

Schlüssel-Wert-Paaren, die die Attribute beschrieben, repräsentiert:

- der Schlüssel **key** hat als Wert den Namen des Jobparameters,
- der Schlüssel **value** hat als Wert des Jobparameters,
- der Schlüssel **qualifier** hat als Wert den zugehörigen Qualifier aus der Scriptlogtabelle.

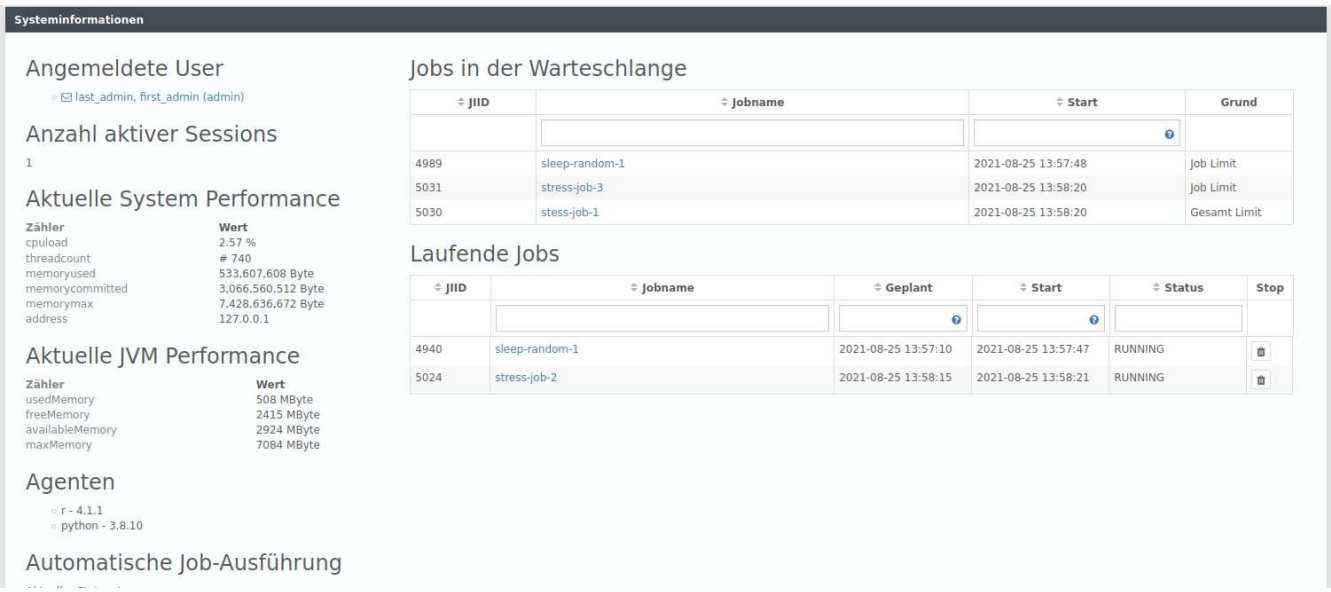
Beispiel für das Format der JSON Antwort

```
[
  {
    "qualifier": "JOB_FIELD",
    "key": "responsible_user",
    "value": "3"
  },
  {
    "qualifier": "JOB_FIELD",
    "key": "issue_ID",
    "value": "13"
  },
  {
    "qualifier": "JOB_FIELD",
    "key": "resourceVersion",
    "value": "1"
  },
  ...
]
```

5.6.8. Systemübersicht (systemInfo)

Die Systemübersicht ist eine View für Systemadministratoren und liefert Informationen über angemeldete Benutzer, Jobs und der Systemperformance. Sie wird über den Widget-Typ_ **"systeminfo"**_ als Widget definiert.

Systeminformationen - Übersicht



Automatische Jobausführung

Jobs können über eine Cron-Expression regelmäßig ausgeführt werden. Um dies zu verhindern, können System-Administratoren die automatische Jobausführung deaktivieren.Damit werden Systemweit Jobs nicht mehr automatisch ausgeführt.

Automatische Job-Ausführung deaktiviert

- Jobs, die über eine Cron-Expression regelmäßig ausgeführt werden, werden **nicht** gestartet.
- Aktuell ausgeführte Jobs werden **nicht** abgebrochen.
- Jobs, die durch das deaktivieren der Automatischen Job-Ausführung nicht gestartet wurden, werden nicht nachgeholt

Beispiel

```
<xml>
  <widget name="template_widget_Systeminfo">
    <!-- Position from left top  -->
    <position>
        <!-- y -->
        <y>0</y>
        <!-- x -->
        <x>0</x>
    </position>
    <!-- Size of the Widget -->
    <size>
      <x>15</x>
      <y>6.5</y>
    </size>
    <!-- Caption: title over the widget and additional features (Contextmenue) -->
    <caption>
      <!-- Display caption -->
      <show>true</show>
      <!-- Title of the Widget -->
      <label>Systeminfo-Template</label>
    </caption>
    <!-- Name of the WidgetType -->
    <widgettype>systeminfo</widgettype>
    <!-- No URL-Trigger-Params needed -->
    <triggerParams/>
    <!-- Interval to refresh the view in ms (-1 for no auto refresh) -->
    <systeminfo>
      <intervall>2000</intervall>
```

```
    </systeminfo>
  </widget>
</xml>
```

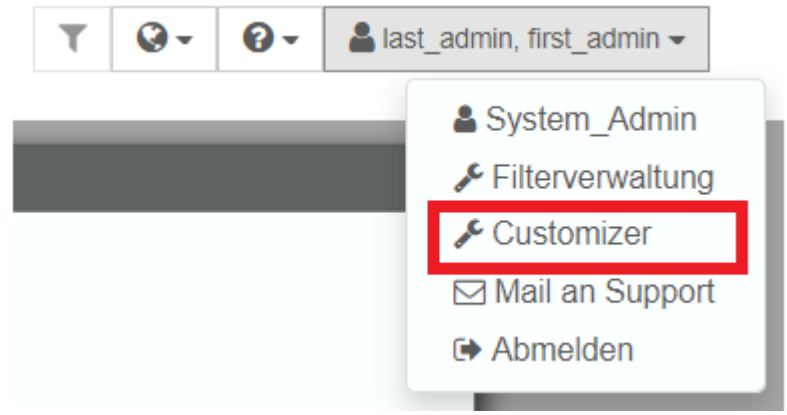
Feld	Mögliche Werte	default	Beschreibung
widgettype	systeminfo		Legt den Widgettyp als Systeminfo fest.
position	Werte für X und Y	0 / 0	Definiert die Position des Widgets im Grid
size	Werte für X und Y	0 / 0	Definiert die Größe des Widgets
caption			Für die Systeminfo kann eine Caption definiert werden, um ihr eine Überschrift zu geben
.interval	Zahl	1000	• Zeitintervall in Milisekunden, in dem die Information automatisch aktualisiert wird. • Negative Werte (-1) für keine Aktualisierung

5.6.9. Themes Manager

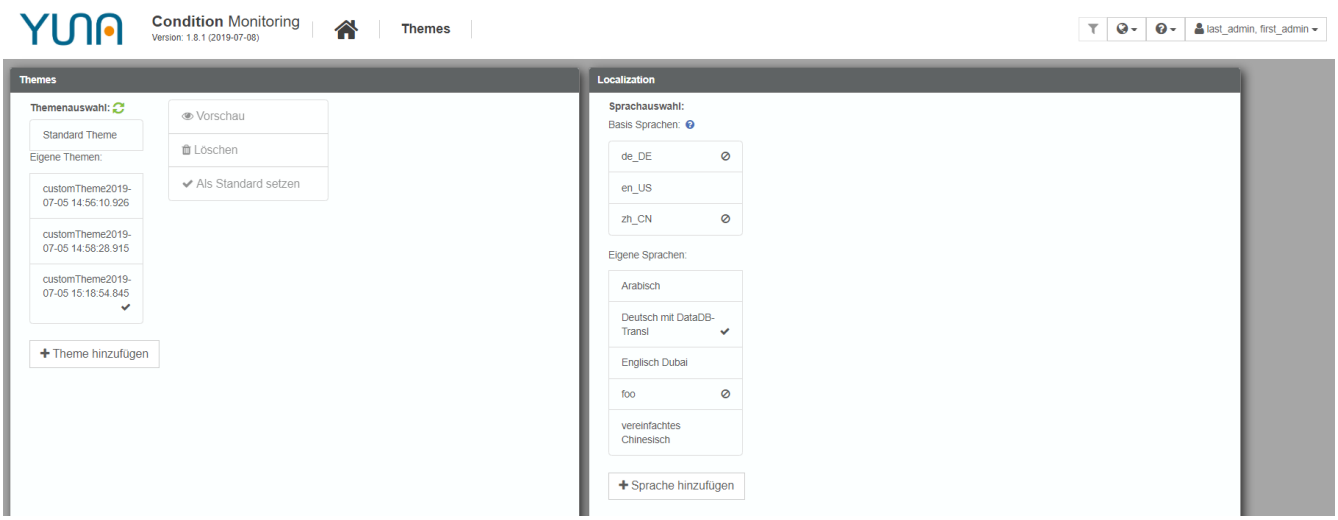
Mit dem Themes Manager können sie die Darstellung Ihres YUNA Portals anpassen. Über ein CSS Sytlesheet können z.B. Formatierungen und Farbgebung der einzelnen Elemente wie Überschriften, Absatztexte oder Hintergrundfarben eingestellt werden. Außerdem können sie über den Themes Manager Ihr Logo anpassen.

Zum Themes Manager navigieren

Klicken sie im Hauptmenü auf Customizer



Themes Manager - Übersicht



Nutzung von bestehender Portal-Themes

Der Themes Manager listet alle derzeit im Portal verfügbaren Themes auf.

Neben dem Hinzufügen und Löschen von Themen, stellt der Theme Manager weitere Funktionen bereit:

- über die Vorschau-Aktion existiert die Möglichkeit, sich ein bestehendes Thema anschauen zu können, ohne das dies Auswirkungen auf andere Nutzer hätte.
- Mittels der Aktion "Als Standard setzen" kann das derzeit für alle Nutzer aktive Thema ausgewählt werden.

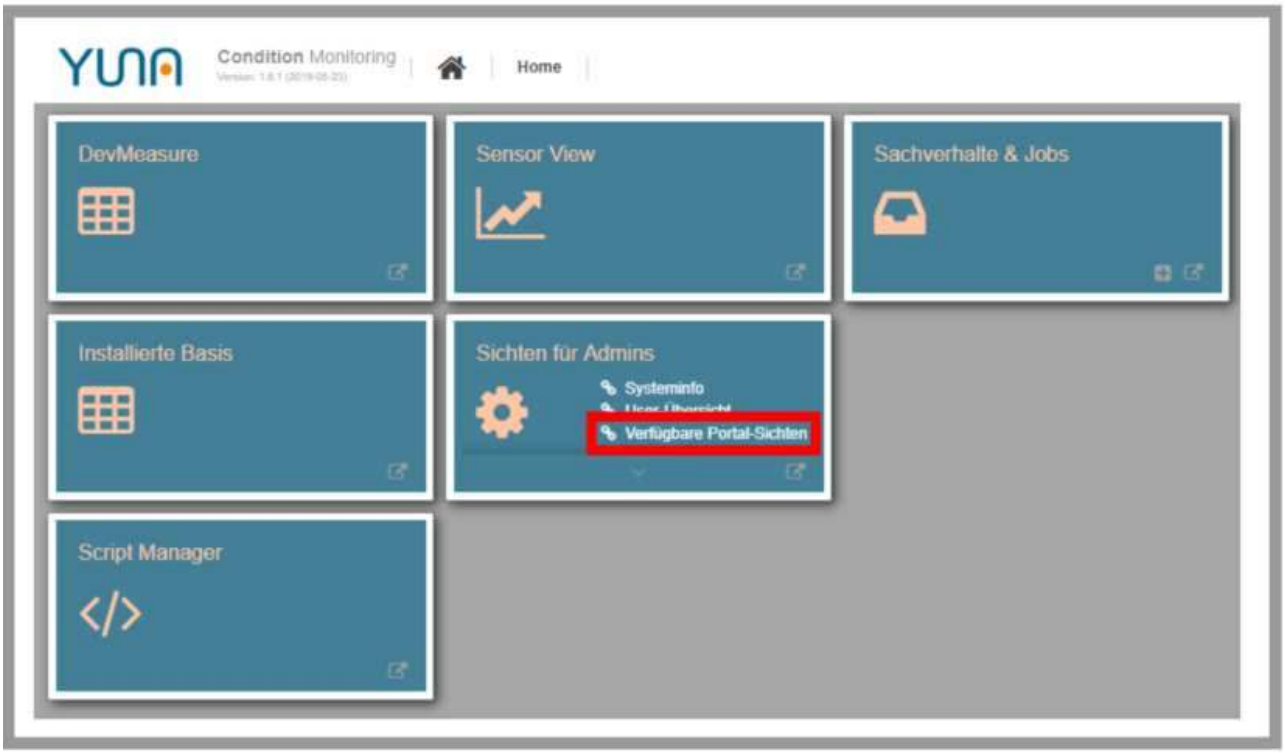
Erstellung neuer Portal-Themes

Über den Button-"Theme hinzufügen" können neue Themes angelegt werden.

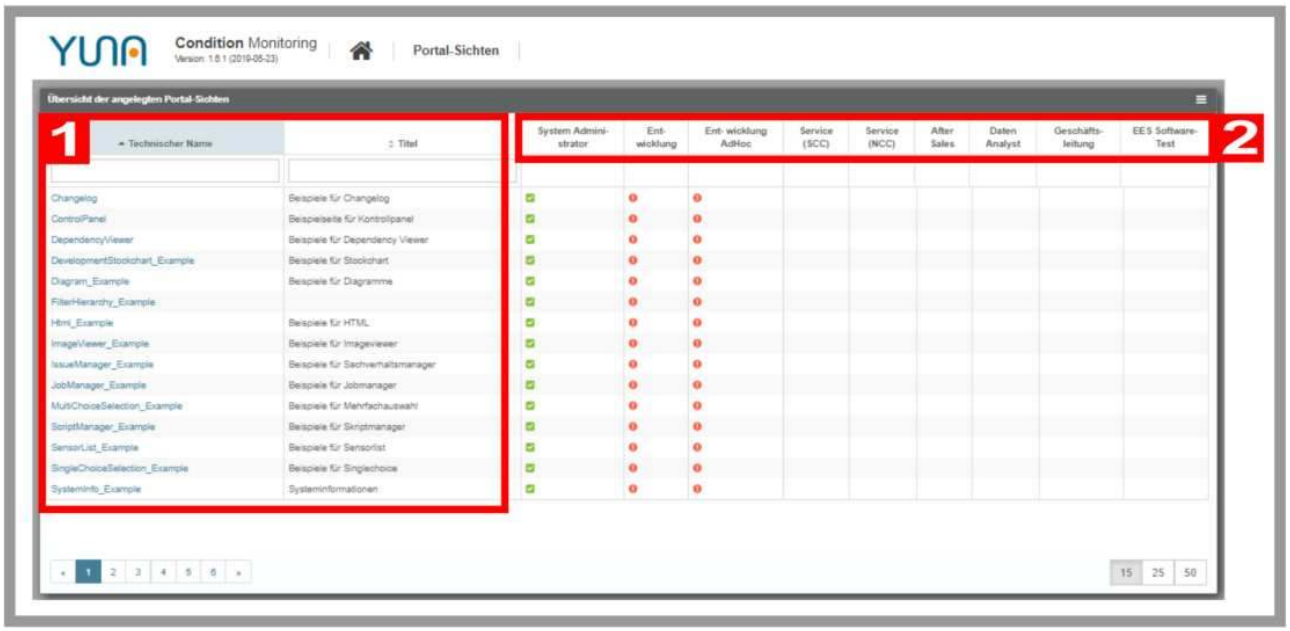
5.6.10. Verfügbare Portal-Sichten

Hier werden alle im Portal verfügbaren Views angezeigt und welche Benutzerrollen auf diese Zugriff haben.

1. **Auf verfügbare Portal-Sichten klicken**



2. **Verfügbare Portal-Sichten**



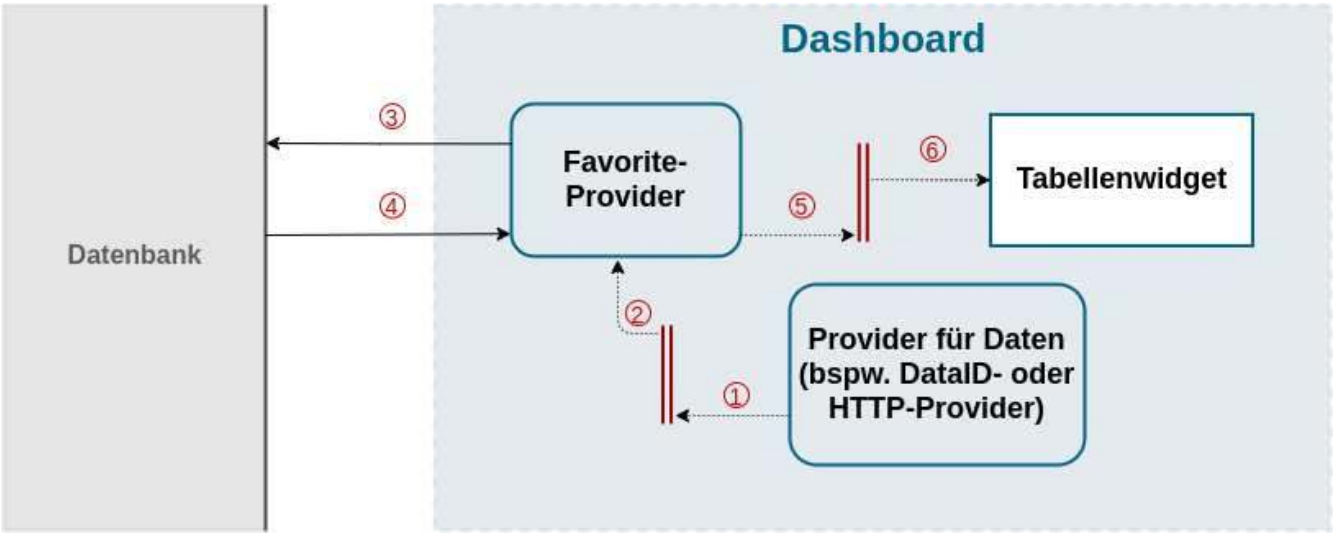
1. Verfügbare Views
2. Benutzerrollen

5.7. IO-Provider

5.7.1. Favorite-Provider

Der Favorite-Provider ermöglicht es beliebige Eingangsdaten mit Favoriteninformationen zu verknüpfen und diese anschließend in einer Tabelle darzustellen.

Beispielgrafik: Favorite-Provider im Zusammenspiel mit einem Tabellenwidget



1. Der für den Datenabruf zuständige Provider ruft Daten ab und publisht diese in seinen Output-Channel
2. Die durch den IO-Channel bereitgestellten Daten werden durch den Favorite-Provider konsumiert, da dieser IO-Channel als Input konfiguriert ist
3. Der Result-Rating-Provider fragt die Datenbank nach bereits markierten Favoriten für den konfigurierten Identifier und den aktuell eingeloggten Benutzer ab
4. Die Datenbank liefert die bereits abgegebenen Ergebnisse an den Favorite-Provider
5. Der Favorite-Provider sendet die in 2. eingegangen Daten zusammen mit den Favoritendaten an seinen konfigurierten Output-Channel
6. Das Tabellenwidget empfängt die in 5. gesendeten Daten auf seinem konfigurierten Input-Channel

YUNAML

Beispielkonfiguration für Favorite-Provider:

```
<io-provider>
    <type>Favorite</type>
    <config>
        <input>InputChannel</input>
        <output>OutputChannel</output>
        <identifierTemplate>{{id}}</identifierTemplate>
        <qualifier>JOB</qualifier>
    </config>
</io-provider>
```

Konfiguration des Favorite-Providers

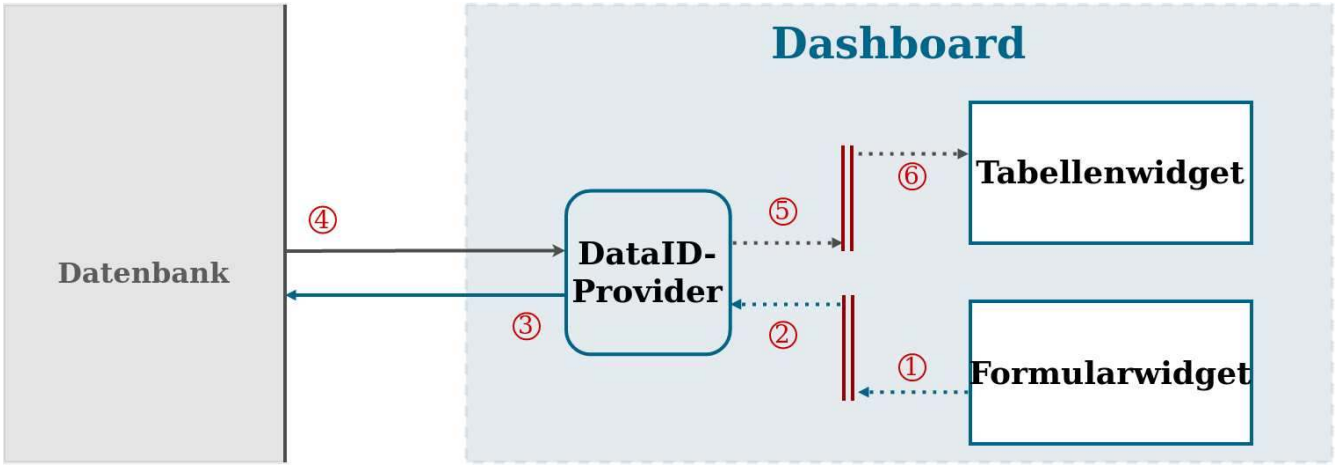
YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
type	Der Typ des IO-Providers. Für den Favorite-Provider muss dieser Parameter auf "Favorite" gesetzt werden.	✅	`<type>Favorite</type>`

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
config	Die verschiedenen Konfigurationsparameter des Result-Rating-Providers.	✅	
config > **input**	Unter <**input**> wird der Name des Input-Channels konfiguriert, welcher die zu bewertenden Daten liefert. Dieser kann beispielsweise über einen DataID- oder Http-Provider bereitgestellt werden.	✅	`<input>InputChannel</input>`
config > **output**	Unter <**output**> wird der Name des Output-Channels konfiguriert, in welchem die mit Favoriteninformationen angereicherten Eingangsdaten veröffentlicht werden.	✅	`<output>OutputChannel</output>`
config > **identifierTemplate**	Angabe eines Handlebar-Templates, um einen bereits im Eingangsdatensatz vorhandenen Schlüssel zur Identifikation einzelner Zeilen zu nutzen. **Wichtig**: Eine nachträgliche Änderung des identifierTemplates kann leicht dazu führen, dass Daten und Ergebnisse nicht mehr zugeordnet werden können.	✅	`<identifierTemplate>{{id}}</identifierTemplate>`
config > **qualifier**	Schlüssel unter welchem die Favoriten-Daten gespeichert werden. Es gibt mehrere Qualifier, die bereits im System verwendet werden, und hier weiterverwendet werden können: • JOB • ISSUE • DASHBOARD Soll einer dieser Favoritentypen verwendet werden, muss der Identifier ebenfalls dazu passend konfiguriert werden und die ID des jeweiligen Jobs, Sachverhalts bzw. Dashboards bezeichnen. Bei Fragen dazu oder Problemen bei der Umsetzung, wenden sie sich bitte an den YUNA-Support	✅	`<qualifier>JOB</qualifier>`

5.7.2. DataID-Provider

Der DataID-Provider ermöglicht es eine DataID auszuführen und das Ergebnis in YUNA-Dashboards zu integrieren.

Beispielgrafik: Ausführung einer DataID gesteuert über ein Formularwidget und anschließende Darstellung in einem Tabellenwidget:



Output-Channel

1. Bei Betätigung des Absenden-Buttons im Formularwidget wird die DataID an den Output-Channel des Formularwidgets übergeben.
2. Die durch den IO-Channel bereitgestellten Daten werden durch den DataID-Provider konsumiert, da dieser IO-Channel als Input konfiguriert ist.
3. Der DataID-Provider führt die DataID auf der Datenbank aus.
4. Die Datenbank liefert das Ergebnis der DataID an den DataID-Provider.
5. Das Ergebnis wird in den Output-Channel des DataID-Providers übergeben.
6. Das Tabellenwidget erhält die neuen Daten aus dem IO-Channel und kann diese darstellen.

YUNA ML

Konfigurationen für DataID-Provider:

Die verschiedenen YUNAML-Elemente sind in der nachfolgenden Tabelle erläutert.

Kursive Elemente beziehen sich auf andere Inhalte ihrer YUNAML-Dashboards, wie z.B. Namen von DataIDs oder IO-Channels.

