SMARTUNIFIER User Manual

Release 1.8.0

Amorph Systems GmbH

October 11, 2023

TABLE OF CONTENTS

Integrate perfectly your
Production-IT using
SMARTUNIFIER
ADVANCED IT-INTEGRATION PLATFORM
AMORPH.pro
SMARTUNIFIER

CHAPTER

ONE

ABOUT SMARTUNIFIER

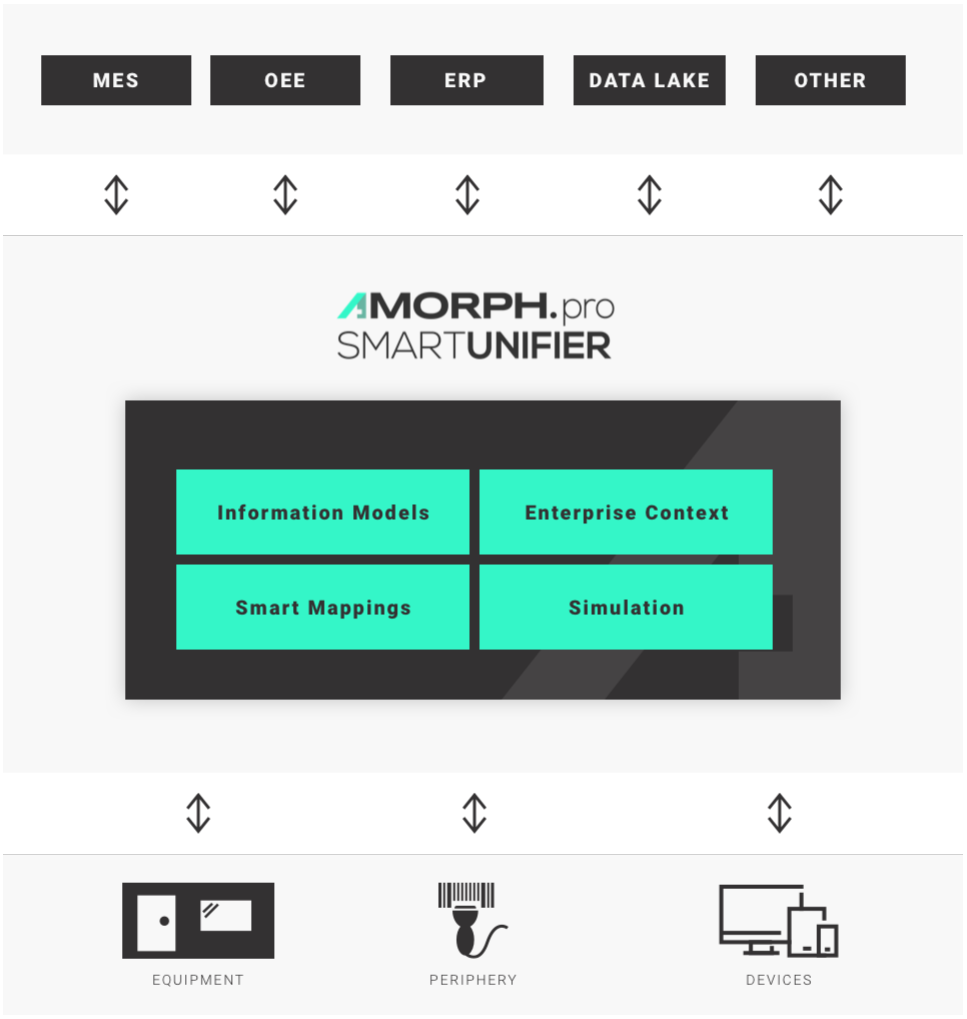
You are new to SMART**UNIFIER**?

- Learn about the *SMARTUNIFIER* connectivity platform
- Learn about the *connectivity use cases* you can address with SMART**UNIFIER**
- Check out the supported *connectivity endpoints and data formats*
- *Get started* with your integration

1.1 What is SMARTUNIFIER

SMART**UNIFIER** represents a powerful but very easy to use decentralized industrial connectivity platform for interconnecting all industrial devices and IT systems including equipment, peripheral devices, sensors/actors, MES, ERP as well as cloud-based IT systems.

SMART**UNIFIER** is the tool of choice for transforming data into real value and for providing seamless IT interconnectivity within production facilities.



1.2 What does SMARTUNIFIER do

- SMART**UNIFIER** provides an easy way to collect data from any *Data Source* and is able to transmit this data to any *Data Target*.
- Data Sources and Data Targets (commonly referred to as Communication Partners) in this respect may be any piece of equipment, device or IT system, communicating typically via cable or Wi-Fi and using a specific protocol like e.g., OPC-UA, file-based, database, message bus.
- With SMART**UNIFIER** several Communication Partners can be connected simultaneously.
- With SMART**UNIFIER** it is possible to communicate unidirectional or bidirectional to each Communication Partner. i.e., messages and events can be sent and received at the same time.
- SMART**UNIFIER** can translate and transform data to any format and protocol that is required by a certain Data Target. This includes different pre-configured protocols and formats, e.g., OPC-UA, file-based, database, message bus, web services and many direct PLC connections. In case a certain protocol or format is currently not available it can be easily added to

SMART**UNIFIER**.

- By applying so called *Information Models*, SMART**UNIFIER** enables the same view to data regardless of the protocol or format being used to physically connect an equipment, device or IT system.
- A big advantage of SMART**UNIFIER** is, that in many cases there is no need for coding when providing interfaces between different Communication Partners – providing a new interface is just drag and drop of data objects between data source(s) and destination(s).

1.3 Important Use Cases with SMARTUNIFIER

SMART**UNIFIER** enables an easy and very efficient realization of many use cases that are crucial for gaining Industry 4.0 Excellence.

In the following subchapters some of the most important SMART**UNIFIER** Use Cases are described. These give a comprehensive overview of the advanced SMART**UNIFIER** Features.

1.3.1 Anything-To-Anywhere IT Interface

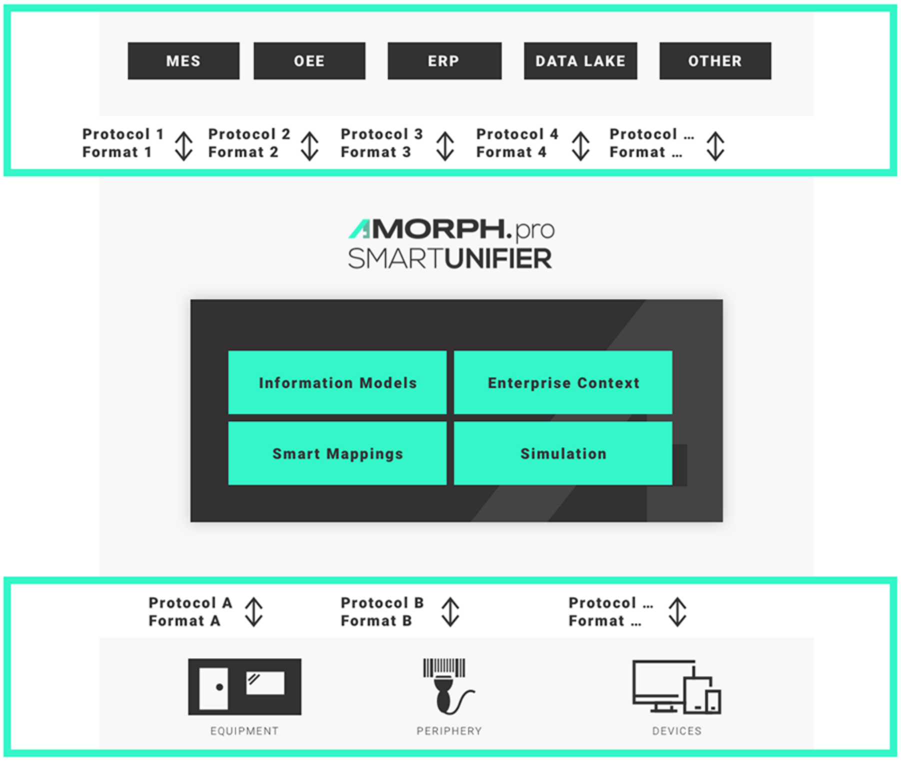
Easy, fast and flexible bi-directional interconnection of multiple IT systems and equipment within a production facility.

Interconnecting heterogeneous shop floor equipment and devices with IT systems and interconnecting different IT systems with each other is a central requirement for a successful transition to modern Industry 4.0 IT landscapes.

SMART**UNIFIER** offers the unique capability to easily interconnect equipment and devices by allowing

- any number of parallel high-speed *Communication Channels* between equipment, devices and IT systems
- high-speed translation between different communication protocols and formats by applying configurable and reusable *Information Models* and *Smart Mappings*
- flexible integration of equipment periphery
- easy integration of enterprise-specific information (e.g., equipment -location/-name/-type/-capabilities) via configurable Enterprise Context
- riskless simulation of interfaces and communication scenarios

Results from renowned reference customers have shown that average equipment integration efforts and **cost can be reduced by up to 90%** using the SMART**UNIFIER** and its advanced technologies to perform powerful IT integration by configuration instead of tedious interface programming.



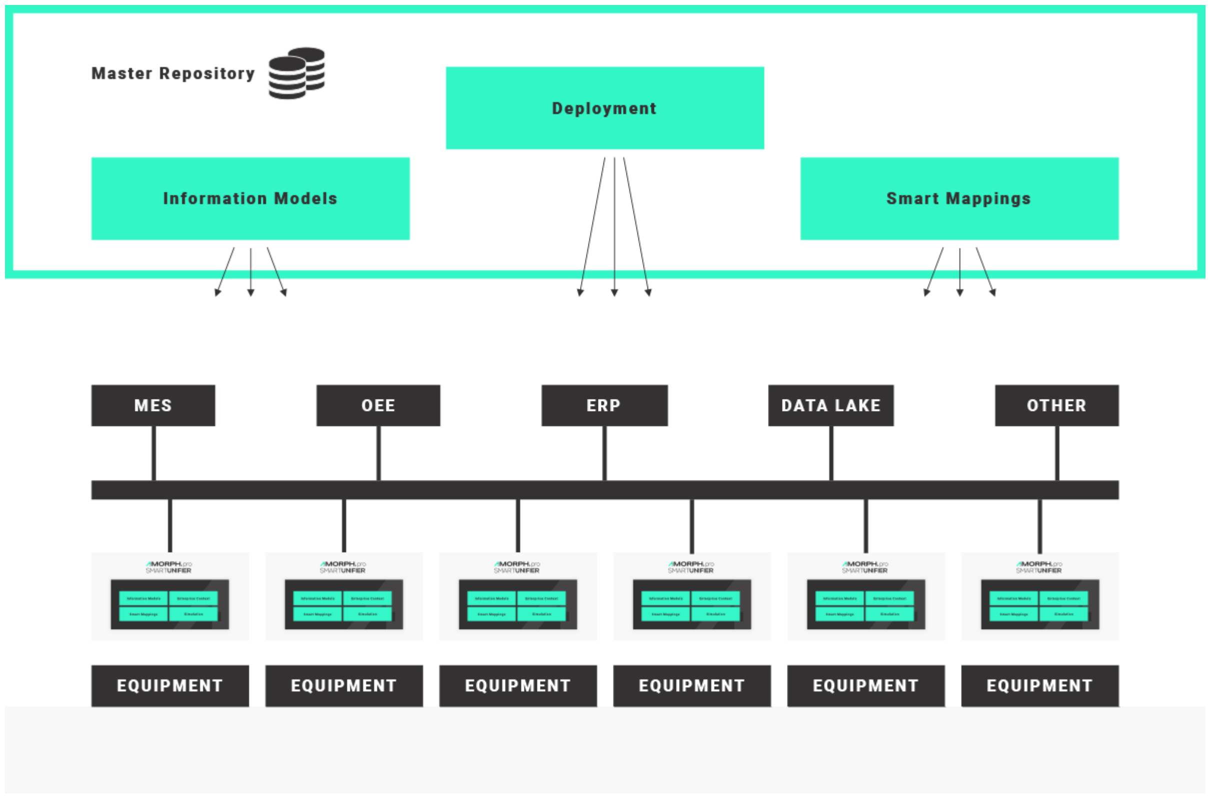
1.3.2 Reusable Interfaces and Interface Models

Reuse interface configurations multiple times with minimum effort.

When running an IT network with a higher number of installed SMART**UNIFIER** *Instances*, all previously created interface configurations (Information Models and Smart Mappings) can be reused easily and shared across the whole installation. This way similar equipment types are integrated using the same connection and translation logic.

Changes and updates of interface configurations can be deployed from a centrally accessible Master Repository, eliminating the need to touch and update each equipment or device individually

Summarized, SMART**UNIFIER** allows a highly comfortable and effective management of very small to very large IT communication environments, creating minimum overhead and letting you reach your main goal: Excellent Manufacturing with a full Industry 4.0 IT infrastructure.



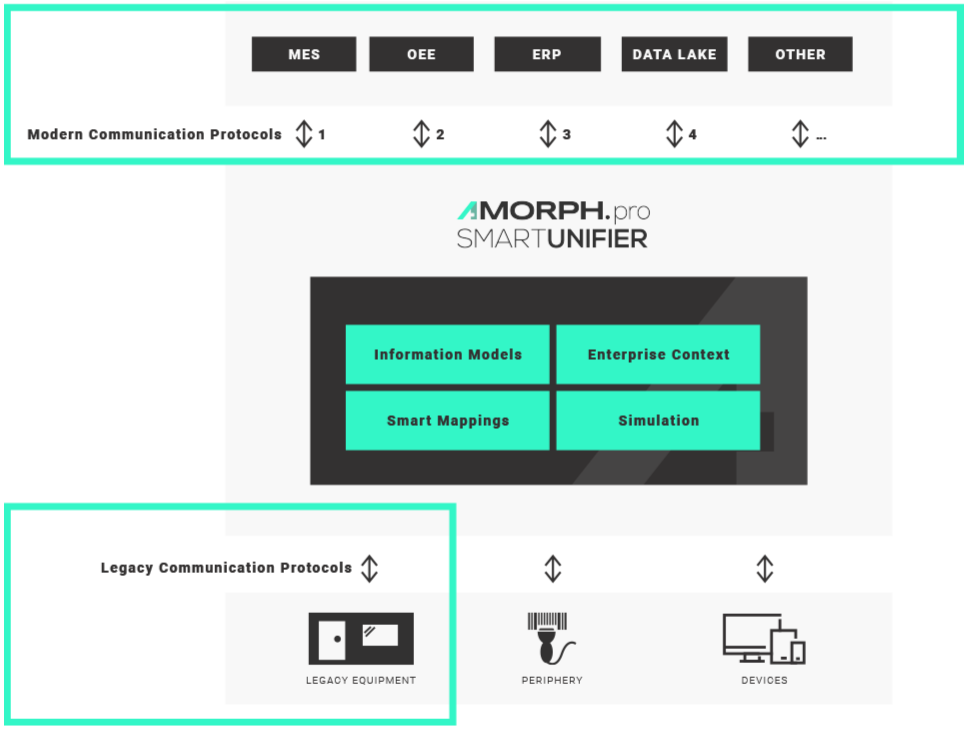
1.3.3 Integrate Legacy Equipment

Fast adaptation of legacy communication protocols and formats to modern enterprise standards.

By applying SMART**UNIFIER** configurable protocol translation (Smart Mappings), modern communication standards like OPC-UA or XML over message bus are fully supported.

SMART**UNIFIER** allows a smooth migration from existing communication protocols and formats (e.g., between existing equipment and MES) to new Industry 4.0 standards.

This unique capability of SMART**UNIFIER** is realized by simply using existing communication channels simultaneously with newly introduced channels. When finishing the migration, the old channels can be switched off without any risk.

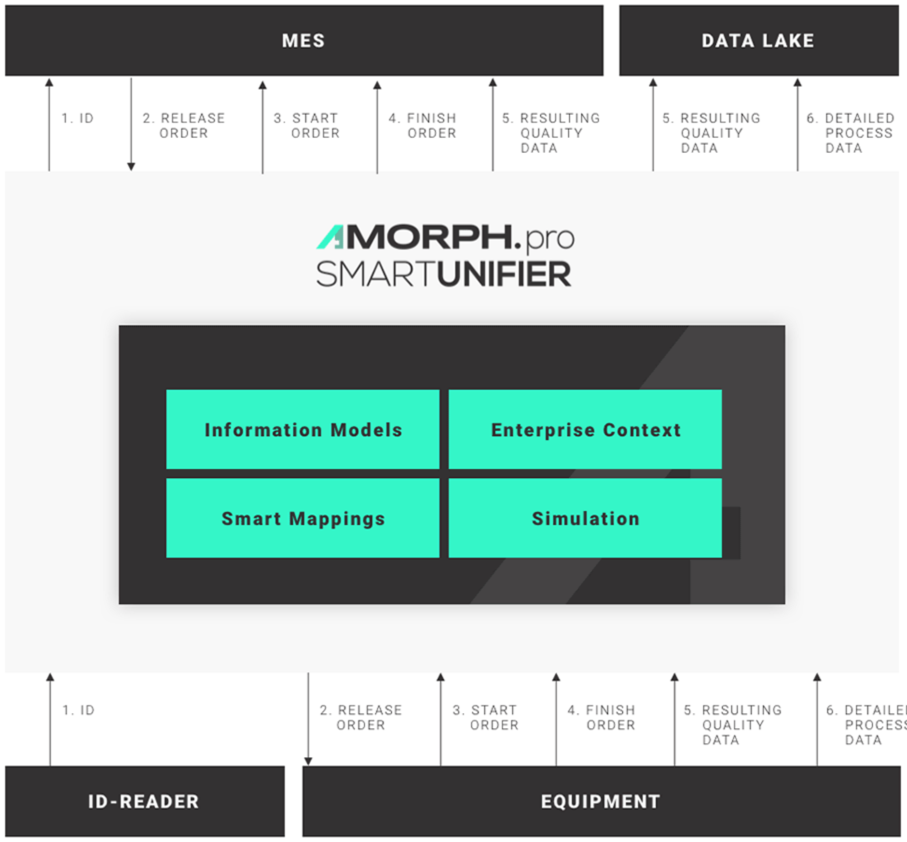


1.3.4 Implement Fab Communication Scenario

Easily implement complete fab communication sequences that cover multiple steps.

With SMART**UNIFIER** it is not only possible to give access to simple equipment or device data and to provide „some data to MES and Cloud", but also with SMART**UNIFIER** complete communication scenarios between equipment to upper-level IT systems can be easily implemented.

The communication scenarios can cover all steps from identification, validation, order start as well as sending results and process data from equipment to MES or Cloud. Of course, it is also possible to provide any parameter data (recipes) from MES or SCADA to equipment.



1.3.5 Provide Base for Remote Maintenance and Health Monitoring

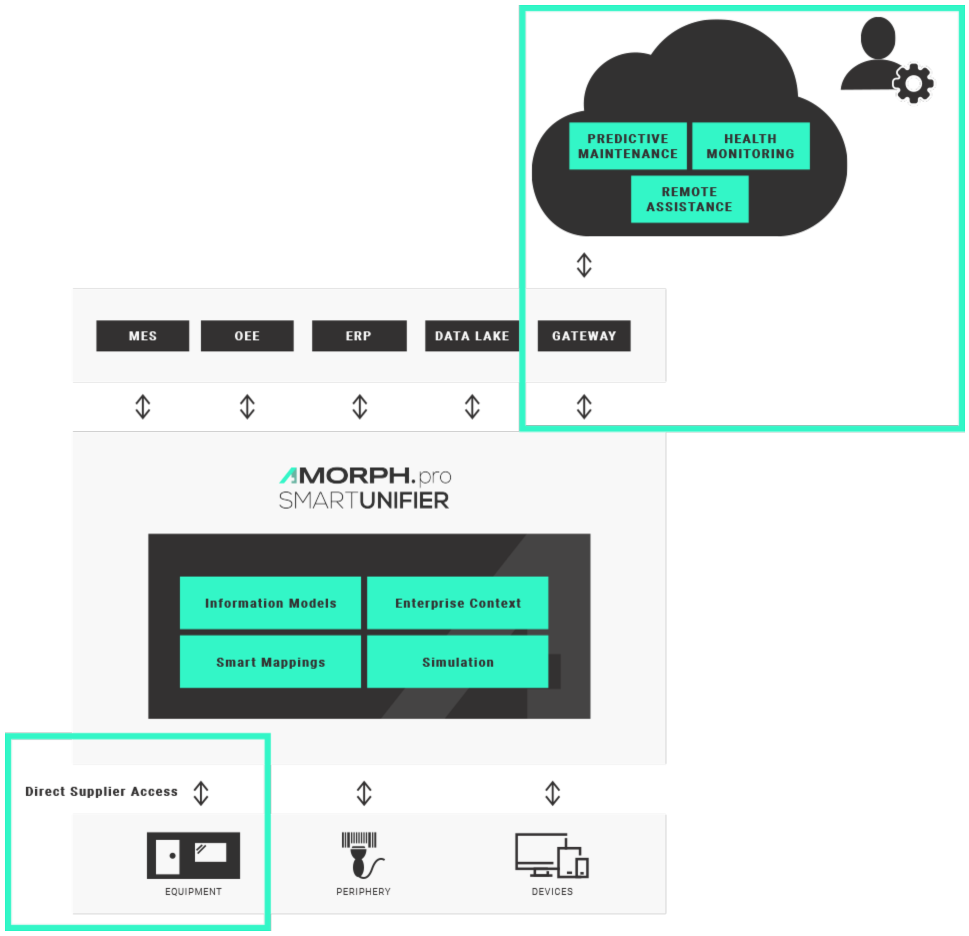
Establish new services and business models by giving secured multi-channel access to equipment and device data in real-time.

Production equipment can be integrated with SMART**UNIFIER** to provide direct access for equipment suppliers or maintenance service providers to relevant equipment data (e.g., equipment status, equipment key parameters) via an equipment supplier's cloud infrastructure.

This way, new innovative business models for equipment suppliers are supported by building the base for "Production as a Service" offerings and remote predictive maintenance.

Also, further advanced business use cases with SMART**UNIFIER** are possible, e.i., by implementing real-time equipment monitoring capabilities in a cloud environment.

Another SMART**UNIFIER** use case is to give Remote Assistance to equipment suppliers to achieve production optimization and to ensure the most efficient usage of equipment resources for customers.



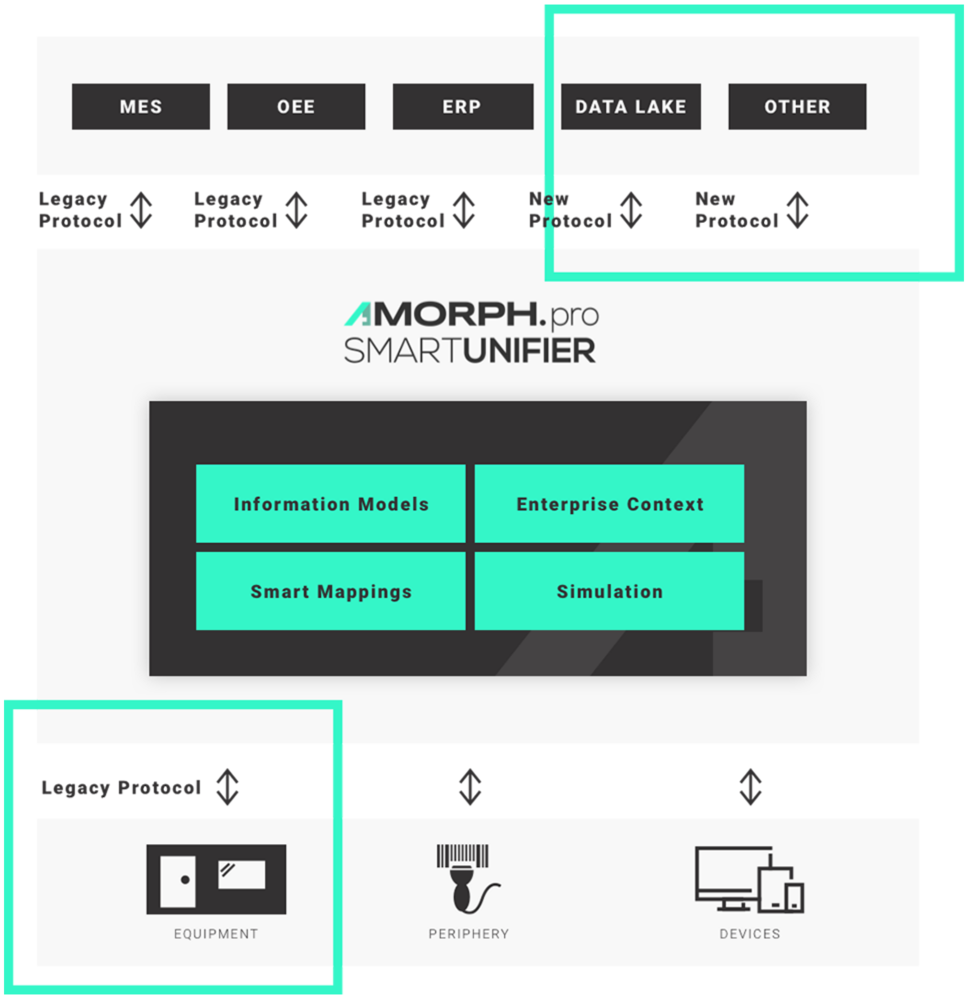
1.3.6 Migrate to Industry 4.0

Migrate step by step to modern communication standards and apply enterprise-wide semantics to data.

A key feature of SMART**UNIFIER** is to open an easy way to integrate new IT systems using modern communication protocols. This is realized by simply adding additional communication channels to the existing legacy channels.

Another feature of SMART**UNIFIER** in this respect is, that all existing IT systems with their legacy protocols and formats can still be operated in parallel with the newly established IT systems (e.g., Data Lake, Advanced Analytics, Cloud).

This way, it is possible to step by step introduce modern communication standards and incrementally migrate to a state-of-the-art Industry 4.0 IT architecture, but still keep the existing IT infrastructure fully operable.



1.3.7 Allow Unlimited Scalability

Rely on unlimited scalability from single equipment and devices to whole facilities.

SMART**UNIFIER** is the first industrial connectivity platform that allows nearly unlimited virtually scalability in terms of number of connected equipment and devices. The SMART**UNIFIER** platform can be applied for integrating one single equipment or device, but with SMART**UNIFIER** hundreds or even thousands of equipment and devices within whole facilities can be integrated to upper-level systems or into the Cloud.

This is because SMART**UNIFIER** is not a traditional middleware having a central limiting message bus. Nor does SMART**UNIFIER** contain any central performance and latency limiting database for providing its communication features.

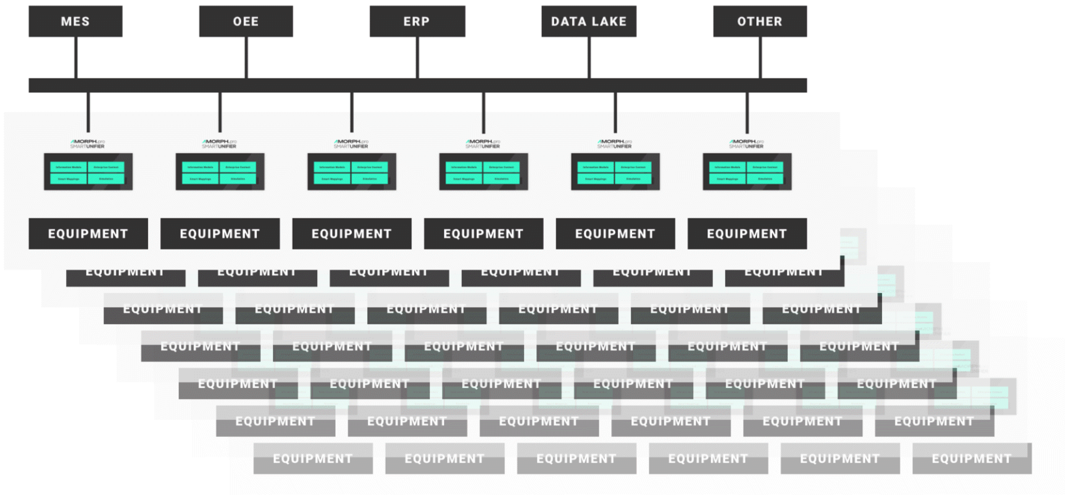
SMART**UNIFIER** works as a distributed environment. Using advanced technologies of distributed computing is the key for enormous scalability.

In a large installation a high number of SMART**UNIFIER** Instances, each with low software footprint, provide the required communication capabilities. These single instances can be deployed to any location within an enterprise IT network – on a server, on an equipment PC, within the Cloud.

Nevertheless, the configuration of all SMART**UNIFIER** Instances can be managed centrally:

- central configuration of Information Models and Smart Mappings
- central Operations Monitoring of installed SMART**UNIFIER** Instances.

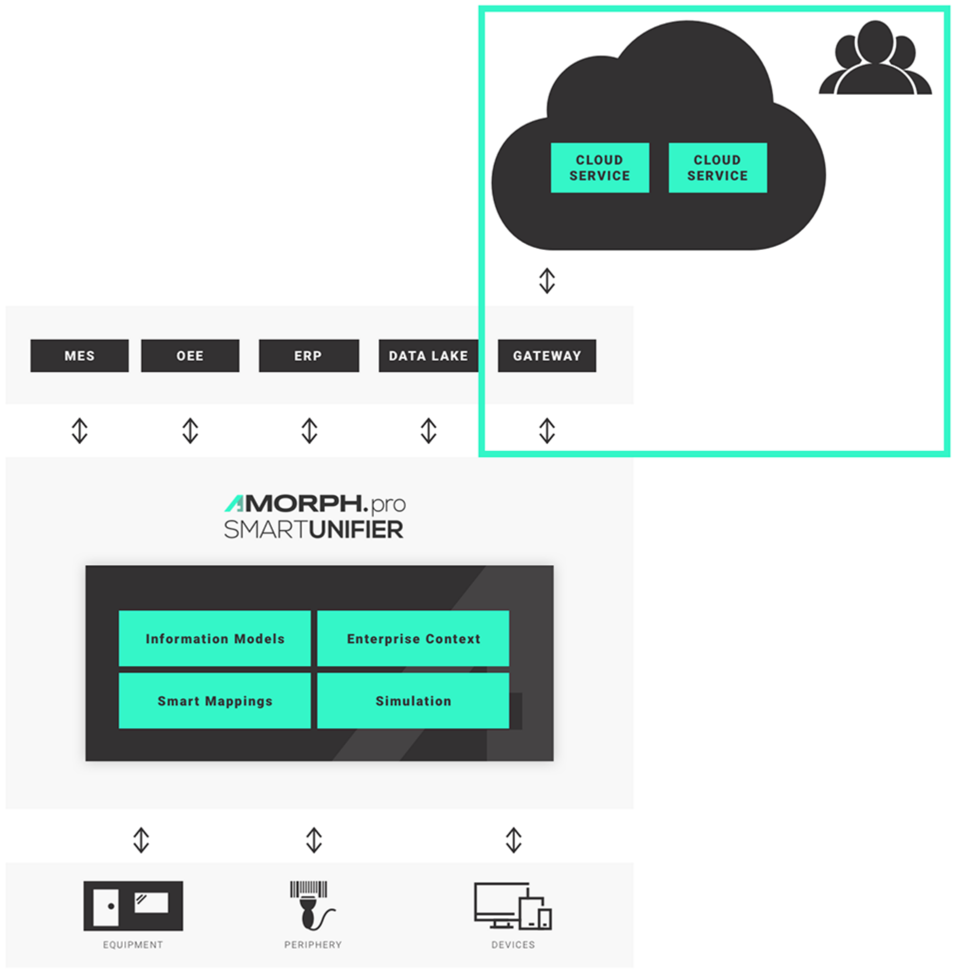
Thus, SMART**UNIFIER** is an essential piece of Industry 4.0 for any manufacturing enterprise – allowing fab-wide and enterprise-wide management of production communication and IT integration infrastructure.



1.3.8 Enable Internet of Things

Out-of-the-box connections between equipment, devices and other IT systems to Cloud infrastructures.

By acting as a translator between equipment and any IOT device precise and secured access of data consumers is possible. The easy connection to any Cloud based infrastructure is also possible (e.g., AWS, Azure).



1.4 Connectivity Endpoints and Data Formats

SMART**UNIFIER** provides comprehensive connectivity support for a variety of equipment, devices and IT systems. This includes many different pre-configured communication protocols and formats. e.g., OPC-UA, file-based, database, message bus, Webservices and direct PLC connections. Preconfigured interfaces are available also for many standard software applications. A number of these connectivity endpoints / communication protocols require a first time customization from Amorph Systems for a specific customer connectivity use case. Please contact Amorph Systems for detailed information.

1.4.1 Connectivity Endpoints / Communication Protocols

The following connectivity endpoints / communication protocols are supported by SMART**UNIFIER**.