YUNA-ML Beispieldefinition für DataID-Provider

```
<io-provider>
    <type>DataId</type>
    <config>
        <dataId>qy_nameOfADataId</dataId>
        <params>
            <parametername>OptionalInputChannel.parameterValue</parametername>
        <params>
        <triggerParams>
            <optional>OptionalInputChannel</optional>
```

```
        </triggerParams>
        <output>OutputChannel</output>
    </config>
</io-provider>
```

Konfiguration des DataID-Providers

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
type	Der Typ des IO-Providers. Für den DataID-Provider muss dieser Parameter auf "DataID" gesetzt werden. Weitere mögliche Werte sind "UrlParams" und "HTTP".	✅	<type>DataId<http>
config	Die verschiedenen Konfigurationsparameter des DataID-Providers.	✅	
config > **dataId**	Die auszuführende DataID des DataID-Providers.	✅	**<dataId>*_dataId Name_**</dataId>*

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
config > **triggerParams** config > triggerParams > **trigger** config > triggerParams > **mandatory** config > triggerParams > **optional**	Um den Zeitpunkt für die Ausführung der DataID zu konfigurieren, werden die Trigger-Parameter in drei YUNAML-Tags definiert: <**trigger**>, <**mandatory**> und <**optional**>. Bei jeder Änderung eines Channels, der in mindestens einem dieser Tags definiert ist, wird geprüft, ob eine Anfrage durchgeführt werden soll. Dabei werden folgende Bedingungen geprüft: 1. Nur wenn Channel in <**trigger**> definiert sind: Wurde ein <**trigger**>-Channel aktualisiert? 2. Nur wenn Channel in <**mandatory**> definiert sind: Wurden alle <**mandatory**> -Channel mit Daten befüllt? Sind weder <**trigger**>- noch <**mandatory**>-Channel definiert, wird bei jeder Änderung eines in <**optional**> angegebenen Kanals eine Anfrage ausgeführt. Ein Channel kann sowohl <**trigger**> als auch <**mandatory**> sein. Sollen mehrere Channel in einem der drei Tags definiert werden, müssen diese in <list>-Tags angegeben werden.	✗	<**triggerParams**> <**trigger**>****someChannel**</trigger>* <**mandatory**> <**list**>****mandatoryChannel**</list>* </**mandatory**> <**optional**> <**list**>****optionalChannel1**</list>* <**list**>****optionalChannel2**</list>* </**optional**> </**triggerParams**>

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
config > **params**	Um Parameter für die Abfrage der DataID zu definieren, wird der Tag <**params**> verwendet. Die einzelnen Parameter, die an die DataID übergeben werden sollen, werden über weitere Tags definiert. Dabei entscheidet der umschließende Tag über den Namen, unter dem der Parameter übergeben wird. Der Wert bezieht sich auf einen der konfigurierten InputChannels und wird als Selektor evaluiert. Das bedeutet, dass Eigenschaften verschachtelter Objekte über "." getrennt werden, während Elemnte einer Liste über eckige Klammern und den jeweiligen Index (startend bei 0) selektiert werden, z.B. "[0]" für das erste Element. **Beispiel für die Übergabe von Parametern an einen DataID-Provider** Daten im Inputchannel "SelectedChannel"**:** Dieser Inputchannel wird mit den selektierten Zeilen einer Tabelle befüllt (→ Outputchannel "selected" des Tabellenwidgets) `[{"ID": 1, "cellValue": "My Value"}]`	✗	<**params**> *<**parametername**>OptionalInputChannel.parameterValue</**parametername**>* </**params**>
config > **output**	Unter <**output**> wird konfiguriert, in welchen IO-Channel das Ergebnis der DataID veröffentlicht wird.	✗	<**output**>**Output Channel*</output>*

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
config > **converter**	Dieser Konfigurationsparameter sollte nur in wenigen Anwendungsfällen angepasst werden und ist daher für erfahrene Anwender vorgesehen. Definiert, in welchem Format das Ergebnis der DataID-Abfrage in den Output-Channel veröffentlicht wird. Es werden zwei Formate unterstützt: 'row' und 'col'. Für Details siehe das <<dataidprovider_beispielzu_converter_formaten, Beispiel zu <converter>-Formaten.	 Default: **row**	<**converter**>col</**converter**>

Konfiguration des DataID-Providers:

Hier wird über "<myParameterName>SelectedChannel[0].cellValue</myParameterName>" konfiguriert, dass ein Parameter mit dem Namen "myParameterName" an die DataID übergeben wird. Dieser hat den Wert "My Value", da aus dem Inputchannel "SelectedChannel" der erste Eintrag der Liste ("[0]") und schließlich die Eigenschaft "cellValue" (".cellValue") selektiert wird.

```
<io-provider>
    <type>DataId</type>
    <config>
        <dataId>qy_nameOfADataId</dataId>
        <params>
            <myParameterName>SelectedChannel[0].cellValue</myParameterName>
        <params>
        <triggerParams>
            <mandatory>SelectedChannel</mandatory>
        </triggerParams>
        <output>OutputChannel</output>
    </config>
</io-provider>
```

Beispiel zu <converter>-Formaten

Selektierte Daten	Die veröffentlichten Daten im 'row'-Format	Die veröffentlichten Daten im 'col'-Format

ID	Name
1	Heinz
2	Sabine
3	Stefan

```
{
    "dataQueryResult": {
        "rows": [
            {"ID": 1, "Name": "Heinz"},
            {"ID": 2, "Name": "Sabine"},
            {"ID": 3, "Name": "Stefan"}
        ],
        "header": ["ID", "Name"]
    }
}
```

```
{
    "dataQueryResult": {
        "columns": {
            "ID": [1, 2, 3],
            "Name": ["Heinz", "Sabine",
"Stefan"]
        },
        "header": ["ID", "Name"]
    }
}
```

YUNA ML

Komplexes Beispiel

In diesem Beispiel wird über den DataID-Provider eine Data-ID ausgeführt. Das Ergebnis wird anschließend in einem Tabellen-Widget dargestellt.

```
<xml>
  <view name="DataID-Provider" roles="System_Admin, AdHoc_Full_Issue">
    <caption>DataID-Provider</caption>
    <description>IO-Provider example</description>
    <userdocu>https://confluence.eoda.de/display/DD1/.IO-Provider</userdocu>

    <!-- DATA-ID IO-Provider -->
    <io-provider>
      <type>DataId</type>
      <config>
        <!-- define the DATA-ID -->
        <dataId>qy_dataid_provider_example</dataId>
        <triggerParams>
          <!-- define the parameters, which will trigger the DATA-ID -->
          <trigger>buttonPressed</trigger>
        </triggerParams>
        <!-- define the outputchannel, where the result from the DATA-ID will go -->
        <output>dataFromQuery</output>
      </config>
    </io-provider>

    <!-- Form-Widget for DATA-ID IO-Provider-->
    <widget>
      <widgettype>formwidget</widgettype>
      <caption>
        <show>true</show>
        <label>DATA-ID Form</label>
      </caption>
      <position>
```

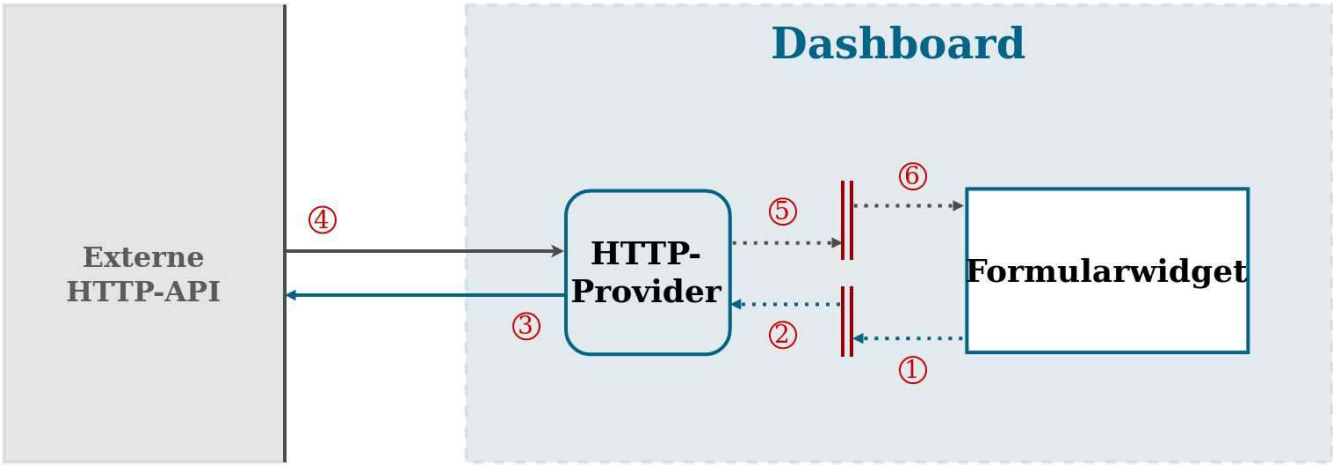
```
        <x>0</x>
        <y>0</y>
      </position>
      <size>
        <x>2</x>
        <y>2</y>
      </size>
      <outputs>
        <!-- on submit, buttonPressed will trigger the DATA-ID -->
        <submit>buttonPressed</submit>
      </outputs>
      <submitButton>
        <label>Run Query!</label>
      </submitButton>
      <formTemplate>
        <![CDATA[
<div>
<p>Run query and show data</p>
</div>
]]>
      </formTemplate>
    </widget>

    <!-- Table-Widget for DATA-ID IO-Provider-->
    <widget name="tbl_dataid_device_basic_info">
      <widgettype>tabledirective</widgettype>
      <caption>
        <show>true</show>
        <label>Data from DATA-ID IO-Provider</label>
      </caption>
      <position>
        <x>2</x>
        <y>0</y>
      </position>
      <size>
        <x>16</x>
        <y>4</y>
      </size>
      <sorting>ProductGroup</sorting>
      <sortingorder>asc</sortingorder>
      <inputs>
        <!-- get the data from the DATA-ID IO-Provider -->
        <!-- takes the data json in row format -->
        <data>dataFromQuery</data>
      </inputs>
      <generalOptions>
        <addColumns>true</addColumns>
        <selectable>false</selectable>
      </generalOptions>
    </widget>
```

```
  </view>
</xml>
```

5.7.3. HTTP-Provider

Der HTTP-Provider ermöglicht es HTTP-Schnittstellen in YUNA-Dashboards zu integrieren und somit Daten von externen Services in Dashboards einzubinden oder umgekehrt die Daten aus einem Dashboard an diese zu senden.Mit Hilfe von Handlebar-Templates können die HTTP-Anfragen dynamisch konfiguriert werden.

Beispielgrafik: Versenden von Formulardaten an eine HTTP-Schnittstelle mit Hilfe des HTTP-Provider:



1. Bei Betätigung des Absenden-Buttons im Formularwidget werden die eingegebenen Daten an den Output-Channel des Formularwidgets übergeben.
2. Die durch den IO-Channel bereitgestellten Daten werden durch den HTTP-Provider konsumiert, da dieser IO-Channel als Input konfiguriert ist.
3. Der HTTP-Provider führt ein mit den bereitgestellten Daten angereichertes Request durch.
4. Die angefragte HTTP-Schnittstelle liefert eine Antwort an den HTTP-Provider.
5. Die Antwort wird in den Response-Output-Channel des HTTP-Providers übergeben. Der Body der Antwort wird entsprechend der Konfiguration des HTTP-Providers ausgelesen.
6. Das Formularwidget erhält die neuen Daten aus dem IO-Channel und kann diese darstellen.

YUNAML

Beispielkonfigurationen für HTTP-Provider:

Durch klick auf die verschiedenen YUNAML-Elemente gelangen sie zu den jeweiligen detaillierten Informationen in den folgenden Tabelle.

Einfaches GET-Request mit minimaler Konfiguration

```
<io-provider>
    <type>HTTP</type>
    <config>
        <outputs>
            <response>OutputChannelName2</response>
```

```
        </outputs>
        <inputs>
            <optional>OptionalChannel</optional>
        </inputs>
        <method>GET</method>
        <url>https://qa.yuna.dev/</url>
        <unsafeCredentialDelegation>true</unsafeCredentialDelegation>
    </config>
</io-provider>
```

Komplexes POST-Request mit allen verfügbaren Konfigurationen

```
<io-provider>
    <type>HTTP</type>
    <config>
        <outputs>
            <request>OutputChannelName1</request>
            <response>OutputChannelName2</response>
        </outputs>
        <inputs>
            <trigger>formWidgetDataWasSentChannel</trigger>
            <mandatory>
                <list>dataChannel</list>
            </mandatory>
            <optional>
                <list>OptionalChannel2</list>
                <list>InputChannel2</list>
            </optional>
        </inputs>
        <requestBodyAs>json</requestBodyAs>
        <responseBodyAs>json</responseBodyAs>
        <method>POST</method>
        <url>https://qa.yuna.dev/</url>
        <unsafeCredentialDelegation>true</unsafeCredentialDelegation>
        <urlParameters>
            <parameterName>parameterValue</parameterName>
        </urlParameters>
        <body>
            {  "name": "{{dataChannel.name}}",
            "password": "{{dataChannel.password}}"  }
        </body>
        <headers>
            <keep-alive>{{paramChannel.keepAlive}}</keep-alive>
        </headers>
    </config>
</io-provider>
```

Konfiguration des HTTP-Providers

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
type	Der Typ des IO-Providers. Für den HTTP-Provider muss dieser Parameter auf "HTTP" gesetzt werden. Weitere mögliche Werte sind "UrlParams" und "DataID".	✓	`<type>HTTP<http>`
config	Die verschiedenen Konfigurationsparameter des HTTP-Providers. Die Konfiguration lässt sich für das leichtere Verständnis in zwei Teile zerlegen: 1. Die Konfiguration der In- und Outputchannels, über die der Provider an die anderen Inhalte eines Dashboards angebunden wird. 2. Die Konfiguration der Anfrage, die gegen eine HTTP-Schnittstelle ausgeführt werden soll.	✓	

Konfiguration der In- und Output-Channels

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel

config > **inputs**
 	
config > inputs > **trigger**
 	
config > inputs > **mandatory**
 	
config > inputs > **optional**
	Die Input-Channel des HTTP-Providers. Die konfigurierten Channel werden für zwei Zwecke verwendet:
	
	1. Das Befülllen der Templates in der Konfiguration der HTTP-Anfrage.
	2. Die Bestimmung, wann eine Anfrage durchgeführt werden soll.
	
	Um den Zeitpunkt für die Anfrage zu konfigurieren, werden die Input-Channel in drei YUNAML-Tags definiert: <**trigger**>, <**mandatory**> und <**optional**>.
	
	Bei jeder Änderung eines Channels, der in mindestens einem dieser Tags definiert ist, wird geprüft, ob eine Anfrage durchgeführt werden soll. Dabei werden folgende Bedingungen geprüft:
	
	1. Nur wenn Channel in <**trigger**> definiert sind: Wurde ein <**trigger**>-Channel aktualisiert?
	2. Nur wenn Channel in <**mandatory**> definiert sind: Wurden alle <**mandatory**> -Channel mit Daten befüllt?
	
	Sind weder <**trigger**>- noch <**mandatory**>-Channel definiert, wird bei jeder Änderung eines in <**optional**> angegebenen Kanals eine Anfrage ausgeführt.
	
	Ein Channel kann sowohl <**trigger**> als auch <**mandatory**> sein.
	
	Sollen mehrere Channel in einem der drei Tags definiert werden, müssen diese in <list>-Tags angegeben werden.
		✗	`<inputs>`
			
			`<trigger>someChannel</trigger>`
			`<mandatory>`
			
			`<list>dataChannel</list>`
			`</mandatory>`
			`<optional>`
			
			`<list>OptionalChannel2</list>`
			
			`<list>InputChannel2</list>`
			`</optional>`
			`</inputs>`

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel

config > **outputs**	Unter <**outputs**> wird konfiguriert, in welche IO-Channel die gesendete Anfrage und die Antwort der abgefragten Schnittstelle veröffentlicht werden.
	
	Das Format, in dem die Daten veröffentlicht werden, kann über **<requestBodyAs>** und **<responseBodyAs>** konfiguriert werden.
		✗	`<outputs`
			`  <request`
			`>OutputChannelName`
			`1</request>`
			
			`<response>OutputCh`
			`annelName2</respon`
			`se>`
			`</outputs>`

config > outputs > **request**	Der <**request**>-Channel wird genutzt um den abgesendeten Request zu veröffentlichen. Dabei wird der Body der Anfrage entsprechend der Konfiguration in **<requestBodyAs>** konvertiert.
	
	Das in den angegebenen Channel veröffentlichte Objekt hat unter anderem folgende Eigenschaften:
	
	• `method`: Die verwendete HTTP-Methode
	• `url`: Die Ziel-URL der Anfrage
	• `headers`: Die Header der Anfrage
	• `body`: Der Body der Anfrage (Für Details zur Formatierung siehe ***<requestBodyAs>**)
		✗	`<outputs>`
			
			`<request>OutputCha`
			`nnelName1</request`
			`>`
			`</outputs>`

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel

config > outputs > **response**	Der <**response**>-Channel wird mit der Antwort der angefragten HTTP-Schnittstelle befüllt. Dabei wird der Body der Anfrage entsprechend der Konfiguration in <**responseBodyAs**> konvertiert.
	
	Das in den angegebenen Channel veröffentlichte Objekt hat unter anderem folgende Eigenschaften:
	
	• `ok`: Ob die Anfrage erfolgreich war.
	• `status`: Der HTTP-Status-Code der Anfrage. Einen allgemeinen Überblick über HTTP-Status-Codes bietet die MDN-Dokumentation. Die Implementierung in einzelnen HTTP-Services kann von dieser allgemeinen Definition abweichen.
	• `statusText`: Die zu dem Status-Code gehörende Status-Nachricht.
	• `url`: Die URL der Antwort.
	• `body`: Der Body der Antwort (Für Details zur Formatierung siehe <**responseBodyAs**>)
	• `headers`: Die Header der Antwort.
		✗	`<outputs>`
			
			`<response>OutputChannelName2</response>`
			`</outputs>`

<table>
<tr><th>YUNAML-Tag</th><th>Beschreibung</th><th>Konfiguration erforderlich?</th><th>Beispiel</th></tr>
<tr>
<td>config > requestBodyAs

config > responseBodyAs</td>
<td>Der Datentyp, in dem der Body des Requests bzw. der Response erwartet wird.

Der HTTP-Provider versucht den Body entsprechend der Konfiguration zu konvertieren.Schlägt dies Fehl, werden keine Daten in den <request>-Channel veröffentlicht.

Unterstützte Datentypen:
<table>
<tr><th>Datentyp</th><th>Erläuterung</th><th>mögl. Anwendung</th></tr>
<tr><td>text</td><td>Text als UTF-8 Zeichenkette</td><td>Anzeige in einem Widget</td></tr>
<tr><td>json</td><td>Objektstruktur im JSON Textformat</td><td></td></tr>
<tr><td>formData</td><td>Schlüssel-/Wert-Paare als Representation von Formularfeldern und ihren Werten</td><td>Senden an/Empfangen von einem Formular-Endpunkt</td></tr>
<tr><td>blob</td><td>Datei ähnliche, rohe Daten</td><td>Down-/Upload zur Weiterverarbeitung</td></tr>
</table>
Für weitere Details zu den verschiedenen Datentypen und ihren Verwendungszwecken, finden die in der MDN-Dokumentation des Body mixin.</td>
<td>✗</td>
<td><pre><requestBodyAs>json</requestBodyAs></pre>
<pre><responseBodyAs>json</responseBodyAs></pre></td>
</tr>
</table>

Konfiguration der HTTP-Anfrage

Die über den HTTP-Provider zu sendende Anfrage kann über mehrere YUNAML-Tags konfiguriert werden. Diese sind in untenstehender Tabelle erläutert.

Die Inhalte aller YUNAML-Tags in diesem Block können mithilfe von Handlebar-Templates dynamisch gesetzt werden. Dabei werden die Daten aus den **<input>-Channels** verwendet, um die Platzhalter in den Handlebar-Templates zu befüllen.

Beispiel für die Ersetzung in Handlebar-Templates:

Der YUNAML-Tag <url> in der Konfiguration der HTTP-Anfrage enthält folgendes Template:

```
<url>https://jsonplaceholder.typicode.com/posts/{{formChannel.[0].number}}</url>
```

Als Beispiel werden hier Daten des Formularwidget Output-Channel verwendet, welche als Liste repräsentiert werden.Mit dem Template wird auf das erste Element der Liste und dessen Feld "number" referneziert.

In der Konfiguration der Input-Channel des HTTP-Providers findet sich folgender Eintrag:

```
<inputs>
    <optional>formChannel</optional>
</inputs>
```

Wird durch ein Formularwidget mit dem Output-Channel “formChannel” nun ein Formular mit dem Feld "number" abgeschickt, welches den Wert 99 enthält, wird das Template aus dem **<url>** -Tag in die folgende URL umgewandelt:

```
<url>https://jsonplaceholder.typicode.com/posts/99</url>
```

Erläuterung zur Deserialisierung von Input-Daten mit mehreren Eigenschaften in Handlebar-Templates

Sollen in einem Handlebar-Template Objekte mit mehren Eigenschaften dargestellt werden, ohne die Eigenschaften einzeln anzugeben, muss ein 'json'-Helper verwendet werden.

Der YUNAML-Tag <body> in der Konfiguration der HTTP-Anfrage enthält folgendes Template:

```
<body>{{formChannel}}</body>
```

Wird durch ein Formularwidget mit dem Output-Channel “formChannel” nun ein Formular mit mehreren Eingabe-Feldern abgeschickt, enthält der Body des Requests folgende Daten:

```
Object [object]
```

Um dies zu vermeiden, muss der 'json'-Helper in dem Handlebar-Template verwendet werden:

```
<body>{{json formChannel}}</body>
```

Dadurch werden die durch das Formularwidget veröffentlichten Daten korrekt zu folgendem Ergebnis deserialisiert:

```
{ "FormularFeld_1":"Wert von FormularFeld 1", "FormularFeld_2":"Wert von FormularFeld 2" }
```

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
config > **method**	Die HTTP-Methode, die für die Anfrage verwendet werden soll. Häufig genutzte Methoden: GET, POST, PUT, DELETE	✅	`<method>GET</method>`
config > **url**	Die Adresse, an die die Anfrage gestellt werden soll. Dies kann entweder eine absolute URL sein, also zum Beispiel mit "https://" beginnen, oder relativ, wenn eine YUNA-REST-Schnittstelle abgefragt werden soll. URL-Parameter können entweder direkt in dem Template angegebem werden, oder über die Einträge im Tag **<urlParameters>** hinzugefügt werden.	✅	`<url>https://jsonplaceholder.typicode.com/posts/{{formChannel.[1].number}}</url>`
config > **urlParameters**	URL-Parameter, die an die URL angehängt werden sollen. Die Angabe erfolgt in XML-Notation. Der Tag definiert den Namen des Headers, der Inhalt den Wert.	❌	`<urlParameters> <parameterName>parameterValue</parameterName> </urlParameters>`
config > **body**	Der Body, der mit der Anfrage versendet werden soll. ⚠️ Wird ein Body definiert, dürfen die HTTP-Methoden 'GET' und 'HEAD' nicht verwendet werden. Der Body kann mit Handlebar-Templates dynamisch konfiguriert werden.Es können dabei Objekte aus dem Input-Channel des HTTP-Providers verwendet werden.Beispiel: \{\{inputChannel.name}} wird durch das Objekt/den Wert aus "name" des Inputchannel ersetzt.	❌	`<body> { "staticTextField": "Hello ", "fieldFromTemplate": "{{inputChannel.name}}" } </body>`

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel

config > **headers**	Header, die für die Anfrage verwendet werden sollen.
	
	Im <**headers**>-Tag werden die Namen der Header in Tags angegeben und die jeweiligen Werte als Inhalt des Tags.
	