Table 1: Connectivity Endpoints

Format	Description
ADLink OpenSplice	Connectivity to ADLink OpenSplice middleware via Data Distribution Service (DDS)
AMQP	Interface to AMQP Message Broker via Active MQ
AODB	Interface to various Airport Operational Database (AODB) Systems that support standard communications via e.g., HTTP, REST, SQL
Apache Active MQ	Interface to Active MQ Message Broker
AWS Elastic Container Service (ECS)	Interface to applications running in AWS ECS
AWS Elastic Compute Cloud (EC2)	Interface to applications running in AWS EC2
AWS IoT	Interface to AWS IoT
AWS IoT Greengrass	Interface to AWS IoT Greengrass via MQTT
AWS IoT Sitewise	Direct interface to AWS IoT SiteWise or via OPC-UA
AWS CloudWatch	Interface to CloudWatch
AWS DynamoDB	Interface to AWS DynamoDB
AWS S3	Interface to AWS S3
AWS SNS	Interface to AWS Simple Notification Service (SNS)
AWS SES	Interface to AWS Simple Email Service (SES)
Barcode Reader	Connectivity to any TCP/IP based barcode reader (or other identification system)
Beckhoff	Interface to Beckhoff PLC via Beckhoff OPC-UA Server
CNC	Connectivity to various CNC controllers (e.g., ABB, Fanuc, Heidenhain, Heller, Sinumerik, Traub, W&T Wiesemann & Theis)
DDS	Connectivity to Data Distribution Service (DDS)
EUROMAP	Connectivity of injection moulding machines via files
File	Read and Write files from arbitrary directories using File Consumer / File Tailer
FIWARE	Interface to FIWARE IoT
FTP	Upload and Download files to/from FTP servers
Flink	Interface to Apache Flink to enable real-time streaming
HTTP	Send request to HTTP servers
HTTPS	Send request to HTTPS servers
InfluxDB	Interface to InfluxDB
InfluxDB v2.0	Interface to InfluxDB v2.0
IBM MQ	Interface to IBM MQ Message Broker
In-Memory	Communication via local machine
ISO-on-TCP	(RFC1006) Connectivity of S7 automation devices with any communication partner
JDBC	Access databases through SQL and JDBC (refer to SQL Databases)

continues on next page

Table 1 – continued from previous page

Format	Description
JMS	Send and receive messages to/from a JMS Queue or Topic using plain JMS
Kafka	Interface to Apache Kafka to enable real-time streaming
MES	Interface to a Manufacturing Execution System (MES) that support standard communications via e.g., HTTP, REST, SQL
Mendix (REST)	Interface to Mendix via REST
Mendix (MQTT)	Interface to Mendix via MQTT
Mendix (Kafka)	Interface to Mendix via Kafka
Modbus-TCP	Communication via Modbus TCP Server / TCP Client
Microsoft Azure (IoT Hub)	Interface to Microsoft Azure Iot Hub via MQTT
MTConnect	Communication Interface to MTConnect compliant agent applications
MQTT	Connectivity by implementing MQTT Client
NoSQL Databases	Cassandra, MongoDB, Hbase
OEE	Interface to various Overall Equipment Efficiency (OEE) Applications that support standard communications via e.g., HTTP, REST, SQL
OPC-UA Client	Connectivity by deploying one or multiple OPC-UA Client instances per SMART**UNIFIER** Communication Instances
OPC-UA Server	Connectivity by deploying one or multiple OPC-UA Server instances per SMART**UNIFIER** Communication Instances
PLC	Connectivity to various PLCs (e.g., Allen-Bradley, B&R, FANUC, General Electric (GE), Hilscher, Honeywell, Krauss Maffei, Mitsubishi, Toshiba, Wago) via TCP/IP
PM	Interface to a various Predictive Maintenance Systems that support standard communications via e.g., HTTP, REST, SQL
POP/IMAP	Receiving emails from a mail server
REST	Communication via REST using REST Server / REST Client (Webservices)
SAP MII	Interface to SAP MII
SAP RFC	Interface to SAP via remote function call (RFC)
SAP Netweaver	Interface to SAP Netweaver via HTTP
SCADA	Interface to various SCADA Systems that support standard communications via e.g., HTTP, REST, SQL
SECS/GEM	Communication with semiconductor or photovoltaic equipment using SECS/GEM interface protocol for equipment-to-host data communications (TCP/IP).
Siemens Industrial Edge	Deployment of SMART**UNIFIER** Communication Instances via Siemens Industrial Edge Platform
Siemens MindSphere (REST)	Interface to MindSphere via REST
Siemens MindSphere (MQTT)	Interface to MindSphere via MQTT
Siemens S7 PLC/TCP	Interface to Siemens S7 1500 / 1200 / 400 / 300 via TCP protocol

continues on next page

Table 1 – continued from previous page

Format	Description
Siemens S7 PLC/OPC-UA	Interface to Siemens S7 1500 / 1200 via OPC-UA protocol
Smart Devices	Interface to various Smart Devices (e.g., Smart Phones, Tablets) that support standard communications via e.g., HTTP, REST, SQL
SMTP	Sending emails from SMTP servers
SOAP	Communication via SOAP (Webservices)
Splunk	Interface to Splunk via HTTP Event Collector
Splunk	Interface to Splunk via Metrics Interface
SQL Databases	Interface to any SQL-based database like e.g., DB2, HSQLDB, MariaDB, MSSQL, OracleDB, PostgreSQL, SQLServer and others
SFTP	Upload and Download files to/from SFTP servers
TCP	Communication from/to any (binary) TCP based protocol
UDP	Communication from/to any (binary) UDP based protocol
VANTIQ	Interface to VANTIQ
VIPA Speed 7	Interface to VIPA Speed 7 PLC
WAGO PLC/IP	Connectivity to WAGO PLCs via OPC-UA
Websocket	Interface to Websocket Server (TCP/IP)

Note: In case a customer requires to connect to other endpoints (e.g., computing devices, PLCs) not listed in the table, please contact Amorph Systems.

1.4.2 Data Formats

The following data formats can be used in conjunction with the above defined connectivity endpoints. The possible formats for a certain connectivity endpoint may be restricted based on the selected communication protocol. For detailed information please contact Amorph Systems.

Table 2: Data Formats

Format	Description
Binary	Handling of any binary communication format (e.g., fixed/variable lengths fields, headers/footers)
CSV	Handle CSV (Comma separated values) payloads
JSON	Encode and decode JSON formats
TEXT	Handling of any text-based communication format
XML	Encode and decode XML formats

Note: In case a customer requires another data format not listed in the table, please contact Amorph Systems.

1.5 Getting Started

Integration of industrial equipment, periphery and devices with IT systems can become a quite complex task. SMART**UNIFIER** delivers a standard way of implementing such integration scenarios. We recommend following the procedures described below.

Step	Action (Phases)	Description
1	Preparation	Collecting the requirements of systems that are going to be integrated
		o Find out what protocols are in use
		o Identify the data structures
		o Identify the overall communication scenarios (creation of sequence diagrams might be useful)
		Testing and validating the communication of the systems that are going to be integrated e.g.,
		o Testing basic connections to the systems (e.g., using MQTT Explorer for testing the connection to an MQTT Broker)
		Define a plan on how to conduct the testing before the rollout
2	Deployment of Manager	The number of SMARTUNIFIER Manager installations depends on the scope of the integration. If several plants are involved, it is recommended to have one installation per plant. Note: Reusable configuration components such as Information Models can be shared across multiple SMARTUNIFIER Manager installations (Backup and Restore)
3	Configuration of the Communication Instance	For each Communication Instance:
		Create Information Models based on the data structure of the system that is going to be integrated
		Create and configure the Communication Channels that are needed to connect to the systems
		Create the Mapping between the Information Models of the systems based on the previously defined communication scenario
		Create the Device Type that acts as a template for the Communication Instance
		Create and configure the Communication Instance
4	Deployment of the Communication Instance	For each Communication Instance:
		Plan the deployment of the Communication Instance
		o Determine the proper deployment type for the Communication Instance (on-premises, edge, or cloud deployment)
		o Deploy the Communication Instance accordingly
5	Testing	Test the communication according to the previously defined test plan
6	Rollout	Rollout and scaling

CHAPTER

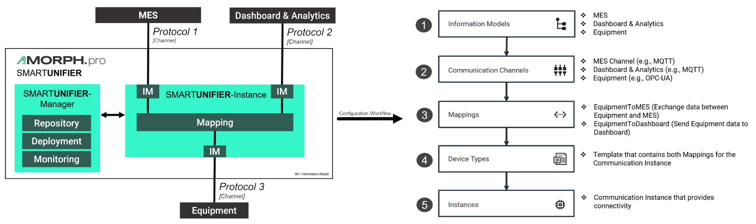
TWO

HOW TO INTEGRATE WITH SMARTUNIFIER

Each integration scenario follows the same workflow, which consists out of 5 steps:

1. *Information Models* - describe and visualize communication related data using hierarchical tree structures
2. *Communication Channels* - describe and configure the protocols needed for the scenario
3. *Mappings* - define when and how to exchange/transform data between Information Models
4. *Device Types* - define templates for Instances
5. *Instances* - define applications that provide the connectivity

Below you can see an example of integration scenario and the necessary steps to establish connectivity with SMART**UNIFIER**:



2.1 Information Models

2.1.1 What are Information Models

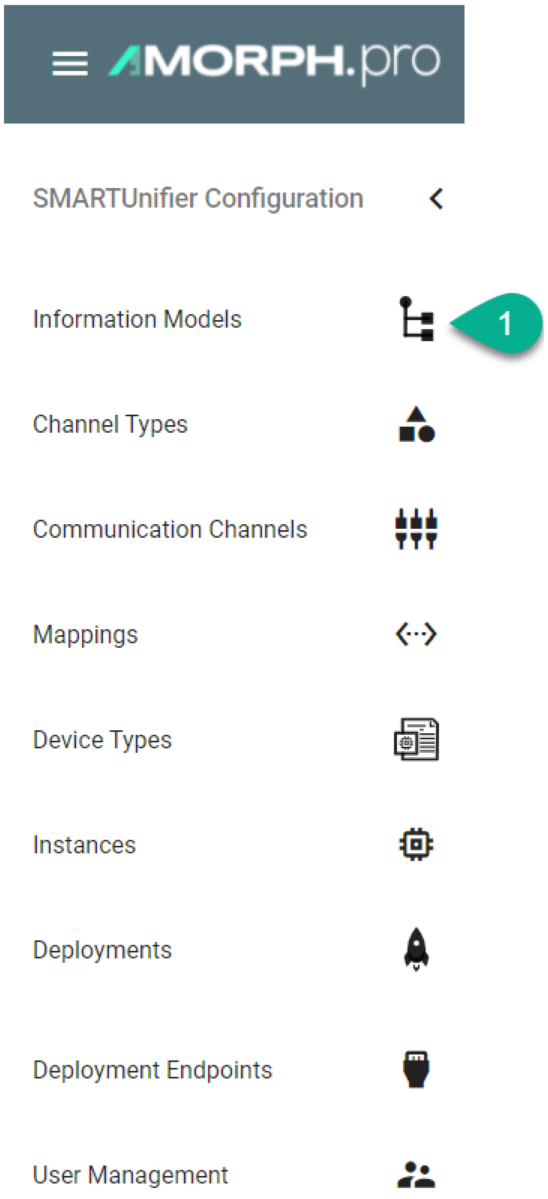
Within the SMART**UNIFIER** an Information Model describes the communication related data that is available for a device or IT system. One device or one IT system therefore is represented by one Information Model. An Information Model consists of so-called *Node Types*. Information Models are built up in a hierarchical tree structure, i.e., elements within the Information Model can contain further elements. This is required to model the data structure of devices as naturally as possible.

Before starting to model the data structure of e.g., an equipment or IT-System the overall use case should be known. Typically, each equipment or IT-system that is going to be integrated gets its own Information Model. If there are equipments and IT-systems of the same type, you can use one common Information Model.

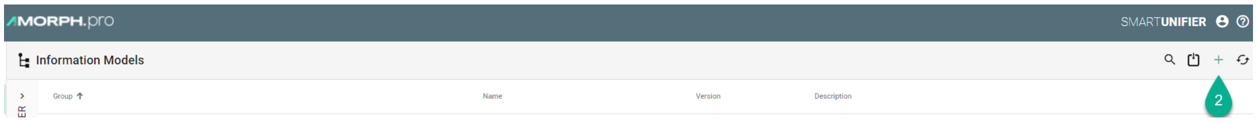
2.1.2 How to create a new Information Model

Follow the steps described below to create an Information Model:

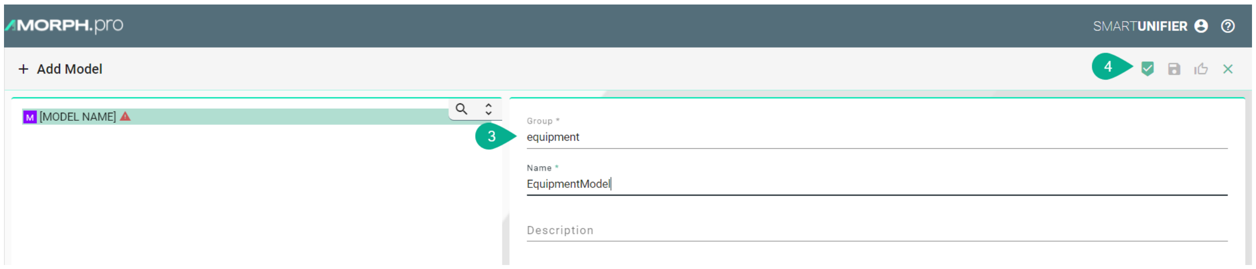
- Select the SMART**UNIFIER** Information Model Perspective **(1)**.



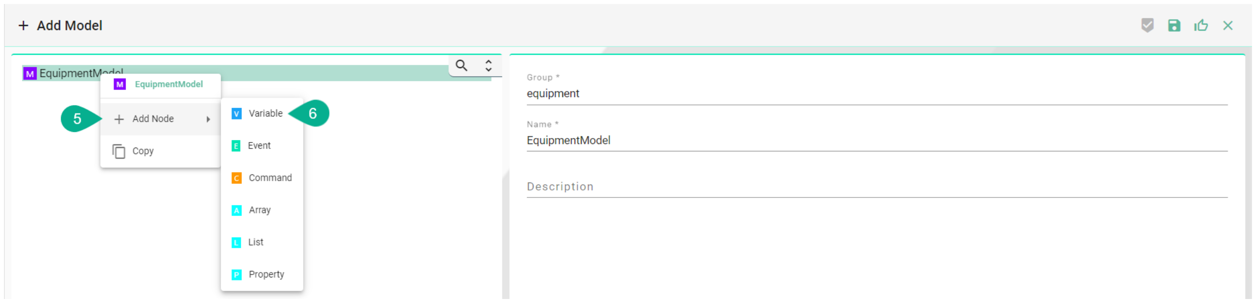
- You are presented with the following screen containing a list view of existing Information Models.
- In order to add a new Information Model, select the "Add Model" button at the top right corner **(2)**.



- On the following screen provide the following mandatory information: Group and Name **(3)**.
- The "Apply" button at the top right corner is enabled after all mandatory fields are filled in. Click the button to generate a new Information Model **(4)**.
- The newly created Information Model is now visible as a node on the left side of the screen.



- After the root model node is created, a new Information Model can be built up using definition types.
- Perform a right click on the root model node and select "Add Node" **(5)**. Select a Definition Type from the dialog **(6)**.



2.1.3 Node Types

Depending on the use case and the Communication Channels you need to choose the proper Node Type. Model node types are elements within an Information Model, like variables, properties, events, commands and also collections such as arrays and lists. Each model node type has a Data Type that defines whether the model node type is a predefined data (e.g.: String, Int, Boolean, etc.) type or a custom data type.

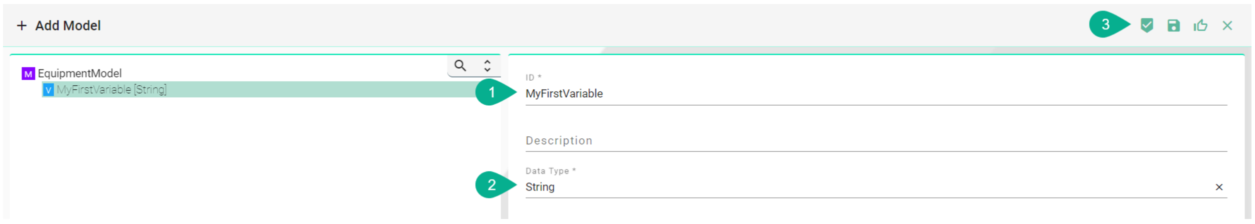
Variables

What are Variables

Variables are used to represent values. Within SMART**UNIFIER** different types of Variables are defined. They differ in the kind of data that they represent and whether they contain other Variables. For example, a file Object may be defined that contains a stream of bytes. The stream of bytes may be defined as a Data Variable that is an array of bytes. Properties may be used to expose the creation time and owner of the file Object.

How to create a Variable

- Enter an ID **(1)**
- Enter a Data Type **(2)**
- Click the "Apply" button **(3)**



ID *
MyFirstVariable

Description

Data Type *
String

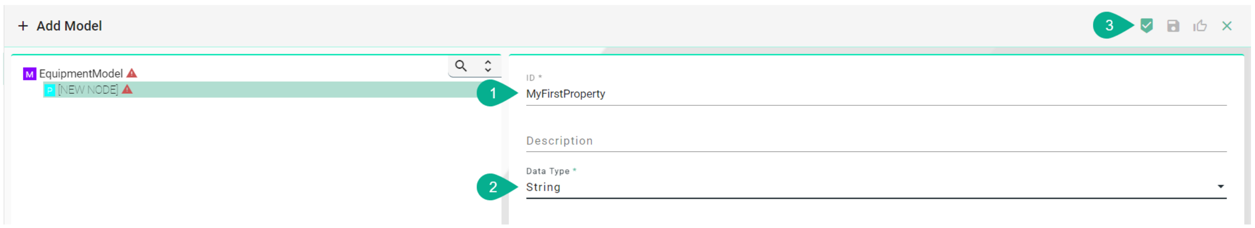
Properties

What are Properties

Properties are working similar to *Variables*. Properties can be used for XML attributes when XML-files are subject to be processed by SMART**UNIFIER**, although XML elements are still represented by Variables in the *Information Model*.

How to create a Property

- Enter an ID **(1)**
- Enter a Data Type **(2)**
- Click the "Apply" button **(3)**



ID *
MyFirstProperty

Description

Data Type *
String

Events

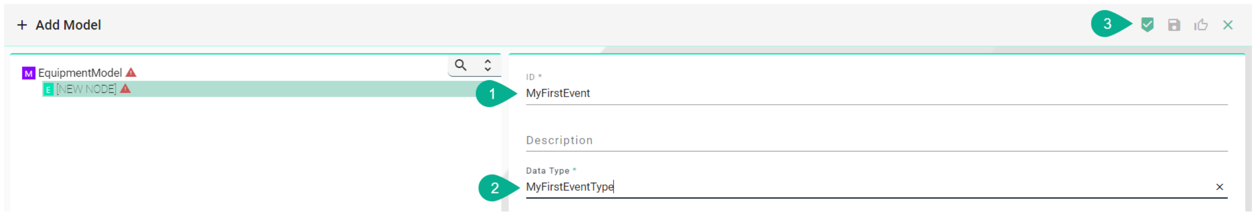
What are Events

SMART**UNIFIER** is an event-driven software. In this context an event is an action or occurrence recognized by SMART**UNIFIER**, often originating asynchronously from an external *data source* (e.g., equipment, device), that may be handled by the SMART**UNIFIER**. Computer events can be generated or *triggered* by external IT systems (e.g., received via a *Communication Channel*), by the SMART**UNIFIER** itself (e.g., timer event) or in other ways (e.g., time triggered event). Typically, events are handled asynchronously with the program flow. The SMART**UNIFIER** software can also trigger its own set of events into the event loop, e.g., to communicate the completion of a task. Each event defined in an *Information Model* has an event type.

An event type consists of one or multiple simple or structured variables. Clients subscribe to such events to receive notifications of event occurrences.

How to create an Event

- Enter an ID **(1)**
- Enter a Data Type for the Event. e.g., "MyFirstEventType" **(2)**
- Click the "Apply" button **(3)**



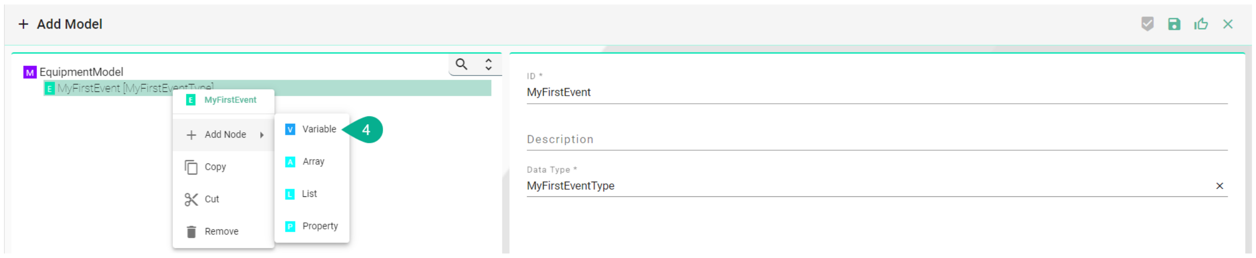
ID *
MyFirstEvent

Description

Data Type *
MyFirstEventType

Within the Event *Variables*, *Arrays* or *Lists* can be added. Follow the steps below to add a Variable:

- Right click the Event node, select "Add Node" and choose a Definition Type **(4)**

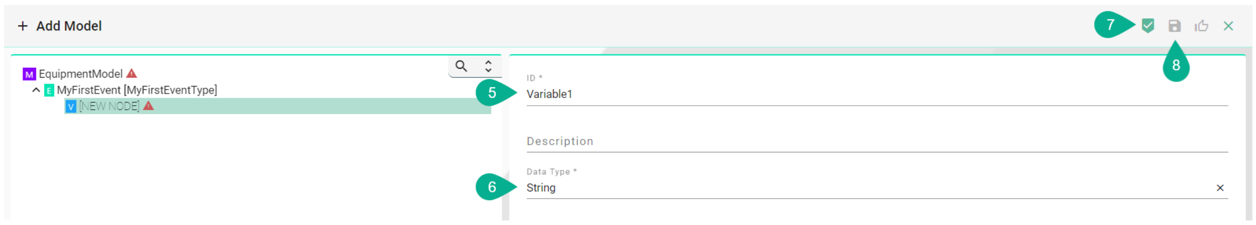


ID *
MyFirstEvent

Description

Data Type *
MyFirstEventType

- Enter an ID **(5)**
- Enter a Data Type **(6)**
- Click the apply button **(7)**
- Click the "Save" button at the top right corner **(8)** to save the Information Model



Commands

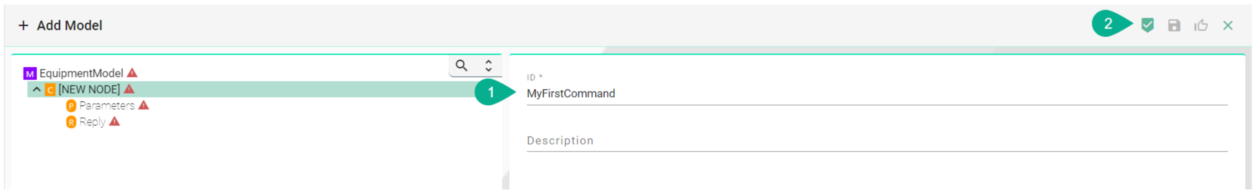
What are Commands

Commands are functions, whose scope is bound by an owning *Information Model*, like the methods of a class in object-oriented programming. Commands within an Information Model are typically invoked by an external IT system (e.g., an equipment) that triggers the command. In addition, commands of a target Information Model (e.g., an MES) can be triggered by the SMART**UNIFIER** through a *Mapping*. A command contains one or multiple simple or structured *Variables*. Also a command has a return parameter that likewise can be a simple or complex *data type*.

The lifetime of the command invocation instance begins when the client calls the command and ends when the result is returned. While commands may affect the state of the owning model, they have no explicit state of their own. In this sense, they are stateless. Each command defined in an Information Model has a command type

How to create a Command

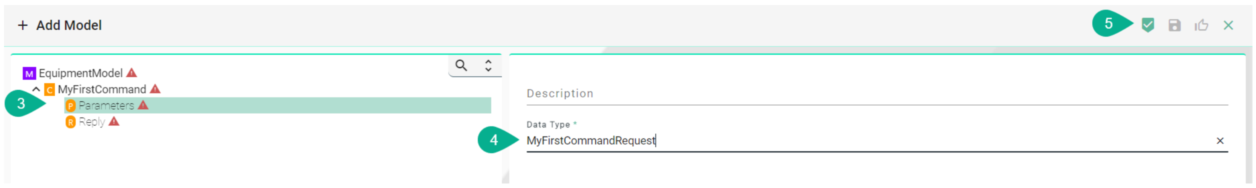
- Enter an ID **(1)**
- Click the "Apply" button **(2)**



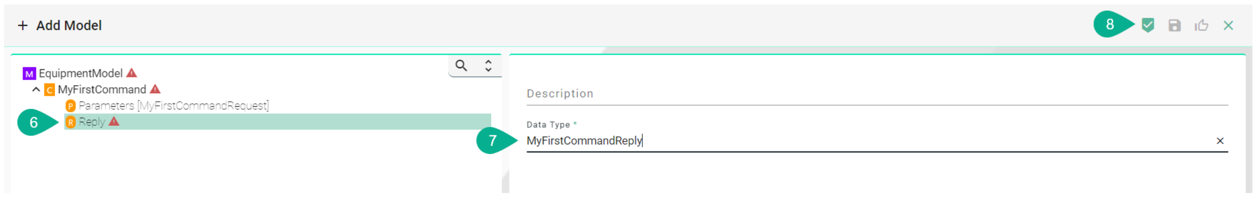
The main two parts of a Command are the Request, referred to as Parameters within the SMART**UNIFIER**, and the Reply. *Variables*, *Arrays* and *Lists* can be added to both of these command parts.

Follow the steps below to add a Variable to Parameters:

- Select the Parameters node from the tree **(3)**
- Enter a Data Type **(4)**
- Click the "Apply" button **(5)**

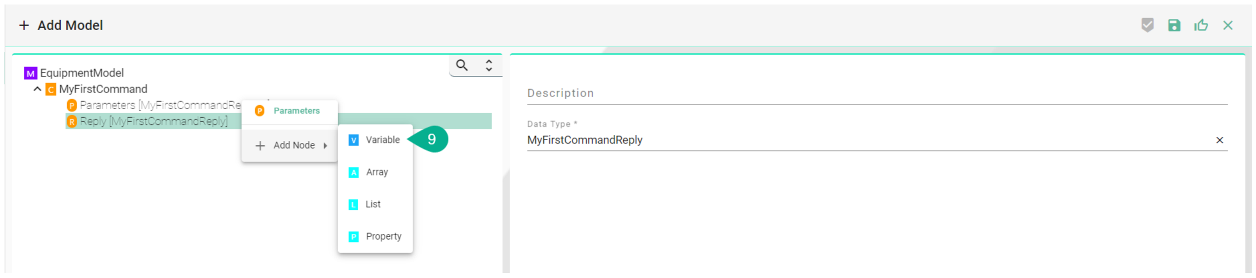


- Select the Reply node from the tree **(6)**
- Enter a Data Type **(7)**
- Click the "Apply" button **(8)**

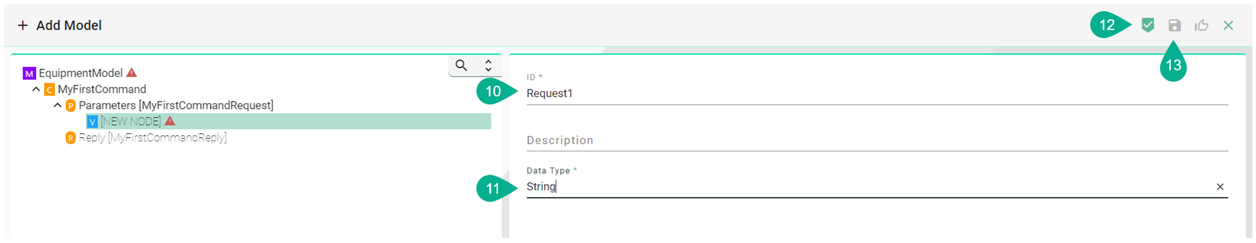


Follow the steps below to add nodes under the Parameter and Reply node:

- Right click the Parameter node, select "Add Node" and choose a Definition Type **(9)**



- Enter an ID **(10)**
- Enter a Data Type **(11)**
- Click the "Apply" button **(12)**
- Click the "Save" button **(13)** to save the Information Model



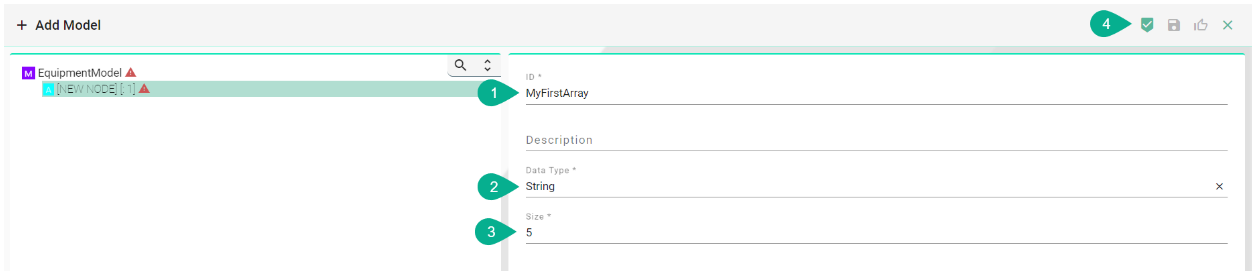
Arrays

What are Arrays

Arrays allow to hold a fixed size collection of elements, which have all the same data type. The size of the array must be defined in the configuration of the Information Model.

How to create an Array

- Enter an ID **(1)**
- Select a Data Type for the Array by clicking the Data Type Drop-Down **(2)**
- Enter the size of the Array **(3)**
- Click the "Apply" button **(4)**



ID *
MyFirstArray

Description

Data Type *
String

Size *
5

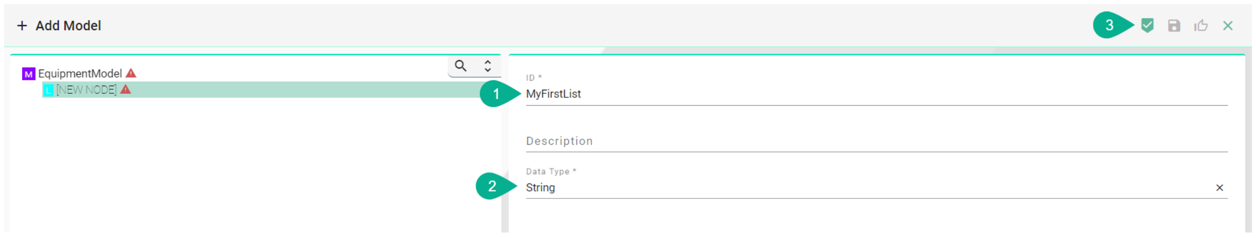
Lists

What are Lists

Lists allow to hold a collection of elements (*Variables*), which can each have different data types.

How to create a List

- Enter an ID **(1)**
- Enter a Data Type for the List. E.g., "String" **(2)**
- Click the "Apply" button **(3)**



ID *
MyFirstList

Description

Data Type *
String

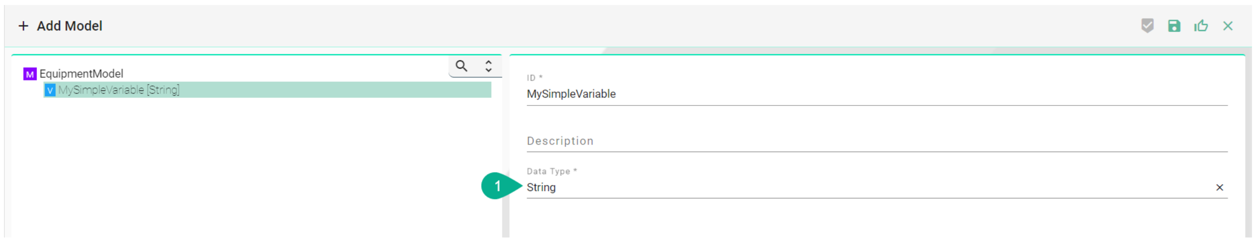
2.1.4 Data Types

There are two kinds of Data Types:

- Predefined Types e.g., String, Integer, Boolean and more. (**Note**: Only available for the definition types - Variables, Properties, Arrays, Lists)
- Custom Types

How to create a Variable as a Simple Type

- Add a new Variable, enter an ID and select a primary data type for the Data Type e.g., "String" **(1)**



ID *
MySimpleVariable

Description

Data Type *
String

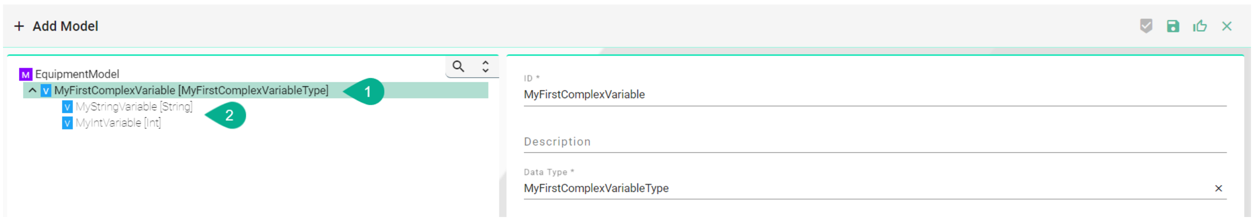
Table 1: Predefined Data Types

Type	Definition
Boolean	true or false
Byte	8 bit signed value (-27 to 27-1)
Int	32 bit signed value (-231 to 231-1)
String	Sequence of characters
Char	16 bit unsigned Unicode character (0 to 216-1)
Double	64 bit IEEE 754 double-precision float
Float	32 bit IEEE 754 single-precision float
Long	64 bit signed value (-263 to 263-1)
Short	16-bit signed integer
Array	Mutable, indexed collections of values.
List	Class for immutable linked lists representing ordered collections of elements.
LocalDate	Immutable date-time object that represents a date, often viewed as year-month-day.
LocalDate-Time	Immutable date-time object that represents a date-time, often viewed as year-month-day-hour-minute-second.
LocalTime	Immutable date-time object that represents a time, often viewed as hour-minute-second.
OffsetDate-Time	Immutable representation of a date-time with an offset.

How to create a Variable as a Custom Type

- Add a new Variable, enter an ID and enter a custom name for the Data Type e.g., “MyFirstComplexVariableType” **(1)**
- Select the Custom Variable - “MyFirstComplexVariableType” - and add a new Variable underneath it **(2)**

Note: Model *Node Types* with custom data types can be easily duplicated throughout the Information Model by selecting the same custom data type for a new model node type.


+ Add Model
EquipmentModel
MyFirstComplexVariable [MyFirstComplexVariableType]
MyStringVariable [String]
MyIntVariable [Int]
1
2

ID *
MyFirstComplexVariable

Description

Data Type *
MyFirstComplexVariableType

Data Types for Properties, Arrays and Lists can be defined as shown above for Variables.

2.1.5 Information Model Structure

The structure of an Information Model depends upon the used Communication Channel in the integraton. Communication Channels can be categorized into data-driven and event-driven communication.

Event-driven refers to an integration triggered by an event that takes place in a system e.g., goods receipt. This event triggers a Rule with in the Mapping of SMART**UNIFIER**.

Data-driven, on the other hand, is spurred by a change in the actual data in a system.

SMART**UNIFIER** also considers command-driven integration were an event takes place in a system and immediatley expects an reply from another system. This is in contrast to the event-driven integration were no reply is expected.

The table below provides some use case examples with its required Information Model structure.

Use Cases	Description	Communication Channel	Information Model Structure
Data driven	Retrieving data from OPC-UA Server	OPC-UA Client	Simple Data Type Variables under the Root node
			Custom Data Type Variables under the Root node
Event driven	Posting data on MQTT Broker	MQTT client	Event under the Root node
			· Simple Data Type Variables under the Event node
			· Custom Data Type Variables under the Event node
			· Lists under the Event node
Command driven	Executing Select request on a database with parameters	SQL Database	Command under the Root node
			· Simple Data Type Variables under the Command Parameters
			· Simple Data Type Variables under the Command Reply

What Information Model structure is required for each communication channel is described in the chapter *Communication Channels*.

Data structures can be imported by using extensions which is especially convenient when dealing with complex structures with a lot of variables: - OpcUa Model Import: Data structures can be imported form a OPC UA server - JSON Model Import: JSON structures can be imported directly

2.2 Communication Channels

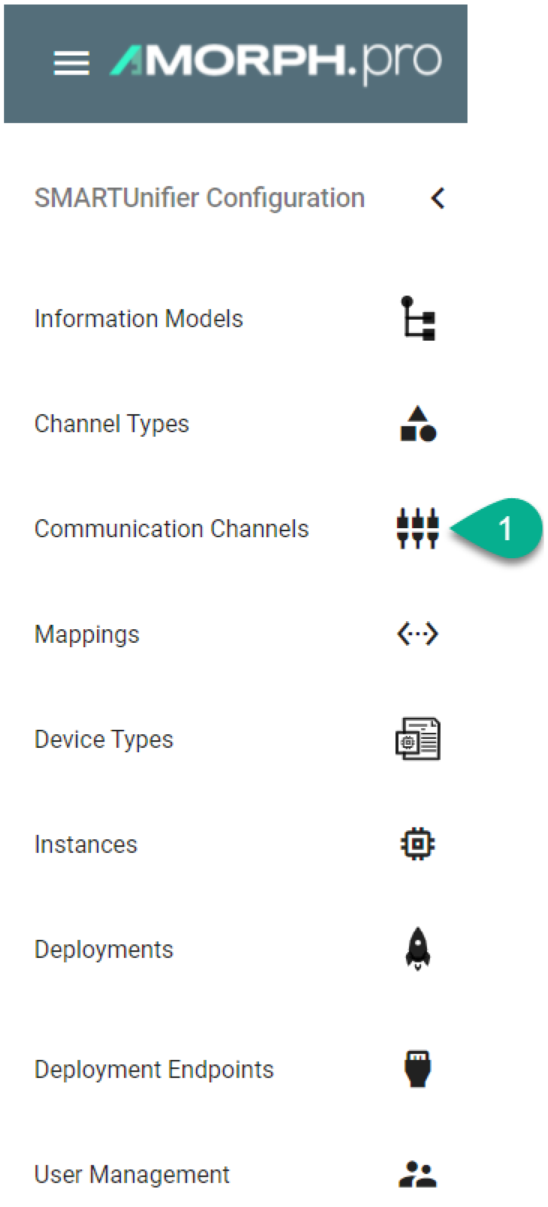
2.2.1 What are Channels

Communication Channel or simply Channel refers to a transmission medium. A Channel is used to convey information from one or several senders (or transmitters). Communicating data from one location to another requires a pathway or medium. These pathways are called Communication Channels, and the information is transmitted with the help of communication protocols. Each *Information Model* can have one Channel or many, and each model can choose which Channels it subscribes to. The information is transmitted through the Communication Channels in both directions: from the external system to the SMART**UNIFIER** application and vice versa.

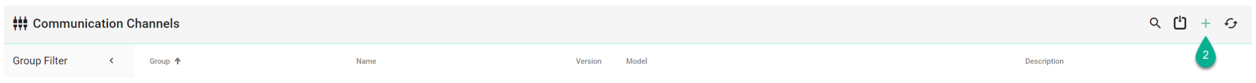
2.2.2 How to create a new Channel

Follow the steps below to create a new Channel:

- Go to the Communication Channels perspective by clicking the “Communication Channels” button **(1)**

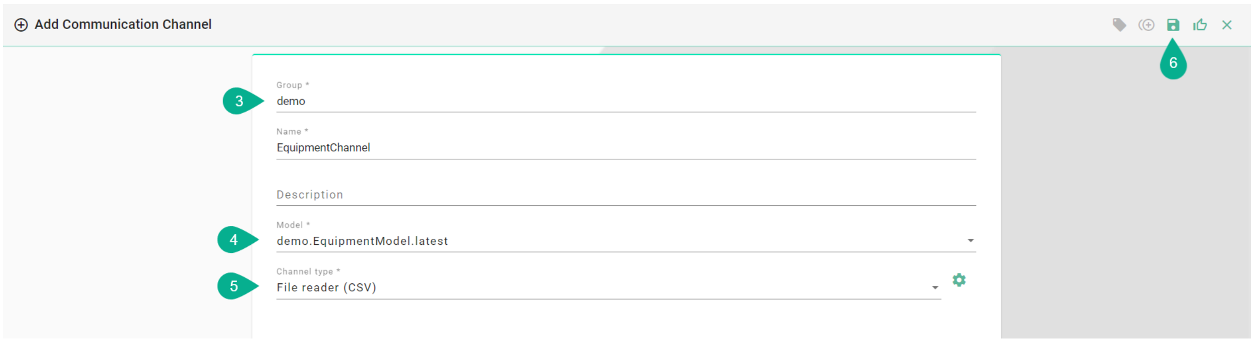


- To create a new Channel, select the “Add Channel” button at the top right corner **(2)**



- The creation of a Communication Channel is split up into two parts. First enter basic information about the new Communication Channel

  - Fill in the information for the Channel identifier such as: Group, Name and Version. Description is optional **(3)**
  - Besides that, associate the Channel with an Information Model **(4)**
  - Select the type this Channel represents from the Drop-Down **(5)**. A list of available Channel Types and a description of how to configure each of them can be found below
- Click the "Save" button **(6)** to save the Channel



Group *
demo
Name *
EquipmentChannel
Description
Model *
demo.EquipmentModel.latest
Channel type *
File reader (CSV)

2.2.3 Channel Types and Configuration

There are several Channel Types available with SMART**UNIFIER**. The supported Communication Channel Types are listed in the chapter *Connectivity Endpoints / Communication Protocols*. If a specific Communicating Channel Type is not available in this product version, please contact Amorph Systems. In many cases the provision of a specific Communication Channel Type can be provided as extension to the standard product.

The configuration of the Communication Channels can be done on Channel, *Device Type* and *Instance* level.

Note: It's important to note that the configuration of a Channel can be overwritten as needed. For example, the configuration made in the Communication Channel view can be changed in the Instance view.

The following paragraphs lay out the configuration process of selected Channel Types. If the Channel Type you want to use is not described, please contact Amorph Systems for configuration guidance.

File-based

File Reader

Characteristics

- File Reader monitors a specified folder - the so-called input folder
- If a file is inserted the following actions take place:
  - The *Trigger* of the specified *Rule* in the Mapping is activated
  - Thus, the Rule is executed
- After successful execution of the rule the file is moved into a so-called output folder
- In case of an exception the file is moved into an error folder

Supported File Formats:

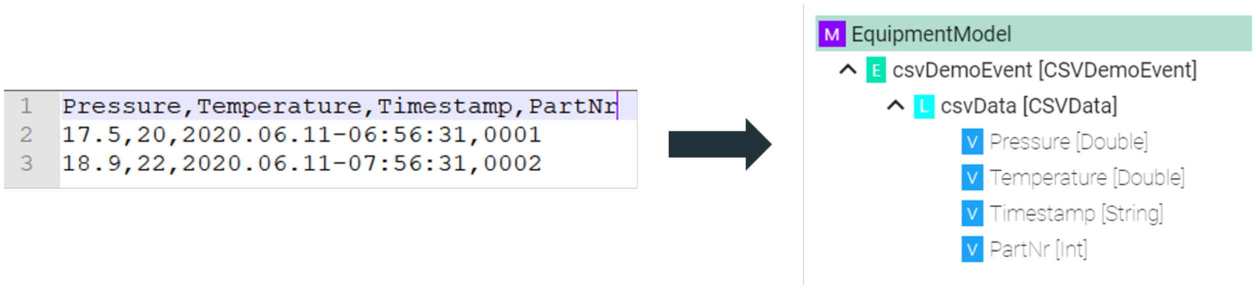
- CSV
- JSON
- XML

Information Model Requirements

The first Node after the root node M must be of type Event E.

CSV

- The node after the Event must be of type *List* L - multiple lines, each representing a data record.
- Fields, which are separated by commas, are represented by the Node Type *Variable* V. Note that the order of fields in the CSV file must match the order of Variables in the Information Model.

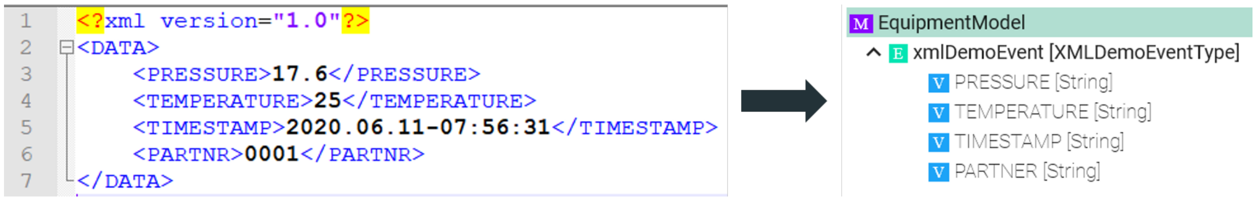


```
Pressure,Temperature,Timestamp,PartNr
17.5,20,2020.06.11-06:56:31,0001
18.9,22,2020.06.11-07:56:31,0002
```

```
M EquipmentModel
  E csvDemoEvent [CSVDemoEvent]
    L csvData [CSVData]
      V Pressure [Double]
      V Temperature [Double]
      V Timestamp [String]
      V PartNr [Int]
```

XML

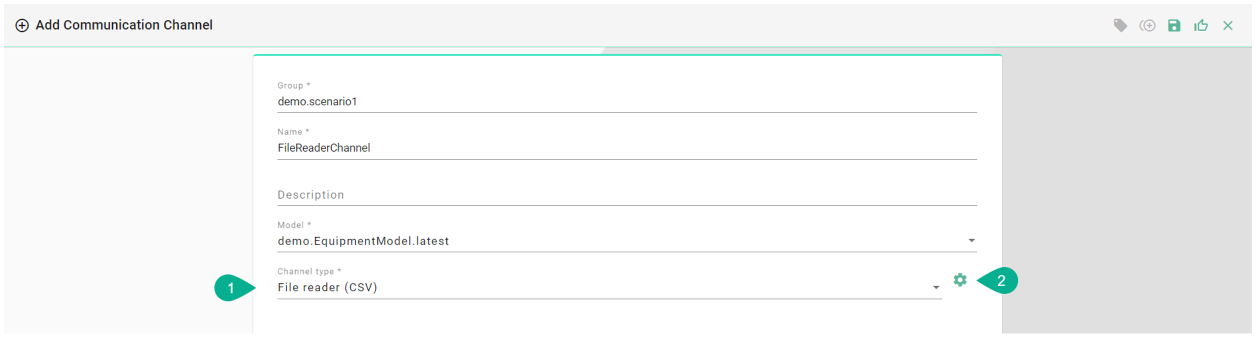
- Elements of the XML file are represented by the Node Type Variable V.
- Attributes of the XML file are represented by the Node Type *Property* P. In order to assign attributes to elements in the Information Model, the element Node Type V must be a *Custom Data Type*.



```
<?xml version="1.0"?>
<DATA>
    <PRESSURE>17.6</PRESSURE>
    <TEMPERATURE>25</TEMPERATURE>
    <TIMESTAMP>2020.06.11-07:56:31</TIMESTAMP>
    <PARTNR>0001</PARTNR>
</DATA>
```

How to use File Reader with CSV

1. Select **File reader (CSV)** from the Drop-Down.
2. Click the **Configure** button.



Add Communication Channel

Group *
demo.scenario1

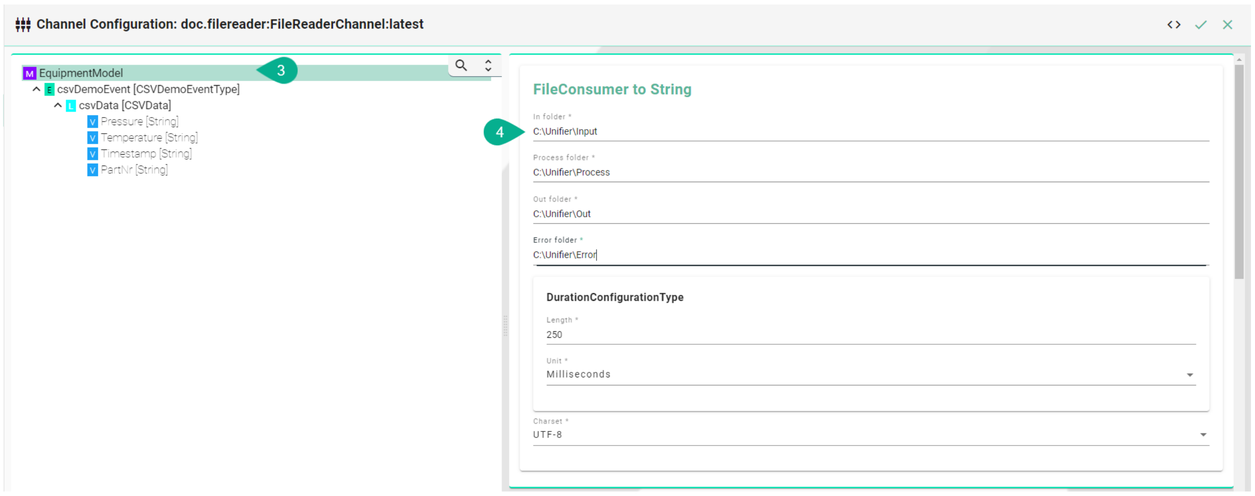
Name *
FileReaderChannel

Description

Model *
demo.EquipmentModel.latest

Channel type *
File reader (CSV)

3. **Make sure** the root model node is selected to configure the File Consumer to String as well as the CSV String to Model.
4. File Consumer to String - Configuration
   - Enter a path for the input folder - **In Folder**
   - Enter a path for the process folder - **Process Folder**
   - Enter a path for the output folder - **Out Folder**
   - Enter a path for the error folder - **Error Folder**
   - Specify the **Polling interval** and select the **Unit**
   - Select the **CharSet** according to the file in use


Channel Configuration: doc.filereader:FileReaderChannel:latest

FileConsumer to String

In folder *
C:\Unifier\Input

Process folder *
C:\Unifier\Process

Out folder *
C:\Unifier\Out

Error folder *
C:\Unifier\Error

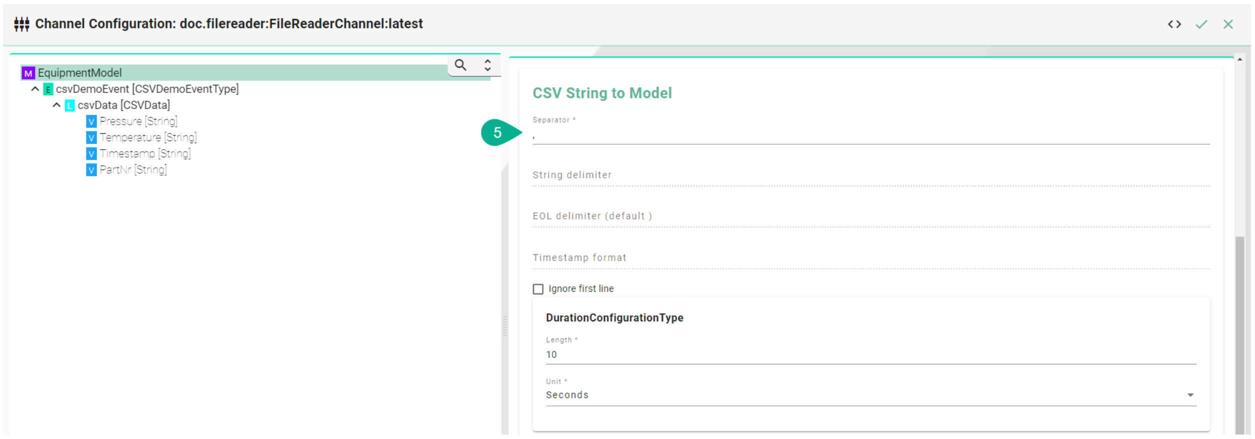
DurationConfigurationType

Length *
250

Unit *
Milliseconds

Charset *
UTF-8

5. CSV Consumer to Model - Configuration
   - Enter the **Separator** which is used in the CSV-file
   - If needed, set **String delimiter**, **EOL delimiter** and the **Timestamp format**
   - If the CSV file contains a header enable **Ignore first line**
   - Specify the **Polling interval** and select the **Unit**


Channel Configuration: doc.filereader:FileReaderChannel:latest

CSV String to Model

Separator *
,

String delimiter

EOL delimiter (default)

Timestamp format

☐ Ignore first line

DurationConfigurationType

Length *
10

Unit *
Seconds

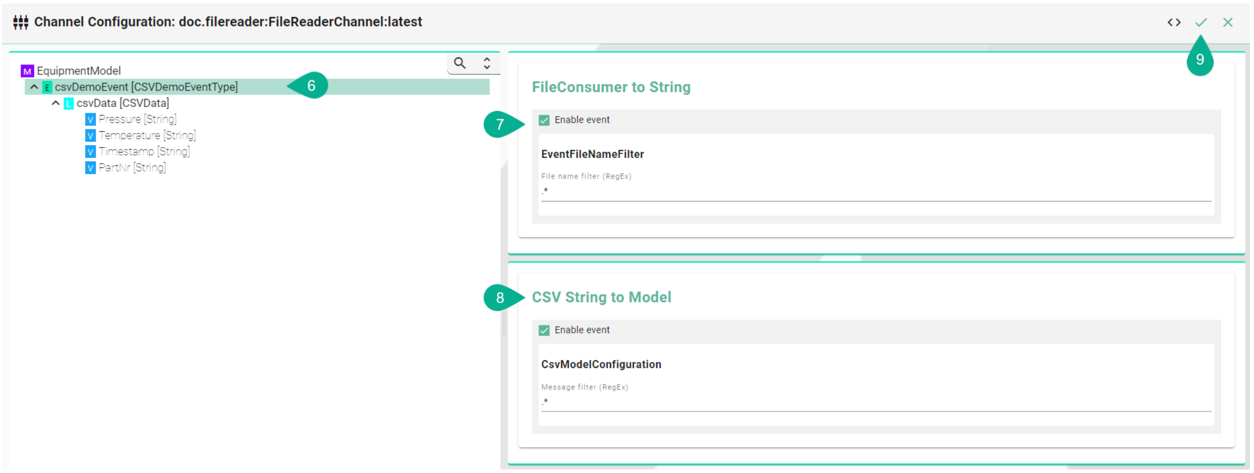
6. Specify the Event used by selecting the **event node** in the tree on the left side

Note: The entries of a CSV-File can only be mapped directly to an Event object and its parameters.

7. File Consumer to String - Configuration
   - Enable the **Event** checkbox for the **File Name Filter**
   - Enter a **Regular expression** in order to determine which file is to be processed in the input folder
8. Csv String to Model - Configuration
   - Enable the **Event** checkbox for the **Csv Model Configuration**

- Start of processing
  - If the entire content of the file is processed on this event enter a wildcard in the **RegEx** field
  - If the processing starts at a specific line enter a regular expression in the **RegEx** field to identify the line

9. Click the **Apply** button, then the **Close** button and save the Channel by clicking the **Save** button


Channel Configuration: doc.filereader:FileReaderChannel:latest

EquipmentModel
csvDemoEvent [CSVDemoEventType]
csvData [CSVData]
Pressure [String]
Temperature [String]
Timestamp [String]
PartNr [String]

FileConsumer to String
✓ Enable event
EventFileNameFilter
File name filter (RegEx)
.*

CSV String to Model
✓ Enable event
CsvModelConfiguration
Message filter (RegEx)
.*

Description of configuration properties:

Property	Description	Example
Separator	Separator type, used in the csv file	, , ;
Delimiter	Values that have an additional delimiter like "Date", "Time"	"
Eol Delimiter	Defining Carriage return and/or Line Feed	`\r`, `\n`
Timestamp format	Format of the timestamp	`YYYY-MM-DD HH:mm:ss`
ignoreFirstLine	Delay between checks of the file for new content in milliseconds	`true`, `false`
TailFromEnd	Set to true to tail from the end of the file, false to tail from the beginning of the file	`true`, `false`
InFolder	Path leading to the Input Folder	`C:\FileConsumer\In`
OutFolder	Path of a node in the Information Model	`C:\FileConsumer\Out`
ErrorFolder	Regular Expression for the message filter used in the implementation	`C:\FileConsumer\Error`
CharSet	Encoding of the file in use	`UTF-8`, `UTF-8 BOM`, etc
ProcessFolder	Regular Expression for the message filter used in the implementation	`C:\FileConsumer\Process`

File Tailer

Characteristics

- File Tailer monitors a given file in a given location.
- Data is processed line by line.
- Note that the File Tailer does not support the definition type **List** in the *Information Model*.

Supported File Formats:

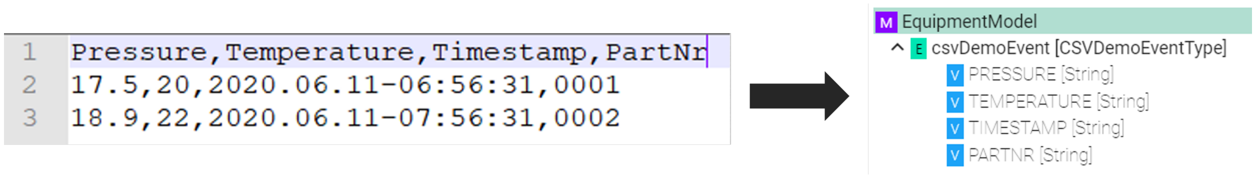
- CSV
- JSON
- XML

Information Model Requirements

The first Node after the root node M must be of type Event E.

CSV

- Fields, which are separated by commas, are represented by the Node Type *Variable* V. Note that the order of fields in the CSV file must match the order of Variables in the Information Model.

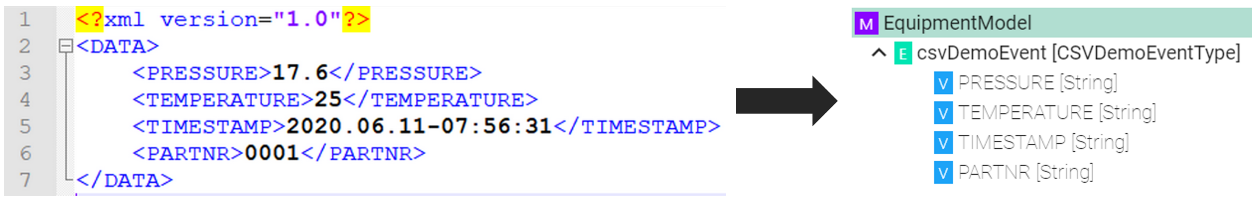


```
Pressure,Temperature,Timestamp,PartNr
17.5,20,2020.06.11-06:56:31,0001
18.9,22,2020.06.11-07:56:31,0002
```

```
M EquipmentModel
  ^ E csvDemoEvent [CSVDemoEventType]
        V PRESSURE [String]
        V TEMPERATURE [String]
        V TIMESTAMP [String]
        V PARTNR [String]
```

XML

- Elements of the XML file are represented by the Node Type Variable V.
- Attributes of the XML file are represented by the Node Type *Property* P. In order to assign attributes to elements in the Information Model, the element Node Type V must be a *Custom Data Type*.

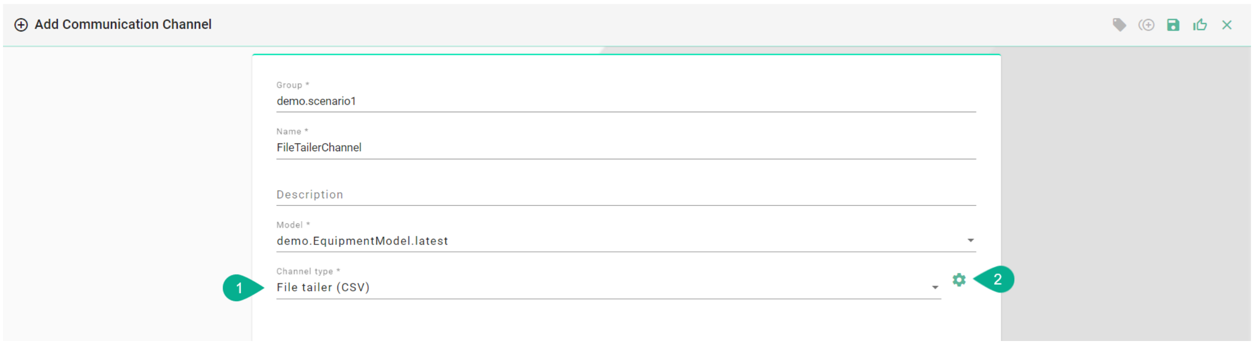


```
<?xml version="1.0"?>
<DATA>
    <PRESSURE>17.6</PRESSURE>
    <TEMPERATURE>25</TEMPERATURE>
    <TIMESTAMP>2020.06.11-07:56:31</TIMESTAMP>
    <PARTNR>0001</PARTNR>
</DATA>
```

```
M EquipmentModel
  ^ E csvDemoEvent [CSVDemoEventType]
        V PRESSURE [String]
        V TEMPERATURE [String]
        V TIMESTAMP [String]
        V PARTNR [String]
```

How to configure the File Tailer (CSV) Channel

1. Select **File tailer (CSV)** from the Drop-Down.
2. Click the **Configure** button.



Add Communication Channel

Group *
demo.scenario1

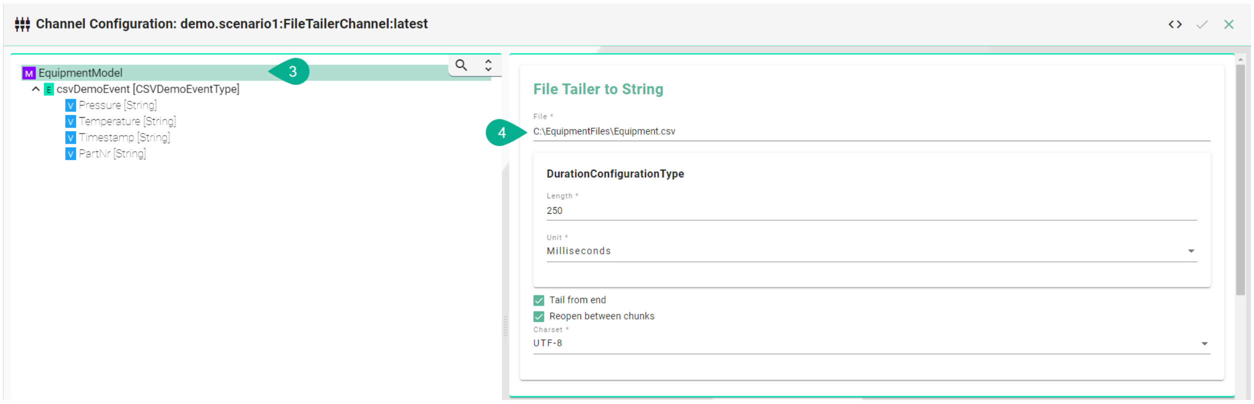
Name *
FileTailerChannel

Description

Model *
demo.EquipmentModel.latest

Channel type *
File tailer (CSV)

3. **Make sure** the root model node is selected to be able to configure the File Tailer to String and CSV String to Model.
4. File Tailer to String - Configuration:
   - Enter the **File path** for the CSV-file on your machine
   - Specify the **Polling interval** and select the **Unit**
   - Enable **Tail from end** if you want to pick up always the last line of the file
   - Enable **Reopen between chunks** if the file should be closed and reopened between chunks
   - Select the **Charset** according to the file in use



Channel Configuration: demo.scenario1:FileTailerChannel:latest

EquipmentModel
csvDemoEvent [CSVDemoEventType]
Pressure [String]
Temperature [String]
Timestamp [String]
PartNr [String]

File Tailer to String

File *
C:\EquipmentFiles\Equipment.csv

DurationConfigurationType

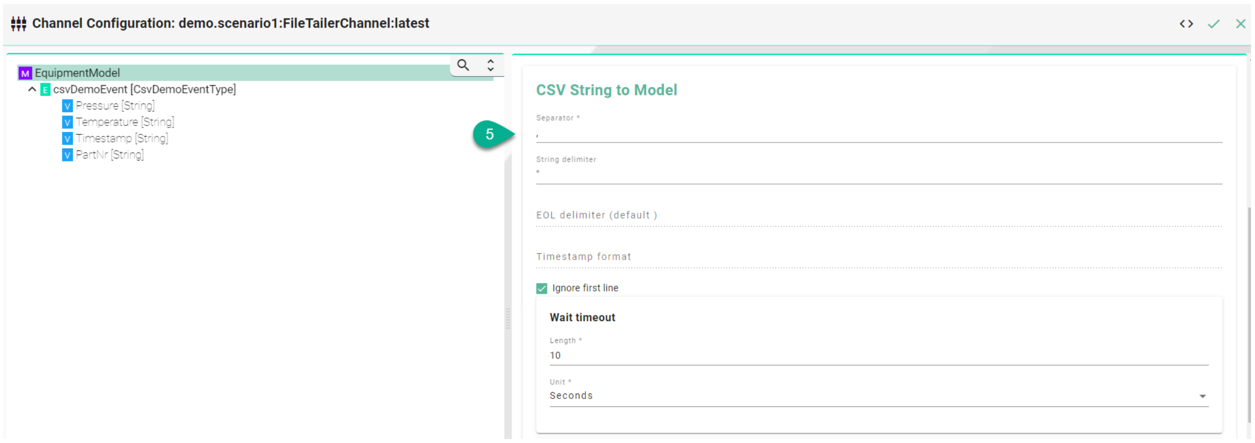
Length *
250

Unit *
Milliseconds

✓ Tail from end
✓ Reopen between chunks

Charset *
UTF-8

5. CSV String to Model - Configuration:
   - Enter the **Separator** which is used in the CSV-file as well as the **String delimiter**
   - Input the **Eol delimiter** and the **Timestamp format** if one is used.
   - If the CSV file contains a header enable **Ignore first line**
   - Input the **Polling interval** and select the **Unit**


Channel Configuration: demo.scenario1:FileTailerChannel:latest

EquipmentModel
csvDemoEvent [CsvDemoEventType]
Pressure [String]
Temperature [String]
Timestamp [String]
PartNr [String]

CSV String to Model

Separator *
,

String delimiter
"

EOL delimiter (default)

Timestamp format

✓ Ignore first line

Wait timeout

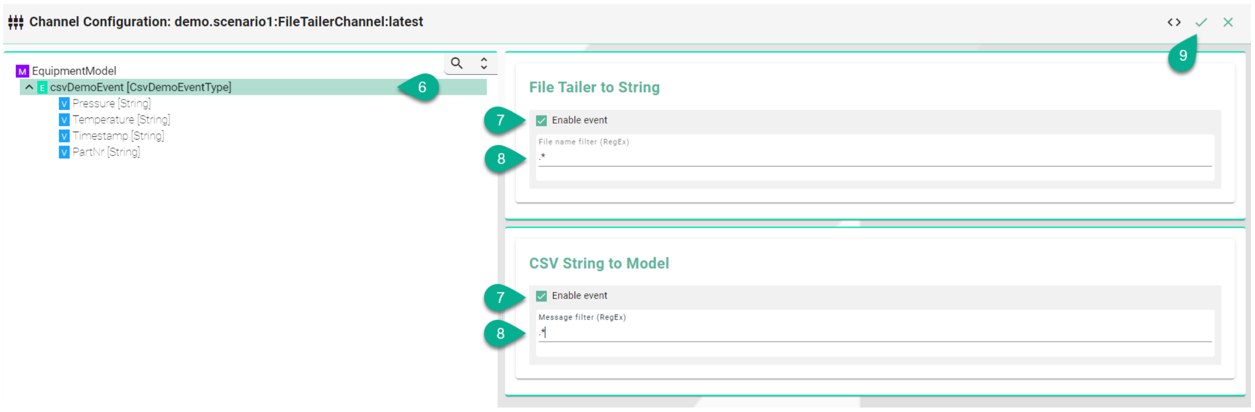
Length *
10

Unit *
Seconds

6. Select the **event node** in the tree on the left side.

Note: The entries of a CSV-File can only be *mapped* directly to an *Event* object and its parameters.