	Schematisch: `<header-key>header-value</header-key>`
		✗	`<headers>`
			`  <keep-alive>{{paramChannel.keepAlive}}</keep-alive>`
			`</headers>`

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
config > **unsafeCredentialDelegation**	Erlaubte Werte: **true** | **false** (Standard = Abgeschaltet) Aktiviert bei einem CORS-Request die automatische Anmeldung des Browsers bei dem Zielserver des HTTP-Request, falls möglich. Standardmäßig verbietet eine Browserrichtlinie die automatische Anmeldung bei einem CORS-Request. Weitere Browser-, Netzwerk-, Sicherheits- oder Systemrichtlinien können den CORS-Request sowie die automatische Anmeldung bei dem Zielserver verhindern. Der verwendete Browser muss möglicherweise für die automatische Anmeldung konfiguriert werden.Die Adresse des Zielservers muss möglicherweise in einer Liste (Whitelist) für die automatische Anmeldung hinterlegt sein. Ist diese Option aktiviert werden Informationen für eine Benutzeranmeldung (Cookies, Nutzername, Passwort, usw.) an den Zielserver gesendet.**Verwenden Sie diese Option nur wenn Sie dem Zielserver vertrauen (z.B. Server innerhalb des Firmennetzwerks).** Der Zielserver muss folgende Header setzen, damit diese Option genutzt werden kann: 1. **Access-Control-Allow-Origin: https://mein-yuna-portal.dev** Der Wert des Header muss der Quelle entsprechen (z.B. https://mein-yuna-portal.de/) und darf nicht auf "*" gesetzt sein. 2. **Access-Control-Allow-Credentials: true**	✗	`<unsafeCredentialDelegation>false</unsafeCredentialDelegation>`

== *Same-Origin-Policy (SOP)* der Browser und *Cross-Origin Resource Sharing (CORS)* bei der Nutzung des HTTP-Provider

> ⚠ **Das Abrufen von Ressourcen von anderen Servern wird durch die Same-Origin-Policy der Browser standardmäßig unterbunden.**
>
> Wenn Yuna z.B. unter der URL **https://yuna.demo.dev** verfügbar ist und über den HTTP-Provider ein Request an **https://weather.example.dev/api/weather?q=Kassel** gesendet werden soll kann dieser Request vom Browser unterbunden werden, da es sich nicht um den Selben Server handelt der Yuna bereitstellt. Dies ist ein Sicherheitskonzept (**Same-Origin-Policy**) bei der Verwendung von JavaScript im Browser. Externe APIs sind üblicherweise für den Zugriff von beliebigen anderen Servern über **Cross-Origin Resource Sharing** konfiguriert.
>
> Durch die Konfiguration vonhttps://de.wikipedia.org/wiki/Cross-Origin_Resource_Sharing[**Cross-Origin Resource Sharing**] auf dem externen Servers kann dieser den Request aus Yuna heraus erlauben.Um den Request an den Server (hier: **weather.example.dev**) zu erlauben muss auf dem Server (hier: **weather.example.dev**) der CORS Header konfiguriert werden.
>
> Weitere Informationen finden Sie auf der Seite **enable cross-origin resource sharing**.Wie der Zielserver des Request konfigriert wird finden Sie für die entsprechende Webserver-Software (Apache, nginX, usw.) unter enable CORS.

YUNA ML

Beispiel

In diesem Beispiel wird über den IO-Provider ein http POST-Request mit Daten aus einem Tabellen-Widget an die Adresse https://postman-echo.com/post gesendet. Die Response (Server-Antwort) wird in einem Formular-Widget dargestellt.

```
<xml>
    <view name="dpe-973" roles="System_Admin, AdHoc_Full_Issue">
        <caption>DPE-973: SIS-Service-Call</caption>
        <description>Dashboard, das ein dem SIS-Service-Call ähnliches Beispiel abbildet</description>
        <userdocu>https://jira.eoda.de/browse/dpe-973</userdocu>
        <widget>
            <position>
                <x>0</x>
                <y>0</y>
            </position>
            <size>
                <x>12</x>
                <y>8</y>
            </size>
            <caption>
                <show>true</show>
                <label>form-widget</label>
            </caption>
            <widgettype>formwidget</widgettype>
            <inputs>
                <response>responseChannel</response>
            </inputs>
            <outputs>
                <submit>formWidgetDataWasSentChannel</submit>
            </outputs>
```



```
            <formTemplate>
                <![CDATA[
                    <div>
                    <h2>Response:</h2>
                    <p>selected: {{inputs.response}}</p>
                    </div>
                    <br>
                ]]>
            </formTemplate>
            <submitButton>
                <dataID>qy_dpe-1387</dataID>
                <label>GO!</label>
            </submitButton>
        </widget>
        <widget name="MultiChoiceSelection_Table_dpe_1387">
            <position>
                <x>0</x>
                <y>9</y>
            </position>
            <size>
                <x>16</x>
                <y>7</y>
            </size>
            <widgettype>tabledirective</widgettype>
            <dataID>qy_dpe-1387</dataID>
            <sorting>ProductGroup</sorting>
            <sortingorder>asc</sortingorder>
            <outputs>
                <selected>selectedData</selected>
            </outputs>
            <cols>
                <list>
                    <field>ProductGroup</field>
                    <title>ProductGroup</title>
                    <sortable>ProductGroup</sortable>
                    <filter>
                        <ProductGroup>text</ProductGroup>
                    </filter>
                    <show>true</show>
                    <width>50</width>
                    <style></style>
                </list>
                <list>
                    <field>EquipmentNo</field>
                    <title>EquipmentNo</title>
                    <sortable>EquipmentNo</sortable>
                    <filter>
                        <EquipmentNo>text</EquipmentNo>
                    </filter>
                    <show>true</show>
                    <width>50</width>
                    <style></style>
                </list>
                <list>
                    <field>ProductionCounter</field>
                    <title>ProductionCounter</title>
                    <type>number</type>
```

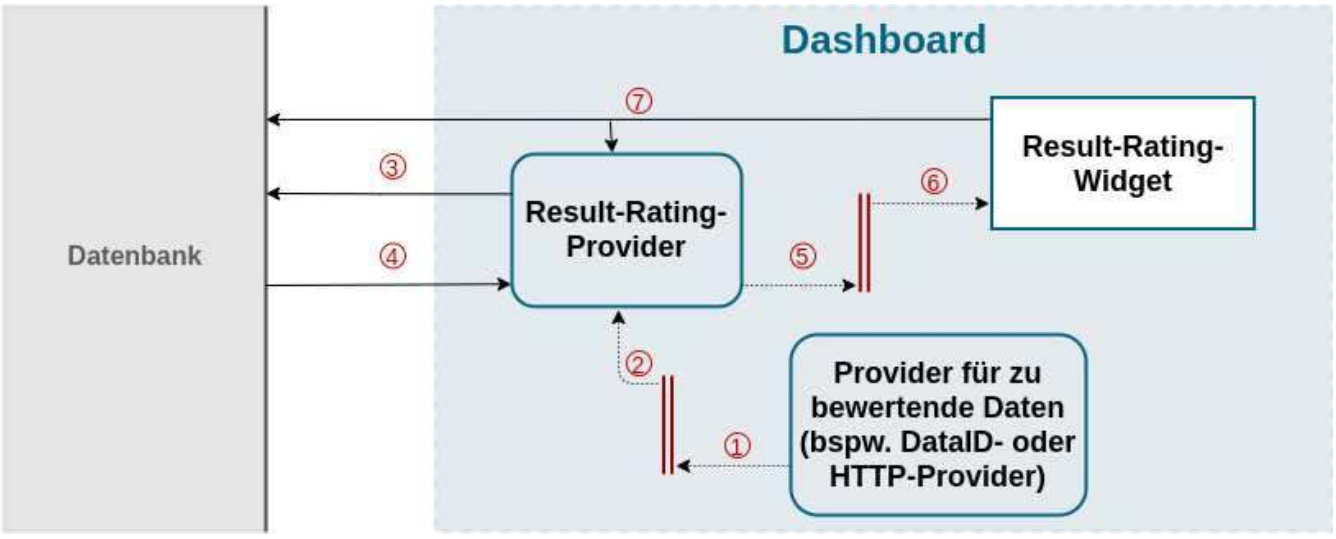
```
                <width>50</width>
                <filter>
                    <ProductionCounter>number-custom</ProductionCounter>
                </filter>
                <show>true</show>
            </list>
            <list>
                <field>ParentEquipmentNo</field>
                <title>ParentEquipmentNo</title>
                <width>50</width>
            </list>
            <list>
                <field>EndCustomer</field>
                <title>EndCustomer</title>
                <width>50</width>
            </list>
            <list>
                <field>Location</field>
                <title>Location</title>
                <width>50</width>
            </list>
            <list>
                <field>LocationCountry</field>
                <title>LocationCountry</title>
                <width>50</width>
            </list>
            <list>
                <field>EquipmentFamily</field>
                <title>EquipmentFamily</title>
                <width>50</width>
            </list>
            <list>
                <field>MachineType</field>
                <title>MachineType</title>
                <width>50</width>
            </list>
            <list>
                <field>LastServiceMission</field>
                <title>LastServiceMission</title>
                <type>genDate</type>
                <width>50</width>
                <filter>
                    <LastServiceMission>date-custom</LastServiceMission>
                </filter>
                <show>true</show>
            </list>
        </cols>
        <generalOptions>
            <selectable>true</selectable>
        </generalOptions>
    </widget>

    <io-provider>
        <type>HTTP</type>
        <config>
            <inputs>
                <mandatory>selectedData</mandatory>
```

```
                    <trigger>formWidgetDataWasSentChannel</trigger>
                </inputs>
                <outputs>
                    <response>responseChannel</response>
                </outputs>
                <method>POST</method>
                <url>https://cors-anywhere.herokuapp.com/https://postman-echo.com/post</url>
                <unsafeCredentialDelegation>false</unsafeCredentialDelegation>
                <body>{{json selectedData}}</body>
                <requestBodyAs>json</requestBodyAs>
                <responseBodyAs>text</responseBodyAs>
                <headers>
                    <Cache-Control>no-cache</Cache-Control>
                </headers>
            </config>
        </io-provider>
    </view>
</xml>
```

5.7.4. Result-Rating-Provider

Der Result-Rating-Provider ermöglicht es, Eingangsdaten mit Bewertungsdaten zu verknüpfen. Diese können dann mithilfe des Result-Rating-Widgets dargestellt und bewertet werden. Alternativ können die Bewertungsdaten aus dem Result-Rating-Provider beispielsweise auch an ein verknüpftes Tabellen-Widget und von diesem dann weiter an ein Result-Rating-Widget gesendet werden, um so das Potenzial der umfangreichen Filter-, Such- und Sortierungsoptionen des Tabellenwidgets nutzen zu können.

Beispielgrafik: Result-Rating-Provider im Zusammenspiel mit einem Result-Rating-Widget



1. Der die zu bewertenden Daten liefernde Provider schreibt diese in seinen Output-Channel
2. Die durch den IO-Channel bereitgestellten Daten werden durch den Result-Rating-Provider konsumiert, da dieser IO-Channel als Input konfiguriert ist
3. Der Result-Rating-Provider fragt die Datenbank nach bereits abgegebenen Bewertungsdaten für den konfigurierten QueryIdentifier und den aktuell eingeloggten Benutzer ab
4. Die Datenbank liefert die bereits abgegebenen Ergebnisse an den Result-Rating-Provider
5. Der Result-Rating-Provider sendet die in 2. eingegangen Daten zusammen mit den Bewertungsdaten an seinen konfigurierten OutputChannel

6. Das Result-Rating-Widget empfängt die in 5. gesendeten Daten auf seinem konfigurierten Input-Channel und stellt diese grafisch dar
7. Bei Abgabe einer Bewertung im Result-Rating-Widget wird diese in die Datenbank geschrieben. Im Zuge dessen wird der Result-Rating-Provider über das Vorhandensein neuer Bewertungen informiert und der Prozess startet erneut mit Punkt 3

YUNAML

Beispielkonfigurationen für Result-Rating-Provider:

Beispiel für einen Result-Rating-Provider

```
<io-provider>
    <type>ResultRating</type>
    <config>
        <input>InputChannel</input>
        <output>OutputChannel</output>
        <rowIdentifierTemplate>resultId:{{resultId}}</rowIdentifierTemplate>
        <queryIdentifier>QueryIdentifier</queryIdentifier>
    </config>
</io-provider>
```

Konfiguration des DataID-Providers

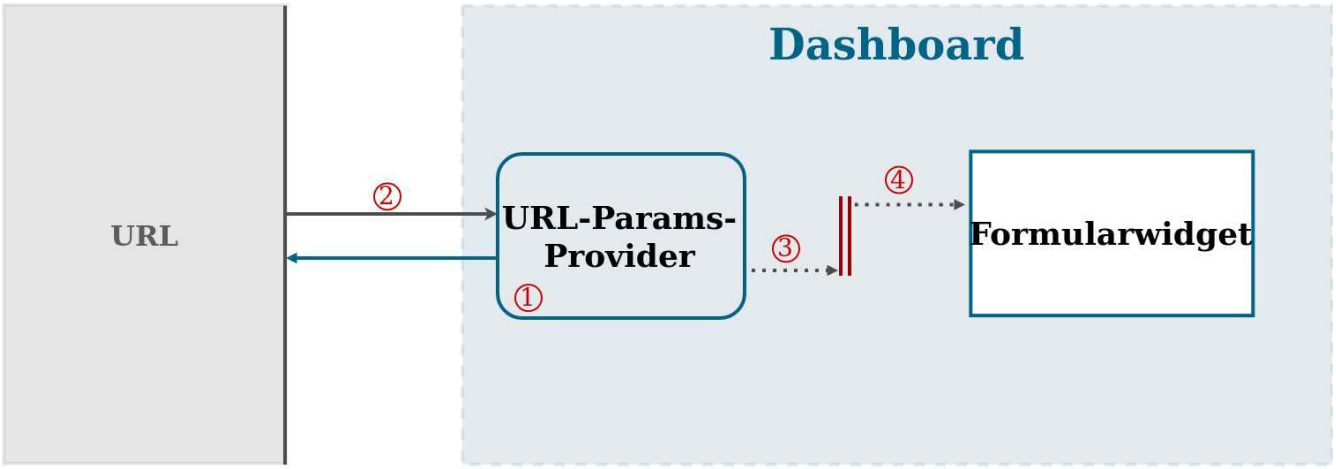
YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
type	Der Typ des IO-Providers. Für den Result-Rating-Provider muss dieser Parameter auf "ResultRating" gesetzt werden. Weitere mögliche Werte sind "UrlParams" und "HTTP".	✅	`<type>ResultRating</type>`
config	Die verschiedenen Konfigurationsparameter des Result-Rating-Providers.	✅	
config > **input**	Unter <**input**> wird der Name des Input-Channels konfiguriert, welcher die zu bewertenden Daten liefert. Dieser kann beispielsweise über einen DataID- oder Http-Provider bereitgestellt werden.	✅	`<input>InputChannel</input>`
config > **output**	Unter <**output**> wird der Name des Output-Channels konfiguriert, in welchem die mit Bewertungsdaten angereicherten Eingangsdaten veröffentlicht werden.	✅	`<output>OutputChannel</output>`

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
config > **rowIdentifierTemplate**	**Optionale** Angabe eines Handlebar-Templates, um einen bereits im Eingangsdatensatz vorhandenen Schlüssel zur Identifikation einzelner Zeilen zu nutzen. Bei nicht vorhandener Angabe eines Templates, wird ein Schlüssel über alle Werte einer Zeile gebildet. **Wichtig**: Eine nachträgliche Änderung des rowIdentifierTemplates kann leicht dazu führen, dass Daten und Ergebnisse nicht mehr zugeordnet werden können.	✗	`<rowIdentifierTemplate>resultId:{{resultId}}</rowIdentifierTemplate>`
config > **queryIdentifier**	Schlüssel unter welchem die Ergebnis-Daten gespeichert werden. **Wichtig**: Muss für jeden Bewertungsdatensatz eindeutig sein.	✓	`<queryIdentifier>QueryIdentifier</queryIdentifier>`

5.7.5. URL-Params-Provider

Der URL-Params-Provider ermöglicht es URL-Parameter aus der URL auszulesen und in Dashboards einzubinden.

Beispielgrafik: Auslesen der URL-Parameter mit Hilfe des URL-Params-Provider:



1. Der URL-Params-Provider prüft bei jeder Aktualisierung der URL, ob die gewünschten URL-Parameter vorhanden sind.
2. Daraufhin liest er die entsprechenden URL-Parameter aus der URL aus.
3. Die Werte der URL-Parameter werden anschließend in die jeweiligen Output-Channel des URL-Params-Provider gegeben.
4. Das Formularwidget erhält die neuen Daten aus dem IO-Channel und kann diese darstellen.

Beispielkonfigurationen für URL-Params-Provider:

Durch klick auf die verschiedenen YUNAML-Elemente gelangen sie zu den jewiligen detaillierten Informationen in den folgenden Tabelle.

Einfaches GET-Request mit minimaler Konfiguration

```
<io-provider>
    <type>UrlParams</type>
    <config>
        <params>
            <list>OptionalChannel</list>
        </params>
    </config>
</io-provider>
```

Konfiguration des URL-Params-Provider

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel
type	Der Typ des IO-Providers. Für den URL-Params-Provider muss dieser Parameter auf "UrlParams" gesetzt werden. Weitere mögliche Werte sind "HTTP" und "DataID".		`<type>UrlParams</type>`
config	Die Konfigurationsparameter des URL-Params-Provider.		

Konfiguration der In- und Output-Channels

YUNAML-Tag	Beschreibung	Konfiguration erforderlich?	Beispiel

config > **params**	In **params** werden die URL-Parameter definiert, die aus der URL ausgelesen werden sollen.
	
	Zur Darstellung in einem Widget muss im Input-Channel der Name des Parameter-Channels angegeben werden.
	
	Sollen mehrere URL-Parameter definiert werden, müssen diese in einem **<list>..</list>** Tag angegeben werden.
		✓	**Beispiel 1 - einzelner Parameter:**
			`<params>paramChannel</params>`
			
			Beispiel 2 - mehrere Parameter:
			`<params>`
			`<list>paramChannel 1</list>`
			`<list>paramChannel 2</list>`
			`</params>`

YUNAML

Beispiel

In diesem Beispiel wird über den URL-Params-Provider ein URL-Parameter aus der URL ausgelesen. Der Wert wird anschließend in einem Formular-Widget dargestellt.

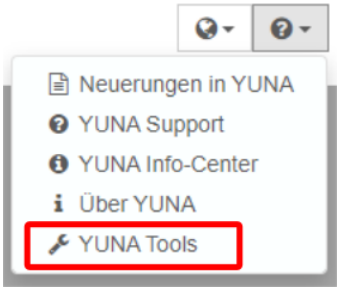
```
<xml>
  <view name="URL-Params-Provider" roles="System_Admin, AdHoc_Full_Issue">
    <caption>URL-Params-Provider</caption>
    <description>IO-Provider example</description>
    <userdocu>https://confluence.eoda.de/display/DD1/.IO-Provider</userdocu>
    <!-- URL-Params IO-Provider -->
    <io-provider>
      <type>UrlParams</type>
      <config>
        <!-- define URL parameters like they appear in the URL -->
        <params>urlParam1</params>
      </config>
    </io-provider>

    <!-- Form-Widget for URL-Params IO-Provider -->
    <widget>
      <widgettype>formwidget</widgettype>
      <caption>
        <show>true</show>
        <label>URL-Params Form</label>
      </caption>
      <position>
```

```
        <x>0</x>
        <y>2</y>
      </position>
      <size>
        <x>2</x>
        <y>2</y>
      </size>
      <inputs>
        <!-- get URL Parameter from URL-Params IO-Provider -->
        <urlParam>urlParam1</urlParam>
      </inputs>
      <submitButton>
        <label>do nothing</label>
      </submitButton>
      <formTemplate>
        <![CDATA[
<div>
<p>set the URL-Param</p>
<span>
value you set: {{inputs.urlParam}}
</span>
</div>
]]>
      </formTemplate>
    </widget>
  </view>
</xml>
```

5.8. Skripte für YUNA erstellen

YUNA verwendet für seine Analysen R-Skripte, die mithilfe einer Analytics IDE erstellt werden können.

Hierfür stellt YUNA die R-Pakete dseconnect und coreconnectivity bereit, die im Bereich YUNA Tools heruntergeladen werden können.



5.8.1. dseconnect

Mit dseconnect können Funktionen des Portals gesteuert und Datenbankzugriffe auf die PortalDB und DataDB ausgeführt werden.

Eine detaillierte Beschreibung der einzelnen Funktionen finden Sie im Paket selbst.

5.8.2. coreconnectivity

Mit coreconnectivity können Funktionen des YUNA Core gesteuert sowie Datenbankzugriffe auf die MongoDB ausgeführt werden.

Eine detaillierte Beschreibung der einzelnen Funktionen finden Sie im Paket selbst.

Installation

1. **rJava installieren**

   install.packages("rJava")

2. **(optional) Falls R die installierte Java Version nicht finden kann: Java Home Path setzen**

   Sys.setenv(JAVA_HOME='C:\\Program Files\\AdoptOpenJDK\\jre-8.0.232.09-hotspot') # for 64-bit version

3. **rJava ausführen**

   library(rJava)

4. **dseconnect installieren**

   install.packages("localPATH/dseconnect_1.15.0.tar.gz", repos = NULL, type = "source")

5. **(optional) Falls R das Paket nicht als x64 installiert**

   install.packages("localPATH/dseconnect_1.15.0.tar.gz", repos = NULL, type = "source", INSTALL_opts=c("--no-multiarch"))

6. **dseconnect ausführen**

   library(dseconnect)

7. **coreconnectivity installieren**

   install.packages("localPATH/coreconnectivity_1.15.0.tar.gz", repos = NULL, type = "source")

8. **coreconnectivity ausführen**

   library(coreconnectivity)

5.8.3. Analytics IDE

Für das Erstellen der Skripte empfehlen wir R-Studio.

Beschreibung:

5.8.4. Beispiele

Mit dem YUNA Portal verbinden

Codebeispiel

```
Beispiel
```

SQL Befehl absetzen

Codebeispiel

```
Beispiel
```

Weiteres...

5.9. dseconnect REST-API

In diesem Kapitel finden Sie Beispiele für die Nutzung der dseconnect REST-API in R-Skripten.

Die Dokumentation zur dseconnect REST-API finden sie als API-Spezifikation unter folgendem Link: **DSEconnect Rest-API**

5.9.1. Beispiel: Verwendung der REST API über R-Skripte

library(httr)

```
portal_url <- "http://0.0.0.0:9000/backend/"
portal_login_endpoint <- "conditionmonitoring/user/login.json"
```

Login in Portal

```
login_credentials = list(name = "admin", password = "12345")
# Note: With httr, Cookies are automatically persisted between requests to the same domain
r <- POST(paste0(portal_url, portal_login_endpoint),
          body = login_credentials,
          encode = "json")

status_code(r)

## [1] 200
```

Make request to auth.apitoken.rest endpoint to get a apitoken

In a Shiny App that runs inside YUNA, the api_token would be delivered in the URL as a

URLParameter e.g.: http://myshinyapp.com/?apitoken=12345-ABCD-THIS-IS-A-API-TOKEN

```
api_token_endpoint = "de.eoda.dse.portal.auth.apitoken.rest/"

api_token_response = POST(paste0(portal_url, api_token_endpoint))

api_token <- content(api_token_response, encoding = "utf-8")

api_token

## [1] "47a6339f-93ec-438c-9118-7d788a1db34f"
```

Use the token to login in the APIToken endpoint

```
login_result <-
  POST(paste0(portal_url, api_token_endpoint, "login"),
       query = list(token = api_token))

status_code(login_result)

## [1] 202
```

Fetch a data query for a given data id, user role and reference tag

```
data_query_endpoint = "de.eoda.dse.portal.dataquery.rest/"

dataid = "qy_InstBasis_Overview"

response_dataid <-
  GET(paste0(portal_url, data_query_endpoint, dataid),
      query = list(role = "System_Admin"))
```

When no refTag is given, it will be used the default reftag. A refTag can be included in the query parameters, e.g.:

```
GET(paste0(portal_url, data_query_endpoint, dataid), query = list(role="System_Admin", refTag=
"Demo"))
```

```
content(response_dataid)

## $id
## [1] 102
##
## $name
## [1] "qy_InstBasis_Overview"
##
## $type
## [1] "StoredProcedure"
##
```

```
## $content
## [1] "<QueryBuilder>\n  <exec>\n    <procedure>DSE-DataDB</procedure>\n    <procedure>data</procedure>\n
<procedure>sp_GetTableDataAccordingToLanguageParameter</procedure>\n    <parameters>\n      <parameter>\n
<id>currentLanguage</id>\n        <parameter>languageParameter</parameter>\n        <type>VARCHAR</type>\n
</parameter>\n      <parameter>\n        <id>filter2</id>\n        <parameter>InstBasis</parameter>\n
<type>VARCHAR</type>\n      </parameter>\n    </parameters>\n  </exec>\n</QueryBuilder>"
##
## $queryTablePath
## [1] "DSE-DataDB data"
##
## $filter
## [1] TRUE
##
## $refTag
## NULL
##
## $cancelable
## $cancelable$present
## [1] FALSE
##
##
## $metaData
## $metaData$typeDefinitions
## $metaData$typeDefinitions$ProductGroup
## $metaData$typeDefinitions$ProductGroup$type
## [1] "Translatable"
##
##
## $metaData$typeDefinitions$MachineType
## $metaData$typeDefinitions$MachineType$type
## [1] "Translatable"
##
##
##
## $metaData$description
## [1] ""
##
##
## $params
## named list()
##
## $roleId
## [1] 1
##
## $queryId
## [1] 47
```