7. Check the **Routes checkbox**.
8. Enter a **Regular expression** for the message filter.
9. Click the **Apply** button, then the **Close** button and save the Channel by clicking the **Save** button on the upper right corner.


Channel Configuration: demo.scenario1:FileTailerChannel:latest

EquipmentModel
csvDemoEvent [CsvDemoEventType]
Pressure [String]
Temperature [String]
Timestamp [String]
PartNr [String]

File Tailer to String

✓ Enable event

File name filter (RegEx)
.*

CSV String to Model

✓ Enable event

Message filter (RegEx)
.*

Description of configuration properties:

Property	Description	Example
Separator	Separator type, used in the csv file	`,` `,` `;`
Delimiter	Values that have an additional delimiter like “Date”, “Time”	`"`
Eol Delimiter	Defining Carriage return and/or Line Feed	`\r`, `\n`
Timestamp format	Format of the timestamp	`YYYY-MM-DD HH:mm:ss`
File	Path to the csv file	`C:\test.csv`
Delay Millis	Delay between checks of the file for new content in milliseconds	`250`
TailFromEnd	Set to true to tail from the end of the file, false to tail from the beginning of the file	`true`, `false`
ReopenBetweenChunks	If true, close and reopen the file between reading chunks	`true`, `false`
routes	Path of a node in the Information Model	`true`, `false`
messageFilterRegEx	Regular Expression for the message filter used in the implementation	`.*`

Databases

SQL Database

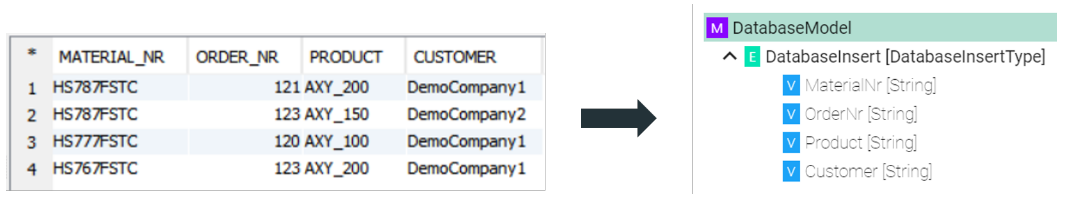
Characteristics - SQL Database

- The SQL Channel can be configured for the following two scenarios:
  - Inserting data
  - Updating data
  - Retrieving data
- When inserting values into the database please **note** that “infinity” values are converted automatically into “null” values.

Information Model Requirements

Insert/Update

- The node after the root model node must be of type *Event* E which represent a database table.
- In case of relational databases: Tables which are dependent on each other require a *List* L.
- Columns of databases are represented by *Variables* V.

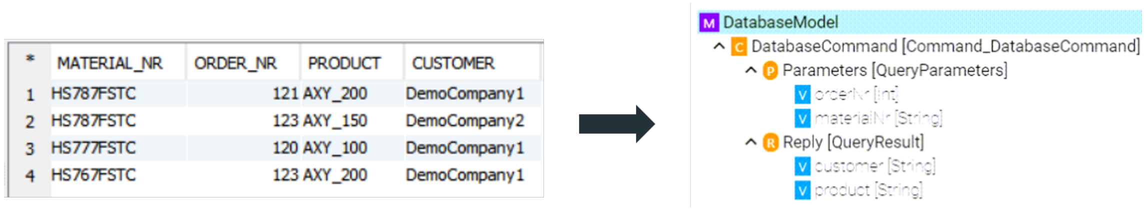


*	MATERIAL_NR	ORDER_NR	PRODUCT	CUSTOMER
1	HS787FSTC	121	AXY_200	DemoCompany1
2	HS787FSTC	123	AXY_150	DemoCompany2
3	HS777FSTC	120	AXY_100	DemoCompany1
4	HS767FSTC	123	AXY_200	DemoCompany1

M DatabaseModel
- E DatabaseInsert [DatabaseInsertType]
  - V MaterialNr [String]
  - V OrderNr [String]
  - V Product [String]
  - V Customer [String]

Select

- The *Command* C defines that after a request is made, a reply with a result is expected.
- Parameters P within a Command represent a collection of query parameter - query parameters are defined as Variables V.
- Reply R within a Command represents the result of the Command - results are defined as Variables V.

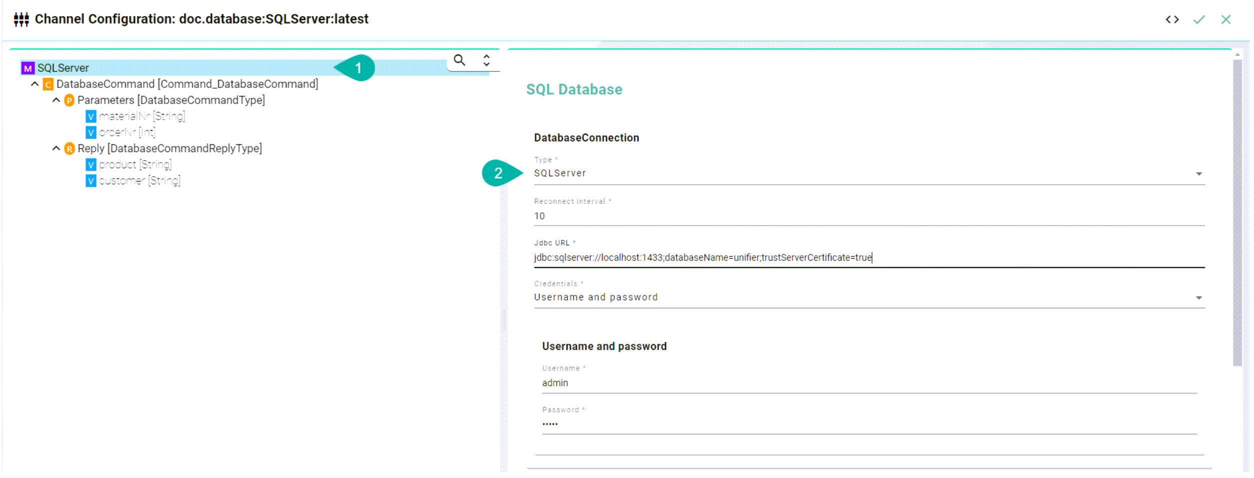


*	MATERIAL_NR	ORDER_NR	PRODUCT	CUSTOMER
1	HS787FSTC	121	AXY_200	DemoCompany1
2	HS787FSTC	123	AXY_150	DemoCompany2
3	HS777FSTC	120	AXY_100	DemoCompany1
4	HS767FSTC	123	AXY_200	DemoCompany1

```
M DatabaseModel
  ^ C DatabaseCommand [Command_DatabaseCommand]
    ^ P Parameters [QueryParameters]
        V orderNr [int]
        V materialNr [String]
    ^ R Reply [QueryResult]
        V customer [String]
        V product [String]
```

How to configure the SQL-Database

1. Select the **root model node** in the tree on the left.
2. Configure the database connection
   - Select the **Database type**.
   - Specify a **Reconnection interval**.
   - Enter the **database connection URL** for the specific database type.
     - DB2: `jdbc:db2:server:port/database`
     - HSQLDB: `jdbc:hsqldb:file:databaseFileName;properties`
     - ORACLE: `jdbc:oracle:thin:prodHost:port:sid`
     - PostgreSQL: `jdbc:postgresql://host:port/database`
     - SQLServer: `jdbc:sqlserver://[serverName[\instanceName][:portNumber]][;property=value[;property=value]]`
     - MariaDB: `jdbc:(mysql|mariadb):[replication:|loadbalance:|sequential:|aurora:]//<host>[:<portnumber>]/[database][?<key1>=<value1>[&<key2>=<value2>]]`
   - Enter the database **Username** and **Password** or select it from the Credentials Manager.



Channel Configuration: doc.database:SQLServer:latest

- SQLServer
  - DatabaseCommand [Command_DatabaseCommand]
    - Parameters [DatabaseCommandType]
      - materialNr [String]
      - orderNr [Int]
    - Reply [DatabaseCommandReplyType]
      - product [String]
      - customer [String]

SQL Database

DatabaseConnection

Type *: SQLServer

Reconnect interval *: 10

Jdbc URL *: jdbc:sqlserver://localhost:1433;databaseName=unifier;trustServerCertificate=true

Credentials *: Username and password

Username and password

Username *: admin

Password *: •••••

Description of configuration properties:

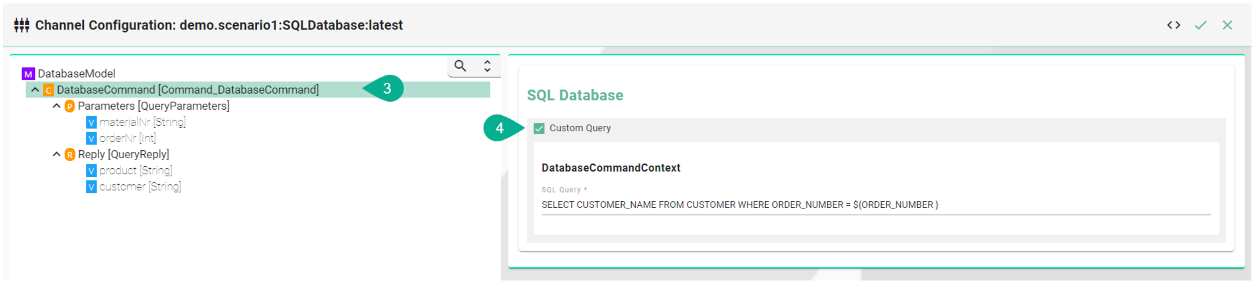
Property	Description	Example
Type	Type of the database	`MariaDB`, `SQLServer`, `ORACLE`, `HSQLDB`, `DB2`, `PostgreSQL`
ReconnectInterval	Time to reconnect if connection fails	`10` (in milliseconds)
JdbcUrl	Url to connect to database	• jdbc:sqlserver://localhost:1433;databaseName=unifier;trustServerCertificate=true • jdbc:mariadb://localhost:3306/unifier?connectTimeout=5000 • jdbc:db2://127.0.0.1:50000/TESTDB • jdbc:hsqldb:file:\protect\T1\textdollardbFileName;shutdown=true • jdbc:oracle:thin:@localhost:1521/MYCDB • jdbc:postgresql://127.0.0.1:5432/postgres
Username and **password**	Credentials of the database	

Note: The configuration of specific *information model nodes* differs whether you want to perform an **insert** or an **select** statement on the database. Inserting data into the database requires an

event node whereas selecting data requires a **command node** in the *Information Model*.

Select Statement

3. Select the **command node** in the tree on the left.
4. Check the **Custom Query** checkbox and enter the **SQL Query**.



Channel Configuration: demo.scenario1:SQLDatabase:latest

DatabaseModel
DatabaseCommand [Command_DatabaseCommand]
Parameters [QueryParameters]
materialNr [String]
orderNr [Int]
Reply [QueryReply]
product [String]
customer [String]

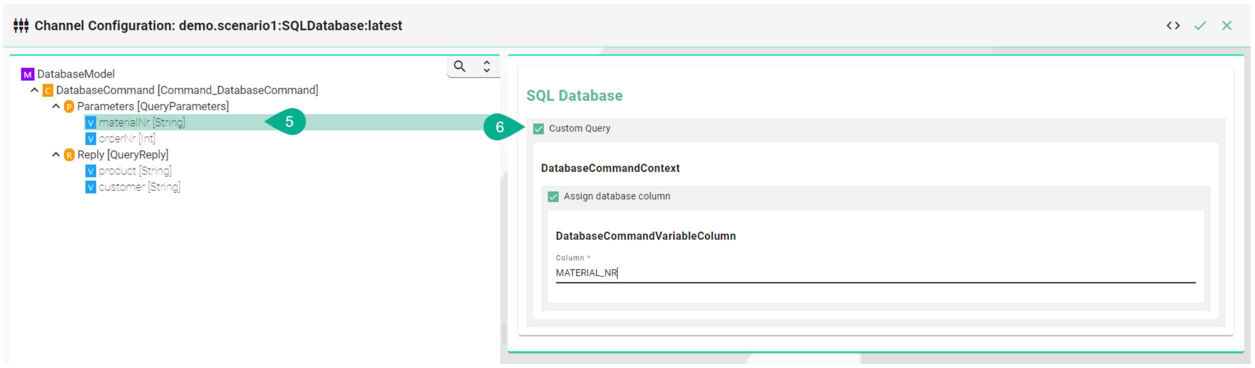
SQL Database

Custom Query

DatabaseCommandContext

SQL Query *
SELECT CUSTOMER_NAME FROM CUSTOMER WHERE ORDER_NUMBER = ${ORDER_NUMBER }

5. Each variable under *Parameters* and *Reply* needs to be assigned to a database column. Select the **variable node** under *Parameters* and in the tree select what needs to be configured.
6. Check the **Assign database column** checkbox and enter the **Column name** as it is defined in the used database.



Channel Configuration: demo.scenario1:SQLDatabase:latest

DatabaseModel
DatabaseCommand [Command_DatabaseCommand]
Parameters [QueryParameters]
materialNr [String]
orderNr [Int]
Reply [QueryReply]
product [String]
customer [String]

SQL Database

Custom Query

DatabaseCommandContext

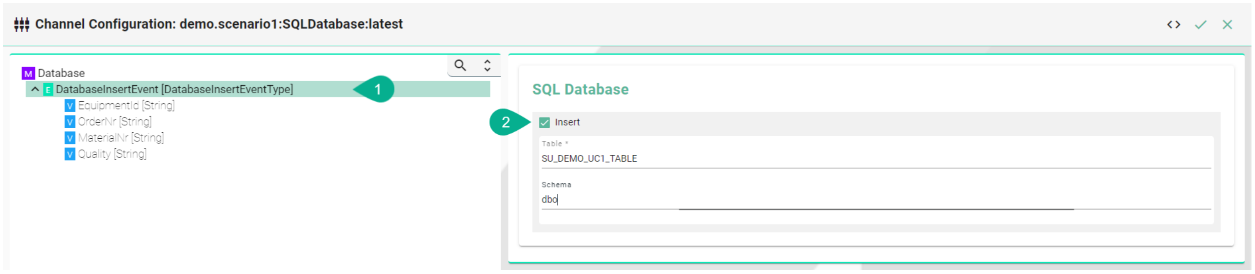
Assign database column

DatabaseCommandVariableColumn

Column *
MATERIAL_NR

Insert Statement

1. Select the **event node** in the tree on the left.
2. Check the **Insert** checkbox and enter the **Table name**. If required enter a **Schema name**.



Channel Configuration: demo.scenario1:SQLDatabase:latest

Database
DatabaseInsertEvent [DatabaseInsertEventType]
EquipmentId [String]
OrderNr [String]
MaterialNr [String]
Quality [String]

SQL Database

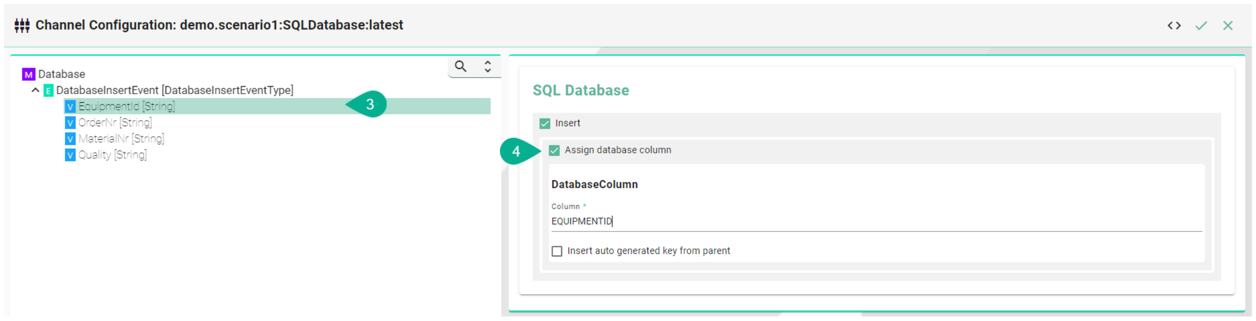
Insert

Table *
SU_DEMO_UC1_TABLE

Schema
dbo

3. Select the **variable node** in the tree on the left
4. Check the **Assign database column** checkbox and enter the **Column name** (Check the **Insert auto generated key from parent** checkbox if the column relates to its parent)

Note: Configuration of the column name is only necessary if the column name in the database is different compared to the variable defined in the Information Model.


Channel Configuration: demo.scenario1:SQLDatabase:latest

- Database
  - DatabaseInsertEvent [DatabaseInsertEventType]
    - EquipmentId [String]
    - OrderNr [String]
    - MaterialNr [String]
    - Quality [String]

SQL Database

- [✓] Insert
- [✓] Assign database column

DatabaseColumn

Column *: EQUIPMENTID

- [ ] Insert auto generated key from parent

InfluxDB v1

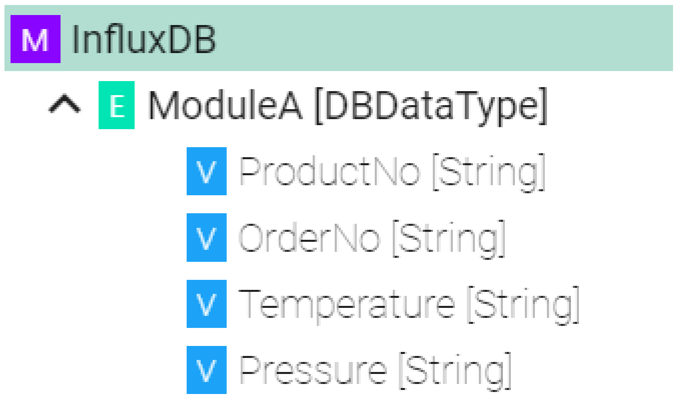
Characteristics - InfluxDB v1

In case of a time series data use case where you need to ingest data in a fast and efficient way you can use InfluxDB.

Information Model Requirements

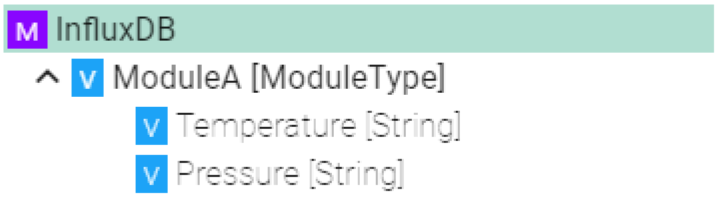
Inserts using Events

- The node after the root model in this case is of the type *Event* E which represent a database table.
- Fields are represented by *Variables* V.

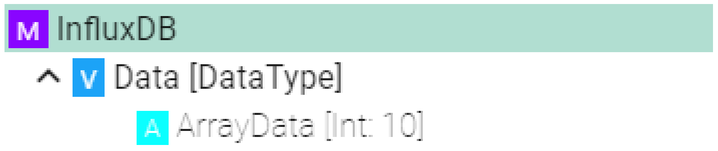

- M InfluxDB
  - E ModuleA [DBDataType]
    - V ProductNo [String]
    - V OrderNo [String]
    - V Temperature [String]
    - V Pressure [String]

Inserts using Custom Data Types

- Complex *Variables* V (ModuleA) represents Measurements
- *Variables* V underneath within the complex variable (Temperature) represents Fields

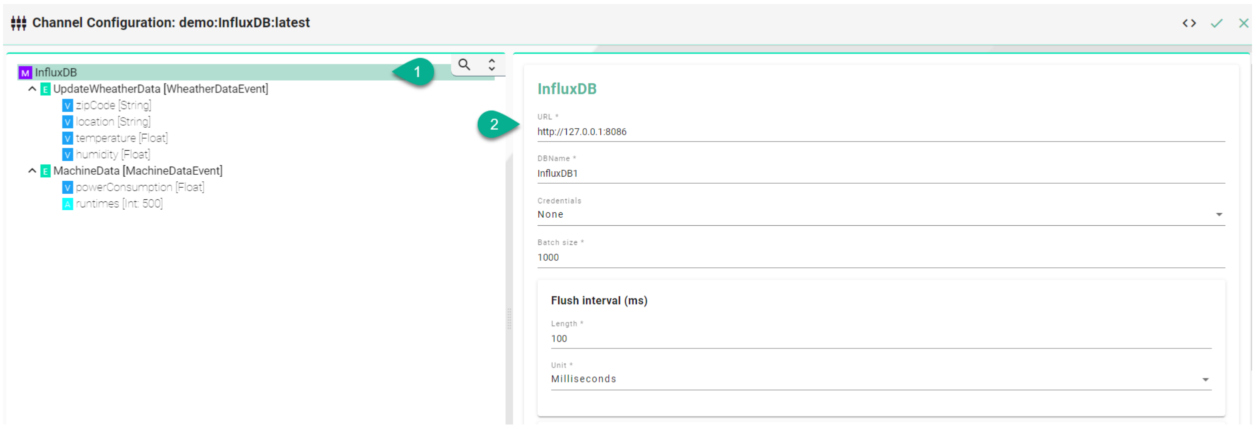


- Arrays A can be used to set use an index



How to configure InfluxDB v1

1. Select the **root model node** in the tree on the left.
2. Configure the InfluxDB.
   - Enter the **URL** to the database
   - Enter the **Database name**
   - Enter the database **Username** and **Password** or select it from the Credentials Manager
   - Enter the **Batch size** - writes data in batches to minimize network overhead when writing data to InfluxDB
   - Enter the **Flush interval** and select the **Unit** (Please note that too short intervalls might cause data loss!)



Channel Configuration: demo:InfluxDB:latest

InfluxDB

URL *
http://127.0.0.1:8086

DBName *
InfluxDB1

Credentials
None

Batch size *
1000

Flush interval (ms)

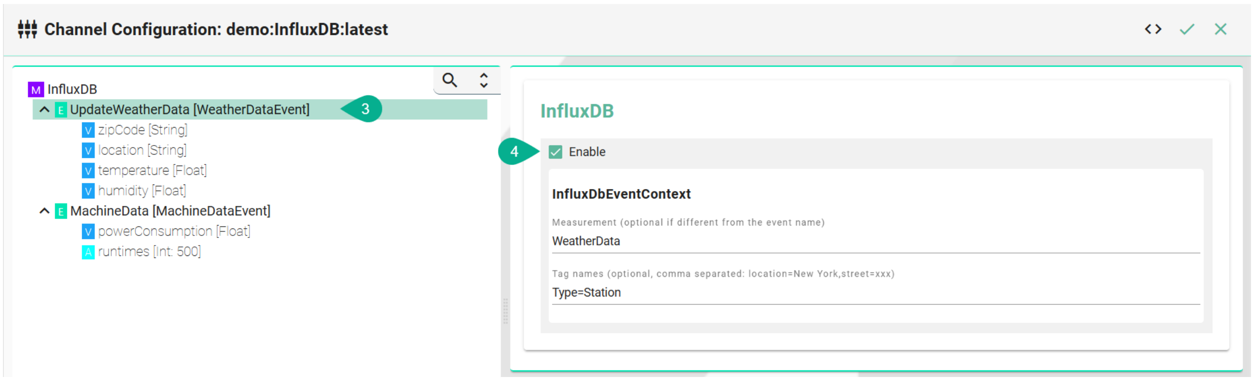
Length *
100

Unit *
Milliseconds

Event Configuration

3. Select the **event node**

4. Enable the checkbox to configure the event
    - Enter the **Measurement** - if it differs from the event name
    - Enter **Tags** - comma separated

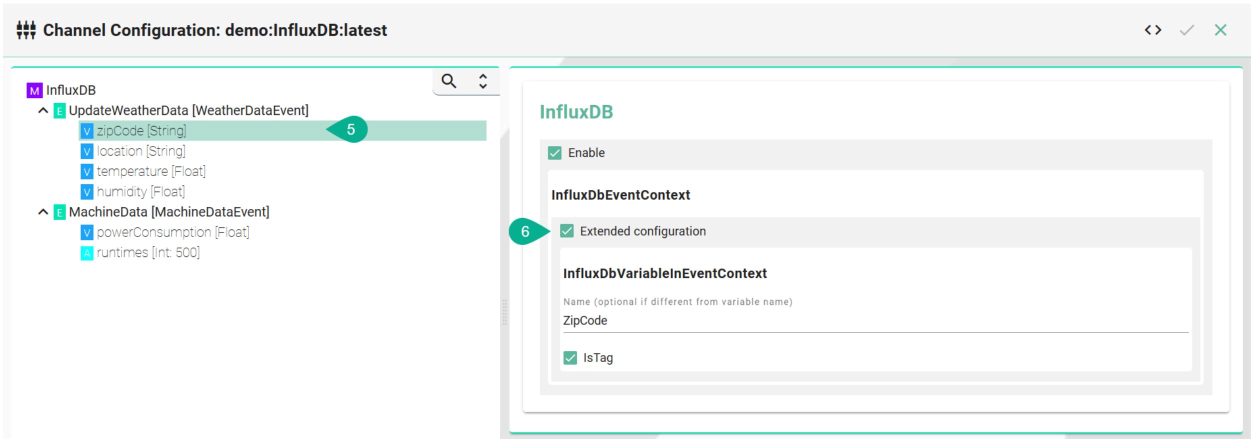


Measurement (optional if different from the event name): WeatherData

Tag names (optional, comma separated: location=New York,street=xxx): Type=Station

Configuration of Tags

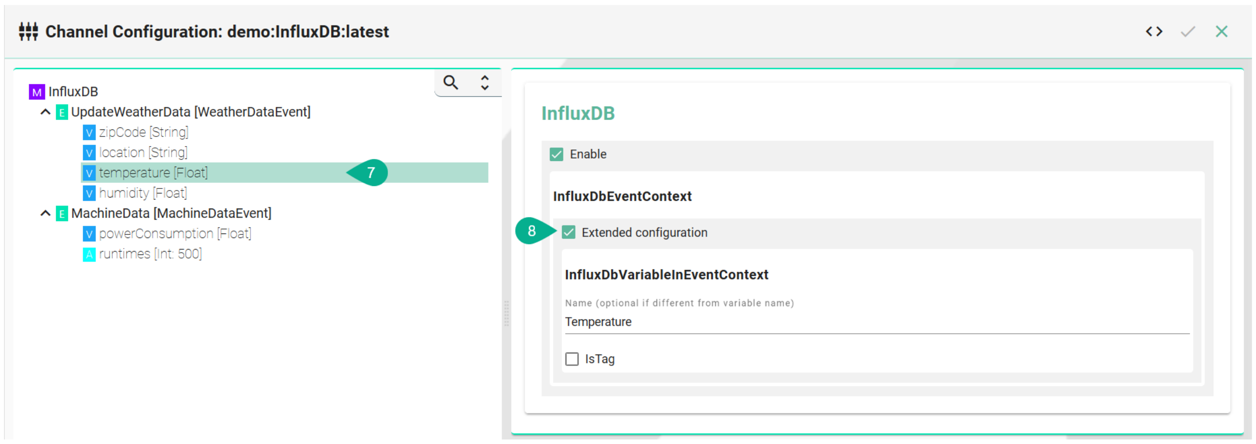
5. Select the variable which should be a **Tag**
6. Enable **Extended configuration**
    - Enter a **Name** - if it differs from the variable name
    - Enable the checkbox **IsTag**



Name (optional if different from variable name): ZipCode

Configuration of fields

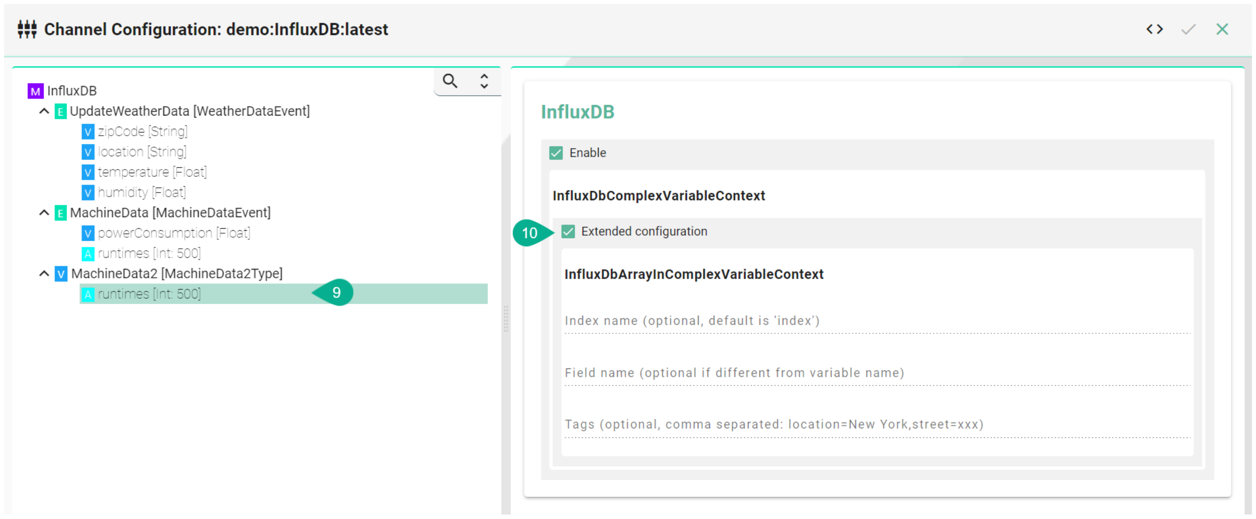
7. Select the variable which should be a **field**
8. Enable **Extended configuration**
    - Enter a **Name** - if it differs from the variable name
    - Leave the checkbox **IsTag** disabled



Array Configuration

9. Select the Array
10. To configure the Array select **Extended Configuration**

- (Optional) Enter an **Index** name
- (Optional) Enter a **Field** name if the event node name differs from the actual name in InfluxDB.
- (Optional) Enter **Tags** separated by commas e.g., (location=NewYork, street=xxx)


Channel Configuration: demo:InfluxDB:latest
InfluxDB
UpdateWeatherData [WeatherDataEvent]
zipCode [String]
location [String]
temperature [Float]
humidity [Float]
MachineData [MachineDataEvent]
powerConsumption [Float]
runtimes [Int: 500]
MachineData2 [MachineData2Type]
runtimes [Int: 500]
9
10
InfluxDB
Enable
InfluxDbComplexVariableContext
Extended configuration
InfluxDbArrayInComplexVariableContext
Index name (optional, default is 'index')
Field name (optional if different from variable name)
Tags (optional, comma separated: location=New York,street=xxx)

Description of configuration properties:

Property	Description	Example
URL	Database URL and port	`http://127.0.0.1:8086`
DB Name	Database name	`InfluxDB`
Credentials	Database credentials	`None`
Batch size	Data written in batches	`1000`
Flush interval	Delay between data flushes in milliseconds, at most batch size records are sent during flush	`1000`
Measurement	Name of the measurement stored in influxdb	`WeatherData`
Tag names	Optional tag to be added to the measurement	`Type=Station`

InfluxDB v2

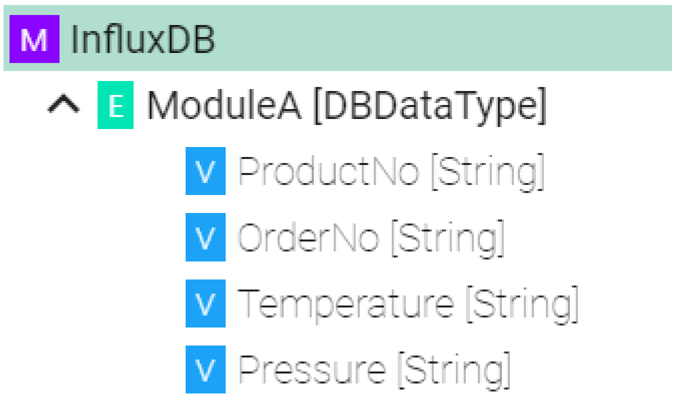
Characteristics - InfluxDB v2

In case of a time series data use case where you need to ingest data in a fast and efficient way you can use InfluxDB.

Information Model Requirements

Inserts using Events

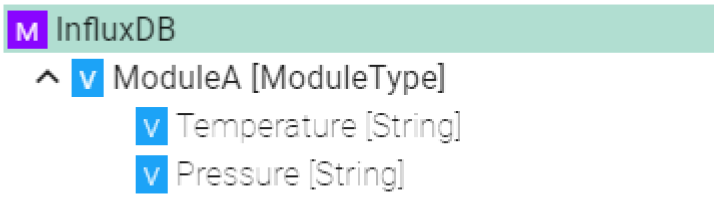
- The node after the root model in this case is of the type *Event* E which represent a database table.
- Fields are represented by *Variables* V.



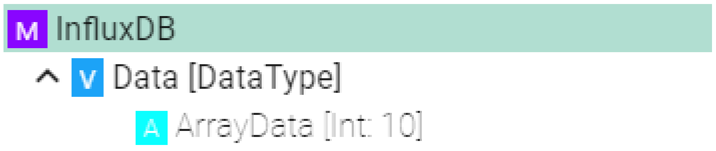
Inserts using Custom Data Types

- Complex *Variables* V (ModuleA) represents Measurements

- *Variables* V underneath within the complex variable (Temperature) represents Fields

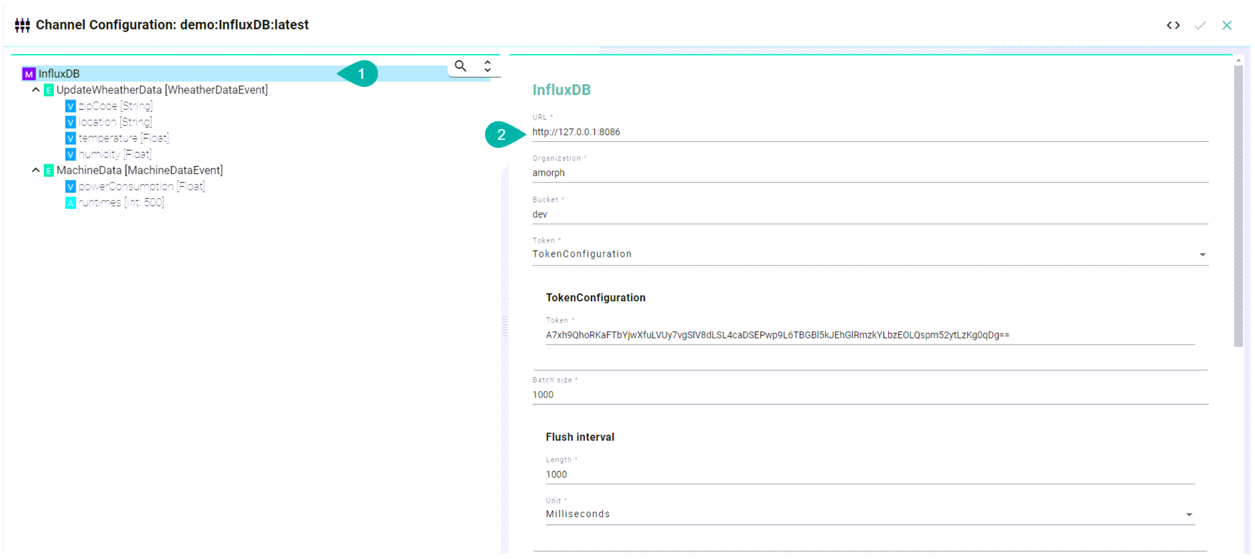


- Arrays A can be used to set use an index



How to configure InfluxDB v2

1. Select the **root model node** in the tree on the left.
2. Configure the InfluxDB.
   - Enter the **URL** to the database
   - Enter the **Organization** defined in the database
   - Enter the **Bucket** defined in the database
   - Enter the **Token** or select it from the Credential Manager
   - Enter the **Batch size** - writes data in batches to minimize network overhead when writing data to InfluxDB
   - Enter the **Flush interval** and select the **Unit** (Please note that too short intervalls might cause data loss!)


Channel Configuration: demo:InfluxDB:latest

- InfluxDB
  - UpdateWheatherData [WheatherDataEvent]
    - zipCode [String]
    - location [String]
    - temperature [Float]
    - humidity [Float]
  - MachineData [MachineDataEvent]
    - powerConsumption [Float]
    - runtimes [Int: 500]

InfluxDB

URL *: http://127.0.0.1:8086

Organization *: amorph

Bucket *: dev

Token *: TokenConfiguration

TokenConfiguration

Token *: A7xh9QhoRKaFTbYjwXfuLVUy7vgSlV8dLSL4caDSEPwp9L6TBGBl5kJEhGlRmzkYLbzEOLQspm52ytLzKg0qDg==

Batch size *: 1000

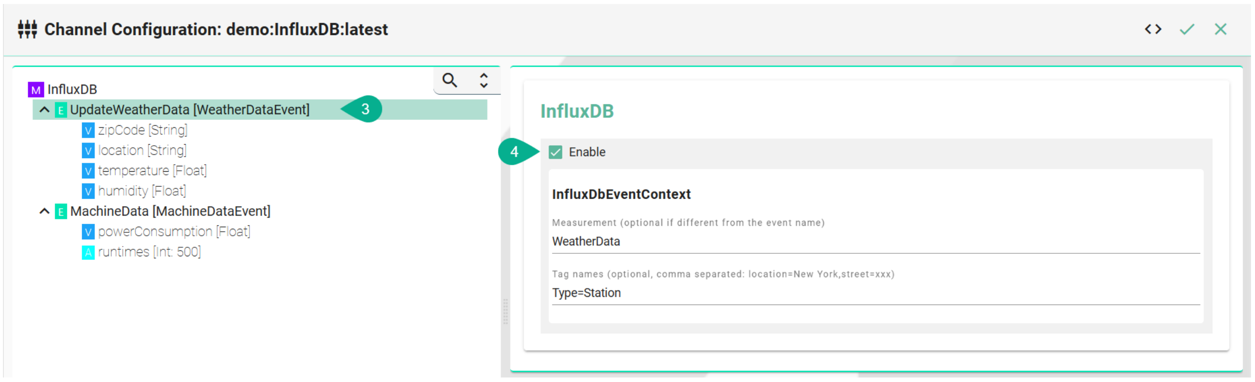
Flush interval

Length *: 1000

Unit *: Milliseconds

Event Configuration

1. Select the **event node**
2. Enable the checkbox to configure the event
   - Enter the **Measurement** - if it differs from the event name
   - Enter **Tags** - comma separated


Channel Configuration: demo:InfluxDB:latest

- InfluxDB
  - UpdateWeatherData [WeatherDataEvent]
    - zipCode [String]
    - location [String]
    - temperature [Float]
    - humidity [Float]
  - MachineData [MachineDataEvent]
    - powerConsumption [Float]
    - runtimes [Int: 500]

InfluxDB

✓ Enable

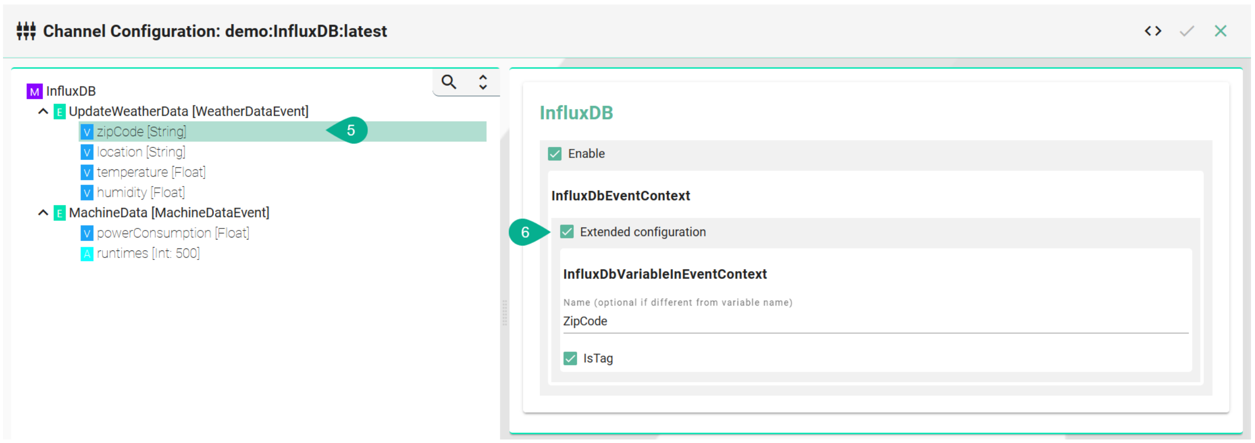
InfluxDbEventContext

Measurement (optional if different from the event name): WeatherData

Tag names (optional, comma separated: location=New York,street=xxx): Type=Station

Configuration of Tags

5. Select the variable which should be a **Tag**
6. Enable **Extended configuration**
   - Enter a **Name** - if it differs from the variable name
   - Enable the checkbox **IsTag**

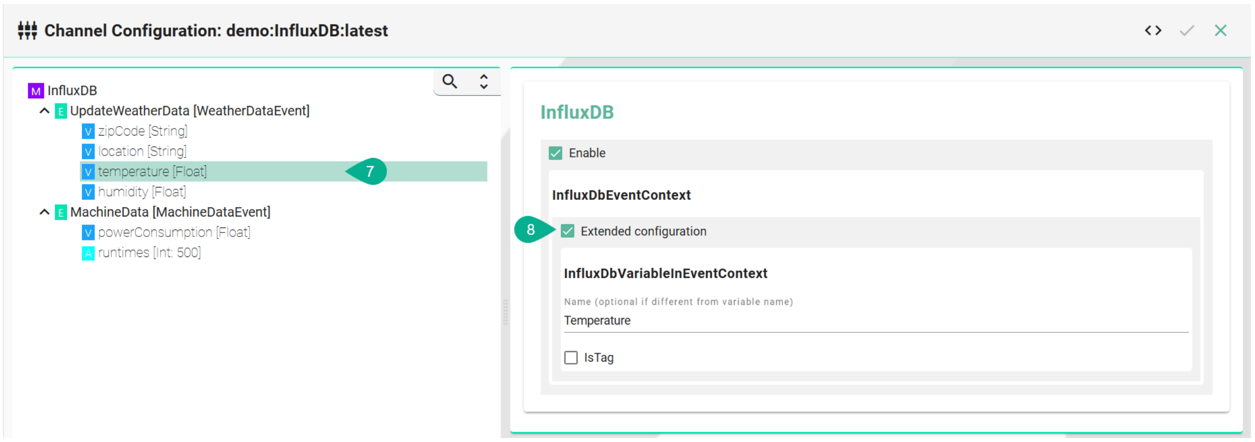

Channel Configuration: demo:InfluxDB:latest

InfluxDB
UpdateWeatherData [WeatherDataEvent]
zipCode [String]
location [String]
temperature [Float]
humidity [Float]
MachineData [MachineDataEvent]
powerConsumption [Float]
runtimes [Int: 500]

InfluxDB
Enable ✓
InfluxDbEventContext
Extended configuration ✓
InfluxDbVariableInEventContext
Name (optional if different from variable name): ZipCode
IsTag ✓

Configuration of fields

7. Select the variable which should be a **field**
8. Enable **Extended configuration**
    - Enter a **Name** - if it differs from the variable name
    - Leave the checkbox **IsTag** disabled


Channel Configuration: demo:InfluxDB:latest

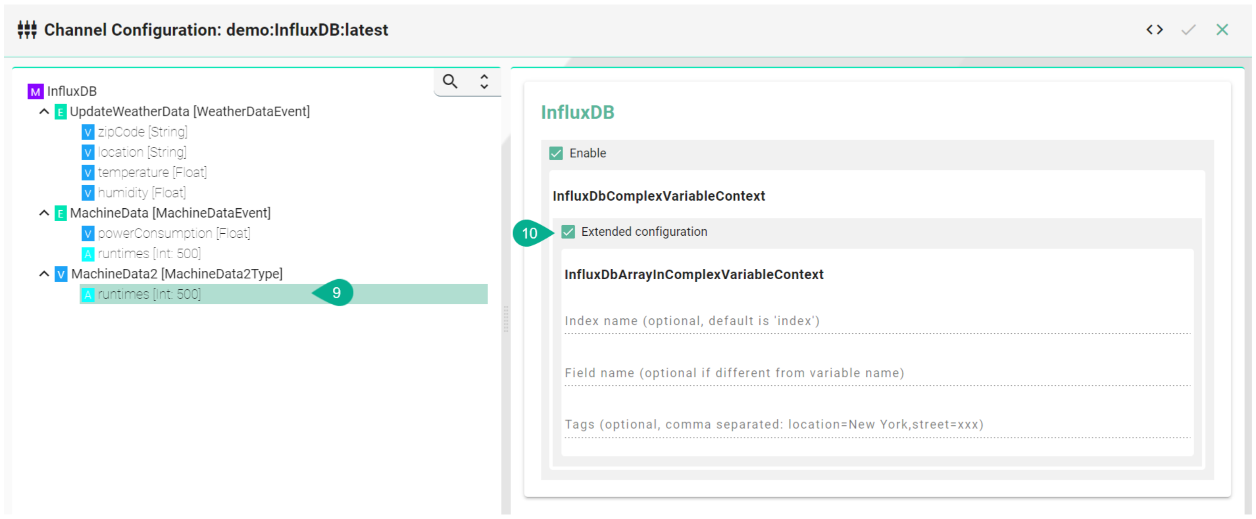
InfluxDB
UpdateWeatherData [WeatherDataEvent]
zipCode [String]
location [String]
temperature [Float]
humidity [Float]
MachineData [MachineDataEvent]
powerConsumption [Float]
runtimes [Int: 500]

InfluxDB
Enable ✓
InfluxDbEventContext
Extended configuration ✓
InfluxDbVariableInEventContext
Name (optional if different from variable name): Temperature
IsTag ☐

Array Configuration

9. Select the Array
10. To configure the Array select **Extended Configuration**

- (Optional) Enter an **Index** name
- (Optional) Enter a **Field** name if the event node name differs from the actual name in InfluxDB.
- (Optional) Enter **Tags** separated by commas e.g., (location=NewYork, street=xxx)


Channel Configuration: demo:InfluxDB:latest

- InfluxDB
  - UpdateWeatherData [WeatherDataEvent]
    - zipCode [String]
    - location [String]
    - temperature [Float]
    - humidity [Float]
  - MachineData [MachineDataEvent]
    - powerConsumption [Float]
    - runtimes [Int: 500]
  - MachineData2 [MachineData2Type]
    - runtimes [Int: 500]

InfluxDB

✓ Enable

InfluxDbComplexVariableContext

✓ Extended configuration

InfluxDbArrayInComplexVariableContext

Index name (optional, default is 'index')

Field name (optional if different from variable name)

Tags (optional, comma separated: location=New York,street=xxx)

Description of configuration properties:

Property	Description	Example
URL	Database URL and port	`http://127.0.0.1:8086`
Organization	Name of the Organization	`CompanyName`
Bucket	Name of the Bucket	`Database_1`
Credentials	Token-based authentication	Token
Batch size	Data written in batches	`1000`
Flush interval	Delay between data flushes in milliseconds, at most batch size records are sent during flush	`1000`
Measurement	Name of the measurement stored in influxdb	`WeatherData`
Tag names	Optional tag to be added to the measurement	`Type=Station`

Protocols

MQTT

Characteristics - MQTT

MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). To learn more about the standard visit the MQTT website.

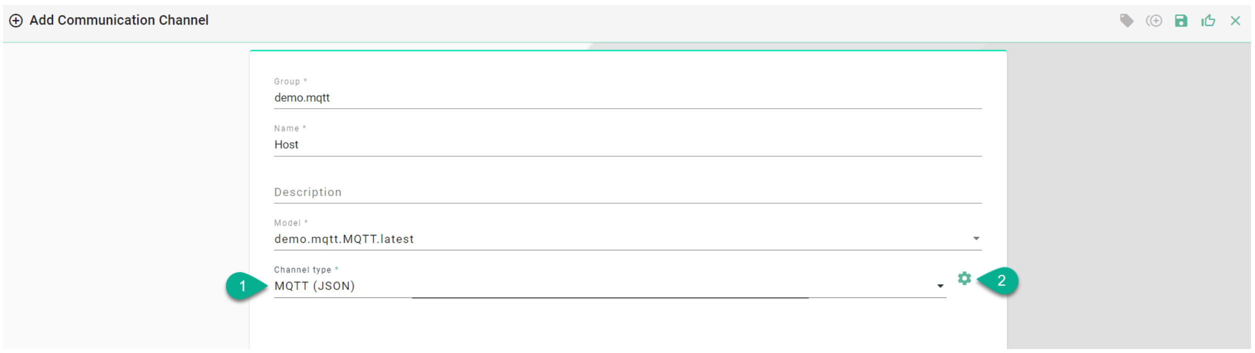
Information Model Requirements

- The first Node after the root node M must be of type *Event* E.

- The following Node Types can be used under the Event Node:
  - *Variables* V with a *Simple Data Type* represents the key-value pairs.
  - *Variables* V with a *Custom Data Type* represent objects that can contain key-value pairs.
  - With *Lists* L you can aggregate multiple variables.
- In case of publishing a topic, the Information Model determines the structure of the payload.
- In case of subscribing to a topic make sure that the Information Model structure matches the payload.

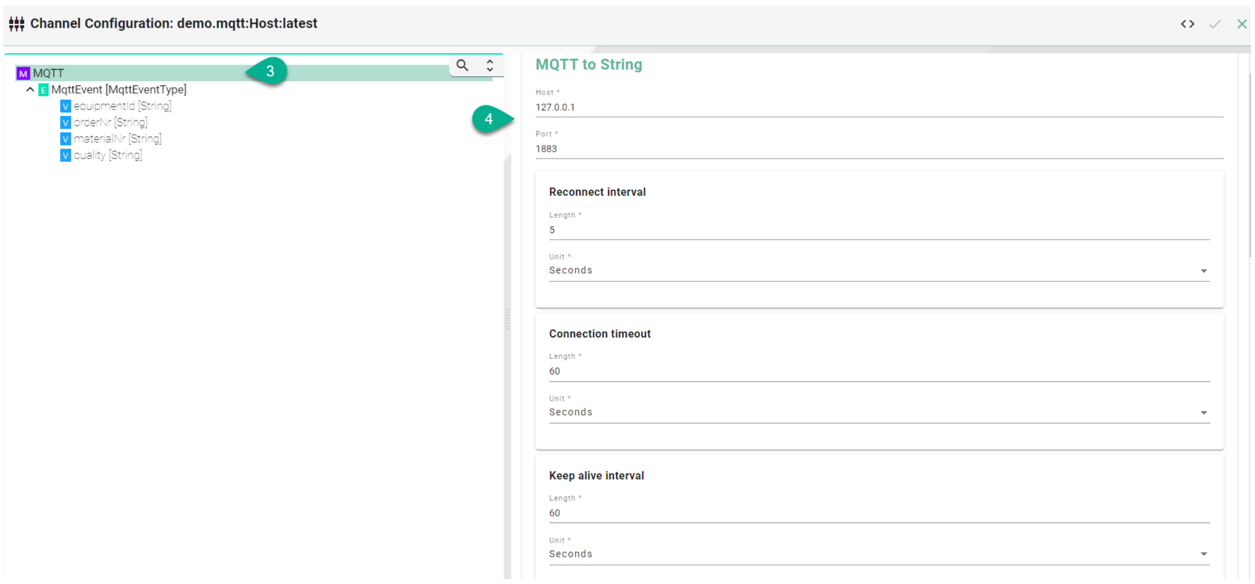
Configuration - MQTT Channel

1. Select the **MQTT (JSON)** as Channel Type.
2. Click the **Configure** button.



Group *
demo.mqtt
Name *
Host
Description
Model *
demo.mqtt.MQTT.latest
Channel type *
MQTT (JSON)

3. Select the **root model node**
4. Configure the MQTT To String configuration:
   - Enter **Host** and **Port** of the MQTT Broker used
   - If required, adjust the default values for **Reconnect interval**, **Connection timeout**, **Keep alive interval** and the **Unit** for each
   - Specify a path to a folder on your local machine. The **temp** directory inside the *SMARTUNIFIER Manager* can be used as well.
   - (Optional) Specify a **Client ID**
   - Set the **Quality of Service (QoS)**
   - (Optional) Enable **Retained** if required
   - Select **Username and password** in order to manually enter the credentials or select **Username and password credentials reference** to add it from the Credentials Manager. If there are no credentials needed (e.g., `test.mosquitto.org`) select **None**.



Channel Configuration: demo.mqtt:Host:latest

MQTT
MqttEvent [MqttEventType]
equipmentId [String]
orderNr [String]
materialNr [String]
quality [String]

MQTT to String

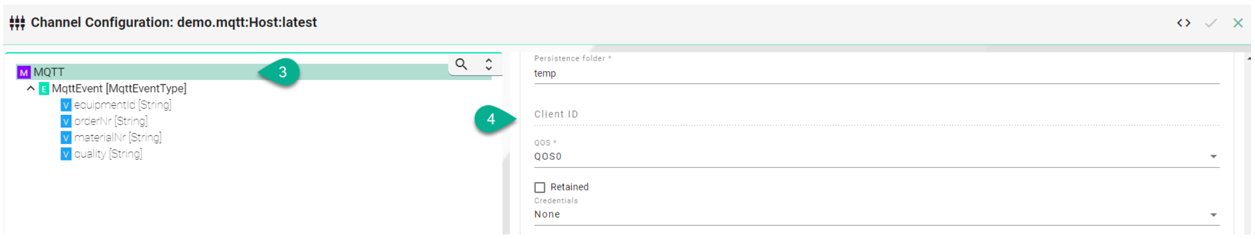
Host *
127.0.0.1

Port *
1883

Reconnect interval
Length *
5
Unit *
Seconds

Connection timeout
Length *
60
Unit *
Seconds

Keep alive interval
Length *
60
Unit *
Seconds



Channel Configuration: demo.mqtt:Host:latest

MQTT
MqttEvent [MqttEventType]
equipmentId [String]
orderNr [String]
materialNr [String]
quality [String]

Persistence folder *
temp

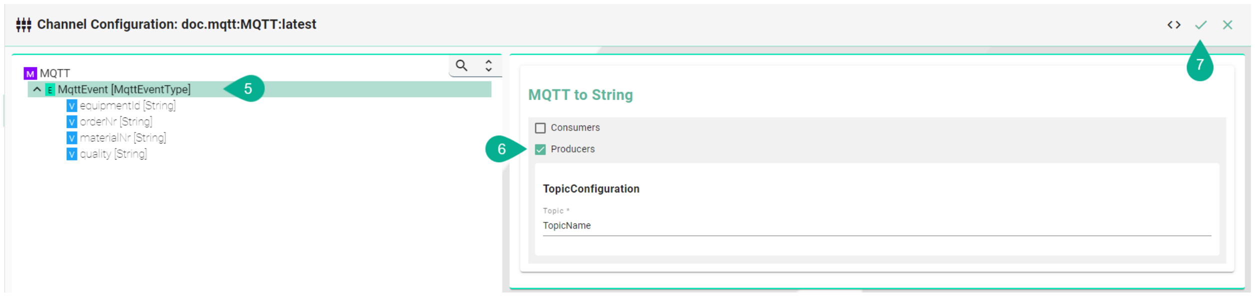
Client ID

QOS *
QOS0

☐ Retained

Credentials
None

5. Select the **event node** in the tree on the left.
6. Enable either **Producer** or **Consumer** depending on the use case and enter a **Topic name**.
7. Click the **Apply** button.



Channel Configuration: doc.mqtt:MQTT:latest

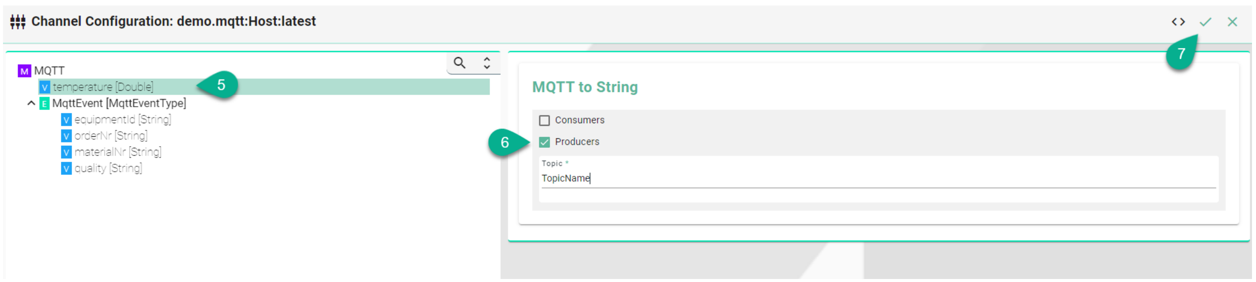
MQTT
MqttEvent [MqttEventType]
equipmentId [String]
orderNr [String]
materialNr [String]
quality [String]

MQTT to String

☐ Consumers
☑ Producers

TopicConfiguration
Topic *
TopicName

The **Producer** or **Consumer** option can be enabled for a **Variable node**.



Channel Configuration: demo.mqtt:Host:latest

MQTT
temperature [Double]
MqttEvent [MqttEventType]
equipmentId [String]
orderNr [String]
materialNr [String]
quality [String]

MQTT to String

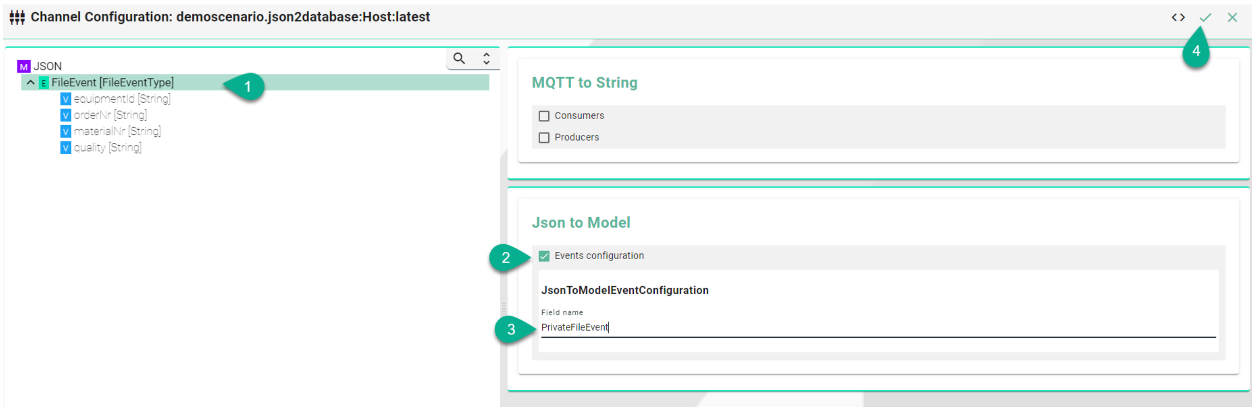
☐ Consumers
☑ Producers

Topic *
TopicName

Json To Model Event Configuration

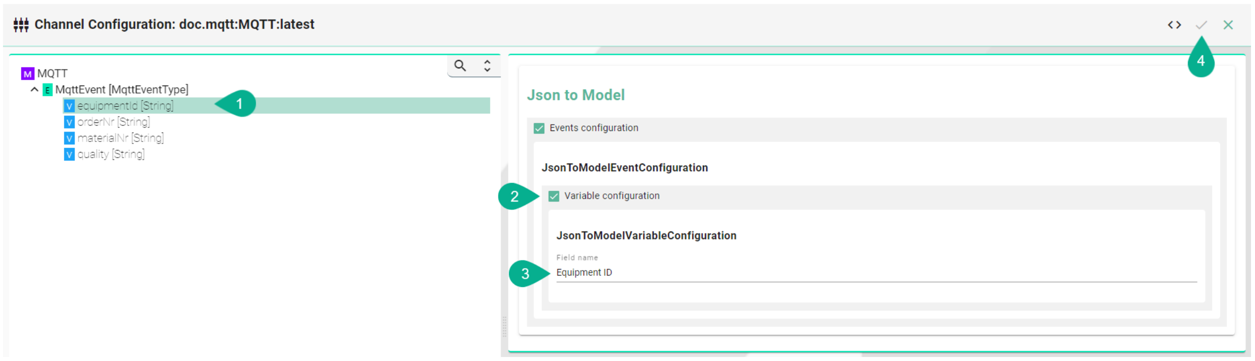
This configuration is used when some keywords or reserved words can't be used in the Information Model.

1. Select the **event node** in the tree on the left.
2. Check the box for the **Events configuration**.
3. Input the **Field name**, representing the reserved word.
4. Click on the **Apply** button.



Json To Model Variable Configuration

1. Select a **variable** in the tree on the left.
2. Check the box for the **Variable configuration**.
3. Input the **Field name**, representing the reserved word.
4. Click on the **Apply** button.



Certificates

Encrypted connection using TLS security is supported. Follow the steps below to encrypt the connection.

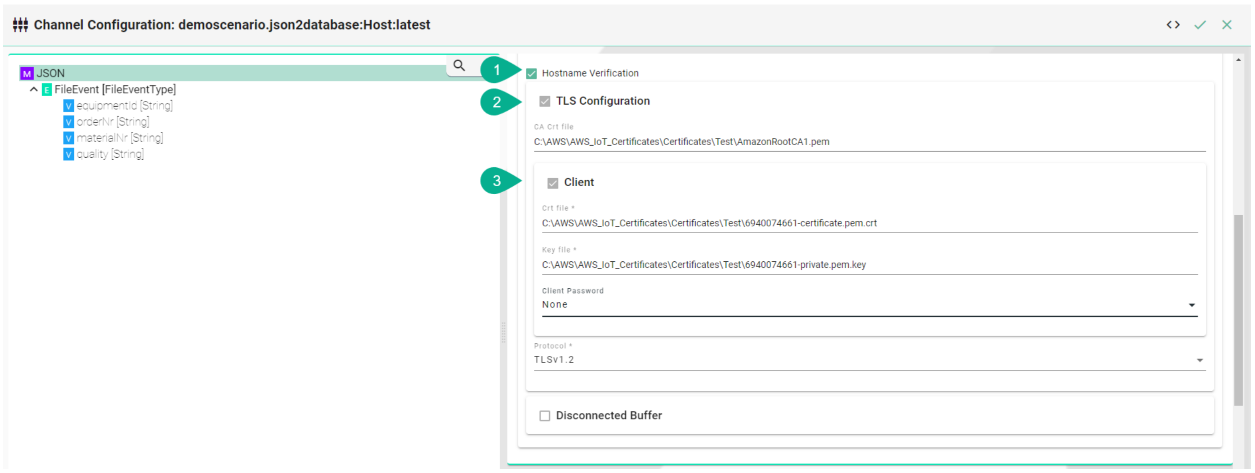
1. Enable **Hostname Verification** (optional)
2. Enable the **Tls Configuration** checkbox

- Enter the **path** to the **CA (certificate authority) certificate** of the CA that has signed the server certificate

Note: Make sure the CA certificate is valid.

3. Enable the **Client** checkbox

- Enter the **path** to the **Client certificate**. The client certificate identifies the client just like the server certificate identifies the server.
- Enter the **path** to the **Private client key**.
- If applicable select to enter a **Password** or to add from the **Credentials Manager**.
- Select the **Protocol** from the Drop-Down.


Channel Configuration: demoscenario.json2database:Host:latest

Hostname Verification

TLS Configuration

CA Crt file: C:\AWS\AWS_IoT_Certificates\Certificates\Test\AmazonRootCA1.pem

Client

Crt file *: C:\AWS\AWS_IoT_Certificates\Certificates\Test\6940074661-certificate.pem.crt

Key file *: C:\AWS\AWS_IoT_Certificates\Certificates\Test\6940074661-private.pem.key

Client Password: None

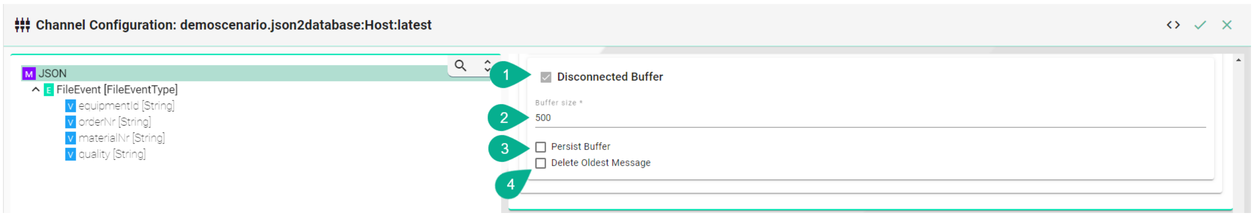
Protocol *: TLSv1.2

Disconnected Buffer

Disconnected Buffer

In case the connection is lost, messages can be buffered offline when the Disconnected Buffer is enabled. Follow the steps below to enable the DisconnectedBuffer.

1. Enable the **Disconnected Buffer** checkbox.
2. Set the **Buffer size** - defines the number of messages being hold e.g., 5000.
3. (Optional) Enable **Persist Buffer**.
4. (Optional) Enable **Delete Oldest Message**.