GET Request to build and execute a query

Getting a data table

```
response_dataid_exec <-
  GET(
    paste0(
      portal_url,
      data_query_endpoint,
      "qy_InstBasis_Overview",
      "/exec"
    ),
    query = list(
      "_role" = "System_Admin",
      "_currentLanguage" = "de_DE",
```



```
      "_converter" = "columnconverter"
    ) # Column converter will deliver data as columns,
    # which is more pratical to convert to R dataframes
  )

get_columns_as_data_frame <- function(httr_response) {
  stop_for_status(httr_response)
  data_columns <-
    content(httr_response, as = "parsed", simplifyVector = TRUE)$dataQueryResult$columns
  return(data.frame(data_columns, stringsAsFactors = FALSE))
}

data_as_df <- get_columns_as_data_frame(response_dataid_exec)

str(data_as_df)

## 'data.frame':    20 obs. of  26 variables:
##  $ ProductionStartDate: num  1.34e+12 1.34e+12 1.43e+12 1.43e+12 1.17e+12 ...
##  $ ProductGroup       : chr  "Airplane Engine" "Airplane Engine" "Airplane Engine" "Airplane Engine" ...
##  $ ProductionCounter  : int  9678 1234 542 963 28 45 386 4328 238 578 ...
##  $ LastUpdate         : num  1.48e+12 1.45e+12 1.48e+12 1.48e+12 1.45e+12 ...
##  $ LocationCountry    : chr  "DE" "DE" "DE" "DE" ...
##  $ LastServiceMission : num  1.41e+12 1.41e+12 1.49e+12 1.49e+12 1.25e+12 ...
##  $ ParentEquipmentNo  : chr  "Airhansa 1234" "Airhansa 1234" "Airhansa5678" "Airhansa5678" ...
##  $ ObjectType         : chr  "BB-1107C" "BB-1107C" "BB-1107C" "BB-1107C" ...
##  $ EndCustomer        : chr  "Airhansa AG" "Airhansa AG" "Airhansa AG" "Airhansa AG" ...
##  $ CommissioningDate  : num  1.34e+12 1.34e+12 1.42e+12 1.42e+12 1.17e+12 ...
##  $ ShippingDate       : num  1.33e+12 1.33e+12 1.42e+12 1.42e+12 1.17e+12 ...
##  $ LastCmCollect      : num  1.49e+12 1.46e+12 1.48e+12 1.49e+12 1.46e+12 ...
##  $ ID                 : int  6 7 15 16 17 18 10015 10031 10032 10033 ...
##  $ UpdateInfoSpecial  : int  -1 2 1 -1 2 1 -1 -1 -1 -1 ...
##  $ EquipmentFamily    : chr  "BB-110" "BB-110" "BB-110" "BB-110" ...
##  $ Status             : chr  "re" "re" "re" "re" ...
##  $ EquipmentNo        : chr  "BB1107_1" "BB1107_2" "BB1107_3" "BB1107_4" ...
##  $ CustomerNo_AG      : chr  "wert" "wert" "wert" "wert" ...
##  $ MachineTypeTK      : chr  "data.deviceBasicInfo.machineType.turboprop" "data.deviceBasicInfo.machineType.turboprop"
"data.deviceBasicInfo.machineType.turboprop" "data.deviceBasicInfo.machineType.turboprop" ...
##  $ ProductionEndDate  : num  1.42e+12 1.42e+12 NA NA 1.25e+12 ...
##  $ LastServiceMessage : num  1.36e+12 1.46e+12 1.48e+12 1.36e+12 1.46e+12 ...
##  $ MachineType        : chr  "Turboprop" "Turboprop" "Turboprop" "Turboprop" ...
##  $ ProductGroupTK     : chr  "data.deviceBasicInfo.productGroup.airplaneEngine"
"data.deviceBasicInfo.productGroup.airplaneEngine" "data.deviceBasicInfo.productGroup.airplaneEngine"
"data.deviceBasicInfo.productGroup.airplaneEngine" ...
##  $ Generation         : int  2 2 3 3 1 1 4 4 5 4 ...
##  $ UpdateInfoGeneral  : int  -1 2 1 -1 2 1 -1 -1 -1 -1 ...
##  $ Location           : chr  "Berlin" "Berlin" "Berlin" "Berlin" ...
```

Getting just equipment list using a predefined dataid and the filter to get only the ones that have sensor data in our test db

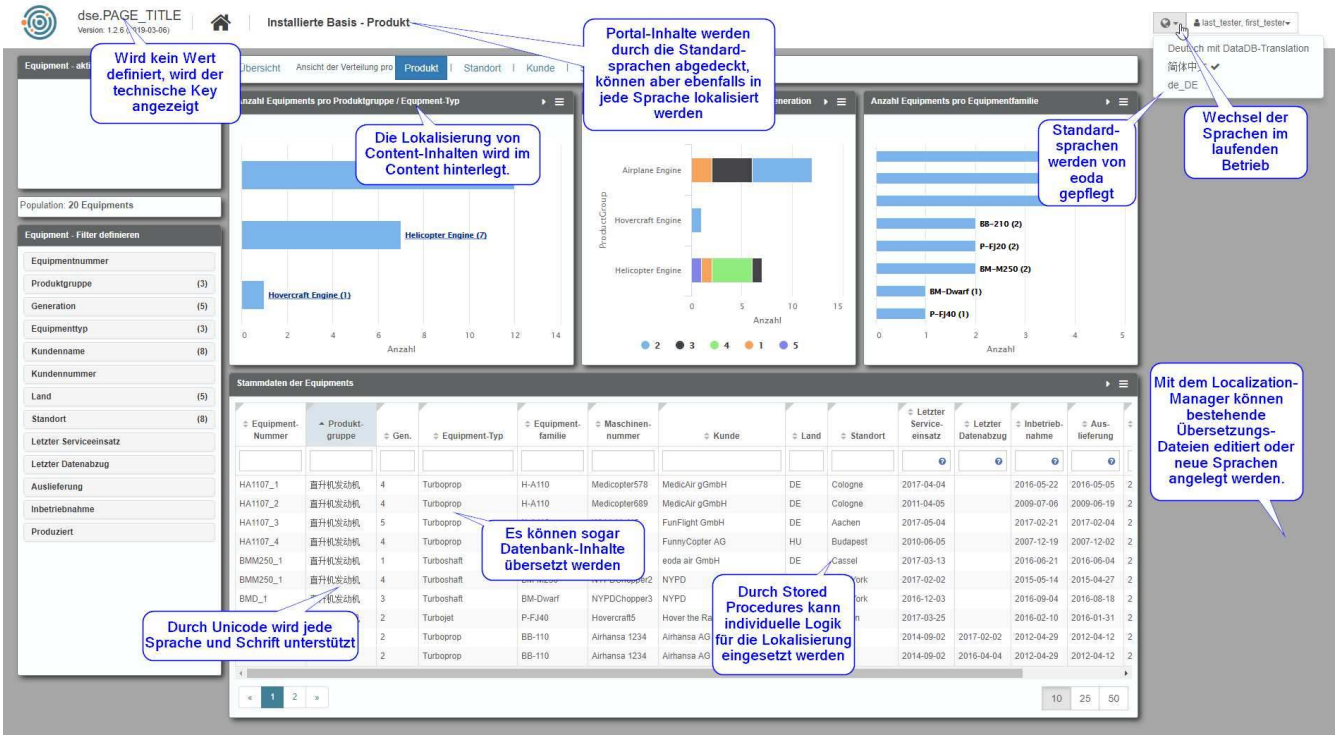
```
response_dataid_equi_list_exec <-
  GET(
    paste0(
      portal_url,
      data_query_endpoint,
      "qy_InstBasis_EquiList",
      "/exec"
    ),
    query = list(
      "_role" = "System_Admin",
```

```
      filter2 = "af9c4ff549003fe39c1b3aad38ca4c21f37c9815f0437bec711d06cd7988d58b",
      "_currentLanguage" = "de_DE",
      "_converter" = "columnconverter"
    ) # Column converter will deliver data as columns,
    # which is more pratical to convert to R dataframes
  )


status_code(response_dataid_equi_list_exec)

## [1] 200

equipment_list <-
  get_columns_as_data_frame(response_dataid_equi_list_exec)

equipment_list

##   EquipmentNo
## 1    HA1107_1
## 2    BMK823_1

for (equipment in equipment_list$EquipmentNo) {
  response_dataid_sensor_data_exec <-
    GET(
      paste0(
        portal_url,
        data_query_endpoint,
        "qy_DataFromEquipments",
        "/exec"
      ),
      query = list(
        "_role" = "System_Admin",
        filter2 = "af9c4ff549003fe39c1b3aad38ca4c21f37c9815f0437bec711d06cd7988d58b",
        "_currentLanguage" = "de_DE",
        "_converter" = "columnconverter",
        sensor = "Rd_CMD_Conductivity_Max_R.EC.01.MAX",
        axis = "0",
        element = equipment,
        index = "100",
        operatinghours = "date"
      )
    )
  stop_for_status(response_dataid_sensor_data_exec)
  sensor_data_df <-
    get_columns_as_data_frame(response_dataid_sensor_data_exec)
  sensor_data_df$Timestamp <-
    as.POSIXct(sensor_data_df$xData / 1000,
               tz = "UTC",
               origin = "1970-01-01")
```

```
  plot(Value ~ Timestamp, data = sensor_data_df, main = equipment)
}
```

6. YUNA-Lokalisierung

Um die Zusammenarbeit zwischen Nutzern mit verschiedenen sprachlichen und kulturellen Hintergründen zu ermöglichen, bietet YUNA die Möglichkeit zur Lokalisierung von Textinhalten.



Im Rahmen von YUNA werden **drei Bereiche** unterschieden, denen sich übersetzbare Inhalte zuordnen lassen:

1. **Portal**: Die Standardoberfläche und die allgemeinen Bedienelemente des Portals
2. **Dashboard Inhalte**: Die benutzerspezifischen Inhalte
3. **Datenbankinhalte**: Die dargestellten Datenbankinhalte

Jede Sprache in YUNA kann einer von **zwei Klassen** zugeordnet werden:

1. **Basissprachen** sind vorgegebene Sprachinhalte für alle Inhalte aus dem Bereich 'Portal'. Die Lokalisierung wird von eoda gepflegt, d.h. sie kann kundenseitig nicht verändert werden. Die Basissprachen enthalten keine Übersetzungen für Dashboard Inhalte und Datenbankinhalte. Aktuell existieren zwei Basissprachen: Deutsch (de_DE) und Englisch (en_US).
2. **Eigene Sprachen** können beliebig definiert werden und alle Bereiche abdecken.

Unabhängig von dem jeweiligen Bereich und der Klasse der Sprache werden Übersetzungen in Form sogenannter **Key-Value-Paare** definiert. Dabei steht ein Übersetzungsschlüssel ('Key') für einen übersetzbaren Inhalt, dem je nach ausgewählter Sprache verschiedene Übersetzungen ('Value') zugewiesen werden.

Der Austausch von Übersetzungen mit dem erfolgt über Textdateien im JSON-Format (JavaScript Object Notation), einem einfachen und verbreiteten Datenformat. Diese Dateien enthalten alle Key-Value-Paare, die zu einer Übersetzung gehören.

6.1. Beispiel für Übersetzungsdateien im JSON Format

JSON-Dateien haben die Dateiendung '.json' und können entweder eine hierarchische oder eine flache Struktur haben.

flache Struktur

```
{
    "dse.general.save":"Speichern",
    "dse.general.cancel":"Abbrechen"
}
```

hierarchische Struktur

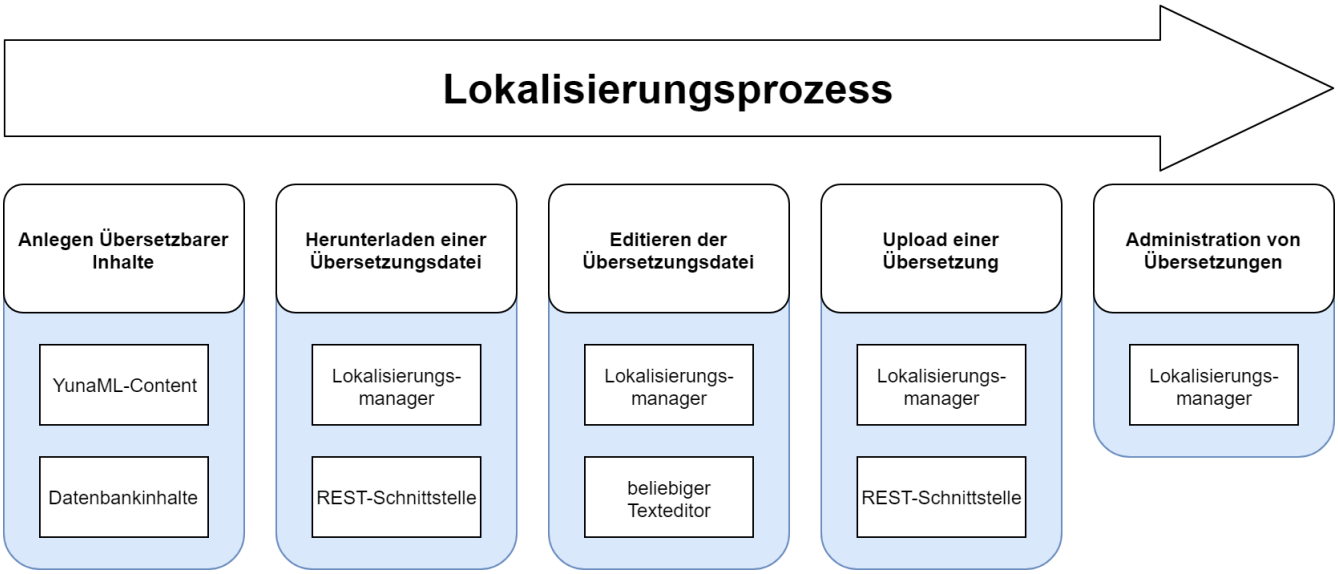
```
{
  "dse":{
    "general":{
       "save":"Speichern",
       "cancel":"Abbrechen"
    }
  }
}
```

Beide Beispiele zeigen dieselbe Übersetzung, jedoch einmal flach und einmal hierarchisch strukturiert.

6.2. Verwaltung der Sprachen

Zur Verwaltung der Sprachen stehen zwei Möglichkeiten zur Verfügung:

1. Über den **Lokalisierungsmanager** können Portaladministratoren Sprachen anlegen, aktualisieren und löschen, sowie die Verfügbarkeit für die Portalbenutzer steuern.
2. Über **Download / Upload von Sprachdateien** können automatisierte Prozesse für das Anlegen und Aktualisieren von Übersetzungen etabliert werden.



6.3. Übersetzbare Inhalte

6.3.1. Typen von Translation-Keys

Nahezu alle Texte, die im YUNA Portal angezeigt werden, können übersetzt werden. Da unterschiedliche Inhalte unterschiedlich angelegt und verwaltet werden, unterscheidet YUNA verschiedene Bereiche, in denen Übersetzungen angewendet werden. Für jeden Bereich gibt es unterschiedliche Möglichkeiten, diese an ihre Bedürfnisse anzupassen. Die folgende Tabelle gibt einen Überblick über die verschiedenen Bereiche.

Typ	Präfix	Beschreibung	Von eoda gepflegt?
Portal-Inhalte	dse.	Übersetzungen für Beschriftungen und Texte in Standard-Sichten und nicht-anpassbaren Widget-Inhalten (z.B. Button-Beschriftungen)	✓
Dashboard Inhalt	content.	Übersetzungen für das Dashboard anpassbare Inhalte, z.B.: Views, Widgets, Überschriften, etc.	✗
Datenbankinhalte	data.	Übersetzungen von Datenbankinhalten	✗

6.3.2. Portal-Inhalte

Alle dargestellten Texte, die von eoda erstellt und nicht individuell angepasst werden, werden im Kontext der Lokalisierung als Portal-Inhalte bezeichnet. Dazu gehören unter anderem Button-Beschriftungen, Tooltips und Standard-Sichten wie die einzelnen Komponenten des Issue-Workflows.

Alle in diesem Kontext gepflegten Keys haben zwei Gemeinsamkeiten:

1. Sie werden von eoda im Rahmen der Basissprachen gepflegt
2. Sie können durch das Präfix 'dse.' identifiziert werden.

6.3.3. Dashboard

Die individuellen Inhalte in YUNA können im Rahmen der Dashboard-Entwicklung mit entsprechenden Übersetzungsschlüsseln versehen werden, sodass auch diese übersetzt werden können.

Für die im Dashboard verwendeten Übersetzungsschlüssel gilt:

1. Da es sich hierbei um individuelle Inhalte handelt, sind die entsprechenden Übersetzungen nicht Teil der Basissprachen, sondern werden ausschließlich in eigenen Sprachen mit Übersetzungen versehen.
2. Sie müssen das gemeinsame Präfix 'content.' haben.

Weiterführende Informationen: Definition übersetzbaren Contents

6.3.4. Datenbankinhalte

Sollen die über eine DataID aus Datenbanken abgerufenen Texte übersetzt werden, müssen zusätzlich zu den ursprünglichen Texten Übersetzungschlüssel in der Datenbank hinterlegt werden. Desweiteren werden Spalten in der DataID-Definition mit dem Metadatentyp 'Translatable' als übersetzbar gekennzeichnet.

Für die zur Übersetzung von Datenbankinhalten genutzten Übersetzungsschlüssel gilt:

1. Da es sich hierbei um individuelle Inhalte handelt, sind die entsprechenden Übersetzungen nicht Teil der Basissprachen, sondern werden ausschließlich in eigenen Sprachen mit Übersetzungen versehen.
2. Sie müssen das gemeinsame Präfix 'data.' haben.

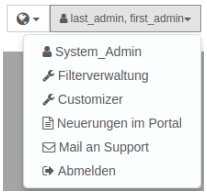
Weiterführende Informationen: Definition von Metadaten in DataIDs

Weiterführende Informationen: Metadatentyp 'Translatable'

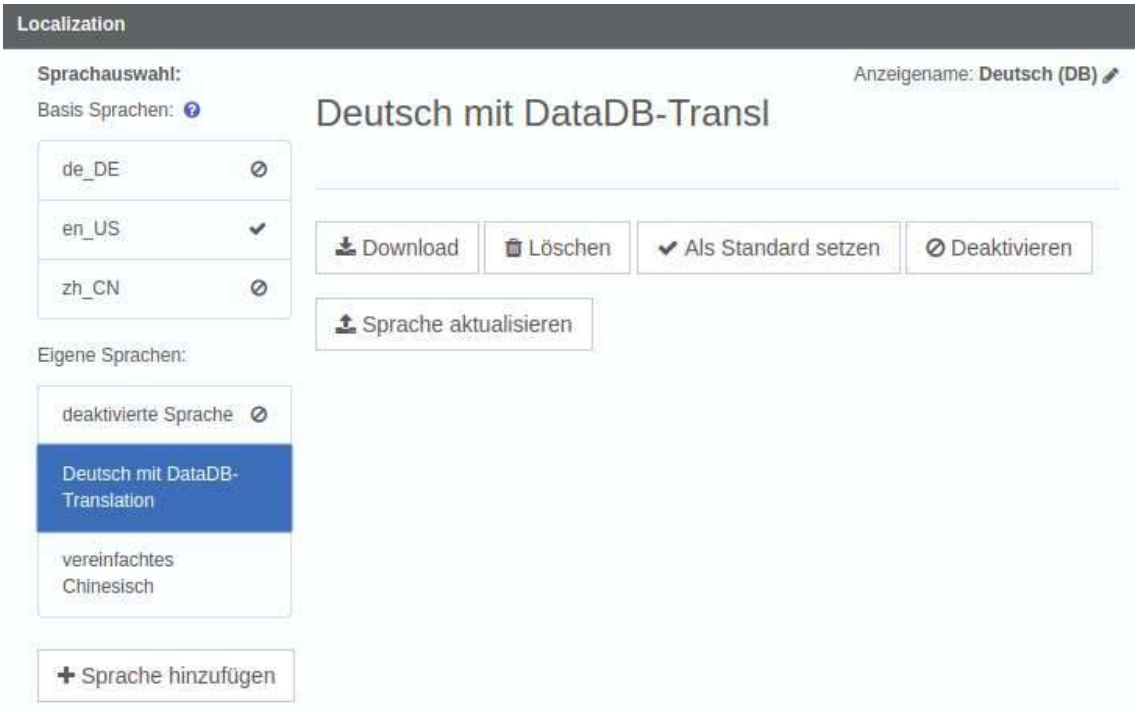
Für die Übersetzung von Datenbankinhalten ist es notwendig, dass zu einer zu übersetzenden Spalte eine Spalte mit dem Suffix 'TK' existiert, die die entsprechenden Übersetzungsschlüssel enthält. Diese sollte den selben Datentyp (derzeit `nvarchar(150)`) haben, wie die Spalte 'portal.localization.Key' der PortalDB.

6.4. Lokalisierungsmanager

Der Lokalisierungsmanager bietet eine komfortable Möglichkeit, die enthaltenen Sprachen an einer zentralen Stelle zu verwalten.



Im Lokalisierungsmanager können alle verfügbaren Sprachen eingesehen, Updates der Sprachen eingespielt oder die globale Sichtbarkeit einer Sprache geändert werden. Um zur Ansicht des Lokalisierungsmanagers zu navigieren, kann das "Customizer"-Element innerhalb der Headerleiste angeklickt werden.



Hier werden alle vorhandenen Sprachen aufgelistet, wobei zu beachten ist, dass Basissprachen nicht aktualisiert oder gelöscht werden können. Jede Sprache ist mit einem Anzeigenamen assoziiert, so hat die Sprache 'Deutsch mit DataDB-Translation' den Anzeigenamen 'Deutsch (DB)'. Der Anzeigename wird zur Darstellung im Sprachauswahl-Menü verwendet und außerdem mit einem Status bezüglich der Sichtbarkeit assoziiert ('Aktiviert', 'Deaktiviert' oder 'Standard'), welcher innerhalb der Schaltfläche angezeigt wird.

Sprachen erstellen

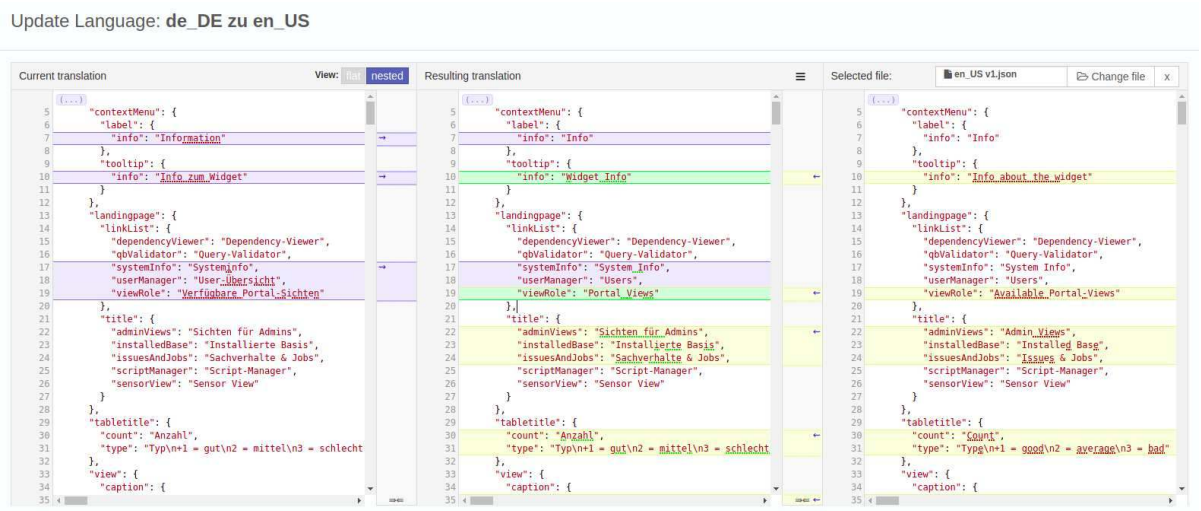
Zum Erstellen einer neuen eigenen Sprache wird die Schaltfläche "Sprache hinzufügen" verwendet. Zunächst muss für die Sprache ein Name, ein Anzeigename, eine Basissprache als Rückfall für nicht existierende Schlüssel und die Datei mit den Übersetzungsschlüsseln für die neue Sprache angegeben werden.

Sprachauswahl anpassen

Die Sichtbarkeit von Sprachen in der Sprachauswahl kann für sowohl Basissprachen als auch eigene Sprachen über die 'Deaktivieren'- oder 'Aktivieren'-Schaltfläche gesetzt werden. Die einzige Ausnahme ist die präferierte Sprache: Diese kann nicht deaktiviert werden ohne gewechselt zu werden.

Sprachen aktualisieren

Über die Schaltfläche "Sprache aktualisieren" können eigene Sprachenaktualisiert werden. Hierfür steht ein Dialog zum Vergleichen der Dateien zur Verfügung: Die Quellsprachdatei und die Übersetzungsdatei werden nebeneinander darstellt und Änderungen an der Übersetzung veranschaulicht.



In der rechten oberen Ecke befindet sich eine Dateiauswahl, mit der eine neue Übersetzungsdatei in das rechte Drittel geladen werden kann (hier 'en_US v1'). Es werden sämtliche Abweichungen zwischen der eingeladenen Datei und dem mittleren Feld, an dem Änderungen an der aktuell genutzten Übersetzung vorgenommen werden können, in gelb dargestellt. Im linken Feld werden Abweichungen zwischen der aktuell hinterlegten Version und den manuellen Änderungen angezeigt.

Die Browserseite muss neu geladen werden, um die geänderten Einstellungen an Sprachen anzuwenden.

6.5. Download / Upload von Sprachdateien

6.5.1. Upload mit dsedep

Mit dem YUNA-Tool dsedep werden Übersetzungsschlüssel (Tkeys) in die Datenbank geschreiben.

YUNA ML Beispiel: HTML Widget mit Übersetzungsschlüssel

```
<xml>
<!--Copyright (c) 2019 eoda GmbH -->
<view name="view_HTML_Basic" roles="System_Admin">
<caption>HTML Widget</caption>
<description>
<tkey>content.demo.tkey_without_default</tkey>
</description>
<widget source="demo_HTML_Basic">
<position>
<x>0</x>
<y>0</y>
</position>
<size>
<x>12</x>
<y>7</y>
</size>
<widgettype>htmlwidget</widgettype>
<htmlwidget>
<template>
<![CDATA[
<h1>
<tkey>content.demo.tkey</tkey>
<default>Default value</default>
</h1>
]]>
</template>

</htmlwidget>
</widget>
</view>
</xml>
```

CDATA-Blöcke müssen als XML geparst werden können (falls nicht möglich, gibt dsedep eine Warnung aus)

Um mit dsedep Übersetzungsschlüssel in die Datenbank zu schreiben, muss zwingend `ohne reftag` deployed werden.

Die Übersetzungsschlüssel werden für die Fallback-Sprache in die Datenbank geschrieben.

6.5.2. Rest-API

Um den Arbeitsablauf für die Erstellung und Pflege von Lokalisierungen im Rahmen des Portals zu vereinfachen, stellt das Portal eine REST-API bereit. Diese kann genutzt werden, um Sprachen im JSON-Format mit der Datenbank auszutauschen.

Name der Sprache

Leerzeichen im Namen der Sprache müssen mit '%20' ersetzt werden. **Beispiel**: `English translation` wird zu `English%20translation`.

Upload

Alle beschriebenen URLs nehmen relativen Bezug auf die DSE Backend-URL. In der Standardeinstellung ist diese "https://<host-name>/backend/". Bei abweichender Systemkonfiguration muss dies bei der Nutzung der REST-API berücksichtigt werden.

Schnittstelle

HTTP-Methode	URL, erfordert Authentifizierung, unterstützt Basic Auth (vgl. https://tools.ietf.org/html/rfc2617)
POST	conditionmonitoring/localization/api/{fallbacklanguage}/{language}

Parameter

Typ	Name	Beschreibung	Datentyp	Optional	Standardwert
Pfad	**fallbacklanguage**	Name der Sprache, die für den Abruf von Übersetzungen als Alternative bei nicht definierten Übersetzungsschlüsseln dienen soll	String	✗	-
Pfad	**language**	Name der Sprache die eingepflegt/aktualisiert werden soll	String	✗	-
Query String	**clean**	Ob die in der Datenbank vorhandenen Einträge der ausgewählten Sprache vor dem Update gelöscht werden sollen.	Boolean	✓	`false`

Typ	Name	Beschreibung	Datentyp	Optional	Standardwert
Body		JSON-Objekt mit Paaren aus Übersetzungsschlüssel und -text	application/json	✗	-

Download

Schnittstelle

HTTP-Methode	URL
GET	conditionmonitoring/localization/api/{language}

Parameter

Typ	Name	Beschreibung	Datentyp	Optional	Standardwert
Pfad	**language**	Name der Sprache, die abgerufen werden soll	String	✗	-

Wenn beim Abruf einer Sprache keine Übersetzungsschlüssel für Inhalte in dieser Sprache vorhanden sind, dann werden (sofern vorhanden) die TKey-Werte aus der Fallback-Sprache verwendet.

Beispiel für die Nutzung der Restschnittstelle

Für die Beispiele wird eine Linux-Kommandozeile (Bash) mit dem Programm curl verwendet.

Abruf einer beliebigen Sprache, der lesende Zugriff benötigt keine Authentifizierung:

```
curl -X GET https://<dse-backend-url>/conditionmonitoring/localization/api/beliebige Sprache
```

Anlegen einer neuen Sprache mit Rückfallsprache 'de_DE' (Authentifizierung erforderlich, wird mit Parameter '-u' aktiviert)

```
curl -X POST https://<dse-backend-url>/conditionmonitoring/localization/api/de_DE/neue Sprache -H "Content-Type:
application/json" -u myUsername -d @"neue Sprache.json"
```

Die neu angelegte Sprache verwendet "de_DE" als Rückfallsprache. Die Datei "neue Sprache.json" enthält die Definition der Sprache in Form von Key-Value-Paaren

Aktualisieren einer existierenden Sprache (Authentifizierung erforderlich)

```
curl -X POST https://<dse-backend-url>/conditionmonitoring/localization/api/de_DE/existierende Sprache -H "Content-Type:
application/json" -u myUsername -d @"aktualisierte Sprache.json"
```

6.6. Lokalisierung via Stored Procedure

Zusätzlich zur Lokalisierung via Key-Value-Paaren ist es ebenfalls möglich, Inhalte aus Datenbanken mit Hilfe einer Stored Procedure zu übersetzen. An die entsprechende Stored Procedure muss dafür ein Parameter übergeben werden, der den `Bezeichner der ausgewählten Sprache` enthält. (Siehe hierzu: Stored Procedures)

Die Bezeichner für Sprache sind durch das YUNA-Backend uvorgegeben werden und können verändert werden. In der YUNAML-Definition muss Name im <parameter>-Tag dem Parameternamen entsprechen, der in der Stored Procedure verwendet wird.

Für ein reibungsloses Funktionieren der Filterung ist es wichtig, dass die Filter der Stored Procedure korrekt übergeben und in der WHERE-Klausel korrekt eingebunden werden. Ein Beispiel dazu findet sich im Kapitel zum Filter anlegen.

6.6.1. Schematische Darstellung

Beispiel: Es existiert die Tabelle "Equipmentstandorte", deren Inhalt abhängig von der ausgewählten Sprache lokalisiert werden soll.

6.6.2. Equipmentstammdaten

ID	Location	Location_en	Location_es
1	Brüssel	Brussels	Bruselas
2	Wien	Vienna	Viena

Abhängig von der ausgewählten Sprache soll die Spalte "Location" in der Tabelle im YUNA-Dashboard den zugehörigen Werte enthalten. Zum Beispiel für die ID 1 als Location `Brussel` wenn Englisch als Sprache ausgewählt ist.

Die aufrufende Prozedur muss dann abgängig vom eingegebenen Sprachparameter die entsprechende(n) Spalte(n) zurückgeben.

6.6.3. Codebeispiel für Aufruf (Auszug):

```
IF languageParameter = 'de_DE'
    SELECT Location FROM Equipmentstammdaten
ELSE IF languageParameter = 'en_US'
    SELECT Location_en AS Location FROM Equipmentstammdaten
ELSE IF languageParameter = 'es_ES'
    SELECT Location_es AS Location FROM Equipmentstammdaten
```

Der Wert der zu übermittelnden Sprachen lässt sich im Fenster Customizer im Benutzermenü im rechten Bereich "Sprachen" ablesen. Der zu prüfende Begriff ist in der Grafik rot umrandet und je nach ausgewählter Sprache unterschiedlich.

6.7. Übersetzungsschlüssel

An dieser Stelle werden die in YUNA verwendeten Übersetzungsschlüssel aufgeführt.

Übersetzungsschlüssel	Default	verwendet von
dse.directive.filter.menu.category.wildcard	Wildcard	Zeitbereichsfilter
dse.directive.filter.menu.date.absolut	Absolut	Zeitbereichsfilter
dse.directive.filter.menu.date.from	Von	Zeitbereichsfilter
dse.directive.filter.menu.date.period.currentDay	aktueller Tag	Zeitbereichsfilter
dse.directive.filter.menu.date.period.currentYear	aktuelles Jahr	Zeitbereichsfilter
dse.directive.filter.menu.date.period.days	Tage	Zeitbereichsfilter
dse.directive.filter.menu.date.period.last	letzte	Zeitbereichsfilter
dse.directive.filter.menu.date.period.lastDays1	letzter Tag	Zeitbereichsfilter
dse.directive.filter.menu.date.period.lastDays30	letzte 30 Tage	Zeitbereichsfilter
dse.directive.filter.menu.date.period.lastDays7	letzte 7 Tage	Zeitbereichsfilter
dse.directive.filter.menu.date.period.lastWeeks13	letzte 13 Wochen	Zeitbereichsfilter
dse.directive.filter.menu.date.period.lastWeeks26	letzte 26 Wochen	Zeitbereichsfilter
dse.directive.filter.menu.date.period.lastWeeks52	letzte 52 Wochen	Zeitbereichsfilter
dse.directive.filter.menu.date.period.noConstraint	keine Einschränkung	Zeitbereichsfilter
dse.directive.filter.menu.date.period.weeks	Wochen	Zeitbereichsfilter
dse.directive.filter.menu.date.relativ	Relativ	Zeitbereichsfilter
dse.directive.filter.menu.date.relativePeriod	Relativer Zeitraum	Zeitbereichsfilter
dse.directive.filter.menu.date.to	Bis	Zeitbereichsfilter

7. Glossar

Begriff	Term	Erklärung
Abfrage	Query	Vgl. Query
Ad-hoc-Analyse	Ad-hoc Analysis	Ad-hoc-Analysen sind eine von zwei unterstützten Arten von Analysen in YUNA. Unter Ad-hoc-Analysen wird verstanden, dass Nutzer von YUNA Daten und Informationen von Equipments (z.B. Produktionsanlagen, Fahrzeugen, Haushaltsgeräten etc.) in Widgets filtern, durchsuchen, visualisieren und explorativ untersuchen können. Bsp: Sensorview, Tabellen filtern und durchsuchen, ...
Administrator	Administrator / System administrator	Vgl. System Administrator
Analyse	Analysis	In YUNA werden zwei Arten von Analysen unterschieden: Ad-hoc-Analysen und automatische Analysen.
Analyseskript	Analytic script	(R-)Skript mit (komplexen) Berechnungen zur Datenanalyse.
API	API	Abkürzung für Application Programming Interface. Eine API ist eine Schnittstelle von Software, die es anderer Software erlaubt, an diese anzudocken und so Daten auszulesen oder in sie hinein zu schreiben.
Appearance	Appearance	Appearance ist im data science portal ein Parameter, den ein Content Developer zur Gestaltung von Widgets nutzen kann.
Authentifizierung	Authentification	Für die Benutzung von YUNA müssen sich Benutzer beim Login authentifizieren. Dies geschieht je nach angelegten Authentifizierungsvariante(n) beispielsweise über LDAP.
automatische Analyse	cyclic evaluation	Automatische Analysen sind eine von zwei Analysetypen von YUNA. Unter automatischen Analysen werden Analysen verstanden, bei denen ein oder mehrere Jobs zyklisch oder manuell gestartet (R-)Analyseskripte ausführen, die einem Sachverhalt zugeordnet sind und so für eine Fachabteilung Analyseergebnisse liefern.
Backend	Backend	Ein Backend ist ein System im Hintergrund, das ein anderes System (im Vordergrund stehend) mit Dienstleistung versorgt.

Begriff	Term	Erklärung
Benutzer	User	Ein Benutzer ist ein Anwender von YUNA. Nachdem sich ein Benutzer sich mit seinen Anmeldedaten (Benutzername und Passwort) im Portal angemeldet hat, kann er die Funktionen nutzen, die für ihn freigegeben wurden. Im Normalfall wird dies durch eine Rolle geregelt, die ihm von einem Systemadministrator zugeteilt wurde.
Berechtigung	Right	Über das Vorhandensein oder Fehlen von Berechtigungen wird geregelt, welche Inhalte ein Benutzer oder eine Rolle sehen, bearbeiten oder löschen darf.
Bereichsauswahl		Eine weitere Möglichkeit zur Auswahl und Filterung von Daten im DSP besteht in der Angabe von absoluten Zeitpunkten oder Zeiträumen (z.B. festes Datum) bzw. relativen Zeiträumen (z.B. letzte 30 Tage). Dies geschieht über Widgets vom Typ DateSubfilterDirective.
Build	Build	Als Build wird ein kompilierter Code bezeichnet. Diese Änderungen an bestehendem Code oder die Erstellung von neuem Code bilden die Basis für Versionsänderungen.
Change Log	Change log	Im Change Log werden Änderungen, Neuerungen und der Wegfall von Softwarekomponenten und deren Funktionalitäten zwischen verschiedenen Versionen protokolliert.
CLI	CLI	Abkürzung für Command Line Interface. Über die Kommandozeile können beispielsweise Befehle zur Ausführung von Jobs eingegeben werden.
CM DataDB	CM DataDB	In der CM DataDB werden alle von den in YUNA eingebundenen Equipments erzeugten Daten gespeichert, die über YUNA analysiert oder erkundet werden sollen.
CM PortalDB	CM PortalDB	In der CM PortalDB werden alle relevanten Informationen zum Betrieb des Portals gespeichert. Dazu zählen beispielsweise Tabelleninhalte wie die Benutzer oder Rollen sowie Content wie Widgets, Views Queries oder DataIDs.
CMP	CMP	Abkürzung für Condition Monitoring Portal.

Begriff	Term	Erklärung
Condition Monitoring Portal	Condition Monitoring Portal	Das Condition Monitoring Portal (CMP) ist ein Produkt der eoda GmbH und stellt eine Konfigurationsmöglichkeit des data science portals (DSP) dar. Im CMP können Daten explorativ analysiert, gefiltert und visualisiert werden. Auch können dort Analysen (im Sinne von Sachverhalten) angestoßen werden, die im core ausgeführt werden.
Content	Content	Content im Sinne des data science portals sind vereinfacht gesagt Inhalte des Portals, die Art der Darstellung des Inhalts oder dessen Elemente, welche die Erzeugung von Inhalten steuern. Content kann statisch oder dynamisch sein. Beispiele für Content sind Widgets, Views, Queries oder DataIDs.
core	core	Vgl. eoda | data science core
cron pattern		Ein Ausführungsplan nach einem bestimmten Muster.
Customization / Custom Solution	Customization	Eine kundenspezifische Entwicklung bzw. Anpassung von Software und/oder deren Modulen.
Data_ID	Data_ID	Eine Data_ID referenziert eindeutig auf Inhalt wie Queries, Bilder oder Skripte. Über die Data_ID werden in Kombination mit einer Role_ID und einer DataType_ID unter anderem die Berechtigungen der einzelnen Rollen für Content-Funktionen gesteuert.
DataDB	DataDB	Vgl. CM DataDB
eoda | data science core	eoda |data science core	Der data science core ist eine Komponente des eoda data science environments. Er dient als zur Steuerung, Kontrolle und Ausführung von Zugriffsberechtigungen und Analyseskripten.
eoda | data science environment	eoda |data science environment	Das data science environment der eoda GmbH besteht aus den Komponenten data science portal und data science core.
eoda | data science portal	eoda | data science portal	Das data science portal (DSP) ist ein Produkt der eoda GmbH und Komponente des data science environments (DSE). Im DSP können Daten explorativ analysiert, gefiltert und visualisiert werden. Auch können dort Analysen (im Sinne von Sachverhalten) angestoßen werden, die im core ausgeführt werden.

Begriff	Term	Erklärung
Deeplink	Deeplink	Deeplinks ermöglichen den Austausch von individuellen Views per Link. Dabei enthält die URL des Links alle notwendigen Informationen wie z.B. Filtereinstellungen, um den Inhalt einer View immer bei jedem Benutzer gleich darstellen zu können.
Default	Default	Default beschreibt die Standardeinstellung eines Parameters.
Deployment	Deployment	Als Deployment wird die Auslieferung von Software bezeichnet. Die Auslieferung kann z.B. neue Module und Funktionen oder Bugfixes beinhalten, die auf die CM PortalDB aufgespielt werden. Ein Deployment bedeutet immer auch eine Änderung der Version des Portals.
dseConnect	dseConnect	R-Paket zur Kommunikation zwischen R und der Datenschicht (CM-DataDB) über das ds portal. Bis Version 0.49 wurde es unter dem Namen spConnect entwickelt.
dseDep	dseDep	Tool zur Verwaltung und zum Deployment von Content in Form von Widgets und Portal Views. dseDep unterscheidet zwischen den drei Entitäten Portal View, DataID und Filter. Dabei extrahiert dseDep zunächst einmalig die Struktur der Datenbank (CM-PortalDB) zur initialen Erstellung einer XML-Datei. Nach einer Bearbeitung des Inhalts wird die aktualisierte XML-Datei ins JSON-Format umgewandelt und erneut zurück in die Datenbank des CMP geladen. Bis Version 0.49 wurde dseDep unter dem Namen spDep entwickelt.
Dynamischer Filter	Dynamic filter	Die Elemente eines dynamischen Filters ändern sich automatisch, sobald neue Elemente hinzukommen, Elemente entfernt oder verändert werden.
Einzelauswahl	Single choice	Bei einer Einzelselektion werden jeweils nur die Daten zu einer ausgewählten Einheit (z.B. ein Gerät) angezeigt, die zuvor aus einer Liste ausgewählt und durchgeschaltet werden kann. Zugehöriger Widget-Typ im CMP ist die SingleChoiceDirective.

Begriff	Term	Erklärung
Equi-Liste	Equi list	Die Equi-Liste bezeichnet die Liste aller Equipments wie z.B. Items oder Maschinen, die in YUNA eingespielt wurden bzw. eine Liste aller in einem durch einen Filter selektierten Subset der in YUNA eingespielten Equipments.
Equipment	Equipment	Als Equipment werden im DSP die einzelnen im Portal eingepflegten Entitäten wie Items, Maschinen, Motoren o.ä. bezeichnet. Die Gesamtheit aller Equipments in der DSP-Instanz bildet die "Installierte Basis" der Portalinstanz.
Ergebnis	Result	Das Resultat des Durchlaufs eines Analysejobs bzw. dessen Analyseskripts im Rahmen eines Sachverhalts.
Erprobung	Test	Phase im Analyseworkflow, in dem der Sachverhalt initial anhand eines Analyseskripts in einem Erprobungsjob getestet wird.
Erweitertes Kachel-Widget	State tile	Das erweiterte Kachel-Widget ist einer von zwei Widget-Typen des DSP, die Inhalte in Kachelform darstellen. Das erweiterte Kachel-Widget kann dynamisch seinen Inhalt ändern und dient beispielsweise zur Visualisierung der Ergebnisse von Analysejobs (z.B. grüne Färbung der Kachel bei zuvor erfolgreichem Jobdurchlauf, rote Färbung bei Fehlern oder Abbrüchen)
ETL	ETL	Akronym für extract, transform, load. Das Funktionsspektrum des DSP beginnt nach dem ETL-Prozess, also wenn bereits Daten in die CM DataDB geladen sind bzw. über Schnittstellen konstant aktualisiert werden.
Feature	Feature	Als Feature werden im Kontext des DSP wahrnehmbare Ausstattungs- und Nutzungsmerkmale wie Komponenten oder Funktionalitäten des DSP bezeichnet, die in ihrer Gesamtheit das Produkt definieren. Beispielhaft können hier die Fähigkeit zur Darstellung von Content in unterschiedlichen Widgets oder die Fähigkeit zur Filterung von Content genannt werden.
Filter	Filter	Mit dem Filter kann ein Benutzer eine Teilmenge einer Datenquelle selektieren, diese speichern und anderen Benutzern über die Weitergabe einer URL zur Nutzung zur Verfügung stellen. Es stehen verschiedene Filtertypen zur Auswahl, die folgender Hierarchie unterliegen: Primärfilter, Sekundärfilter, Subfilter, (lokale) Tabellenfilter.

Begriff	Term	Erklärung
Filterhierarchie	Filter hierarchy	Über die Filterhierarchie werden Abhängigkeiten von Filtern geregelt. Die Hierarchie folgt dem Muster Primärfilter → Sekundärfilter → Subfilter → (lokale) Tabellenfilter.
Filterverwaltung	Filter management	Die Filterverwaltung ist ein Feature des DSP. Durch sie können Benutzer Filter anlegen und als globale oder eigene Filter definieren sowie von anderen Benutzern angelegte globale Filter ansehen, nutzen oder ändern.
Frontend	Frontend	Das Frontend ist die dem Benutzer sichtbare Oberfläche des DSP, in der er über Contentelemente wie Widgets Inhalte und Daten eingeben, sich diese anzeigen oder damit interagieren kann, die wiederum im Backend verarbeitet werden.
Funktion	Function	Vgl. Funktionalität.
Funktionalität	Functionality	Funktionalitäten im DSP sind die Eigenschaften und Fähigkeiten einzelner Komponenten wie beispielsweise das Sortieren von Spalten im Tabellen-Widget oder auch widgetübergreifende Funktionalitäten wie die Möglichkeit zur Filterung und Funktionalitäten im Rahmen von Workflows. Einzelne Funktionalitäten oder das Zusammenwirken verschiedener Komponenten und Funktionalitäten ergeben zusammen ein Feature.
Globaler Filter	Global filter	Ein globaler Filter ist für alle Benutzer des DSP sicht- und nutzbar.
Grid	Grid	Das Grid ist die virtuelle Grundfläche einer Portal View im DSP, in der Widgets zu einer Portal View angeordnet werden.
Installierte Basis	Installed basis	Als Installierte Basis werden im Kontext des CMP alle in der DSP-Instanz angelegten Equipments bezeichnet. Auf diese Datenbasis werden die anwendbaren Filterfunktionen, Queries und andere Funktionen des Portals ausgerichtet bzw. daran angepasst.
Issue	---	vgl. Sachverhalt
Job	Job	Ein Job im Sinne des Condition Monitoring Portals ist zuständig für das Anstoßen der Ausführung von Analyseskripten und die Ausgabe der Ergebnisse. Jobs können automatisiert (zyklisch, Zeitgesteuert) oder manuell ausgeführt werden

Begriff	Term	Erklärung
Jobformular	Job formular	Im Jobformular können Jobs angelegt, gestartet und beendet werden. Desweiteren können Jobverantwortliche und die Jobparameter und -ausführungszyklen definiert werden.
Julia	Julia	Julia ist eine Sprache zur Erstellung von Analyseskripten.
Kachel	Tile	Als Kachel werden rechteckige Widgets mit frei definierbarer Größe im DSP bezeichnet, die beispielsweise als dynamische Informationsfläche und/oder Link zu anderen Inhalten des DSP dienen können. Im Kontext des DSP existieren zwei Typen von Widgets in Form von Kacheln: Das Kachel-Widget und das erweiterte Kachel-Widget.
Kachel-Widget	Tile widget (Default Link)	Ein rechteckiges Widget vom Typ Kachel-Widget mit frei definierbarer Größe. Es dient gleichzeitig als Informationsfläche und Link zu einer weiteren, ihm untergeordneten Portal View. Beispielsweise besteht die Landing Page des DSP aus Kachel-Widgets. Neben dem klassischen Kachelwidget wie auf der Startseite des
Kategorialer Filter	Categorial filter	Vgl. Dynamischer Filter
Komponente (im Kontext des DSP)	Component	Eine Komponente zeichnet sich dadurch aus, dass sie ein Element einer komponentenbasierten Anwendung darstellt und definierte Schnittstellen zur Verbindung mit anderen Komponenten besitzt. Die genaue Form einer Komponente ist abhängig vom jeweiligen Komponentenmodell. Komponenten im Sinne des DSP sind als Benutzer wahrnehmbare, abgrenzbare Elemente (z.B. ein Tabellen-Widget). In der technischen Realisierung kann eine Komponente Funktionalitäten aus anderen technischen Komponenten vererbt bekommen (z.B. Widget-Header) oder um weitere Komponenten (Caching, Stored Procedure, ...) erweitert werden.
Landing Page	Landing page	Nach der Anmeldung im DSP befindet man sich auf seinem Startbildschirm, der „Landing Page“. Je nachdem, welche Berechtigungen für die eigene Benutzerrolle gesetzt sind, sieht man Kacheln, also Objekte vom Typ Kachel-Widget, die zu weiteren Portal Views führen.

Begriff	Term	Erklärung
LDAP	LDAP	Akronym für Lightweight Directory Acces Protocol. Das LDAP ist eine mögliche Form der Authentifizierung zur Anmeldung am DSP.
Live	Live	Als Status → Nicht mehr Test, sondern live geschaltet (Live System). Als Synonym kann der Begriff "Produktiv" genutzt werden.
Lokaler Filter	Local filter	Ein lokaler Filter ist im Gegensatz zu einem globalen Filter nur für den Benutzer sicht- und nutzbar, der ihn angelegt hat.
Mehrfachauswahl	Multi selection	Bei der Mehrfachselektion kann ein Benutzer aus einer Liste mehrere Elemente auswählen, indem er die zu den Listenelementen gehördenden Checkboxen anklickt. Der zugehörige Widget-Typ im DSP ist die FilterDirective.
Modell (statistisches, bzw. Analysemodell)	Model (analytic model)	(Ausprägungen: Continious, Batch, Live)
Paket	Package	Über Pakete können YUNA oder seine Komponenten um Funktionalitäten aus anderen Softwaretools ergänzt werden oder Verbindungen zwischen YUNA und den Funktionalitäten anderer Software hergestellt werden. Beispiele sind R-Pakete für Analysen wie spConnect zur Verbindung von YUNA mit R-Studio.
Paramenter (für Analysen)	Parameter	Über Parameter lassen sich die Eigenschaften von Jobs definieren.
Population	Population	Als Population wird im Kontext des DSP die Menge aller Equipments bezeichnet, die sich in der installierten Basis befinden bzw. in einem durch einen Filter erzeugten Subset aller Equipments der installierten Basis.
Population Filter	Population filter	Ein Filter zur Erzeugung einer (Equipment-)Population, also eines Subsets von Equipments der installierten Basis für Jobs und Sachverhalte.