Channel Configuration: demoscenario.json2database:Host:latest

Disconnected Buffer

Buffer size *: 500

Persist Buffer

Delete Oldest Message

Description of configuration properties:

Property	Description	Example
host	URL of the MQTT Broker.	`test.mosquitto.org`
port	Port of the MQTT Broker.	`1883`
reconnectInterval	Time interval to reconnect to the MQTT Broker after loss of connection in seconds	`5`
connectionTimeout	Time interval the connection times out in seconds	`60`
keepAliveInterval	Time the session persists in seconds	`60`
persistenceFolder	Path to a folder for the persistence store of the MQTT	`temp`
clientId	Identifies an MQTT client which connects to an MQTT Broker	`MyClientID`
username	Client username	`Username`
password	Client password	`Password`
hostnameVerification	Hostname Verification	`true`, `false`
tls	Encryption	`true`, `false`
producers	Data producer	`true`, `false`
consumer	Data consumer	`true`, `false`
protocol	TLS protocol version	`TLSv1.1`, `TLSv1.2`
disconnectedBuffer	Offline buffering of data	`true`, `false`
bufferSize	Amount of message allowed in the buffer	`5000`
persistBuffer	Buffer persistence	`true`, `false`
deleteOldestMessage	Delete oldest message in buffer	`true`, `false`

Modbus

Characteristics - Modbus

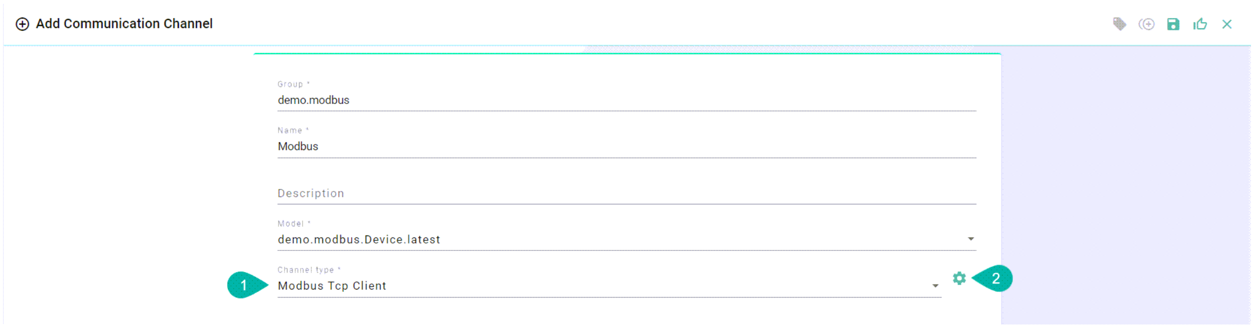
MODBUS is an application-layer messaging protocol, positioned at level 7 of the OSI model. It provides client/server communication between devices connected on different types of buses or networks. To learn more about the standard visit the MODBUS website.

Information Model Requirements

- The following Node Types can be used to model a register:
  - *Variables* V with a *Simple Data Type*.
  - *Variables* V with a *Custom Data Type*.

Configuration - Modbus

1. Select **Modbus/Tcp Client** as Channel Type.
2. Click the **Configure** button.



Add Communication Channel

Group *: demo.modbus

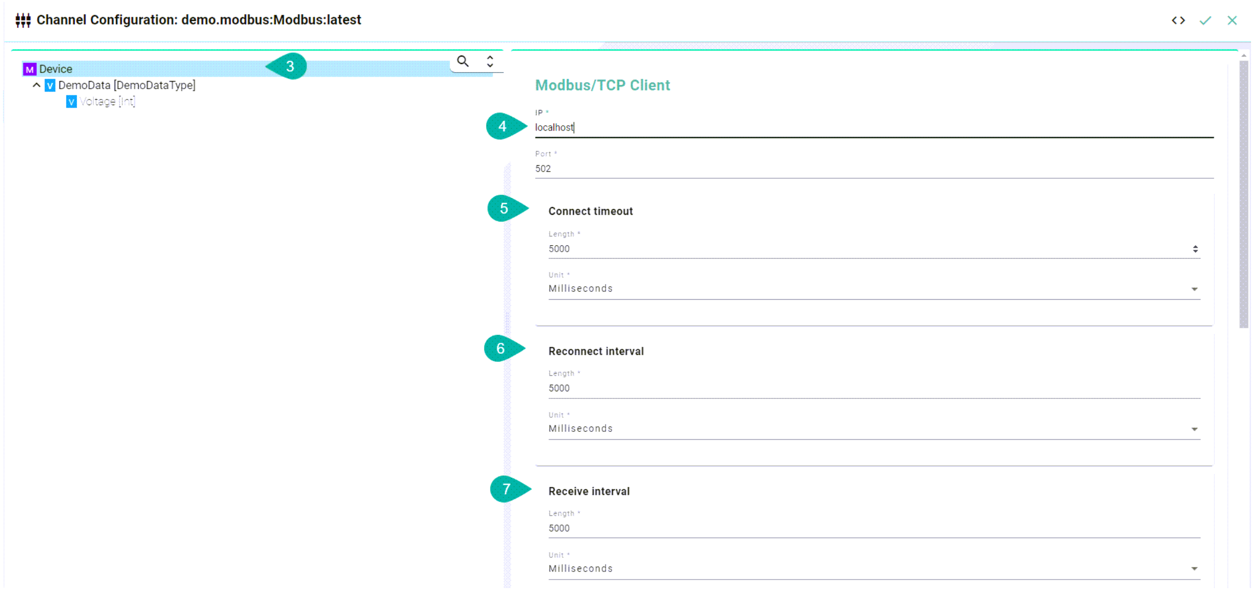
Name *: Modbus

Description

Model *: demo.modbus.Device.latest

Channel type *: Modbus Tcp Client

3. Make sure the root model node is selected to configure the Modbus/TCP Client
4. Enter the **IP** address and the **port**
5. (Optional) Change the **Connect interval** if needed
6. (Optional) Change the **Reconnect interval** if needed
7. (Optional) Change the **Receive interval** if needed



Channel Configuration: demo.modbus:Modbus:latest

Device
DemoData [DemoDataType]
Voltage [Int]

Modbus/TCP Client

IP *: localhost

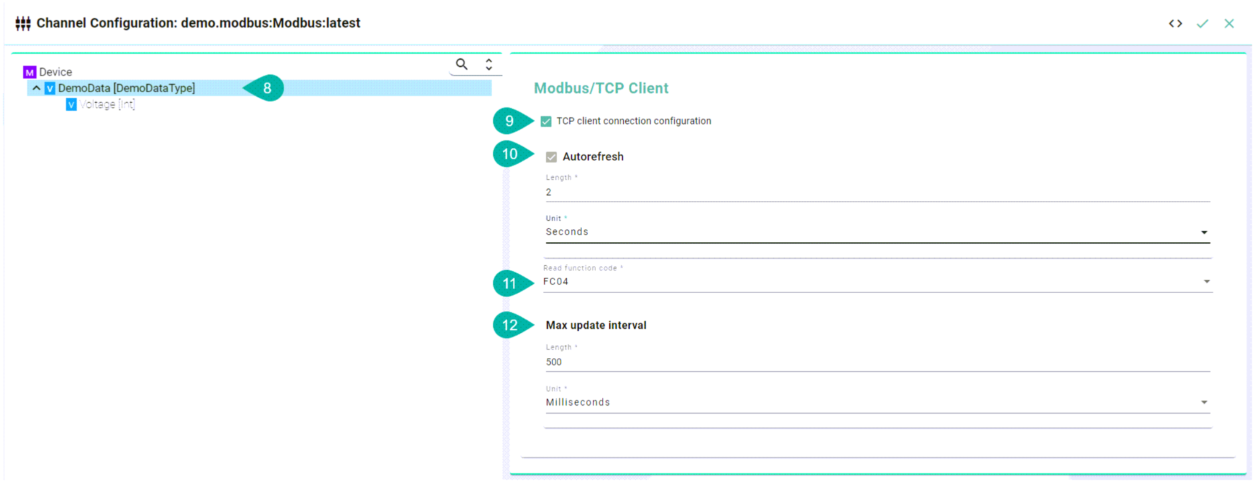
Port *: 502

Connect timeout
Length *: 5000
Unit *: Milliseconds

Reconnect interval
Length *: 5000
Unit *: Milliseconds

Receive interval
Length *: 5000
Unit *: Milliseconds

8. Select the complex variable node
9. Enable the checkbox **TCP Client connection configuration**
10. (Optional) Enable **Autorefresh** to specify the retrieval rate
11. Select the **Function Code**
12. (Optional) Change the **Max update interval** if needed


Channel Configuration: demo.modbus:Modbus:latest

Device
DemoData [DemoDataType]
Voltage [Int]

Modbus/TCP Client

TCP client connection configuration

Autorefresh

Length *: 2

Unit *: Seconds

Read function code *: FC04

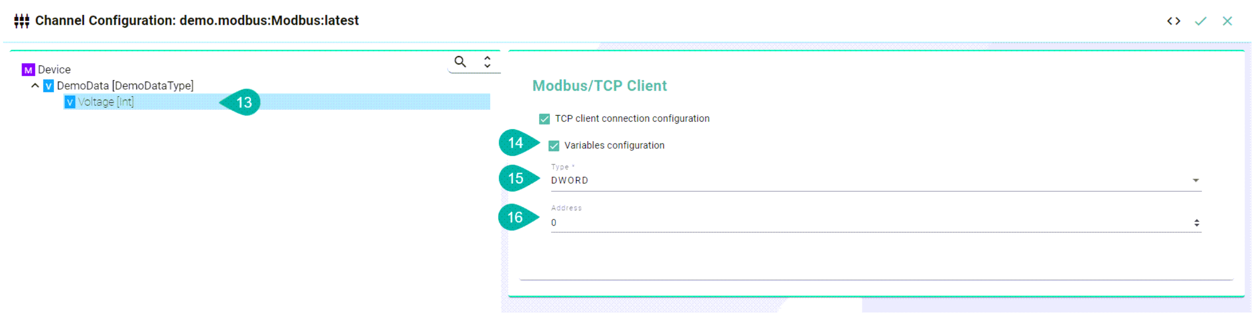
Max update interval

Length *: 500

Unit *: Milliseconds

13. Select the complex variable node
14. Enable the checkbox **Variables configuration**
15. Select the **Data type**
16. (Optional) Enter the **register address**

Note: If **address** is left empty, SMART**UNIFIER** assumes that the Information Model structure is in line with the register addresses.


Channel Configuration: demo.modbus:Modbus:latest

Device
DemoData [DemoDataType]
Voltage [Int]

Modbus/TCP Client

TCP client connection configuration

Variables configuration

Type *: DWORD

Address: 0

Description of data type format:

Data Type	Size	Range
BYTE, USINT, UInt8	8 Bit	`0 - 255`
WORD, UINT, UInt16	16 Bit	`0 - 65.535`
DWORD,UDINT, UInt32	32 Bit	`0 - 4.294.967.295`
LWORD,ULINT, UInt64	64 Bit	`0 - 2^64-1`
SINT, Int8	8 Bit	`-128 - 127`
INT, Int16	16 Bit	`-32.768 - 32.767`
DINT, Int32	32 Bit	`-2.147.483.648 - 2.147.483.647`
LINT, Int64	64 Bit	`-2^63 - 2^63-1`
REAL, Float32	32 Bit	`-3,402823e+38 - 3,402823e+38`
LREAL, Float64	64 Bit	`-1,7976931348623158e+308 - 1,7976931348623158e+308`

Description of configuration properties:

Property	Description	Example
IP	Client IP	`localhost`
Port	Client port	`502`
Connection timeout	Time interval the connection times out	`60`
Reconnect interval	Time interval to reconnect to the client after loss of connection	`5`
Receive interval	TCP/IP receive timeout	`50`
Autorefresh	Automatic polling of Modbus server	`2`
Read function code	Function code used for reading variables from a modbus server	`FC04`
Max update interval	Minimum time between requests to the Modbus server (if autorefresh is not used)	`60`
Variable configuration Type	Format of variable	`DWORD`
Variable configuration Address	Address of the variable on the modbus server	`0`

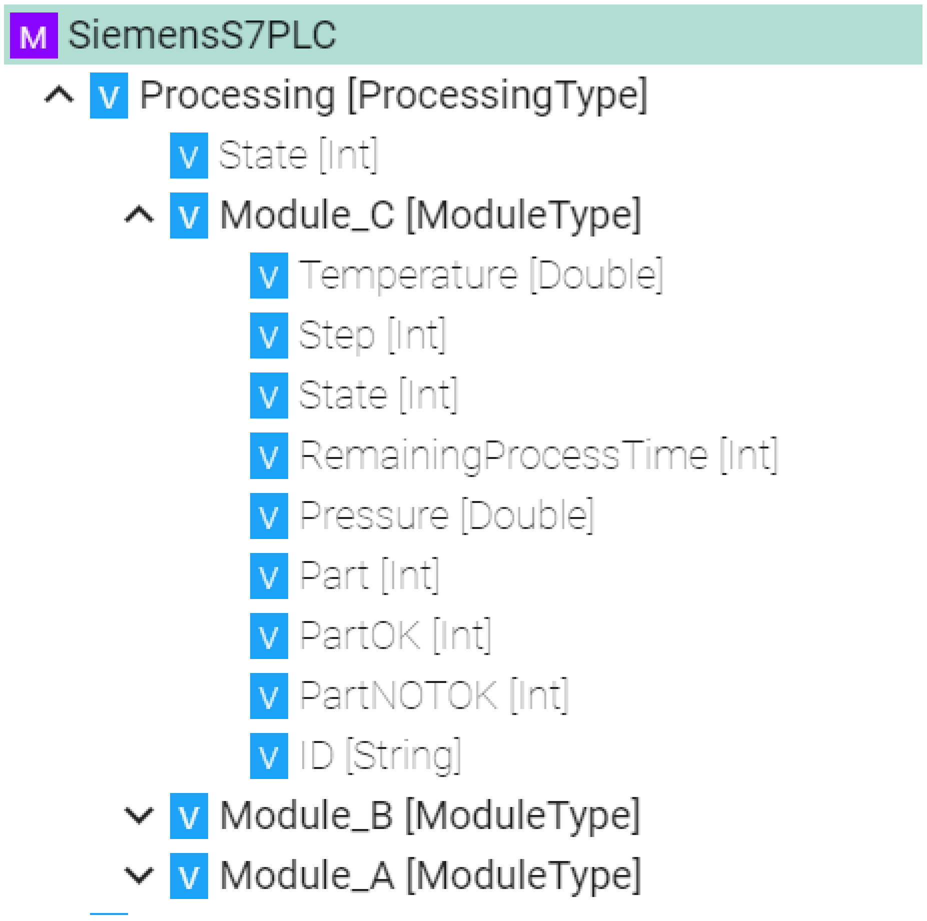
OPC-UA

Characteristics - OPC-UA

OPC (Open Platform Communications) enables access to machines, devices and other systems in a standardized way. To learn more about the standard visit the OPC-UA website.

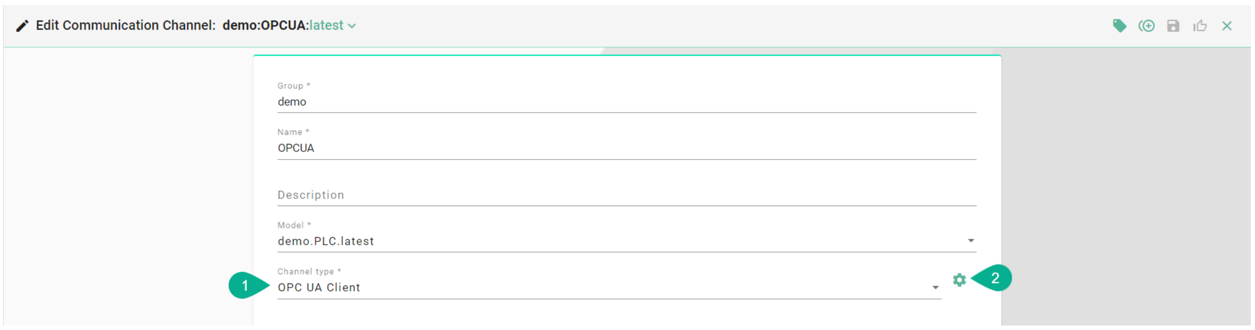
Information Model Requirements

- The following Node Types can be used to model data structures:
  - *Variables* V with a *Simple Data Type*.
  - *Variables* V with a *Custom Data Type*.



Configuration - OPC-UA Client

1. Select **OPC UA Client** as Channel Type.
2. Click the **Configure** button.



Group *
demo

Name *
OPCUA

Description

Model *
demo.PLC.latest

Channel type *
OPC UA Client

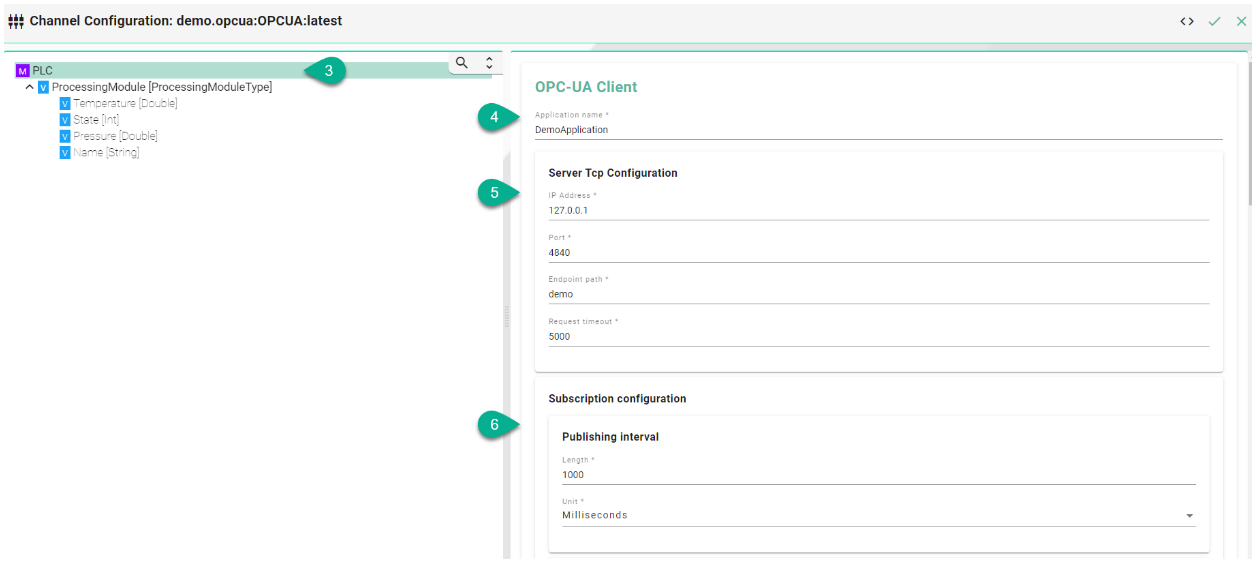
3. **Make sure** the root model node is selected to configure the OPC-UA Client
4. Enter an **Application name**
5. Input the TcpConfiguration

- Enter an **IP address**

- Enter the **Port**
- Define an **Endpoint**
- Set a **Request timeout**

6. Configure the defaultSubscriptionAttribute

- Define a **Publishing interval** and select the **Unit**


Channel Configuration: demo.opcua:OPCUA:latest
PLC
ProcessingModule [ProcessingModuleType]
Temperature [Double]
State [Int]
Pressure [Double]
Name [String]
3
4
5
6

OPC-UA Client

Application name *
DemoApplication

Server Tcp Configuration

IP Address *
127.0.0.1

Port *
4840

Endpoint path *
demo

Request timeout *
5000

Subscription configuration

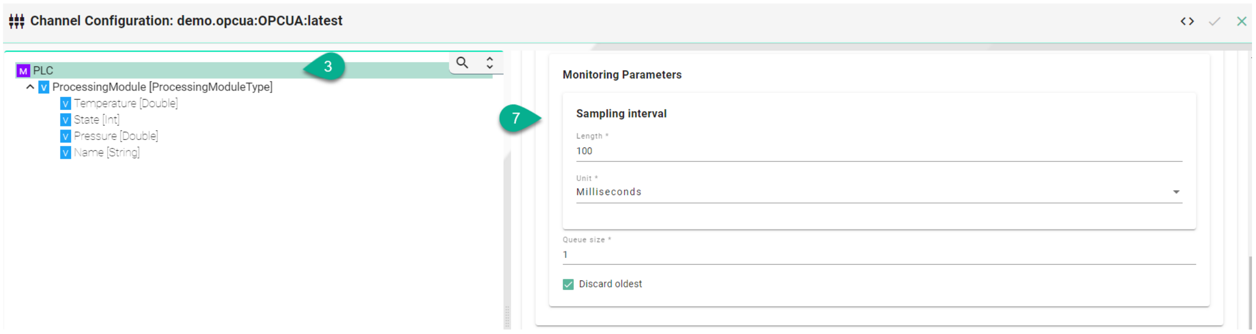
Publishing interval

Length *
1000

Unit *
Milliseconds

7. Configure monitoringParameters

- Set a **Sampling interval** and the **Unit**
- Enter a **Queue size**
- Enable **Discard oldest** depending on the use case


Channel Configuration: demo.opcua:OPCUA:latest
PLC
ProcessingModule [ProcessingModuleType]
Temperature [Double]
State [Int]
Pressure [Double]
Name [String]
3
7

Monitoring Parameters

Sampling interval

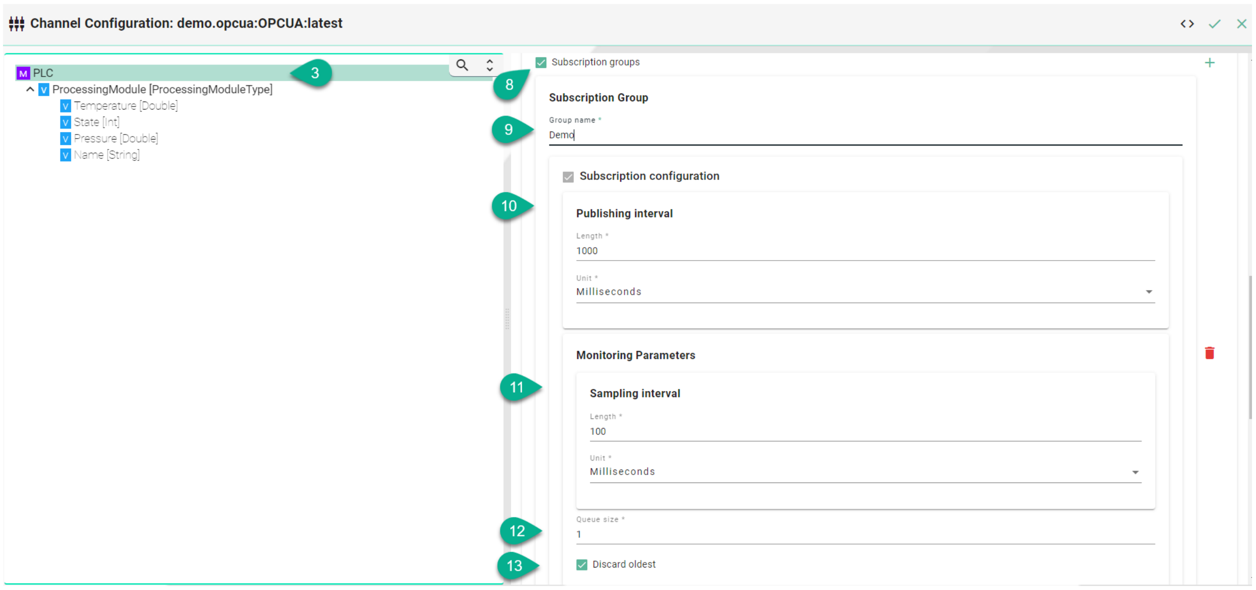
Length *
100

Unit *
Milliseconds

Queue size *
1

✓ Discard oldest

8. Enable **Subscription Groups** depending on the use case
9. Input the **Group name**
10. Define a **Publishing interval** and select the **Unit**
11. Set a **Sampling interval** and the **Unit**
12. Enter a **Queue size**
13. Enable **Discard oldest** depending on the use case


Channel Configuration: demo.opcua:OPCUA:latest
PLC
ProcessingModule [ProcessingModuleType]
Temperature [Double]
State [Int]
Pressure [Double]
Name [String]
3
8
9
10
11
12
13

Subscription groups

Subscription Group

Group name *
Demo

Subscription configuration

Publishing interval

Length *
1000

Unit *
Milliseconds

Monitoring Parameters

Sampling interval

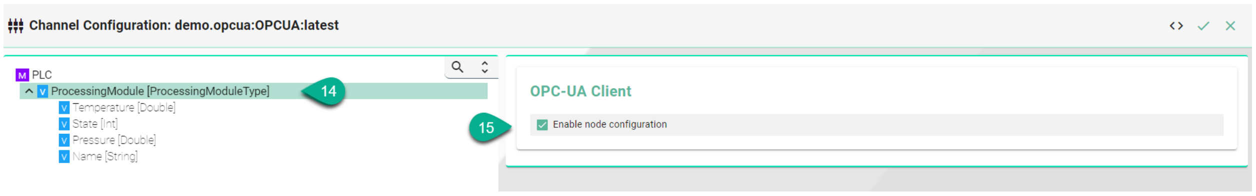
Length *
100

Unit *
Milliseconds

Queue size *
1

Discard oldest

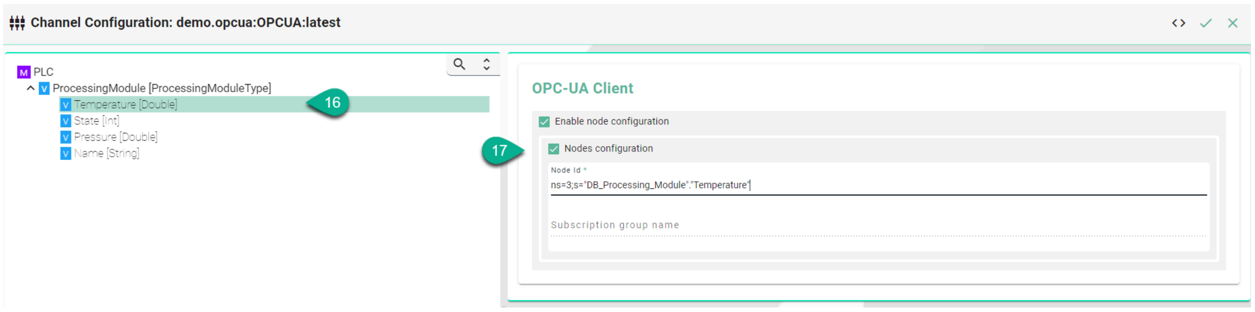
14. Select the complex variable node.
15. Enable the Node configuration


Channel Configuration: demo.opcua:OPCUA:latest
PLC
ProcessingModule [ProcessingModuleType]
Temperature [Double]
State [Int]
Pressure [Double]
Name [String]
14
15

OPC-UA Client

Enable node configuration

16. Assign OPC-UA data block variables to corresponding variables in the Information Model by selecting the variable in the tree
17. Assign data block
    - Enable the **Nodes configuration** checkbox
    - Enter the **Node Id**


Channel Configuration: demo.opcua:OPCUA:latest
PLC
ProcessingModule [ProcessingModuleType]
Temperature [Double]
State [Int]
Pressure [Double]
Name [String]
16
17

OPC-UA Client

Enable node configuration

Nodes configuration

Node Id *
ns=3;s="DB_Processing_Module"."Temperature"

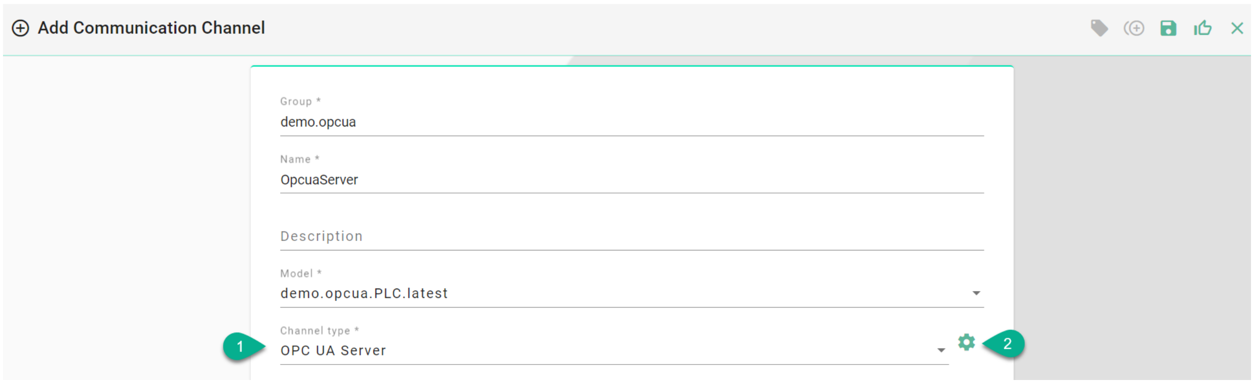
Subscription group name

Description of configuration properties:

Property	Description	Example
IP Address	Client IP	`127.0.0.1`
Port	Client port	`4840`
Endpoint path	Service name at the server endpoint	`demo`
Publishing interval	Interval in which Notification Messages are sent	`1`
Sampling interval	Sampling interval of monitored items	`10`
Queue size	Max number of messages stored in the publish queue	`1`
Node id	Id of the item	`s='DB_Processing_Module'`

Configuration - OPC-UA Server

1. Select **OPC UA Server** as Channel Type.
2. Click the **Configure** button.


Add Communication Channel

Group * demo.opcua

Name * OpcuaServer

Description

Model * demo.opcua.PLC.latest

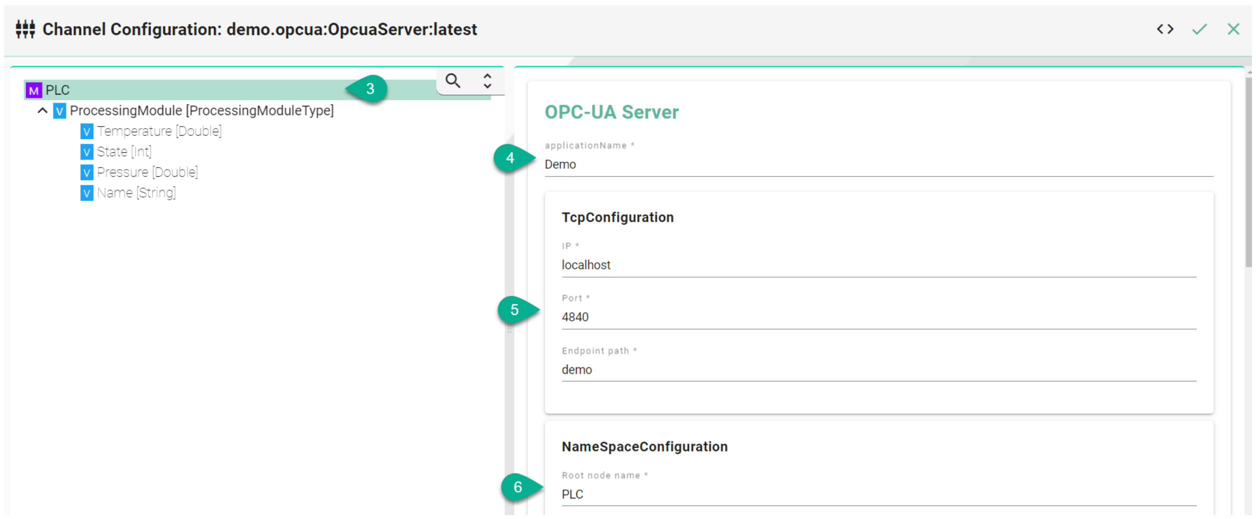
Channel type * OPC UA Server

3. Make sure the root model node is selected to configure the OPC-UA Server
4. Enter an **application Name**
5. Configure TCP

- Enter an **Ip Address**
- Enter the **Port**
- Define an **Endpoint**

6. Configure the **NameSpace**

- Provide a **Root node name**


Channel Configuration: demo.opcua:OpcuaServer:latest

PLC
ProcessingModule [ProcessingModuleType]
Temperature [Double]
State [Int]
Pressure [Double]
Name [String]

OPC-UA Server

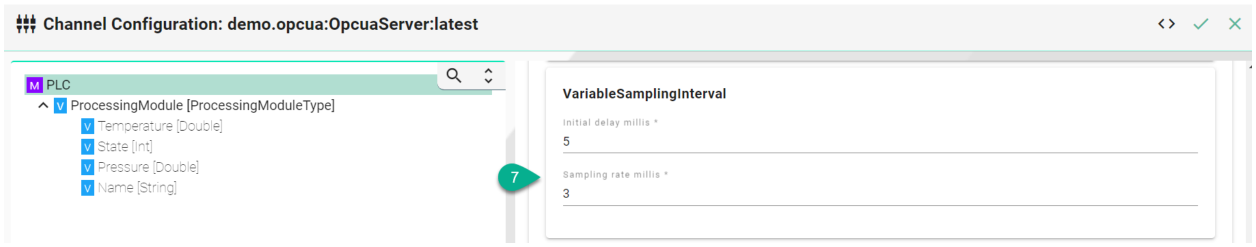
applicationName *: Demo

TcpConfiguration
IP *: localhost
Port *: 4840
Endpoint path *: demo

NameSpaceConfiguration
Root node name *: PLC

7. Configure the variable sampling interval

- Set the **Initial delay** in milliseconds
- Input the **Sampling rate** in milliseconds


Channel Configuration: demo.opcua:OpcuaServer:latest

PLC
ProcessingModule [ProcessingModuleType]
Temperature [Double]
State [Int]
Pressure [Double]
Name [String]

VariableSamplingInterval
Initial delay millis *: 5
Sampling rate millis *: 3

Description of configuration properties:

Property	Description	Example
IP Address	Server IP	`127.0.0.1`
Port	Server port	`4840`
Endpoint path	Service name at the server endpoint	`demo`
Root node name	The name of the top-level node in the OPC UA server's address space	`PLC`
Initial delay	The time the OPC UA client/server should wait before attempting to connect to the OPC UA server	`10`
Sampling rate	The rate at which the OPC UA client/server requests data from the OPC UA server	`1`

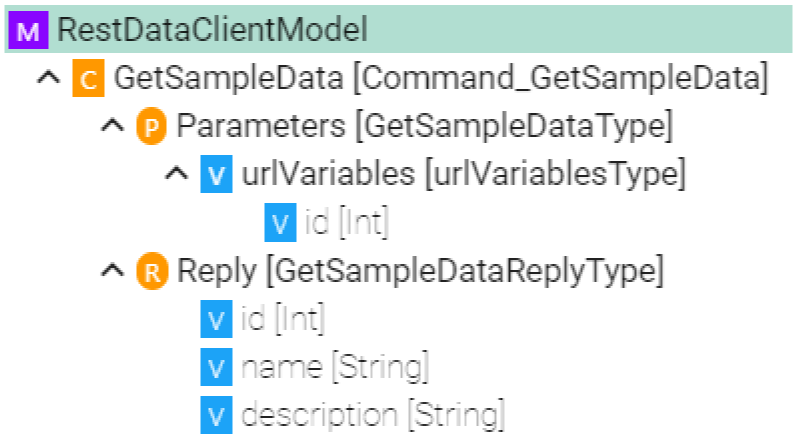
REST

Characteristics - REST

Representational state transfer (REST) is a software architectural style that describes a uniform interface between decoupled components in the Internet in a Client-Server architecture. To learn more about the standard visit the REST section in Wikipedia website.

Information Model Requirements

- The first Node after the root node M can be of type *Event* E, *Command* C or *Variable* V
- The following Node Types can be used under the Event Node:
  - *Variables* V with a *Simple Data Type* represents the key-value pairs.
  - *Variables* V with a *Custom Data Type* represent objects that can contain key-value pairs.


M RestDataClientModel
C GetSampleData [Command_GetSampleData]
P Parameters [GetSampleDataType]
V urlVariables [urlVariablesType]
V id [Int]
R Reply [GetSampleDataReplyType]
V id [Int]
V name [String]
V description [String]

Configuration - REST Server

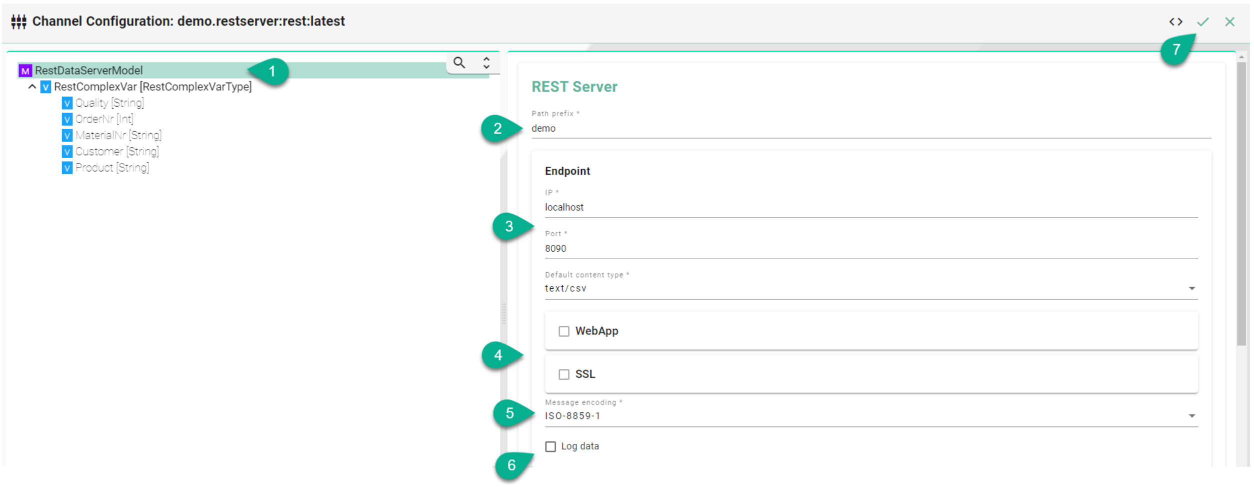
The following sample configuration shows how variables can be made accessible over a REST server.

1. Select the **root model node** in the tree on the left.
2. Enter a **path prefix**.
3. Configure the *REST Server* endpoint.

- Enter the **IP**.
- Enter the **port**.
- Enter the **Content-Type**.

4. Check the **webapp** checkbox and provide the **WAR-file** if you want to host an application.
5. Select the **Message encoding** standard.
6. Check the box to **Log data** (E.g., the body of a request).

7. Click the **Apply** button and save the Channel by clicking the **Save** button on the upper right corner.



Channel Configuration: demo.restserver:rest:latest

RestDataServerModel
- RestComplexVar [RestComplexVarType]
  - Quality [String]
  - OrderNr [Int]
  - MaterialNr [String]
  - Customer [String]
  - Product [String]

REST Server

Path prefix *: demo

Endpoint

IP *: localhost

Port *: 8090

Default content type *: text/csv

☐ WebApp

☐ SSL

Message encoding *: ISO-8859-1

☐ Log data

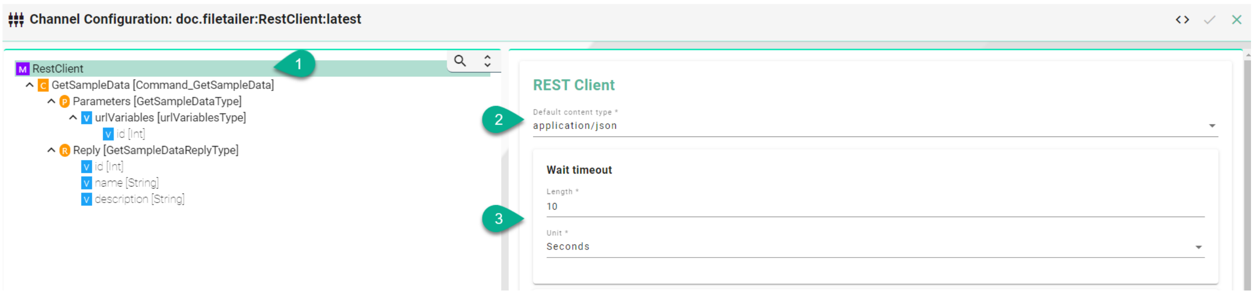
Description of configuration properties:

Property	Description	Example
pathPrefix	Prefix for the URL	`e.g., demo`
Port	Port of the REST server	`e.g., 9002, 9000, ...`
IP	IP address of the REST server	`http://localhost`
DefaultContentType	Is used to indicate the media type of the resource	`application/json`, `application/xml`, `text/html`, `text/csv`
webapp	Possibility to host an application	`true`, `false`
Message encoding	Encoding standard for messages	`ISO-8859-1`, `UTF-16`

Configuration - REST Client

The following sample configuration shows a GET request using url parameters.

1. Select the **root model node** in the tree on the left
2. Select the **content type** - defines the media type of the associated representation
3. Set the **wait timeout**



Channel Configuration: doc.filetailer:RestClient:latest

RestClient
- GetSampleData [Command_GetSampleData]
  - Parameters [GetSampleDataType]
    - urlVariables [urlVariablesType]
      - id [Int]
  - Reply [GetSampleDataReplyType]
    - id [Int]
    - name [String]
    - description [String]

REST Client

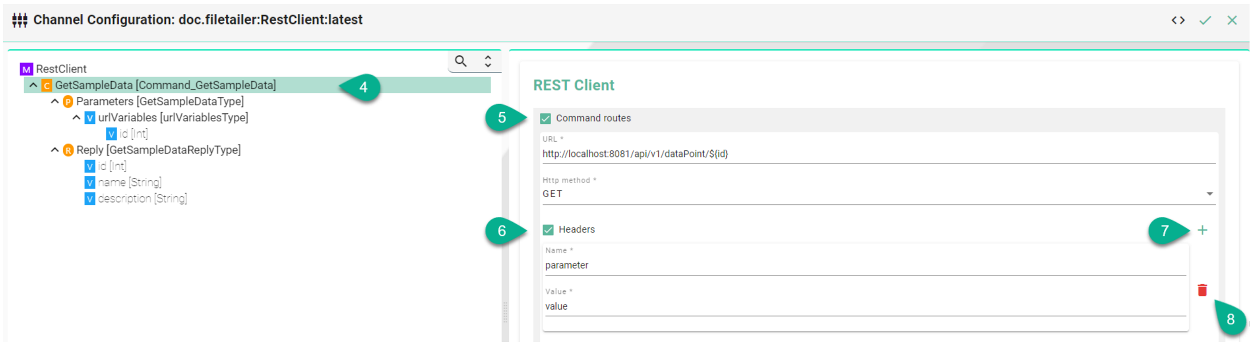
Default content type *: application/json

Wait timeout

Length *: 10

Unit *: Seconds

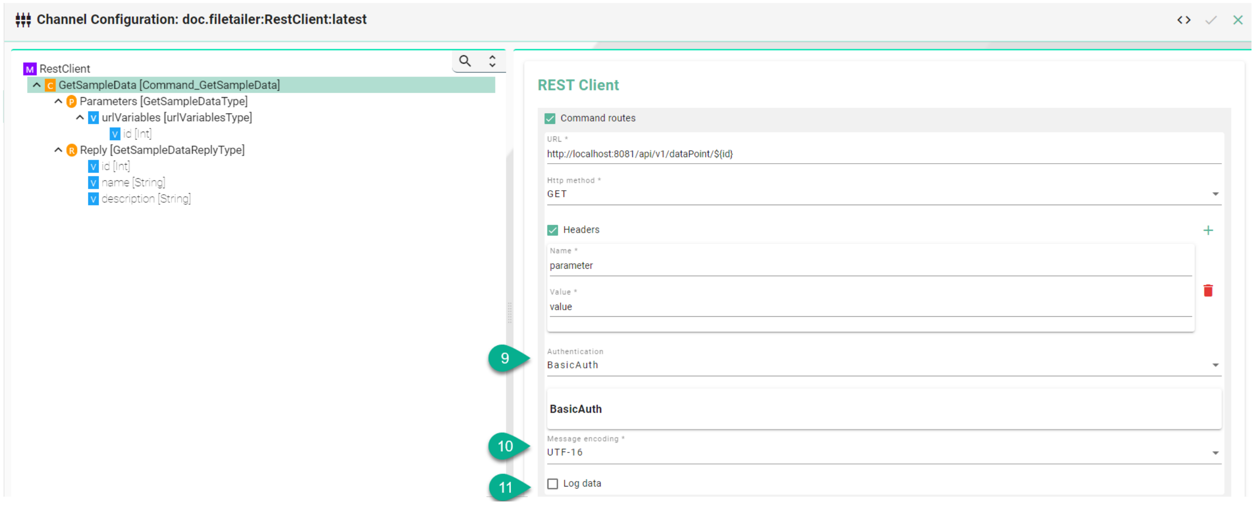
4. Select the **Command** node
5. Enable the **Command routes** checkbox for the configuration of the following fields:
   - Enter the **URL** - If URL parameters are used then add each parameter in the following syntax `${id}`
   - Select the **HTTP method**.
6. Headers (Optional) - Enable the checkbox **Headers** for the configuration of the following fields:
   - Enter the name of the header
   - Enter the value
7. Add multiple header entries by clicking the **Add** button
8. Delete a header by clicking the **Delete** button


Channel Configuration: doc.filetailer:RestClient:latest

RestClient
GetSampleData [Command_GetSampleData]
Parameters [GetSampleDataType]
urlVariables [urlVariablesType]
id [Int]
Reply [GetSampleDataReplyType]
id [Int]
name [String]
description [String]

REST Client
Command routes
URL *: http://localhost:8081/api/v1/dataPoint/${id}
Http method *: GET
Headers
Name *: parameter
Value *: value

9. Select the **Authentication** type
10. Select the **Message encoding** standard
11. Check the box to **Log data** (E.g., the body of a request).

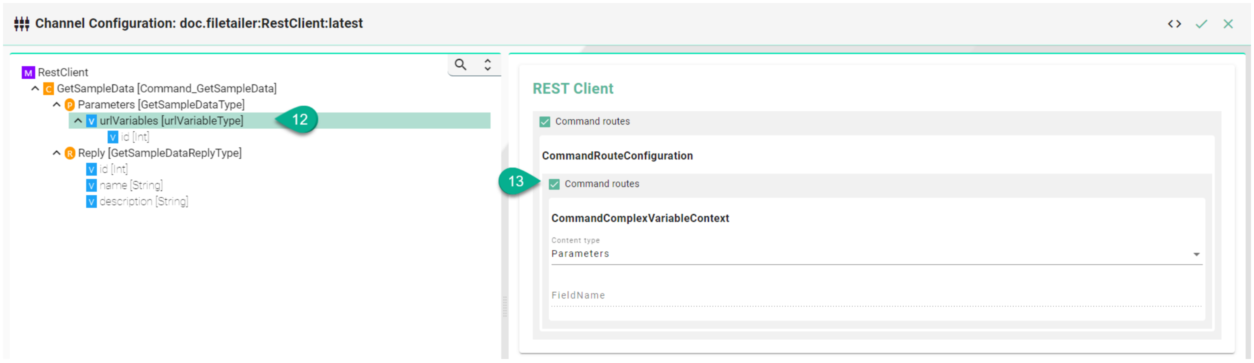

Channel Configuration: doc.filetailer:RestClient:latest

RestClient
GetSampleData [Command_GetSampleData]
Parameters [GetSampleDataType]
urlVariables [urlVariablesType]
id [Int]
Reply [GetSampleDataReplyType]
id [Int]
name [String]
description [String]

REST Client
Command routes
URL *: http://localhost:8081/api/v1/dataPoint/${id}
Http method *: GET
Headers
Name *: parameter
Value *: value
Authentication: BasicAuth
BasicAuth
Message encoding *: UTF-16
Log data

12. URL Parameters (Optional) - Select a custom variable node
13. Enable the **Command routes** for the configuration of the following fields:

- Select the **Content Type**
- (Optional) Enter a **Field Name** in case the *Information Model Node* is not matching the REST API


Channel Configuration: doc.filetailer:RestClient:latest

RestClient
GetSampleData [Command_GetSampleData]
Parameters [GetSampleDataType]
urlVariables [urlVariableType]
id [Int]
Reply [GetSampleDataReplyType]
id [Int]
name [String]
description [String]

REST Client
✓ Command routes
CommandRouteConfiguration
✓ Command routes
CommandComplexVariableContext
Content type: Parameters
FieldName

Description of configuration properties:

Property	Description	Example
URL	URL of the REST API.	`http://localhost:8081/api/v1/dataPoint/${id}`
HttpMethod	HTTP method for the action performed by the Client.	`GET`, `POST`, `PUT`
HeaderName and Header Value	To provide server and client with additional information	`Retry-After: 12`
Default Content Type	Is used to indicate the media type of the resource.	`application/json`
RouteHeaderConfiguration	Headers represent the meta-data associated with the API request	`Name, Value`
Authentication Type	Type of the Authentication	`Basic`, `Digest`, `Kerberos`, `NTLM`, `SPNEGO`
Content Type of Parameter Nodes	Type of the Parameter	`Parameters`, `Body`, `Header`, `None`
Field Name	For non-matching Information Model nodes and API spelling	`String`
WaitTimeoutDuration	Timeout in seconds until request is failing	`10`
Message encoding	Encoding standard for messages	`ISO-8859-1`, `UTF-16`

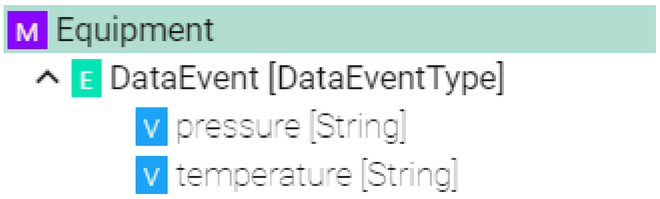
SECS/GEM

Characteristics - SECS/GEM

The SECS/GEM is the semiconductor's equipment interface protocol for equipment-to-host data communications. In an automated fab, the interface can start and stop equipment processing, collect measurement data, change variables and select recipes for products. To learn more about the standard visit the SECS/GEM section in Wikipedia website.

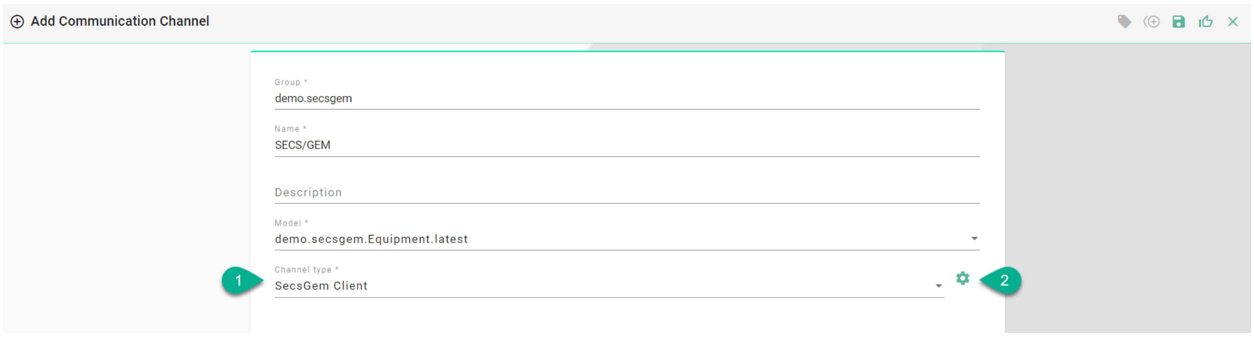
Information Model Requirements

- The first Node after the root node M can be of type *Event* E, *Command* C or *Variable* V
- The following Node Types can be used under the Event Node:
  - *Variables* V with a *Simple Data Type* represents the key-value pairs.
  - *Variables* V with a *Custom Data Type* represent objects that can contain key-value pairs.


M Equipment
E DataEvent [DataEventType]
V pressure [String]
V temperature [String]

Configuration - SECS/GEM Client

1. Select **Secs Gem Client** as Channel Type.
2. Click the **Configure** button.


Add Communication Channel

Group *
demo.secsgem

Name *
SECS/GEM

Description

Model *
demo.secsgem.Equipment.latest

Channel type *
SecsGem Client

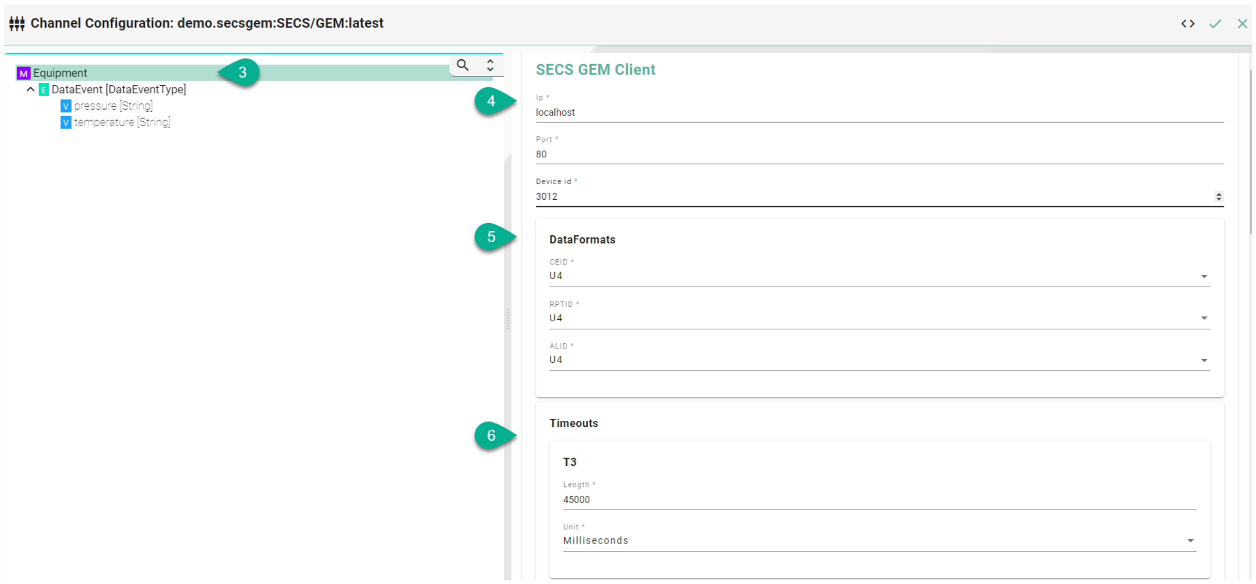
3. **Make sure** the root model node is selected to configure the SECS/GEM Client
4. Enter the device configuration:

- input the equipment-to-host **Ip** address
- type in the TCP **Port** for the communication

- input the **Device Id**

5. Enter the **Data Formats**

- Input **CEID** - format for event Ids
- Enter **RPTID** - format for report Ids
- Input **ALID** - format for alarm Ids



Channel Configuration: demo.secsgem:SECS/GEM:latest

Equipment
DataEvent [DataEventType]
pressure [String]
temperature [String]

SECS GEM Client

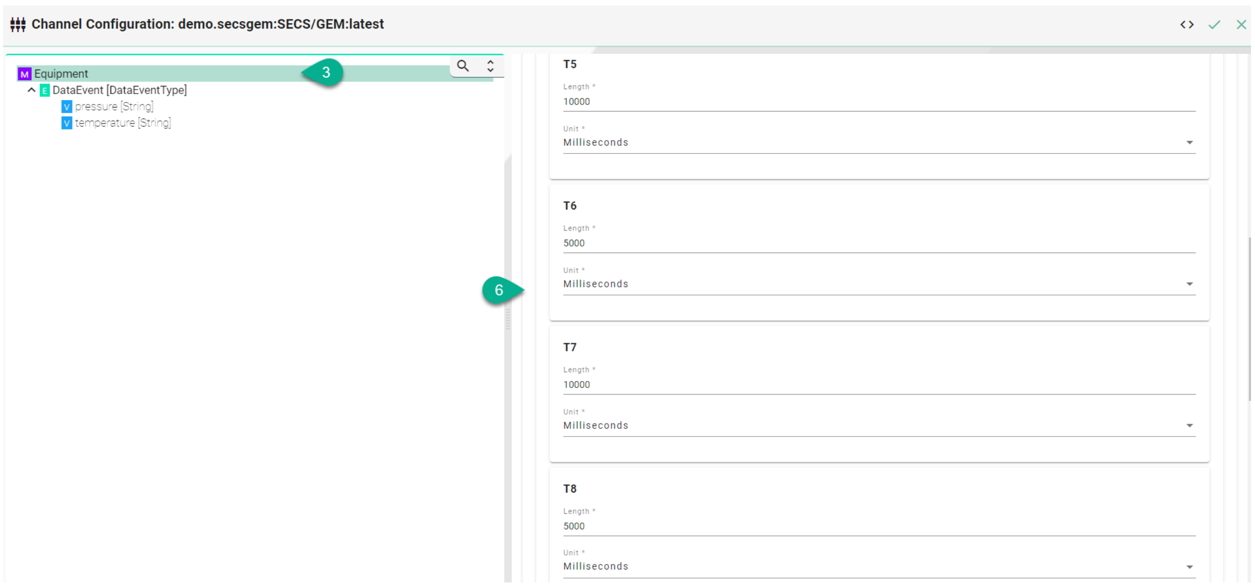
Ip * localhost
Port * 80
Device id * 3012

DataFormats
CEID * U4
RPTID * U4
ALID * U4

Timeouts
T3
Length * 45000
Unit * Milliseconds

6. Input timeout for:

- **T3** - Reply Timeout in the HSMS protocol.
- **T5** - Connect Separation Timeout in the HSMS protocol used to prevent excessive TCP/IP connect activity by providing a minimum time between the breaking, by an entity, of a TCP/IP connection or a failed attempt to establish one, and the attempt, by that same entity, to initiate a new TCP/IP connection.
- **T6** - Control Timeout in the HSMS protocol which defines the maximum time an HSMS control transaction can remain open before a communications failure is considered to have occurred. A transaction is considered open from the time the initiator sends the required request message until the response message is received.
- **T7** - Connection Idle Timeout in the HSMS protocol which defines the maximum amount of time which may transpire between the formation of a TCP/IP connection and the use of that connection for HSMS communications before a communications failure is considered to have occurred.
- **T8** - Network Intercharacter Timeout in the HSMS protocol which defines the maximum amount of time which may transpire between the receipt of any two successive bytes of a complete HSMS message before a communications failure is considered to have occurred.


Channel Configuration: demo.secsgem:SECS/GEM:latest
M Equipment
E DataEvent [DataEventType]
V pressure [String]
V temperature [String]
3
6

T5
Length *: 10000
Unit *: Milliseconds

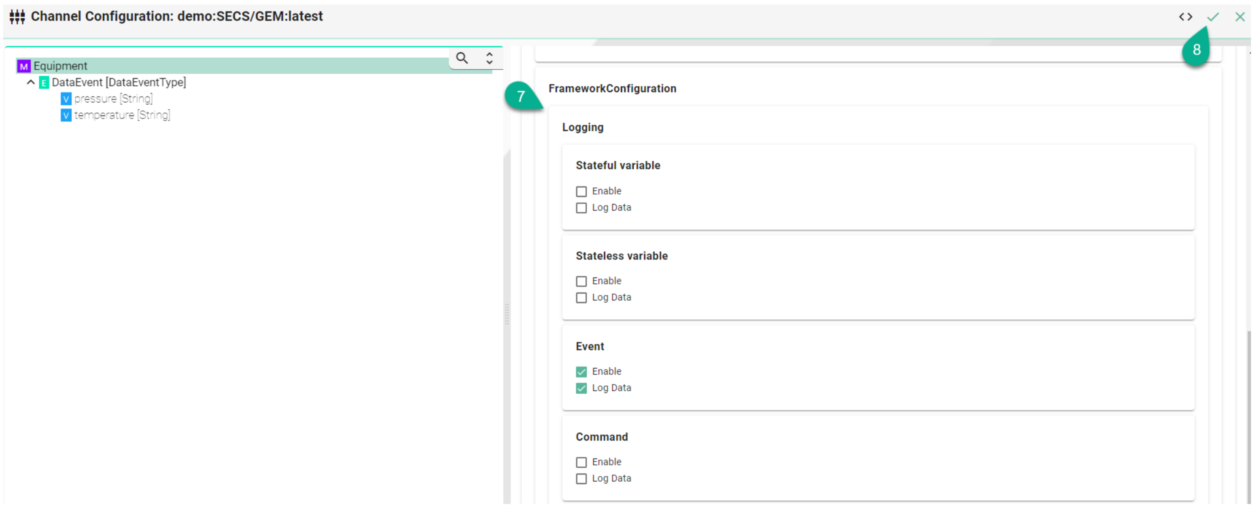
T6
Length *: 5000
Unit *: Milliseconds

T7
Length *: 10000
Unit *: Milliseconds

T8
Length *: 5000
Unit *: Milliseconds

7. Select the logging type for the required Node Types:

- Check the **Enable** box
- Check the **Log Data** box


Channel Configuration: demo:SECS/GEM:latest
M Equipment
E DataEvent [DataEventType]
V pressure [String]
V temperature [String]
7
8

FrameworkConfiguration

Logging

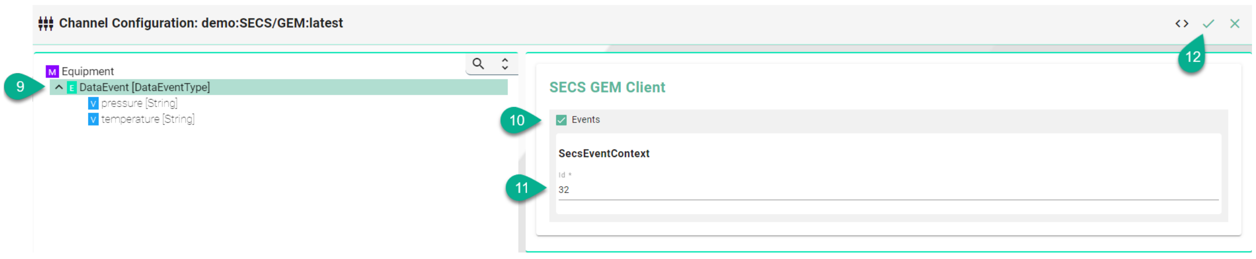
Stateful variable
- [ ] Enable
- [ ] Log Data

Stateless variable
- [ ] Enable
- [ ] Log Data

Event
- [x] Enable
- [x] Log Data

Command
- [ ] Enable
- [ ] Log Data

8. Click on the **Apply** button
9. Select the **Event** node to configure the event context

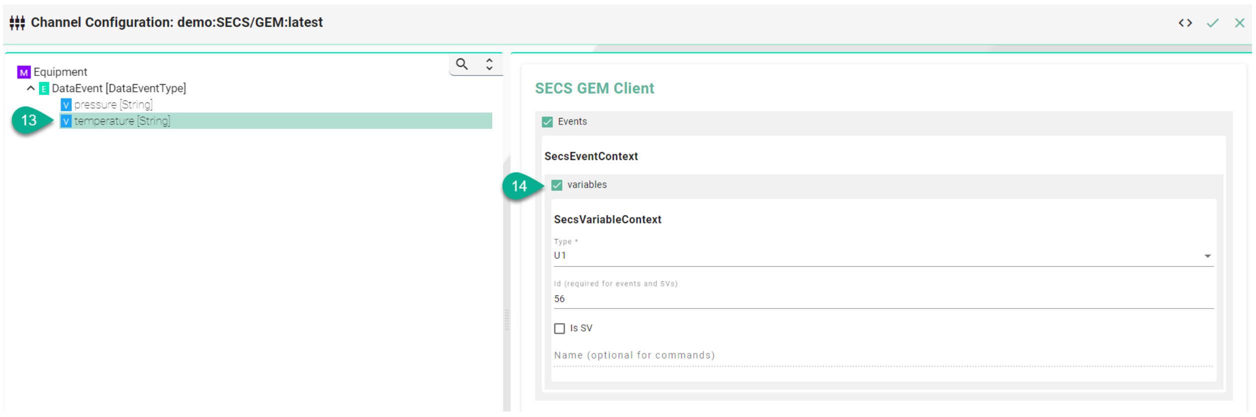

Channel Configuration: demo:SECS/GEM:latest
M Equipment
E DataEvent [DataEventType]
V pressure [String]
V temperature [String]
9
10
11
12

SECS GEM Client
- [x] Events

SecsEventContext
Id *: 32

10. Click to check the **Events** box

11. Enter the event context **Id** which will trigger the event in the Information Model
12. Click on the **Apply** button
13. Select the variable in the tree



SecsEventContext

SecsVariableContext

Type *: U1

Id (required for events and SVs): 56

☐ Is SV

Name (optional for commands)

14. Click to check the **variables** box and configure the Secs variable context
    - select the variable **Type**
    - enter the variable **Id**
    - click the **Is SV** box to check if the variable is a SV
    - input the variable **Name**

Description of configuration properties:

Property	Description	Example
Ip	IP address of the Equipment	`http://localhost`
Port	TCP port for the communication	`5000`
Device Id	Id of the equipment	`NJ-300`
CEID	Format for event Ids	`U4`
RPTID	Format for report Ids	`U4`
ALID	Format for alarm Ids	`U4`
Timeouts	Time interval the connection times out in milliseconds	`45000`
T3	Reply timeout in the HSMS protocol	`10000`
T5	Connect Separation Timeout in the HSMS protocol	`5000`
T6	Control Timeout in the HSMS protocol	`10000`
T7	Connection Idle Timeout in the HSMS protocol	`5000`
T8	Network Intercharacter Timeout in the HSMS protocol	`10000`
Id	Id of the equipment event which will trigger the event	`E32`
Type	Type of variable	`U1`
Id	Variable Id	`V56`
Type	Commands - Type of the message	`S2F41`
Id	Commands Id	`C33`
RCMD	Name of command if it is different from the command Id	`C1`

Email

Characteristics - Email

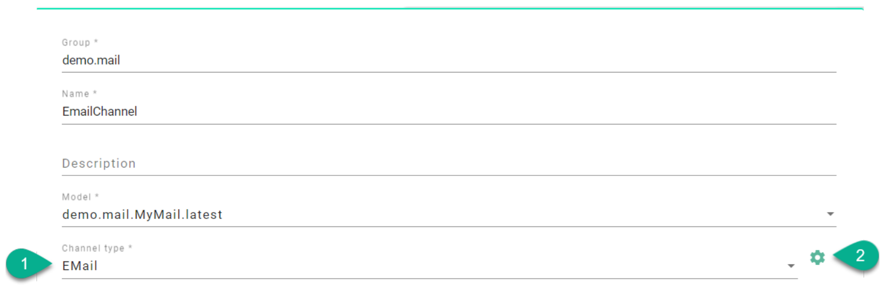
SMART**UNIFIER** provides the capability of integrating the email protocols. The email protocols define the mechanism of the email exchange between servers and clients. An email protocol is a group of rules which ensure that emails are properly transmitted over the Internet.