Begriff	Term	Erklärung
Portal View	Portal view	Eine Portal View ist ein Grid-Element, dass 1 bis n Widgets zur Analyse einer Fragestellung vereint. Um eine eindeutige Unterscheidung zu Datenbank-Views zu schaffen, wird in Dokumentationen der Begriff Portal View genutzt. Durch die Deeplink-Funktionalität des DSP kann ein Benutzer eine ihm vorliegende View (inklusive seiner Filterselektionen) an einen anderen DSP-Benutzer versenden, indem er ihm die URL seiner Portal View zukommen lässt. Sofern der Empfänger über ausreichende Berechtigungen verfügt wird ihm exakt die gleiche View inklusive der angewandten Filter angezeigt wie dem Versender.
PortalDb	PortalDB	vgl. CM PortalDB
Primärfilter	Primary filter	Der Primärfilter ist die erste Filterebene für die Daten der installierten Basis im DSP. Geladene Primärfilter werden in der URL als eindeutiger Hashwert gespeichert und können so anderen Benutzern durch Weitergabe eines Links zu exakt dieser Portal View zugänglich gemacht werden. Ein Primärfilter kann beispielsweise in der Filterverwaltung erstellt, gespeichert und anschließend im Portal geladen werden.
Produktion	Poduction	Phase im Sachverhaltsworkflow, in dem die Analyse produktiv verwendet wird. Nachdem die Test- und Verfikationsphasen des Sachverhalts abgeschlossen wurden, wird in der Produktivphase der Sachverhalt mit einem oder mehreren Skripten untersucht. Anschließend können Rückschlüsse aus den Ergebnissen gezogen werden.
Python	Python	Programmiersprache, die u.a. zur Erstellung von Analyseskripten genutzt werden kann.
Query	Query	Als Query werden Datenbankabfragen bezeichnet. Im Kontext des DSP können diese beispielsweise in der XML-Konnotation QueryBuilder geschrieben werden.
R	R	R ist eine Sprache zur Erstellung von Skripten, mit denen statistische Analysen durchgeführt werden können.
REST	REST	Schnittstelle / API

Begriff	Term	Erklärung
Rohdaten	Raw data	Unter Rohdaten werden unveränderte Daten von Datenquellen wie beispielsweise Messwerte von Sensoren verstanden.
Rolle	Role	Benutzer einer Anwendung sind einer Rolle zugeordnet, die mit verschiedenen Rechten zur Ansicht, Bearbeitung oder Löschung von Content verbunden ist. Rollen und deren Berechtigungen können von Administratoren festgelegt und vergeben werden. Weitere typische Nutzerrollen im Kontext von YUNA sind neben den Systemadministratoren Entwickler (eoda-intern), Content Developer (eoda-intern oder externe beim Kunden), Data Scientists und Endnutzer (z.B. Qualitätsmanager, Geschäftsleitung, Sales, Controlling,...)
Sachverhalt	Issue	Fragestellung, zu der Informationen anhand von Analysen bereitgestellt werden sollen.
Sekundärfilter	Secondary filter	Filtert die Daten einer spezifischen Tabelle. Ein Sekundärfilter ist im Grunde ein Primärfilter, der allerdings hierarchisch einem anderen Primärfilter unterstellt ist und von diesem abhängt. Er basiert demnach auf den durch den Primärfilter vorgefilterten Daten und wird zur weiteren logischen Einschränkung der anzuzeigenden Anlageninformationen genutzt. Sekundärfilter werden ebenfalls in der URL als Hashwerte gespeichert und ermöglichen es so, dass die ausgewählten Filtereinstellungen über das Versenden von Links an weitere Benutzer weitergegeben werden können. Beispiel: Primärfilter = Installierte Basis, Sekundärfilter = Meldungsspeicher → im Meldungsspeicher werden nur noch Meldungen zu Equipments betrachtet, die im Primäfilter ausgewählt wurden.
Selektion	Selection	Beschreibt sowohl das Setzen von Auswahlmöglichkeiten in einem Filter als auch die anschließend gefilterte, ausgewählte Menge.

Begriff	Term	Erklärung
Session	Session	Eine Session, auch genannt Sitzung, bezeichnet eine stehende Verbindung eines Clients mit einem Server. Den Zustand dieser Sitzung gilt es zu erhalten, was insbesondere bei der Nutzung zustandsloser Protokolle (etwa HTTP) wichtige Bedeutung hat. Sessionbehandlung stellt für Intra- und Internetsysteme eine kritische Herausforderung dar und beeinflusst häufig die Performance eines Systems.
Sitzung	Session	Vgl. Session
Skript	Script	Vgl. Analyseskript
Skriptsprache	Script language	Unterstützte Sprache für die Analyseskripte (R, Julia, Python, ...)
spConnect	spConnect	R-Paket zur Kommunikation zwischen R und der Datenschicht (CM-DataDB) über das ds portal. Seit Version 0.49 wird es unter dem Namen dseConnect weiterentwickelt.
spDep	spDep	Tool zur Verwaltung und zum Deployment von Content in Form von Widgets und Portal Views. spDep unterscheidet zwischen den drei Entitäten Portal View, DataID und Filter. Dabei extrahiert spDep zunächst einmalig die Struktur der Datenbank (CM-PortalDB) zur initialen Erstellung einer XML-Datei. Nach einer Bearbeitung des Inhalts wird die aktualisierte XML-Datei ins JSON-Format umgewandelt und erneut zurück in die Datenbank des CMP geladen. Seit Version 0.49 wird spDep unter dem Namen dseDep weiterentwicklt.
Statischer Filter	Static filter	Statischer Filter sind Filter, die aus einer Liste von fest definierten Elementen bestehen. Diese Liste von Filterelementen wird nicht automatisch ergänzt oder reduziert, sobald in einer Tabelle, auf die sich der Filter bezieht, ein Element hinzugefügt, entfernt oder geändert wird. Die Anpassung der Liste der Filterelemente muss aktiv durch einen Benutzer erfolgen. Das Gegenteil sind dynamische Filter.
Store	Store	vgl. Analytic Store

Begriff	Term	Erklärung
Stored Procedure	Stored procedure	Eine Stored Procedure ist eine Funktion für Datenbankmanagementsysteme, die es ermöglicht, komplexe Folgen von SQL-Anweisungen im System unter einem Namen zu speichern und zu jeder Zeit ausführen zu können.
Subfilter	Sub filter	Lokaler Filter auf View-Ebene. Einzel- oder Multiauswahl einer vordefinierten Variable, die zur Veränderung eines spezifischen URL-Parameters führt. Bei Einzelauswahl durchschaltbar (Bspw. Sensorliste / Equipmentliste). Kann gesteuert werden durch den Widget-Typ Einzelselektion (Singlechoicedirective).
System Administrator	System administrator	Ein System Administrator ist eine Rolle in YUNA. Er verfügt über vielfältige Rechte, installiert und verwaltet YUNA auf Informationstechnologischer Seite, legt neue Benutzer an, vergibt diesen Rechte etc.
Tabelle	Table	In einer Tabelle werden Informationen in Zellen dargestellt, die durch eine Kreuzung aus Spalten und Zeilen entstehen. Die Darstellung einer Tabelle erfolgt im DSP bislang durch ein Tabellen-Widget.
Tabellen-Widget	Table widget	Ein Tabellen-Widget ist ein interaktiver und hochindividuell gestaltbarer Widget-Typ des DSP, der abgefragte Daten in strukturierter Form zeilen- und spaltenbasiert wiedergeben kann. Die Inhalte können durch Filter bereits vorgefiltert sein, durch zusätzliche Filter weiter gefiltert oder spaltenweise sortiert bzw. durchsucht werden. Weiterhin gibt es vielfältige Exportfunktionen. Auch können Typen von Spalte einer Tabelle frei definiert werden, die jeweils unterschiedliche Eigenschaften aufweisen und Funktionalitäten bereitstellen.
Tabellenfilter	Table filter	Mit einem (lokalen) Tabellenfilter lassen sich Inhalte eines Tabellen-Widgets durch die Eingabe von Suchbegriffen im Eingabefeld einer Spalte weiter filtern. Diese Tabellenfilter sind nicht speicherbar und somit können Selektionen über Tabellenfilter nicht über die Kopie von URLs weitergeleitet werden.
Test	Test	Vgl. Erprobung

Begriff	Term	Erklärung
Token	Token	Ein Token wird zur Authentifizierung der Zugangsberechtigungen gegenüber der Rest-API der Service Plattform verwendet.Er ist Base64 encoded hat den folgenden Aufbau:[32b:Sha256Hmac der folgenden 16 byte mit 128bit Secret][8byte:(Unix TimeStamp)*1000 des Ablaufzeitpunkts][8byte:Bitmaske der Permissions]Der Token kann damit 64 Zugangsberechtigungen speichern.
Truncate	Truncate	Truncate ist eine Funktionalität des Tabellenwidgets. Im Fall von langem Text in schmalen Spalten einer Tabelle wird der Text (je nach Einstellung) abgeschnitten und somit nur ausschnittsweise in verkürzter Form angezeigt. Truncate sorgt dafür, dass der abgeschnittene bzw. verkürzt dargestellte Inhalt der Tabellenzelle in voller Länge angezeigt wird, sobald der Benutzer mit dem Mauszeiger für einige Zeit über der Tabellenzelle verharrt.
Update	Update	Bugfix, Hotfix, reguläres, major,...
URL	URL	Die URL stellt die Adresse einer eindeutigen Portal View dar. In der URL einer Portal View werden von einigen Widgets Parameter (wie z.B. ausgewählte Filter) angehängt, welche widerum von anderen Widgets ausgelesen und so zur Abfrage sowie Ausgabe von Daten führen. Das DSP nutzt eine Deeplink-Funktionalität, wodurch die Weitergabe der URL an einen anderen Benutzer dazu führt, dass der Empfänger exakt die gleiche Portal View inkl. der vorgenommenen Filterungen angezeigt bekommt wie der versendende Benutzer.
User	User	vgl. Benutzer
Variante	Variant	Als Variante werden unterschiedliche Ausprägungen eines Produkts oder eines Software-Elements bezeichnet. Die Ausprägungsvarianten können sich z.B. kundenspezifisch unterscheiden oder im Umfang von Modulen, Features oder Funktionen.
Verifizierung	Verification	Phase im Analyse Workflow, in dem eine Analyse und ihre Ergebnisse fachlich durch Domain Experten überprüft und verifiziert werden.

Begriff	Term	Erklärung
Version	Version	Eine Version kennzeichnet den Entwicklungsstatus einer Software, Dokumentation o.ä. Bei der Versionierung spricht man von einer neuen Major Version, wenn inkompatible API-Änderungen vorgenommen werden (V 1.0 auf V 2.0). Von Minor Version wird gesprochen, wenn Funktionen in einer rückwärtskompatiblen Art hinzugefügt werden (V. 1.1 auf V. 1.2). Von einer Patch version ist die Rede, wenn rückwärtskompatible Bug Fixes implementiert werden (V. 1.1.1 auf V. 1.1.2).
View	View	vgl. Portal View
Widget	Widget	Das CMP verfügt über vielfältige Widgets, wie z.B. das Tabellen-Widget, Charts, Kacheln oder das Image Viewer-Widget. Diese Widgets können Informationen darstellen, sind teilweise interaktiv und dynamisch und können untereinander interagieren. Einzelne Widgets können zu einer View zusammengefasst werden.
Workflow (Analyseworkflow)	Workflow	Automatisierte Analysen im Kontext des DSP durchlaufen einen mehrstufigen, kundenindividuell gestaltbaren Analyseworkflow, in dem der zu untersuchende Sachverhalt nebst der zu untersuchenden Sachverhaltspopulation sowie die Verantwortlichen für den Sachverhalt, die Analysen und die zugehörigen Jobs definiert werden. Desweiteren werden im Rahmen des Sachverhaltsworkflows die unterschiedlichen Erprobungs, Verifikations und Produktivjobs sowie deren Ausführungsthytmen zur Untersuchung des Sachverhalts festgelegt.
XML	XML	Akronym für Extensible Markup Language. XML ist eine Sprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien und wird im Kontext des DSP zur generierung von Content verwendet.

8. Changelog

YUNA Version 2.6

Verschlüsselte Zugangsdaten im Sachverhalt

Zugangsdaten können im Sachverhalt sicher hinterlegt werden

- Werte werden verschlüsselt in der Portal-DB gespeichert
- Entschlüsselung nur mit Schlüssel des jeweiligen YUNA-Servers[1] möglich
- Werte können nicht über REST-Schnittstelle abgerufen werden
- Die Zugangsdaten können nur innerhalb von Skripten abgerufen werden
- Zugangsdaten werden nicht geloggt

Jeder YUNA-Server hat einen eigenen Schlüssel. Wird ein Server neu aufgesetzt, können bereits gespeicherte Zugangsdaten nicht mehr abgerufen werden

Arbeitsverzeichnis der Skriptausführung

Die Arbeitsverzeichnisse werden standardmäßig im temporären Verzeichnis des Betriebssystems angelegt:

- Unter Linux /tmp/
- Unter Windows ~/AppData/Local/Temp

Damit wird dem Betriebssystem ermöglicht, temporäre Dateien zu bereinigen. Über die Agent-Config kann der Ordner überschrieben werden.

Verbesserungen der Log-Ausgabe

Alle Ausgaben von YUNA wurden überprüft und folgende Log-Level gewählt

- ERROR für alle Fehler, die den Betrieb von YUNA beeinträchtigen könnten
- WARN für sonstige Fehler, z.B. Bedienfehler
- INFO für sonstige Ausgaben
- DEBUG für Testausgaben

Als Standard wird das Log nun in zwei Dateien erfasst:

- dse.log für alle Ausgaben mit verkürzten Fehlermeldungen
- dse-error.log für ERROR-Level Ausgaben

Das Logging kann nun über die Datei logback.xml konfiguriert werden.

Bisherige Konfiguration des Loggers über config.yaml hat keine Auswirkungen mehr!

Überarbeitung des Zeitfilters

Verschiedene Erweiterungen des Zeitfilters:

- Konfiguration des Zeitformats und der Darstellung über YUNALang
- Möglichkeit auch auf Millisekunden zu filtern
- Neben Textfeldern jetzt auch Nummerblöcke als Input
- Nutzereinstellung des Zeitfilter wird im Browser gespeichert

YUNA Version 2.5

YUNA Lang: Ready for Production

YUNA Lang, die VSCode Erweiterung für das Erstellen und Bearbeiten von Dashboards in YUNA, wurde weiter verbessert und hat ab sofort den Status "Ready for Production". Die neunen Features umfassen:

- Passwörter für das Deployment können nun sicher hinterlegt werden und müssen nicht mehr bei jedem Deploymentvorgang neu eingegeben werden.
- Eine Projektdatei wird jetzt in jedem YUNA Lang Projekt hinterlegt. So ist sichergestellt, dass beim Öffnen eines Projekts der richtige Ordner gewählt wurde.
- Vor dem Deployment wird nun ein Versionsabgleich zwischen der Erweiterung und der YUNA-Instanz durchgeführt, um mögliche Kompatibilitätsprobleme zu erkennen.
- Für jedes Projekt lassen sich Standards für das Deployment-Profil und den RefTag einstellen.

Gruppenspalten im DataGrid Widget

Im DataGrid Widget wurde ein neuer Spaltentyp angelegt, mit dem sich mehrere Spalten gruppieren lassen. Die Gruppen können so konfiguriert werden, dass sie auf- und zugeklappt werden können und dabei unteschiedliche Werte anzeigen.

YUNA Version 2.4

Konditionelles Einfärben von Zellen im DataGrid Widget

Im DataGrid Widget lassen sich einzelne Zellen anhand von vorher definierten Bedingungen gezielt einfärben, um diese hervorzuheben.

Übersetzungen im Kategoriefilter

Die Übersetzung der Filter wurde für die Einträge des Kategoriefilters erweitert. Die Einträge können jetzt im aktiven Filter und der Mehrfachauswahl übersetzt angezeigt werden.

YUNA Lang Deployment & Auslieferung

Die YUNA Lang Extension, sowie das CLI Tool für automatisiertes Deployment wurden in die RPM Auslieferung übernommen. Beide Tools lassen sich in der Anwendung über YUNA Tools herunterladen.

YUNA Version 2.3

Manuelle Eingabe im DataGrid-Widget

Es wurde die Funktion implementiert Felder im Datagrid-Widget editierbar zu machen. Sobald eine Spalte geändert wurde, wird der Inhalt über einen Output-Channel zur Verfügung gestellt. Um die Änderungen zu persistieren, kann ein DataIDProvider mit einer Stored Procedure verwendet werden.

YUNA Version 2.2

Vollbildmodus

Es gibt ein neues Symbol in der Caption eines Widgets  . Darüber kann der Vollbildmodus ein- oder ausgeschaltet werden.

Der Vollbild Modus ist standardmäßig deaktiviert und muss über die Dashboard-Definition aktiviert werden.

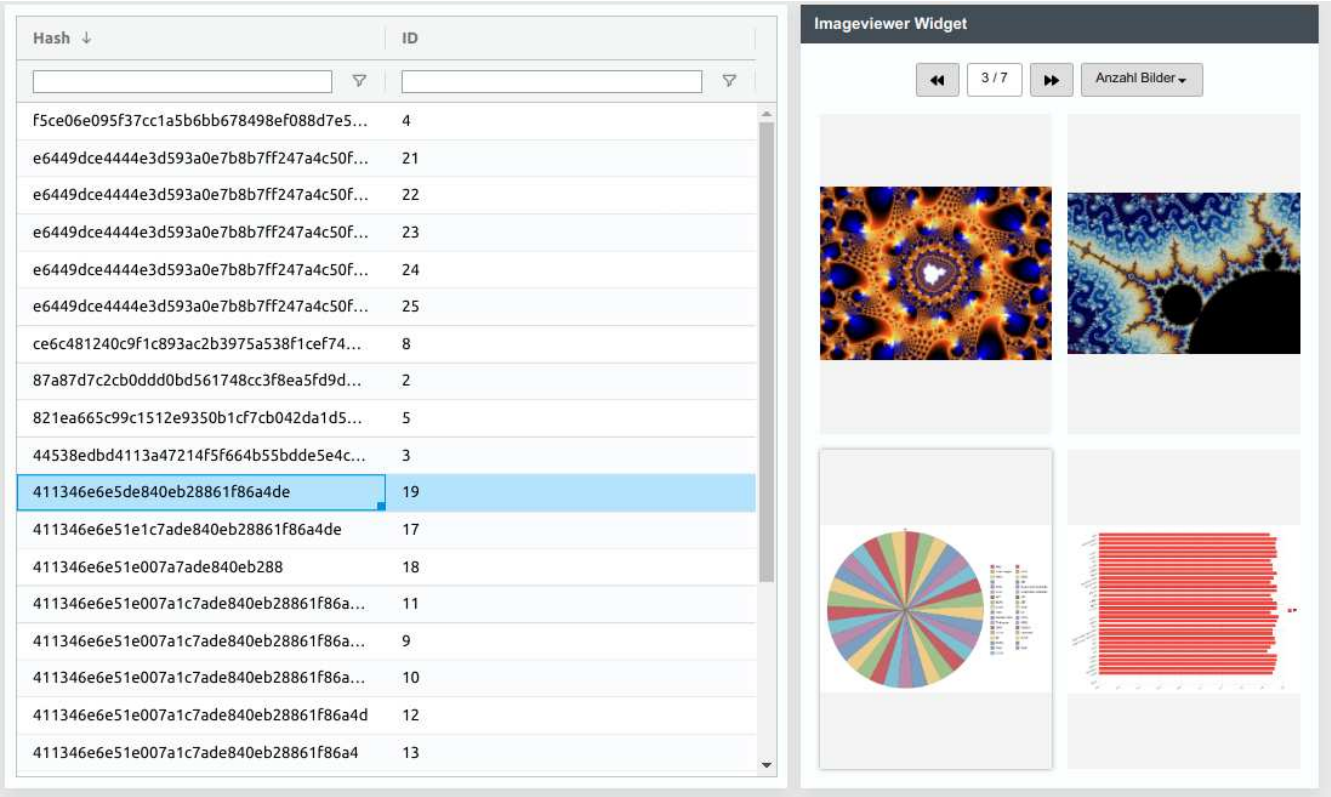
[image]

ImageViewer

Über einen Input-Channel (Zum beispiel mithilfe des DataGrid-Widgets) des ImageViewers können gezielt Bilder angezeigt werden.Damit in der Multi-Ansicht es besser ersichtlich ist, welches Bild ausgewählt wurde, wird das Bild im ImageViewer nun leicht umrandet.

Dadurch ist es leichter zu erkennen, auf welches Bild sich der Datensatz bezieht, den man gerade ausgewählt hat.

Im Screenshot ist die Umrandung in der unteren linken Ecke des ImageViewers zu erkennen.



YUNA Version 2.1

Filter-API

Es wurde eine Möglichkeit bereitgestellt, einen oder mehrere Filter anhand des Namens zu finden.Dabei gibt es verschiedene Möglichkeiten um Filter über den Namen zu finden.

In der folgenden Tabelle wird der dafür benötigte Endpunkt genauer beschrieben.

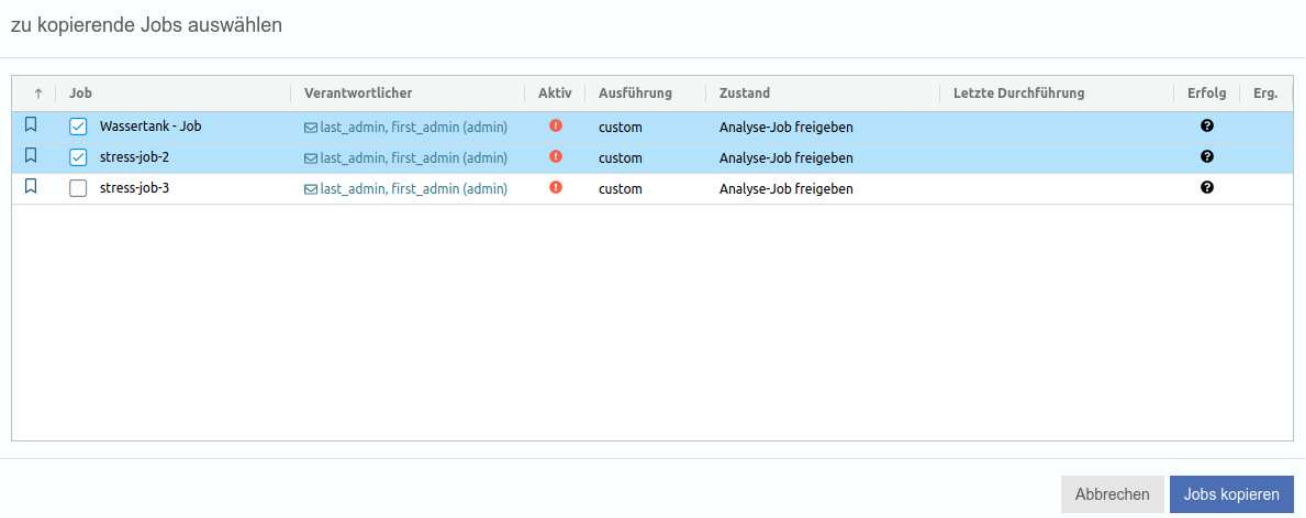
Methode?		**URL?**
GET?		.../de.eoda.dse.portal.filter.rest/info/all?
Parameter?	**Werte?**	**Beschreibung?**
name?	Beliebiger String?	Der Name nach dem Filter gesucht werden sollen?
matcher? *optional*?	contains? *default*?	Parameter *name* ist im Filternamen enthalten?
	prefix?	Parameter *name* ist der Anfang der Filternamens?
	exact?	Parameter *name* ist der exakte Filtername?

Duplizieren von Jobs im Sachverhaltsprozess

Die Tabellen in den Sachverhaltsphasen ***Analyse erproben***, ***Sachverhalt verifizieren*** und ***Sachverhalt produktiv*** wurden grundlegend erneuert.
Als Basis dient jetzt AG-Grid, das auch im neuen Datagrid zum Einsatz gekommen ist.

In der Tabelle gibt es einen duplizieren Button, um einen **einzelnen** Job zu duplizieren.

Desweiteren ist in den Phasen ***Sachverhalt verifizieren*** und ***Sachverhalt produktiv*** ein Button zum duplizieren der Jobs aus der vorherigen Sachverhaltsphase hinzugekommen.Dieser öffnet ein Modal-Fenster in dem die Jobs, die dupliziert werden sollen, ausgewählt werden können.



YUNA Version 2.0

Neuer Technologie-Stack mit Java 11

Filter-API: Auslesen aller Filter eines Filterbezeichners

YUNA Version 1.9.0

Neue Funktionen

Formularwidget

Formularwidget hinzugefügt. Es können Formulare definiert und ein Endpunkt, an den die Daten geschickt werden, konfiguriert werden.

JavaScript im HTML Widget

Scriptmode in HTML-Widget eingebaut. Javascript kann ohne Einschränkungen benutzt werden.

Fehlerlogging

Script Logging erweitert. Bei Fehlern wird das Script Log jetzt um eine konfigurierbare Anzahl vorhergehender Zeilen erweitert.

Zurück-Button

In Dashboards funktioniert der Zurück-Button des Browsers wieder.

Außerdem weitere kleinere Anpassungen und Bugfixes

YUNA Version 1.8.0

Neue Funktionen

Job-Parameter Logging

Im Scriptlog werden die Parameter eines Jobs geloggt. Auf diese Weise wird die Nachvollziehbarkeit von Jobergebnissen sowie die Reproduzierbarkeit derselben erhöht.

API Authentifizierung

Die Userverwaltung kann nun über Basic-Authentication genutzt werden.

Info-Button

Hinter dem neuen Info-Button verbergen sich Verweise zum YUNA Support sowie zum YUNA Changelog. Auf diese Weise können sich Nutzer ein besseres Bild über die Funktionalität und Veränderungen in neuen Versionen machen oder Hilfe bei Problemen finden.Die Menüeinträge sind leicht erweiterbar und konfigurierbar, so dass zum Beispiel ein Verweis auf eine Sicht für Änderungen in den Dashboards durch den Kunden eingebunden werden kann.

Hinweisfenster nach Aktualisierung

Seit dieser Version wird dem Nutzer nach einem Upgrade ein Informationsfenster angezeigt, das ihn auf die neue Version aufmerksam macht.

Weitere kleinere Anpassungen und Bugfixes

YUNA Version 1.7.0

Versionshinweise - YUNA - Version 1.7

- Features
  - [DPE-616] - Mit einem roten Badge wird die Anzahl aktivierter Filterkriterien im Aktive-Filter-Widget visualisiert
  - [DPE-816] - Unter dem Menüpunkt "Neuerungen im Portal" werden die YUNA Release Notes angezeigt
  - [DPE-901] - Das Logging von benutzergesteuerten Abbrüchen von Datenbankabfragen ist erweitert worden
- Bug

- [DPE-883] - Das Logging wurde so erweitert, dass auch invalide Filter dort identifiziert werden können

YUNA Version 1.6.1

Versionshinweise - YUNA - Version 1.6.1

- Bug
  - [DPE-749] - BUG - Aktuelle Anzahl zu analysierender Equipments bleibt bei "Laden" im Neue Job
  - [DPE-848] - Anzahl Items - bleib bei "Laden" wenn Filter gelöscht
  - [DPE-876] - Population Filter von Issue kann nicht gelöscht werden
  - [DPE-900] - Hotifx 1.6.1 - Anforderungen

YUNA Version 1.6.0

Changelog YUNA 1.6

- Features
  - Neues Feld: Beschreibung von Rollennamen.
  - Verbessertes Anwendungslog.
  - Der Abbruch von einem Job (manuell oder durch Agentenentkopplung) wird jetzt auch im Cyclic Evaluation Log eingetragen.
  - Schnittstelle (REST-API) zur Userverwaltung.
  - Das automatische Anlegen neuer LDAP-User in der Datenbank kann per Konfiguration jetzt abgeschaltet werden.
  - Bei der Definition einer Single-Choice-Directive mit freier Eingabe im Content kann jetzt einen Default-Wert gesetzt werden.
  - Die Single-Choice-Directive kann jetzt ohne Query als reines Eingabefeld und ohne Definition einer Listenansicht verwendet werden.

YUNA Version 1.35.0

Datagrid Widget - Nachfolger des Table-Widget

Neben deutlichen Performance-Verbesserungen bei der dynamischen Darstellung großer Datensätze bringt das DataGrid Widget die Grundlage für eine Vielzahl neuer, spannender Features für die Anwendung mit:

1. **Interaktion**Integrierte Volltext-Suche in den jeweiligen SpaltenVerbessertes Zusammenspiel mit anderen Widgets, so lassen sich z.B. verschiedene Formatierungen und Funktionen z.B. durch Filter-Widgets ändern
2. **Erweiterte Filter**Die abgebildeten Daten können nun auch mittels Vergleichsoperatoren (equals, lower than, higher than, in range etc.) gefiltert und eingegrenzt werden.
3. **Übersichtlicher**Kombination verschiedener Stammdaten z.B. aus verschiedenen Datenbank-Spalten in einer einzigen SpalteSpalten lasen sich beliebig in ihrer Breite verändern

Schneller Zusammenhänge erkennen – Integrated Charts

Das neue DataGrid bietet außerdem die Möglichkeit, Daten aus der Tabelle heraus direkt zu visualisieren! Dazu können die gewünschten Zellen einfach markiert werden, dann kann per Rechtsklick ein geeigneter Diagrammtyp gewählt werden, der sich auch im Nachhinein noch modifizieren lässt. Entsprechend lassen sich Verläufe aber auch Zusammenhänge noch schneller erkennen – natürlich lassen sich diese Diagramme dann wieder mit anderen YUNA-Usern teilen oder exportieren.

Vorfilter

- Möglichkeit implementiert, damit ein Vorfilter auch auf einen Join wirken kann.
- Die Vorfilter API um einen Endpunkt erweitert, mit dem die einzigartigen Filter-Definitionen abgefragt werden können

YUNA Version 1.34.0

DataID-Provider: Verbesserte Definition von Parametern

Es ist nun einfacher verschiedene Parameter, wie z.B. URL-Parameter, einzelne Zellen einer selektierten Tabellenzeile oder Filter, an einen DataID-Provider zu übergeben.

Siehe DataID-Provider

YUNA Version 1.33.0

Kennzeichnung von Sachverhalten & Jobs als "Favoriten"

Jobs und Sachverhalte können nun über die Job- und Sachverhalts-Listen und in ihren jeweiligen Editoren als Favoriten gekennzeichnet werden. Mithilfe der Sortierung von Tabellen können diese schnell gefunden werden.

Werden Job- und Favoritendaten in eigenen Tabellenwidgets dargestellt, können mit dem Favorite-Provider und dem Spaltentyp Favorit (favorite), die Favoriten auch in beliebigen Tabellenwidget angezeigt werden. Ebenso können beliebige andere Daten um die Kennzeichnung von Favoriten ergänzt werden.

Neu markierte und entfernte Favoriten werden erst nach einer Aktualisierung der Seite korrekt sortiert.

Visualisierung aktiver Ausprägungen im Filterwidget

In den Filterwidgets (active filter und Mehrfachauswahl) werden die einzelnen Filterkategorien, nach denen gefiltert wird, nun farblich hervorgehoben, sodass leicht erkennbar ist, nach welchen Kategorien aktuell gefiltert wird und welche Kategorien Teil eines geladenen Filters sind.

YUNA Version 1.32.1

Formatierung von numerischen Spalten im Tabellenwidget

Über den Parameter "options" kann die Darstellung numerischer Tabellenspalten umfangreich konfiguriert werden, um z.B. Währungen anzuzeigen.

Siehe numerische Tabellenspalten

YUNA Version 1.32.0

Kopieren von Rohdaten aus Tabellenspalten

Über den Parameter "copy" kann das Kopieren der Werte einer Tabellenspalte aktiviert werden?. Dadurch wird ein Icon im Header der Spalte hinzugefügt, über das die Werte der Spalte in die Zwischenablage kopiert werden können. Beim Kopieren werden die aktuelle Sortierung und Filterung der Tabelle beachtet.

Siehe Erweiterte Eigenschaften

Job-Warteschlangen-Management

Neben der maximalen Anzahl laufender Instanzen pro Job, kann über den Parameter "maxOverallParallelJobExecutionCount" nun auch die maximale Anzahl an Jobs insgesamt konfiguriert werden. Wird der Wert überschritten landet der Job in der Warteschlange und wird erst ausgeführt, wenn die Ausführung möglich ist.

Siehe Parallele Jobausführung

Hinweis auf fehlende Auswahl in Widgets

In Widgets, deren Hauptfunktion das Anzeigen von Daten ist (Tabellen-Widget, Chart-Widget und Stockchart-Widget), wird ein Hinweis in der Caption angezeigt, wenn nicht alle notwendigen Triggerparameter gesetzt sind. Dieser Hinweis kann durch den Dashboardentwickler konfiguriert werden, um für das jeweilige Dashboard individuelle Hinweise zu geben.

In Widgets, die zur Auswahl von Parameter dienen (Singlechoice-Widget, Datesubfilter-Widget und Sensorlist-Widget), kann über den Parameter "mandatory" festgelegt werden, dass eine Auswahl notwendig ist. Ist der Paramter gesetzt, wird das Widget hervorgehoben, solange keine Auswahl getroffen wurde.

YUNA Version 1.31.0

Bereichsselektion in kategorialen Filtern

In kategorialen Filtern (Type "category") kann nun durch drücken der Shift-Taste ein Bereich zwischen zwei Werten ausgewählt werden.

YUNA Version 1.30.0

Usability-Verbesserungen

- Im Sachverhalt kann nun über einen zweiten Button zwischen Sachverhaltsphasen gewechselt werden, ohne einen Kommentar eingeben zu müssen.
- Im Job-Editor sind freigegebene Jobs (Phase 2) standardmäßig aktiv und müssen nicht erst manuell aktiviert werden
- Im Job-Editor werden Änderungen beim Wechsel in die nächste Phase automatisch gespeichert.
- Im Tabellenwidget kann die Suche alternativ dauerhaft angezeigt werden. Siehe Analytische Funktionen → Search
- Durch Doppelklick auf deaktivierte Filterkategorien und Auswahlmöglichkeiten, kann der geladene Filter direkt aufgelöst werden um die entsprechenden Optionen verändern zu können. Der dazugehörige Warndialog kann über eine Konfiguration im Configstore deaktiviert werden.

YUNA Lang als Early Access verfügbar

Die Erstellung von Dashboards wird mit der Einführung von YUNA Lang signifikant vereinfacht. Validierung, Auto-Completion, Auto-Formating, Komfortunktionen wie Onhover-Hilfetexte und ein erleichtertes Sourcing sind nur einige der Funktionen, die den Nutzern nun in Form eines Early Access erstmals zur Verfügung gestellt werden.

> Mit Version 1.30 werden Leertags (s.u.) innerhalb von Widget-Definitionen nicht mehr offiziell unterstützt.
>
> Bis einschließlich Version 1.32 werden Widget-Definitionen mit diesen Tags weiterhin funktionieren.
>
> Ab Version 1.33 werden diese Widget-Definitionen nicht mehr funktionieren
>
> **Beispiele für Leertags**
>
> ```
> <tag></tag>
> </tag>
> ```

YUNA Version 1.29.0

Result Rating an IO-Channel angebunden

Über die IO-Channel kann das Result Rating mit anderen Widgets gekoppelt werden. So ist es zum Beispiel möglich, Tabellen-Widget und Result-Rating zu koppeln, um immer die aktuell ausgewählte Zeile zu bewerten. Die Daten werden zwischen Tabellenwidget und Result Rating synchronisiert, sodass z.B. Sortierung und Filterung in beiden Widgets immer identisch sind.

Dafür wurde das Result Rating in ein Widget und einen IO-Provider aufgetrennt: Der Result-Rating-Provider ergänzt die abgerufenen Daten um Bewertungen und das Widget stellt die Oberfläche zur Bewertung der Daten bereit.

Mehr Informationen unter: Result-Rating-Widget und Result-Rating-Provider

YUNA Version 1.28.0

Vorfilter

Es ist jetzt möglich für einen Vorfilter mehrere ODER-verknüpfte Filterhashes zu definieren. Dadurch werden alle Daten angezeigt, die die Bedingungen des ersten Filterhashes oder die Bedingungen des zweiten Filterhashes erfüllen, sodass die Gesamtmenge aller zutreffenden Einträge und nicht nur wie zuvor die Schnittmenge (durch eine UND-Verknüpfung) dargestellt werden kann. Zudem besteht nun die Möglichkeit die Vorfilter über eine REST-API zu setzen und zu aktualisieren.

Siehe: YUNA-Vorfilter

Erweitertes Server-Log

Das Server-Log wurde erweitert, sodass in Fehlerfällen bei einer Data ID die Ursache besser zurückverfolgt werden kann und somit das Debugging vereinfacht wird.Um dies zu gewährleisten werden von nun an u.a. Informationen wie der Name der Data ID und die dazugehörigen Parameter geloggt.

Sowie kleinere Änderungen und Bugfixes

YUNA Version 1.27.0

Überarbeiteter Tabellenexport

UX verbessert: * Mit weniger Klicks zum selben Ergebnis * Weniger "unnötige" Schalter * Übersichtlicher und eindeutiger

Die zuletzt verwendete Konfiguration kann im Browser und optional über verschiedene Dashboards hinweg gespeichert werden.

Es können automatisch generierte Dateinamen konfiguriert werden, um bei mehreren aufeinanderfolgenden Exports sofort passende Namen zu vergeben.

Siehe Analytische Funktionen

Neues Widget: Dashboard-Übersicht

Die Dashboard-Übersicht zeigt alle für den jeweiligen Nutzer unter dem aktuellen Referenz-Tag verfügbaren Dashboards an.

Siehe Dashboard-Übersicht

Konfigurierbarer RememberMe-Zeitraum

Die Gültigkeitsdauer des RememberMe Zeitraums kann nun über die config.yaml konfiguriert werden. Der Standardwert beträgt 30 Tage.

Siehe Remember-Me

Sowie kleinere Änderungen und Bugfixes

YUNA Version 1.26.0

Absicherung der Core-API

Für die Core-API verfügen nur noch Systemadministratoren über Schreibberechtigungen. Alle anderen Rollen verfügen nur über Leseberechtigungen.

Die Dokumentation der Core-API kann unter {Server-Adresse}/swaggerui abgerufen werden, sofern der Proxy, i.d.R. Apache HTTP Server, entsprechend konfiguriert wird.

Benachrichtigungsschnittstelle - MVP

Die Benachrichtigungsschnittstelle steht zur Evaluation als MVP und Opt-In-Feature zur Verfügung. Das Feature kann über den Eintrag "message.active" im Configstore aktiviert werden.

Da durch vergangene Jobausführungen bereits große Mengen an Nachrichten vorhanden sein können, kann es sein, dass das Feature initial langsam ist. Um alte Daten zu bereinigen und die Geschwindigkeit zu erhöhren, steht ein Aufräumjob zur Verfügung, der in der config.yaml konfiguriert werden kann. Siehe Script Logging

UX-Verbesserungen

Es wurden einige UX-Verbesserungen vorgenommen:

- Skripte im Skriptmanager können nach Namen gefiltert werden.?
- Standardfilter werden nicht mehr aufgelöst dargestellt.
- Bei dem Versuch Änderungen am System-Standardfilter zu speichern, wird eine Warnmeldung angezeigt.?
- Im Tabellenwidget werden irreführende Statusinformationen ("keine Daten") vor und während des initialen Ladens der Daten ausgeblendet?
- Für GenLinks kann über die Option "openInNewTab" konfiguriert werden, ob ein Link in einem neuen Tab oder im aktuellen Fenster geöffnet wird.

Sowie kleinere Änderungen und Bugfixes

YUNA Version 1.25.1

Skriptmanager: Auswahl gespeicherter Skripte und Sortierung

Im Skriptmanager bleibt ein gespeichertes Skript ausgewählt und kann sofort in einem Sachverhalt verwendet werden. Außerdem wurde die Sortierung der Skriptliste verbessert.

YUNA Version 1.25.0

Überarbeitetes Result Rating

Das Result Rating wurde wesentlich überarbeitet. Das Widget kann nun direkt in Dashboards integriert werden und wird nicht mehr in einem Modal angezeigt.

Desweiteren erfolgt die Datenanbindung nun über DataIDs.

Siehe Result-Rating-Widget (resultrating)

Systeminfo: Anzeige des Startzeitpunkts laufender Jobs

Ín der Systeminfo wird nun wieder der Startzeitpunk laufender Jobs angezeigt.

Git-Skriptmanager: Branch-Referenz in Git-Origins

Beim Anlegen und Aktualisieren von Git-Origins kann nun über den Parameter "branchReference" ein Branch angegeben werden, der angebunden werden soll.

Siehe Git Repository anbinden

Sowie kleinere Änderungen und Bugfixes

YUNA Version 1.24.0

Neue Funktionen

Skriptmanager: Git-Anbindung

Über die Git-REST-API können Skripte zur Auswahl und Ausführung in YUNA angebunden werden.

Siehe Git-Anbindung

Überarbeiter Job-Lifecycle

Zur besseren Nachvollziehbarkeit verschiedener Job-Ausführungen, wurde der Job-Lebenszyklus um verschiedene Status ergänzt.

Scriptlog-Cleanup überarbeitet

Das Aufräumen der Skriptlogs wurde derart überarbeitet, sodass sie freier konfiguriert werden kann und geringere Systemlast erzeugt wird.

Siehe Script Logging

Systeminfo: Anzeige wartender Jobs

In der Systeminfo werden jetzt auch Informationen zu wartenden Jobs angezeigt.

Siehe Systeminfo

dsedep: Übersichtlicheres Logging

Beim Deployment von Dashboard-Inhalten mit dsedep wurden unnötige und doppelte Informationen entfernt. Fehlermeldungen beim Parsing werden nur unter Verwendung der Option "–verbose" bzw "-v" angezeigt.

Siehe Dashboard Inhalte deployen

GenLink: Neues Flag im Label

Im Label von GenLinks kann das Sonderzeichen "\" vorangestellt werden, um zu vermeiden, dass das Label als HTML verarbeitet wird. So können auch Texte, die fälschlicherweise als HTML interpretiert werden könnten, als Label verwendet werden.

Siehe GenLink

Sowie kleinere Änderungen und Bugfixes

YUNA Version 1.23.3

Neue Funktionen

Aufräumen von Log-Einträgen

Log-Einträge in den Tabellen scriptlog und cyclicEvalutationLog können jetzt automatisch aufgeräumt werden.Möglich wird dies durch eine Konfiguration in der configStore Tabelle.

Siehe: Configstore, Skriptlog

Abrufen von Jobparameter einer Jobinstanz

Die Jobparameter einer Jobinstanz können jetzt sowohl über eine Datenbank-View als auch über einen Rest-Endpunkt abgerufen werden.

Siehe: Skriptlog

"YUNA Support" Eintrag im Informationsmenü

Die Sichtbarkeit des Eintrags "YUNA Support" im Informationsmenü kann jetzt konfiguriert werden.

Siehe: Konfigurationen in der env.json

Sowie kleinere Änderungen und Bugfixes

YUNA Version 1.22.0

Neue Funktionen

Zeitplanung von Jobs

Der Jobmanager hat einen neuen Zeitplanungsassistenten für die automatische Jobausführung

bekommen.

Dieser hat eine neue Benutzeroberfläche und die Möglichkeit mehrere Tage,Stunden und Minuten zu wählen.

Siehe: Jobmanager, Zyklische Ausführung

Sowie kleinere Änderungen und Bugfixes

YUNA Version 1.21.3

Neue Funktionen

Skriptsprache jetzt im Skriptmanger einstellbar

Es ist jetzt möglich die Skriptsprache (Python oder R) im Skriptmanager festzulegen. Bei der Jobausführung wird dann automatisch ein passender Agent ausgewählt.

Ein Beispiel für die Konfiguration des Python Agenten ist hier zu finden

Modernisierter Imageviewer

siehe: Imageviewer

Der Imageviewer wurde vollständig neu implementiert und stellt nun zahlreiche neue Features bereit:

- Navigation mit Tastaturkürzeln
- Navigation zu beliebigen Bildern mit wenigen Interaktionen
- Über den zusätzlichen Input-Channel 'src' können Bilder nun aus weiteren Quellen geladen werden, z.B. URLs
- Moderneres Userinterface

Die YUNAML-Definition des Widgets hat sich nicht verändert. Es sind keine Änderungen notwendig.

config.yaml: Mehrere Wartungsfenster

siehe: Konfiguration von Wartungsfenstern

Die Konfiguration eines Wartungsfensters bewirkt, dass in dem definierten regelmäßigen Zeiträumen keine Jobs ausgeführt werden.

Nun können Mehrere Wartungsfenster definiert werden.

Prototyp der Prometheus Schnittstelle

siehe: Monitoring

Es wurde ein Prototyp für eine Prometheus Schnittstelle zum Monitoring von Yuna aktiviert.

YUNA Version 1.20.1

Kleinere Änderungen und Bugfixes

YUNA Version 1.20.0

Neue Funktionen

Gruppenfilter für Teams

siehe: Aktive Filter, Filterverwaltung

Um die Teamarbeit mit Filtern zu verbessern, haben wir Gruppenfilter eingeführt. Somit ist es möglich, dass ein Filter mehrere Besitzer hat. Besitzer von Filtern haben die gleichen Rechte, wie der ursprüngliche Erzeuger.

Hinweis Für Dashboard-Entwickler und Datenbank-Administratoren

In der Tabelle portal.FilterInfo wird die Spalte "User_ID" zu "Creator_ID" umbenannt. Referenzen auf diese Spalte müssen entsprechend angepasst werden.

Übergabe der aktuellen URL-Parameter bei Links (genLink)

siehe: Links (genLink) vYUNA_DOC_1.20

Um Dashboard-Entwicklern mehr Flexibilität bei der Widget-Gestaltung zu bieten, haben wir die Konfiguration von GenLinks so erweitert, dass die aktuell aktiven URL-Parameter zusätzlich zu den explizit konfigurierten Parametern übergeben werden können.

Sowie kleinere Änderungen und Bugfixes.

YUNA Version 1.19.0

Kleinere Änderungen und Bugfixes

YUNA Version 1.18.0

Neue Funktionen

dseconnect als REST-API

Funktionen von dseconnect werden als REST-API bereit gestellt.

- Login
- Ausführen von DataIDs (inkl. Filter und Vorfilter)
- Abrufen von Queries zu DataIDs

Deaktivieren automatischer Jobs

Über das Systeminfo-Widget können automatische Jobs deaktiviert werden.

Sachverhalt löschen als Verantwortlicher

Der bei einem Sachverhalt eingetragene Verantwortlicher darf jetzt den Sachverhalt löschen.

Sowie kleinere Änderungen und Bugfixes.

YUNA Version 1.17.0

Neue Funktionen

Escaping in genLinks

Über einen Backslash ("\") kann nun verhindert werden, dass Werte in genLinks escapt werden.

Entfernen von YUNAML-Inhalten unter Referenz-Tags

Über die Option '–clean' können alle Inhalte unter einem Referenz-Tag entfernt werden.

Parameter zur automatischen Anmeldung an Zielservern bei CORS-Requests

Der Parameter 'unsafeCredentialDelegation' des HTTP-Providers erlaubt es die automatische Anmeldung an Zielservern bei CORS-Requests zu aktivieren

Sowie kleinere Änderungen und Bugfixes

YUNA Version 1.16.1

Kleinere Änderungen und Bugfixes

YUNA Version 1.16.0

Neue Funktionen

Unterschiedliche Datenbankzugriffe aus verschiedenen Sachverhaltstypen

Je Sachverhalt kann die zu verwendende DataSource ausgewählt werden, um die Datenbankanfragen restringieren zu können.

Verschlüsselte Connections auf Daten-DBs

Alle Client-Verbindungen aus dem YUNA-Produktumfeld können so konfiguriert werden, dass sie nur noch über verschlüsselte Verbindungen erfolgen:

- Änderungen in der dse.conf.xml
- Änderungen in der config.yaml

JQuery im HTML-Widget

Es ist möglich, JQuery im HTML-Widget einzubinden.

Sowie kleinere Änderungen und Bugfixes sowie Erweiterungen in der Dokumentation

YUNA Version 1.15.0

Neue Funktionen

Abbrechen von Stored Procedures

Es können jetzt auch Stored Procedures abgebrochen werden.

Konfiguration für das Abbrechen von Datenbankabfragen

Standardmäßig können alle Datenbankabfragen, sofern der jeweilige Datenbanktreiber dies zulässt abgebrochen werden. Über die Service-Konfiguration kann dies global geändert werden, über den <cancelable>-Tag für einzelne DataIDs.

DataID-Provider

Der DataID-Provider ermöglicht den Aufruf von DataIDs und die Bereitstellung der jeweiligen Ergebnisse über IO-Channel.

Sowie kleinere Änderungen und Bugfixes

YUNA Version 1.14.1

Kleinere Änderungen und Bugfixes

YUNA Version 1.14.0

Neue Funktionen

HTTP-Provider

Neue YUNAML-Komponente, die die Einbindung von externen HTTP-APIs in Dashboards ermöglichen.

Erweitertes Logging

Die Ausgabe von bestimmten Returnwerten aus Funktionsaufrufen im Skriptlog wird nicht mehr unterdrückt.

Sowie kleinere Änderungen und Bugfixes

YUNA Version 1.13.0

Neue Funktionen

Shiny Widget-Authentifizierung gegen YUNA

Das in der letzten Version vorgestellte iFrame-Widget kann nun verwendet werden, um Shiny-Apps einzubinden und diese gegen YUNA zu authentifizieren, dh. auf die in YUNA vorliegenden Daten zuzugreifen.

ResultRating: Bewerten ist jetzt über die Tastatur möglich

Durch die Belegung von Tastaturtasten können Ergebnisse jetzt komfortabel mit der Tastatur bewertet werden.

FormularWidget: Daten können an eine Stored Procedure übergeben werden

Ab jetzt ist es möglich, die Daten, die über das Formularwidget exportiert werden an eine Stored Procedure zu übergeben.

Lokalisierung erweitert

Es ist jetzt möglich, den Inhalt von CDATA-Blöcken, Popups, allen Tooltips und Items im Filterwidget zu übersetzen

Sonderzeichen in gesourcten Objekten

Gesourcte Objekt im htmlParams-Tag unterstützen jetzt Sonderzeichen

Meldung bei fehlerhaftem Anmeldeversuch

Meldet sich ein Nutzer mit falschen Anmeldedaten an, wird eine entsprechende Meldung angezeigt.

Sowie kleinere Änderungen und Bugfixes

YUNA Version 1.12.0

Neue Funktionen

IFrame-Widget

Einführung eines einfachen iFrame-Widget's zur Einbettung von Shiny-Apps in YUNA.

URL-Parameter

URL-Parameter werden jetzt auch innerhalb von <htmlParams> Elementen interpretiert.

YUNA Version 1.11.1

kleinere Anpassungen und Bugfixes

YUNA Version 1.11.0

Neue Funktionen

ResultRating

Mit dem neuen Tool "YUNA ResultRating" können Analyseergebnisse komfortabel von Nutzern bewertet werden.

Die bewerteten Ergebnisse können zum Beispiel zum Überprüfen existierender Algorithmen dienen oder zum Trainieren von völlig neuen Algorithmen verwendet werden.

DataSource Widget Kommunikation

Es ist möglich, Widgets so zu konfiguieren, dass diese über den DataSource miteinander kommunzieren und etwa die angezeigten Inhalte im Imageviewer von der Anzeige im Tablewidget abhängt.

YUNA-Tools

Die YUNA-Tools dseconnect (bzw. spconnect), dsedep und das Connectivity Paket befinden sich an einer zentralen Stelle in der RPM-Datei und können zusätzlich über den Menüpunkt "YUNA Tools" im Info-Menü heruntergeladen werden.

Maximale Zahl paralleler Jobs

Die maximale Zahl parallel ausführbarer Jobs kann konfiguriert werden.

Weitere kleinere Anpassungen und Bugfixes

YUNA Version 1.10.0

Neue Funktionen

Core-API

Die Schnittstelle zur Core ist ausführlich dokumentiert und der direkte Zugriff darauf ist nun möglich.

Kartenwidget

Das Darstellen von Geodaten auf einer Kartenansicht ist nun möglich.

SQL-Statements werden schneller abgebrochen

Der Abbruch von SQL-Statements ist verbessert worden.

Markierbare Zeilen im Tabellenwidget

Das Tabellenwidget kann nun eine Konfiguration erhalten, damit Zeilen ausgewählt werden können.

Standardwerte im Zeitbereichsfilter

Der Zeitbereichsfilter kann nun mit einem Standardwert versehen werden, um initial eine eingeschränkte Datenmenge zu laden.

YUNA-Infocenter im Info-Menü

Über das Info-Menü kann nun auf das YUNA-Infocenter zugegriffen werden.

Länge von Job-Parametern

Job-Parameter können jetzt länger als 100 Zeichen sein.

Anzeige von Logo und CSS bei der Theme-Erstellung

Bei der Theme-Erstellung werden sowohl das ausgewählte Logo als auch die ausgewählte CSS-Datei wieder angezeigt.

DSE Version 1.5.1

Versionshinweise - DSE Produkt-Entwicklung - Version 1.5.1

- Bug
  - [DPE-851] - Jobformular landet in einem Deadlock

DSE Version 1.5.0

Vorfilter

Seit Version 1.5 ist das Feature "**Vorfilter**" in YUNA enthalten. Mit einem Vorfilter ist es dem Administrator möglich, die verfügbaren Daten für einzelne Nutzer im Vorfeld einzuschränken. Ein Nutzer sieht damit nur die für ihn relevanten Daten und kann diese mit eigenen Filtern weiter bearbeiten. In der Dokumentation finden sich in den Abschnitten für User und für Administratoren alle benötigten Informationen.

Versionshinweise - DSE Produkt-Entwicklung - Version 1.5

- Features
  - [DPE-550] - Vorfilter schränkt das Ergebnis von Queries ein
  - [DPE-555] - Visualisierung, dass ein Vorfilter aktiv ist
  - [DPE-704] - Erstellung einer neuen Rolle für Vorfilterverwaltung

    - [DPE-213] - spconnect: Liste aller Parameter, welche das Portal standardmäßig an ein Skript übergibt, dokumentieren
- Change Request
    - [DPE-679] - Script_ID wieder befüllen
- Bug
    - [DPE-801] - CM-Portal 1.4.3: Speicherleck
    - [DPE-560] - DateSubFilterDirective funktioniert mit Rolle "AdHoc_Full_Issue" nicht
    - [DPE-799] - Im Issue- und Job-Formular kann keine Kopie eines Filters übernommen werden
    - [DPE-110] - Filter-Details: Liste der Jobs und Issues werden nicht geladen

DSE Version 1.4.5

Versionshinweise - DSE Produkt-Entwicklung - Version 1.4.5

- Bug
    - [DPE-791] - Sensorsicht: Meldungsanzeige funktioniert nur, wenn man alle anwählt und nach Abwahl einzelner Kategorien werden diese angezeigt
    - [DPE-790] - Sensorsicht: Es lassen sich keine Bilder mehr generieren, es läuft nur ein Ladespinner.
    - [DPE-787] - CM-Portal 1.4.3: System startet nach Deployment auf Test nicht richtig
    - [DPE-748] - CM-Portal: Doppelter User bei Selbstregistrierung
    - [DPE-803] - Keine Paginierung im Internet Explorer
    - [DPE-764] - PNG-Export Stockchart funktioniert nicht

DSE Version 1.4.4

Versionshinweise - DSE Produkt-Entwicklung - Version 1.4.4

- Bug
    - [DPE-772] - Die Definition von Filterkriterien der Population war fehlerhaft
    - [DPE-789] - Sekundärfilter sind zum Teil ohne Inhalt, obwohl die Anzahl vor dem Aufklappen angegeben ist

DSE Version 1.4.3

Versionshinweise - DSE Produkt-Entwicklung - Version 1.4.3

- Bug
    - [DPE-769] - Minor Liquibase Fixes

DSE Version 1.4.1

Versionshinweise - DSE Produkt-Entwicklung - Version 1.4.1

- Bug
    - [DPE-763] - Portal lädt nicht zuverlässig im Internet-Explorer

DSE Version 1.4.0

Versionshinweise - DSE Produkt-Entwicklung - Version 1.4

- Features
  - [DPE-614] - Zoomfunktion in Diagramm-Widgets (basechart) ermöglichen* [DPE-554] - Anlegen eines Vorfilters
  - [DPE-681] - Lokalisation über Stored Procedure
- Bug
  - [DPE-729] - Neues Logging bei Fehlern nutzlos
  - [DPE-737] - Frontend funktioniert nicht bei Pfaden in der URL
  - [DPE-721] - Auszählung der Population im JobFormular ist fehlerhaft

DSE Version 1.3.0

Versionshinweise - DSE Produkt-Entwicklung - Version 1.3

- Features
  - [DPE-553] - Vorfilter schränkt die Ausprägungen im FilterMenu ein
- Bug
  - [DPE-696] - Sachverhalt Fehlermeldung
  - [DPE-726] - Support: CM-Portal 1.2.7: Fehler beim Liquibase Update
  - [DPE-541] - Lokalisierung funktioniert nicht wie beschrieben in der Doku

DSE Version 1.2.8

Versionshinweise - DSE Produkt-Entwicklung - Version 1.2.8

- Bug
  - [DPE-728] - Job Manager Stunden und Minute wird auf 0 gesetzt

DSE Version 1.2.7

Versionshinweise - DSE Produkt-Entwicklung - Version 1.2.7

- Bug
  - Minor Bugfixes

DSE Version 1.2.6

Versionshinweise - DSE Produkt-Entwicklung - Version 1.2.6

- Bug
  - [DPE-612] - R-Agent: Verbindung über HTTPS/Apache Proxy funktioniert nicht [Bei Betrieb über ein Proxy muss die Proxy-Adresse in der Konfiguration angegeben werden, siehe Dokumentation]
  - [DPE-684] - Eigene Filter in der Filterverwaltung fehlen [Sporadisch auftretender Fehler wurde behoben]

DSE Version 1.2.5

Versionshinweise - DSE Produkt-Entwicklung - Version 1.2.5

- Bug
  - [DPE-690] - Verlinkung auf Skripte funktioniert nicht mehr korrekt [Zukünftige Einträge im CyclicEvualationLog enthalten nun die korrekte ID zur Skript-Version]
  - [DPE-689] - Jobs können nicht gelöscht werden [Fehlerursache behoben, Verbesserung der Systemstabilität und Logging erweitert]
  - [DPE-677] - Spalten mit NULL-Werten werden bei einer Sortierung zuerst angezeigt [Sortierung wie gewünscht angepasst, NULL-Werte werden immer nach unten sortiert]
  - [DPE-611] - Jobs: "Monatliche Ausführung" führt zu Fehler [Es kann nun ein Wartungsfenster in der Portalkonfiguration definiert werden. Die Eingabe zur monatlichen Ausführung von Jobs wurde durch die Definierbarkeit des gewünschten Wochentags verbessert]
  - [DPE-607] - Jobs: Jobnamen können nicht geändert werden [Berechtigungsfehler wurde behoben]
  - [DPE-606] - Sachverhaltsergebnisbilder werden nicht mehr im ImageViewer angezeigt [Konfigurationsfehler wurde behoben]
  - [DPE-592] - Spalte Script_ID in CM-PortalDB.portal.EvaluationInfo wird nicht mehr befüllt [Fehlerursache in Data Science-Skripten, wurde behoben]
- Features
  - [DPE-681] - Lokalisation über Stored Procedure [Wir haben die bestehende Lokalisierung erweitert. Inhalte können über Stored Procedures lokalisiert werden. Hierdurch können Übersetzungen mit eigener Logik eingebunden werden.]
  - [DPE-610] - Validierung von Cron-Patterns und Logging [Cron-Patterns der Jobs werden nun validiert und bei Fehlern wird ein entsprchender Eintrag im Log erzeugt]

DSE Version 1.2.4

Versionshinweise - DSE Produkt-Entwicklung - Version 1.2.4

- Bug
  - [DPE-607] - Jobnamen können nicht geändert werden* [DPE-606] - Sachverhaltsergebnisbilder werden nicht mehr im ImageViewer angezeigt

DSE Version 1.2.3

Versionshinweise - DSE Produkt-Entwicklung - Version 1.2.3

- Bug
  - [DPE-585] - Monatliches Schedule für Jobs deaktiviert, da z.Zt. fehlerhaft

DSE Version 1.2.2

Versionshinweise - DSE Produkt-Entwicklung - Version 1.2.2

- Bug
  - [DPE-590] - BugFix Timestamp-Migrationsskript für Null Fall

DSE Version 1.2.1

Versionshinweise - DSE Produkt-Entwicklung - Version 1.2.1

- Bug
  - [DPE-589] - Jenkins Release 1.2.0 fehlgeschlagen, Docker basierter dseconnect Test blockiert build

DSE Version 1.2.0

Es werden einige Einstellungen unter dem *raas* Tag in der *dse.conf.xml* ersetzt bzw. gestrichen:

Vorher:

```
<raas>
    <version>1.0</version>
    <vendor>eoda GmbH</vendor>
    <vendor_url>https://www.eoda.de</vendor_url>
    <debuguser>false</debuguser>
    <debuguserlogin>test@test</debuguserlogin>
    <debuguserpassword>YourPassword</debuguserpassword>
    <localdata>/var/raasimages/</localdata>
    <spconnectversion>45</spconnectversion>
    <spconnect>/opt/virgo/rpackges/spconnect_0.4.tar.gz</spconnect>
    <cron>true</cron>
</raas>
```

Es entfallen:

- debuguser
- debuguserlogin
- debuguserpassword
- cron

Stattdesse kommt hinzu:

```
<connectpackage>spconnect</connectpackage>
```

DSE Version 1.1.6

Versionshinweise - DSE Produkt-Entwicklung - Version 1.1.6

- Bug
  - [DPE-581] - Script wird nach 30 Zeichen gekürzt
  - [DPE-579] - Link zum Sachverhaltsformular aus JobEdior ist falsch

- [DPE-575] - Jobs mit exportResults laufen nicht durch
- [DPE-576] - Logging ist unübersichtlich geworden
- [DPE-569] - Internal Server Error" bei Sprache "Aktualisierien" im Frontend
- [DPE-577] - Im IssueFormular wird ein falsches Script ausgewählt
- [DPE-580] - Neues Script kann nicht ausgewählt werden

DSE Version 1.1.5

Versionshinweise - DSE Produkt-Entwicklung - Version 1.1.5

- Bug
  - [DPE-564] - Spaltenname in View portal.script createdat statt changedat

DSE Version 1.1.4

Versionshinweise - DSE Produkt-Entwicklung - Version 1.1.4

- Bug
  - [DPE-564] - Spaltenname in View portal.script createdat statt changedat

DSE Version 1.1.3

Versionshinweise - DSE Produkt-Entwicklung - Version 1.1.3

- Bug
  - [DPE-549] - LX03: Daten in Sichten werden nicht mehr geladen

DSE Version 1.1.2

Versionshinweise - DSE Produkt-Entwicklung - Version 1.1.2

- Bug
  - [DPE-541] - Lokalisierung funktioniert nicht wie beschrieben in der Doku

DSE Version 1.1.1

Versionshinweise - DSE Produkt-Entwicklung - Version 1.1.1

- Bug
  - [DPE-449] - .xlsx export in Sensorview funktioniert nicht
  - [DPE-487] - ORGINFCMPBL-830 - Core-Integration: Feedback beim manuellen Starten eines Jobs erforderlich
  - [DPE-532] - Spconnect zeigt bei Nutzung in RStudio die LDAP Credentials im Klartext
  - [DPE-502] - R Ausführung in v1.1.0 auf QA System funktioniert nicht

DSE Version 1.1.0

Versionshinweise - DSE Produkt-Entwicklung - Version 1.1.0

- Bug
  - [DPE-168] - Contextmenü in der Singlechoice
  - [DPE-282] - Spaltensortierung wenn NaN enthalten
  - [DPE-297] - Kontextmenü in der Sensorview befüllt
  - [DPE-316] - Dependency Viewer funktioniert wieder
  - [DPE-330] - Spalten Filter Tooltip bleibt stehen
  - [DPE-331] - Spalter Filter tooltip bei Single-Choice Widget zeigt "false" active
  - [DPE-335] - Es können nicht mehr als 1000 Punkte in ein Basechart angezeigt werden
  - [DPE-339] - '0' werte wurden nicht angezeigt im Timeseries plot mit groupBy
  - [DPE-433] - Deeplink-Funktionalität auf Sicht funktioniert wieder richtig
  - [DPE-456] - Job wird als nicht erfolgreich angezeigt, obwohl der letzte Durchlauf erfolgreich war
  - [DPE-457] - Analyseskript wird wieder mehr angezeigt
  - [DPE-458] - Skript wird gespeichert
  - [DPE-459] - Skript kann nicht verwendet werden
  - [DPE-460] - Job kann nicht manuell gestartet werden
  - [DPE-461] - Parameterbeschreibung wird nicht angezeigt
  - [DPE-462] - Jobformular / Sachverhaltsformular sieht manchmal komisch aus
  - [DPE-463] - Job-Scheduling funktioniert nicht richtig
  - [DPE-480] - Agent entkoppelt sich
  - [DPE-482] - Core-Integration: Alte DB-Tabellen stehen noch nicht vollständig als Views auf die neuen Core-Tabellen zur Verfügung
  - [DPE-483] - Core-Integration: Debugging-Möglichkeit der aktuellen Jobs steht nicht mehr zur Verfügung
  - [DPE-484] - Core-Integration: Logging hat sich komplett verändert
  - [DPE-485] - Core-Integration: Standardsichten in Sachverhalten passen nicht
- Story
  - [DPE-337] - Automatische Aktualisierung lokalisierter Inhalte bei Sprachenwechsel
  - [DPE-421] - Implementierung eines Metadaten-Typs (MDT) zur Übersetzung von DataID-Abfrage-Ergebnissen
- Aufgabe
  - [DPE-432] - dseconnect für stored procedures öffnen

DSE Version 1.0

Die Version 1.0 des eoda DSE stellt eine Integration von eoda Data Science Core und eoda Data Science Portal dar, das bedeutet, dass die ehemals getrennten Komponenten nun über eine gemeinsame Codebasis sowie über eine angeglichene, modularisierte Architektur verfügen.

Vorteile der Integration

Dieser Schritt von getrennten Produkten hin zu einer umfassenden Umgebung bietet mehrere Vorteile, darunter die folgenden:

- Erweitertes Logging

- Skalierbarkeit (Lastenverteilung auf Agenten)
- Erhöhte Stabilität
- Schnittstelle (API)
- Module können nach Bedarf angebunden oder deaktiviert werden
- Unterschiedliche Paketversionen durch verschiedene Agenten nutzbar
- Skriptsprachen Unterstützung (R, Python, Julia)
- Daten aus Datenbanktabellen können in Skripten über Schlüsselwörter referenziert werden

Versionshinweise - DSE Produkt-Entwicklung - Version 1.0.0

- Story* [DPE-148] - IssueFormular: ResultView und ResultViewJob einstellbar machen* [DPE-318] - Extrahieren einer JSON-Datei mit Portal- und Content-Übersetzung* [DPE-321] - Core-Integration: Developmentserver für Tests der Analyse-Skripte vorbereiten
- Aufgabe* [DPE-358] - Entwickler-Systeme auf dscRuntime umstellen* [DPE-382] - DPE-318, DPE-175 und alle Bugfixes sind in DSE 1.0 integriert

Anpassungsbedarf bei der Migration

Typ	Zusammenfassung	Anpassungsbedarf nach Upgrade
Agent	Agent installieren	
Konfiguration	Konfiguration anpassen	conf.yaml, dse.conf.xml
Konfiguration	Customsichten statt Defaultsichten	ConfigStore: Defaultsichten ggf. ersetzen durch aktuelle Customsichten
User	Stored Procedures anpassen	Für das Anlegen von Usern muss ggf. die Stored Procedure angepasst werden
Datenbanken	Rechte anpassen	Für den DB-User müssen gegebenenfalls die Berechtigungen angepasst werden
Datenbanken	DB Migration	

Technische Änderungen

Typ	Zusammenfassung
Architekturänderung	Analyse-Jobs werden nun in Projektordnern organisiert, die betreffenden Skripte wurden dazu in die entsprechenden Projekte überführt. (STATT: Jobs führen Projekte aus UND Skripte wurden in Projekte überführt)
Technische Änderung	Rechte und Rollen werden im Core verwaltet
Technische Änderung	Issuejobresults- und Jobresultssichten können im Issueformular definiert werden
Datenbankmigration	

Typ	Zusammenfassung
Datenbankmigration	Jobtabelle und Jobinstancestabellen aufgelöst
Testing	Implementierung umfangreicher automatisierter Tests

Breaking Changes

Kategorie	Zusammenfassung
Authentifizierung	Standard ist Datenbank, LDAP muss aktiviert bzw. konfiguriert werden
Job-Handling	Die "eine Instanz pro Job"-Policy greift auch bei manuell gestarteten Jobs.
Issueformular	Im Issueformular gibt es jetzt zwei neue Felder: "Issue Result View" und "Job Result View". Beim Erstellen eines Issues werden diese Standardmäßig auf die in der ConfigStore Tabelle unter den keys *viewConfiguration.issueResult* bzw. *viewConfiguration.jobResult* hinterlegten Werten gesetzt. Sind die keys nicht vorhanden, werden die Werte *#/common/issueresult* bzw. *#/common/jobresult* als initial Werte genutzt.
R-Connectivität	Das R-Paket "spconnect" wurde in "dseconnect" umbenannt und muss auf dem R-Server (Agenten) neu installiert werden. Um eine Abwärtskompatibilität zu gewähren, kann die Verwendung des Packages konfiguriert werden.

DSE Version 0.53.2

Typ	Zusammenfassung
Bug	Scrollbalken bei leerem ImageViewer und Einzelbildansicht
Bug	Bei Fehlern liefert das Log nur noch die Fehlermeldung und keine weiteren Informationen
Story	Der Quellcode wird nun mit jedem Release in der DMZ ausgeliefert
Story	Extrahieren einer JSON-Datei mit Portal- und Content-Übersetzung

DSE Version 0.53

Typ	Zusammenfassung
Bug	Ausführungszeiten von Jobs stimmen nicht mit den definierten Scheduling-Zeiten überein
Bug	ng-click wird bei Klick auf Link in Tabellen zusätzlich ausgeführt
Story	Core-Integration: Skripte sollen Ergebnisse ins Portal zurückschreiben
Story	Usability: Vereinheitlichung der Anzeige von UserNamen

DSE Version 0.52

Typ	Zusammenfassung
Bug	Warnfeld zeigt "undefind" statt der Equi-Nummer
Bug	Parameter deren Value=0 ist werden nicht an die URL übergeben
Bug	ToolTipp bleibt im Edge stehen
Bug	"Tabelle durchsuchen" im IE nicht schließbar
Bug	Portal im IE nur mit geöffneter Konsole erreichbar
Bug	Dynamische Links im HTML-Widget funktionieren nicht
Bug	Gesourcte Childknoten in gesourcten Knoten werden nicht in die DB geschrieben
Story	Möglichkeit zur Aktualisierung bestehender Sprachen schaffen
Story	Seitentitel: Zeilenumbrüche in Titelleiste entfernen
Story	Portal: Ort der Konfigurationsdatei parametriesierbar machen

DSE Version 0.51

Zusammenfassung
Bug wurde beseitigt, durch den die Kommentarfunktion bei Sachverhaltsstatusübergang fehlte.
Die Dokumentation zum Bereich "DSP neu starten wurde aktualisiert.
Default-Werte werden bei doppelten tkeys nicht mehr überschrieben.
Der Link von sp.conf.xml funktioniert wieder.
RPM 0.49.3 ruft wieder korrekt die sp.conf.xml auf.
Die Links zur PDF-Doku im Seitenheader sind wieder vorhanden und aufrufbar.

Zusammenfassung
Eine Datenabfrage wurde überarbeitet, wodurch nun mit RefTag deployter Content weiterhin jederzeit im Portal-Frontend aufrufbar ist.
Das "Spaltenfilter ist aktiv"-Symbol wurde wieder in die Caption aufgenommen.
Referenzen innerhalb von dseconnect auf com.sun.jna wurden entfernt.
Der Core-Python-Agent-Integrationstest wurde vereinfacht.
Der "Tabelle durchsuchen"-Dialog im Tabellen-Widget wird zukünftig innerhalb der Captionleiste angezeigt.
Die bedingte Formatierung von Zellen wurde überarbeitet und erweitert, so dass soe nun wieder in allen Fällen funktioniert.
Für die Darstellung von Dezimalzahlen wird nun die Einstellung des Client-Systems verwendet.
Icons im Button des DateFilter werden wieder korrekt angezeigt.
Das Logging zwischen R und dem Portal wurde überarbeitet, so dass in der CyclicEvaluationLog nun sicher Start-/Stop-Einträge und Fehler-Meldungen abgelegt werden.

DSE Version 0.50

Zusammenfassung
Ein Bug wurde beseitigt, durch den Python-Skripte mit leeren Zeilen nicht einheitlich behandelt wurden.
Ein Fehler wurde in raas-utils beseitigt: Serialisierung von Insert - und Update-Builder funktionierte nicht.
Das Konzept zur Übersetzung von Content wurde umgesetzt, so dass Content nun lokalisierbar ist.
Es wurde eine API für den Import der Übersetzungs-JSONs geschaffen.
Es ist nun möglich, über DSEdep eine JSON-Datei mit den Übersetzungskeys und den zugehörigen Arbeitstexten zu erzeugen.
Ab sofort ist ein Zugang zum Portal auch ohne AD-Anbindung möglich durch setzen eines Parameters in der Portalkonfiguration.
Ein Bug wurde beseitigt, durch den im Scheduler ein laufender bzw. hängender Job alle anderen Jobs blockieren konnte.
Bedingtes formatieren von Zellen funktioniert nun wieder in allen Fällen korrekt.
Es fand eine Umbenennung von Komponenten statt: spDep -→ DSEdep, spConnect -→ DSEconnect, sp.conf.xml -→ dse.conf.xml. Durch Wrapper wurde eine Abwärtskompatibilität geschaffen, so dass die sp-Varianten noch nutzbar sind.
Ein Problem bei der Eingabe von Spaltenfiltern wurde beseitigt, bei dem der Fokus der Input-Fields teils verloren gehen konnte.
Das Datenbank-Passwort kann in der sp.conf.xml / dse.conf.xml nun verschlüsselt werden.
Es wurde eine View für die User-Tabelle eingerichtet.
Es ist nun möglich, über das Portal im core eincheckten R-Skripte auszuführen.