Information Model Requirements

- The following Node Types can be used to model a register:
  - *Variables* V with a *Simple Data Type*.
  - *Variables* V with a *Custom Data Type*.

Configuration - Email

1. Select **EMail** as Channel Type.
2. Click the **Configure** button.



Group *
demo.mail

Name *
EmailChannel

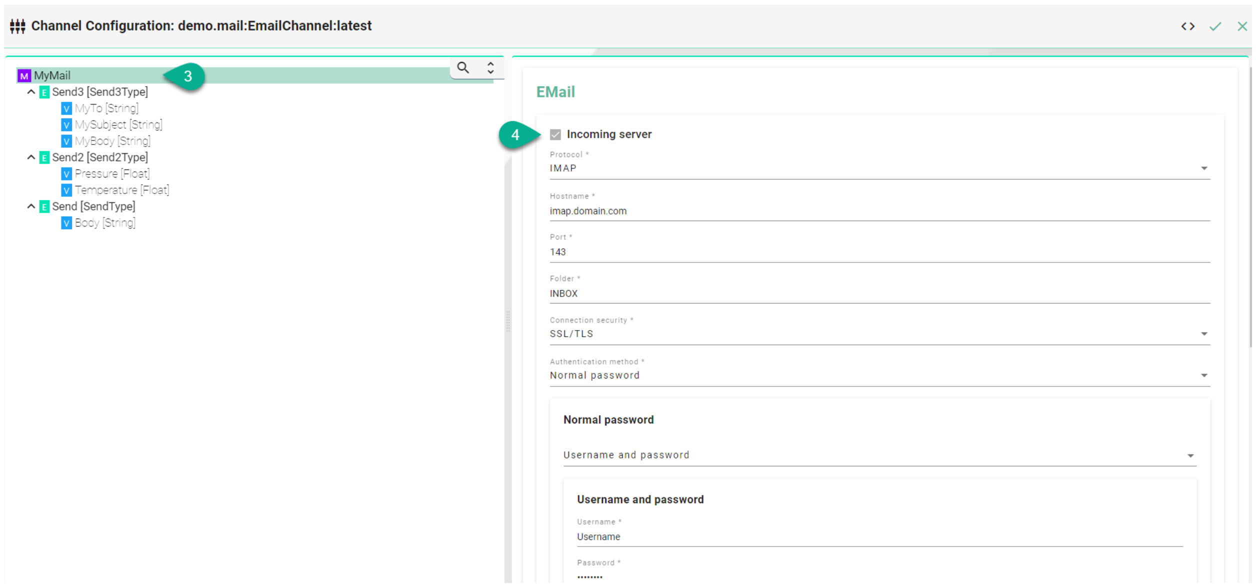
Description

Model *
demo.mail.MyMail.latest

Channel type *
EMail

3. Make sure the root model node is selected to configure the Email Client.
4. Enable **Incoming server** for configuration, based on the email provider:

- Select the **Protocol**
- Input the **Hostname**
- Provide the **Port**
- Input the **Folder** name
- Select the **Connection security**
- Choose the **Authentication method**
- Input credentials
- Configure the **Polling interval** for checking new emails
- Configure the **Timeout** length



Channel Configuration: demo.mail:EmailChannel:latest

MyMail
- Send3 [Send3Type]
  - MyTo [String]
  - MySubject [String]
  - MyBody [String]
- Send2 [Send2Type]
  - Pressure [Float]
  - Temperature [Float]
- Send [SendType]
  - Body [String]

EMail

☑ Incoming server

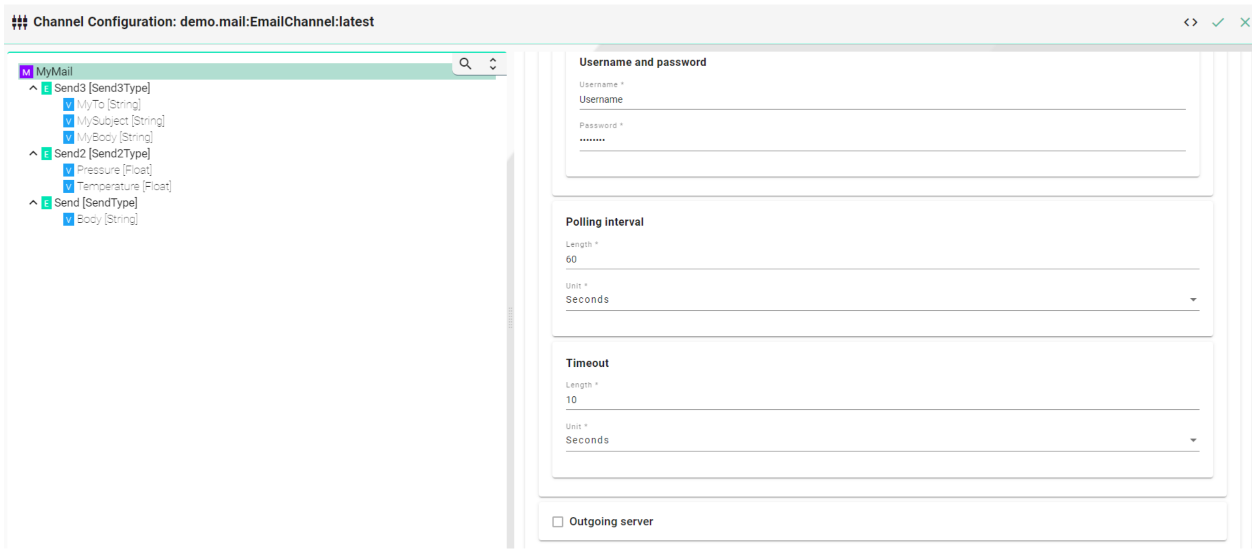
Protocol *: IMAP
Hostname *: imap.domain.com
Port *: 143
Folder *: INBOX
Connection security *: SSL/TLS
Authentication method *: Normal password

Normal password

Username and password

Username and password

Username *: Username
Password *: ········



Channel Configuration: demo.mail:EmailChannel:latest

MyMail
- Send3 [Send3Type]
  - MyTo [String]
  - MySubject [String]
  - MyBody [String]
- Send2 [Send2Type]
  - Pressure [Float]
  - Temperature [Float]
- Send [SendType]
  - Body [String]

Username and password

Username *: Username
Password *: ········

Polling interval

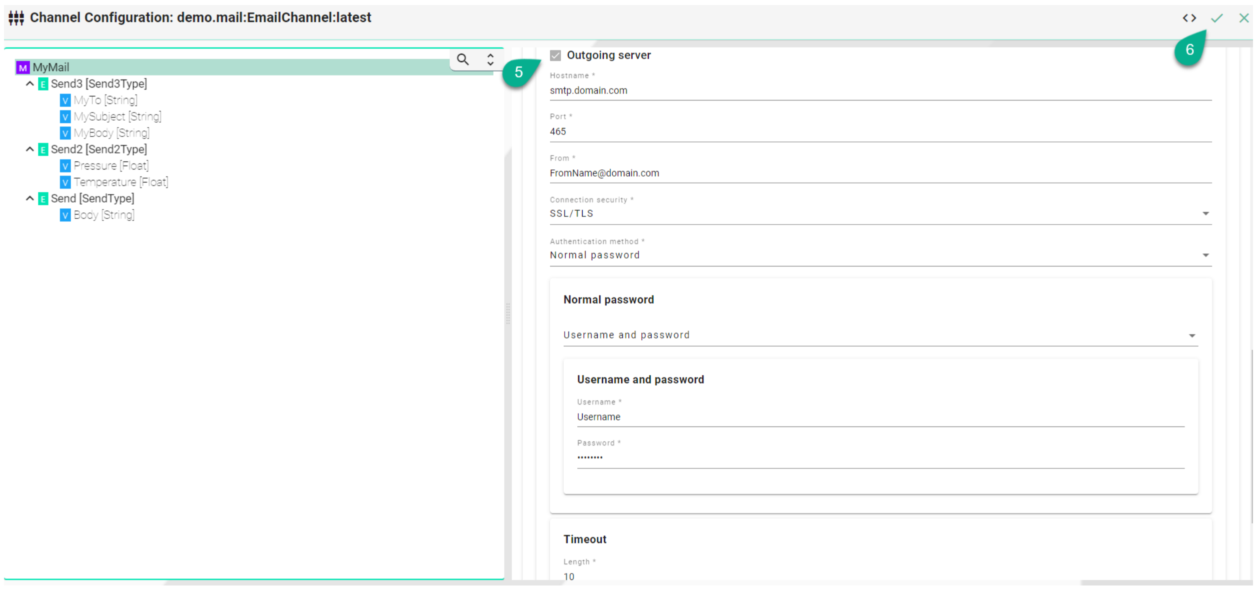
Length *: 60
Unit *: Seconds

Timeout

Length *: 10
Unit *: Seconds

☐ Outgoing server

5. Enable **Outgoing server** for configuration, based on the email provider:
   - Input the **Hostname**
   - Provide the **Port**
   - Input the **From** hostname
   - Select the **Connection security**
   - Choose the **Authentication method**
   - Input credentials
   - Configure the **Polling interval** for checking new emails
   - Configure the **Timeout** length
6. Click on the **Apply** button.


Channel Configuration: demo.mail:EmailChannel:latest

Outgoing server

Hostname *: smtp.domain.com

Port *: 465

From *: FromName@domain.com

Connection security *: SSL/TLS

Authentication method *: Normal password

Normal password: Username and password

Username and password

Username *: Username

Password *: ••••••••

Timeout

Length *: 10

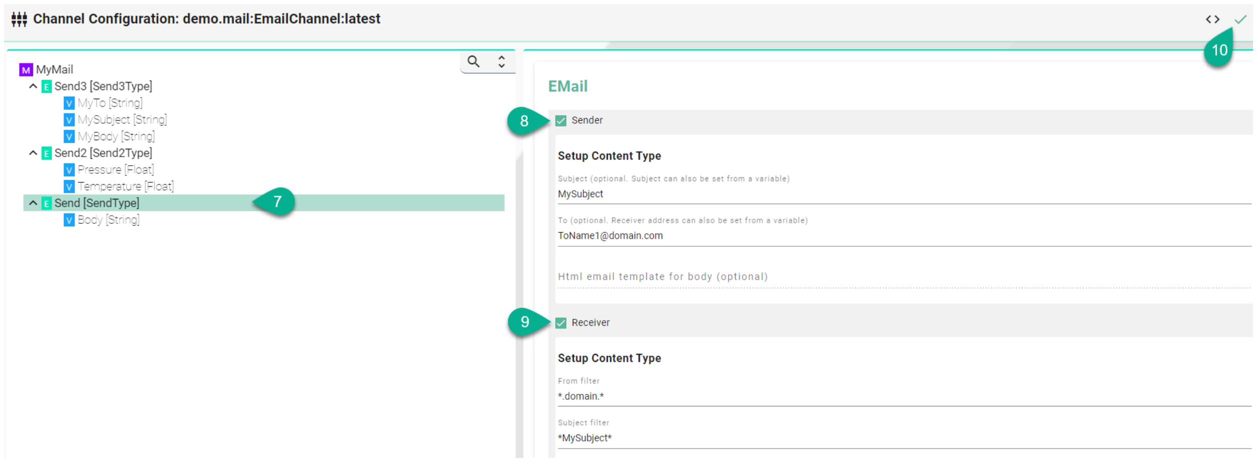
7. Select the Event node.
8. Enable **Sender** for configuration:

- Input the **Subject** if not using a **Subject** variable under the Event node
- Provide the **Receiver address** if not using a **To** variable under the Event node
- Input the **Html email template** (optional)

9. Enable **Receiver** for configuration:

- Input filter based on the Sender
- Provide filter based on the Subject

10. Click on the **Apply** button.


Channel Configuration: demo.mail:EmailChannel:latest

EMail

Sender

Setup Content Type

Subject (optional. Subject can also be set from a variable): MySubject

To (optional. Receiver address can also be set from a variable): ToName1@domain.com

Html email template for body (optional)

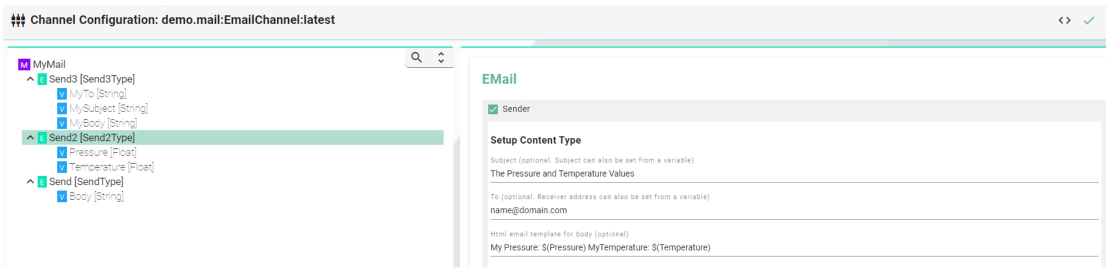
Receiver

Setup Content Type

From filter: *.domain.*

Subject filter: *MySubject*

- Example of sending the value of a Variable:



Channel Configuration: demo.mail:EmailChannel:latest

- MyMail
  - Send3 [Send3Type]
    - MyTo [String]
    - MySubject [String]
    - MyBody [String]
  - Send2 [Send2Type]
    - Pressure [Float]
    - Temperature [Float]
  - Send [SendType]
    - Body [String]

EMail

✓ Sender

Setup Content Type

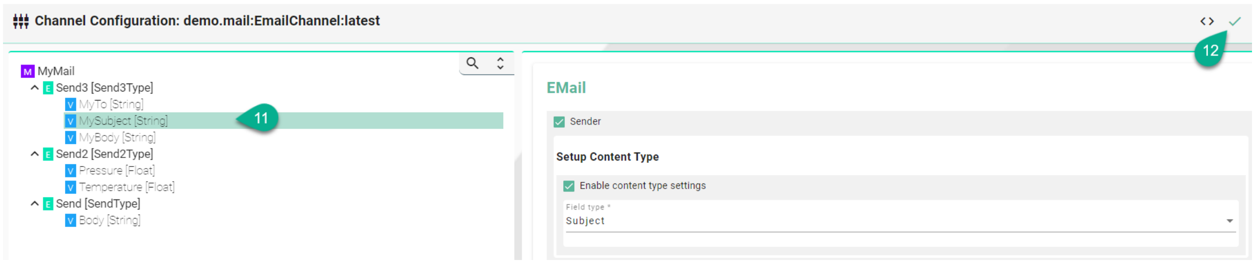
Subject (optional. Subject can also be set from a variable): The Pressure and Temperature Values

To (optional. Receiver address can also be set from a variable): name@domain.com

Html email template for body (optional): My Pressure: $(Pressure) MyTemperature: $(Temperature)

11. Select the Variable:

- If the variable is using a key name (To, From, Subject, Body) no additional configuration is needed
- Example of Variable used as **Subject**:



Channel Configuration: demo.mail:EmailChannel:latest

- MyMail
  - Send3 [Send3Type]
    - MyTo [String]
    - MySubject [String]
    - MyBody [String]
  - Send2 [Send2Type]
    - Pressure [Float]
    - Temperature [Float]
  - Send [SendType]
    - Body [String]

EMail

✓ Sender

Setup Content Type

✓ Enable content type settings

Field type *: Subject

12. Click on the **Apply** button to finish.

Description of data type format:

Data Type	Size	Range
BYTE, USINT, UInt8	8 Bit	`0 - 255`
WORD, UINT, UInt16	16 Bit	`0 - 65.535`
DWORD,UDINT, UInt32	32 Bit	`0 - 4.294.967.295`
LWORD,ULINT, UInt64	64 Bit	`0 - 2^64-1`
SINT, Int8	8 Bit	`-128 - 127`
INT, Int16	16 Bit	`-32.768 - 32.767`
DINT, Int32	32 Bit	`-2.147.483.648 - 2.147.483.647`
LINT, Int64	64 Bit	`-2^63 - 2^63-1`
REAL, Float32	32 Bit	`-3,402823e+38 - 3,402823e+38`
LREAL, Float64	64 Bit	`-1,7976931348623158e+308 - 1,7976931348623158e+308`

Description of configuration properties:

Property	Description	Example
Protocol	Incoming server protocol	`IMAP`
Incoming Hostname	Incoming server address	`imap.domain.com`
Port	Server port	`143`
Folder	Incoming emails folder	`INBOX`
Connection security	Communication security standards	`SSL/TLS`
Polling length	Automatic polling of Email server	`60`
Timeout length	Time interval the connection times out	`10`
Outgoing Hostname	Outgoing server address	`smtp.domain.com`
Outgoing From	Sender Email address	`name@domain.com`
Subject	Email subject	`MySubject`
To	Receiver address	`name@domain.com`
Html email template for body	HTML code for email body	`MyTemperature: $(temperature)`
From filter	Filter by sender using regex	`*.domain.*`
Subject filter	Filter by subject using regex	`*MySubject*`

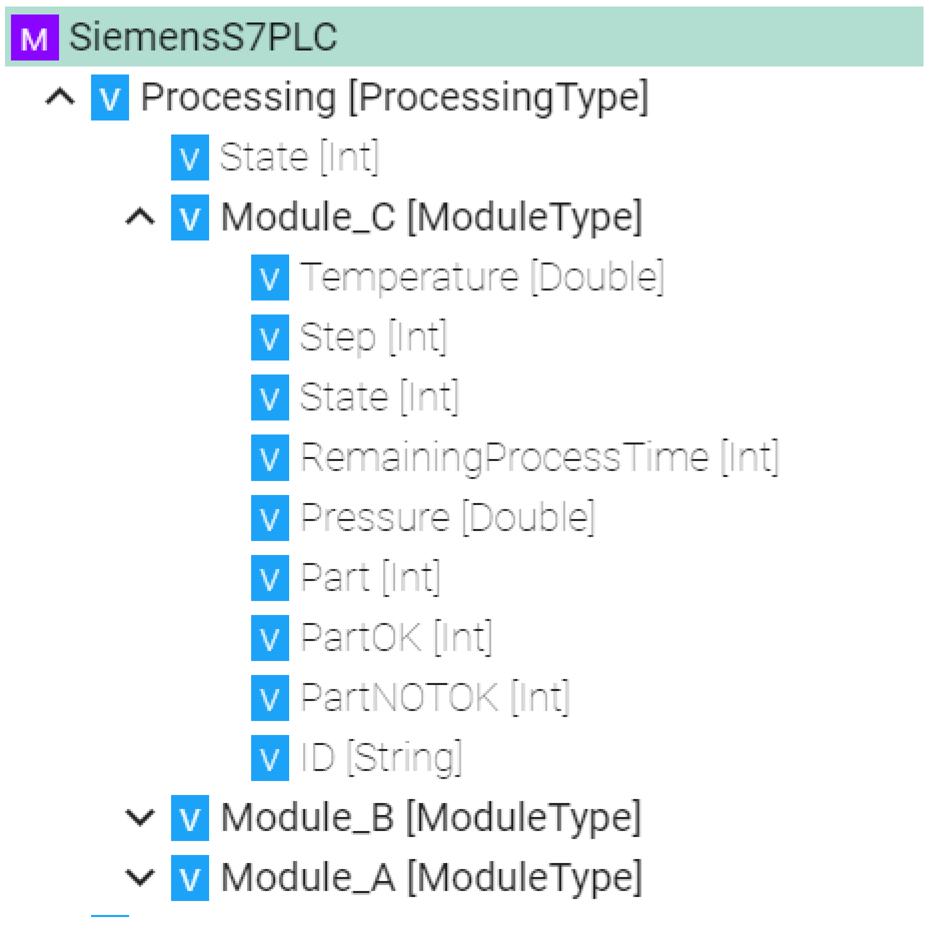
AWS SiteWise IoT

Characteristics - AWS IoT SiteWise

The AWS IoT SiteWise Channel enables you to send data directly to assets measurements via the AWS IoT SiteWise API.

Information Model Requirements

- The first Node after the root node M can be of type *Event* E or *Variable* V.
- The following Node Types can be used under the Event Node or Variable Node:
  - *Variables* V with a *Simple Data Type* represent measurements.
  - *Variables* V with a *Custom Data Type* represent asset models.
- The following measurement data types can be used when creating a variable of a *Simple Data Type*:
  - String
  - Int
  - Double
  - Boolean

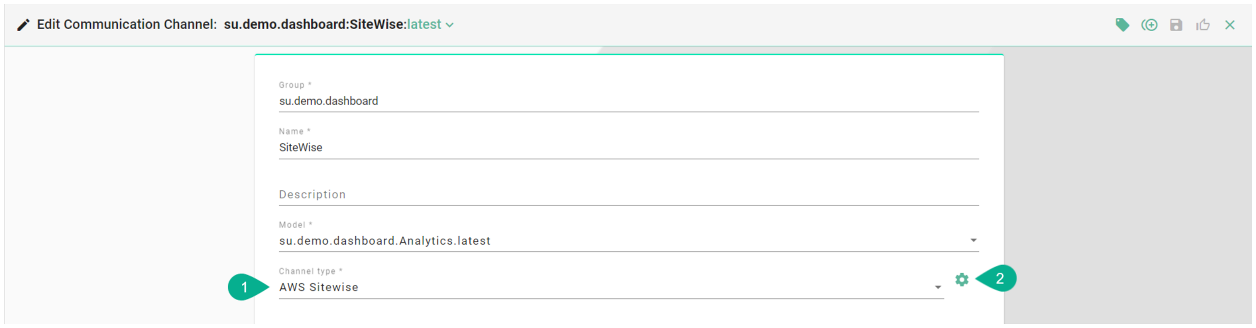


Note: Make sure that the Information Model is available in the AWS IoT SiteWise service. You can use the AWS SiteWise extension in order to export an SMARTUNIFIER Information Model to AWS IoT SiteWise.

Configuration - AWS IoT SiteWise

The following sample configuration shows how a AWS IoT SiteWise Channel is created.

1. Select **AWS Sitewise** as Channel Type.
2. Click the **Configure** button.


Edit Communication Channel: su.demo.dashboard:SiteWise:latest

Group *
su.demo.dashboard

Name *
SiteWise

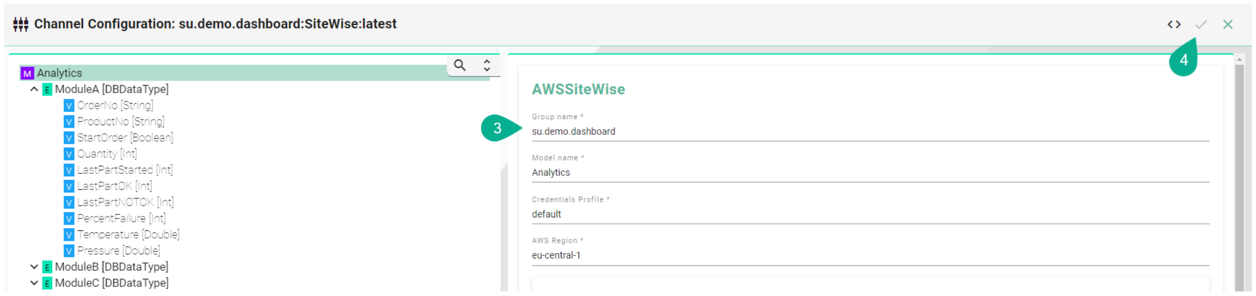
Description

Model *
su.demo.dashboard.Analytics.latest

Channel type *
AWS Sitewise

3. Enter the SiteWise configuration:

- Enter the group of the *Information Model*
- Enter the name of the *Information Model*
- Enter the profile from the credential file that should be used
- Enter the region of the AWS Iot SiteWise service you are using


Channel Configuration: su.demo.dashboard:SiteWise:latest

AWSSiteWise

Group name *
su.demo.dashboard

Model name *
Analytics

Credentials Profile *
default

AWS Region *
eu-central-1

Description of configuration properties:

Property	Description	Example
Group name	Information Model group name	`demo`
Model name	Information Model name	`Analytics`
Credentials Profile	Profile from the credential file	`default`
AWS Region	Region of the AWS Iot SiteWise service	`eu-central-1`

2.2.4 General Configurations

These configurations apply for all Communication Channel Types.

Framework Configuration

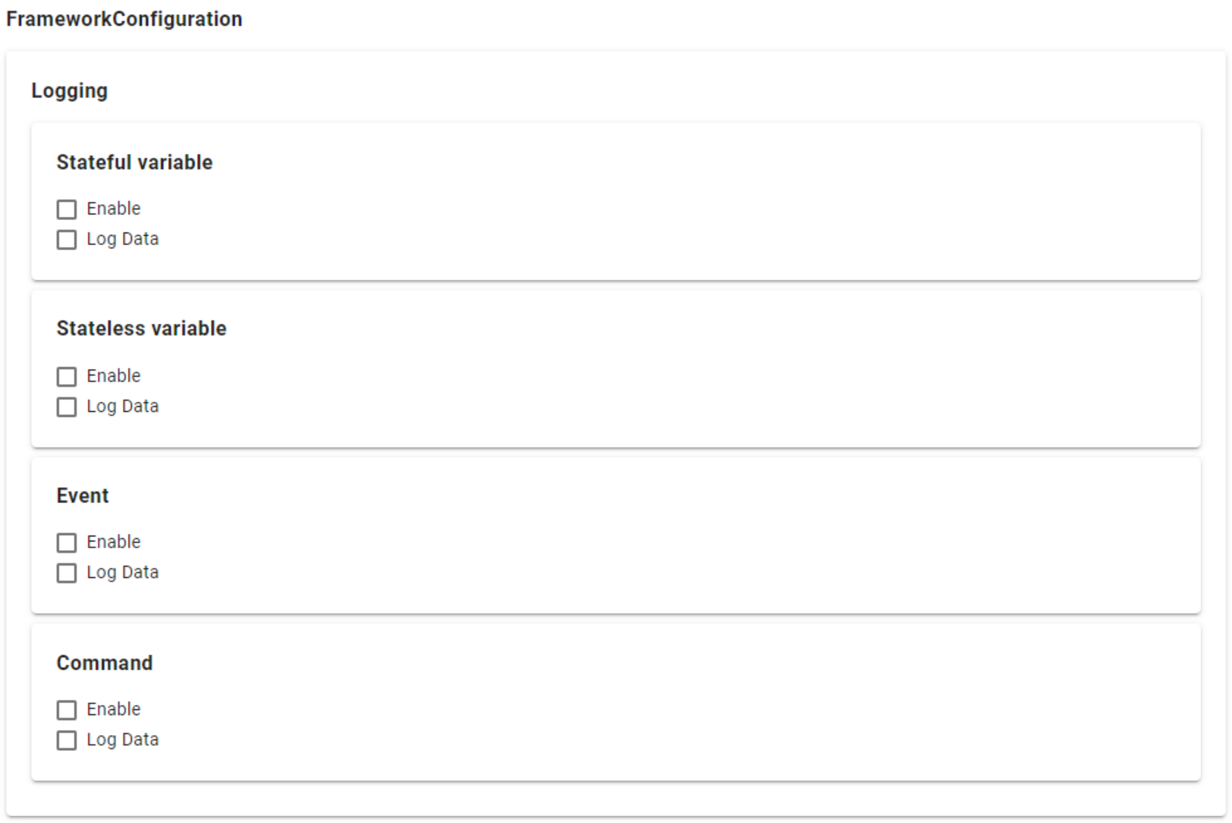
The Framework Configuration enables insights into data handled by *Mapping Rules*. If enabled, logs will be generated once Rules are triggered and executed. These logs are visible then by default in the **INFO** *Log Level* as well as in the *Log Viewer*.

The following Framework Logging Configurations are available:

- Stateful Variable
- Stateless Variable
- *Event*
- Command

For each configuration there are two ways to use logging:

- **Enable**: Logs out information about the *Node Type* that was executed by the Rule.
- **Log Data**: Logs out in JSON-format the actual data of the *Node Type* that was executed by a Rule.


FrameworkConfiguration

Logging

Stateful variable
☐ Enable
☐ Log Data

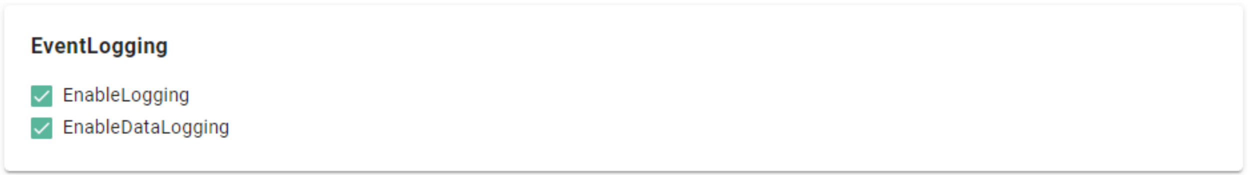
Stateless variable
☐ Enable
☐ Log Data

Event
☐ Enable
☐ Log Data

Command
☐ Enable
☐ Log Data

Event Logging

To use the Event Logging enable the checkbox **EventLogging** and for more detailed logging **Event-DataLogging**.


EventLogging

✓ EnableLogging
✓ EnableDataLogging

Event Logging Output

```
[INFO ] - EventDefinition - Received Event: /Model/bcdbbfd3-cdbe-4ade-8a73-
↪3788e6815c46/Event/ReleaseOrder
```

Event Data Logging Output

```
[INFO ] - EventDefinition - Received Event: /Model/bcdbbfd3-cdbe-4ade-8a73-
↪3788e6815c46/Event/ReleaseOrder={"Quantity":10,"ProductNumber":"Mv5","OrderNumber":
↪"Ord154","EquipmentId":"4-SWC2"}
```

2.3 Mappings

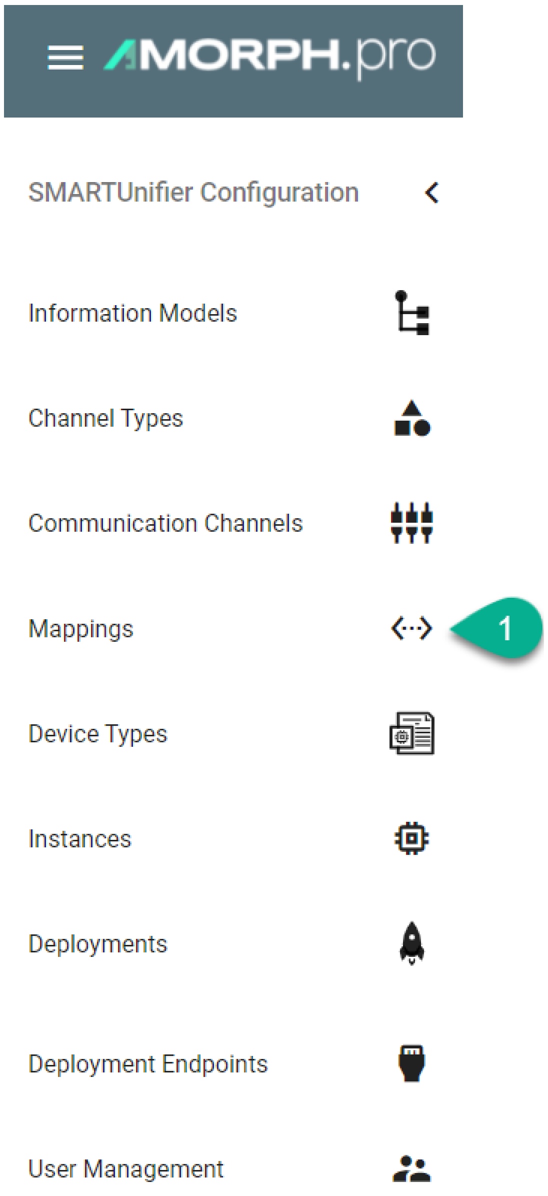
2.3.1 What are Mappings

Mappings represent the SMART**UNIFIER** component that define when and how to exchange/transform data between two or multiple *Information Models*. In other words, it is acting as a translator between the different Information Models. One Mapping consists of one or multiple Rules. A Rule contains a Trigger, which defines when the exchange/transformation takes place, and a list of actions that are defining how the exchange/transformation is done.

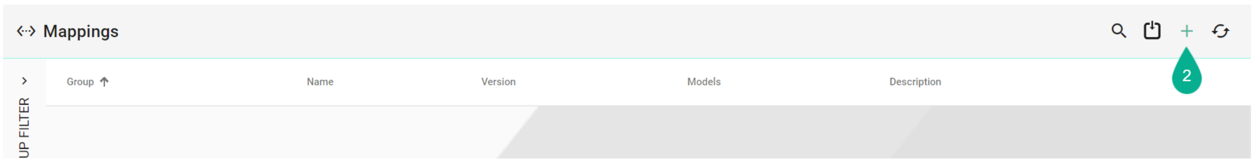
2.3.2 How to create a new Mapping

Follow the steps below to create a new Mapping definition:

- Go the Mappings perspective by clicking the "Mappings" button **(1)**



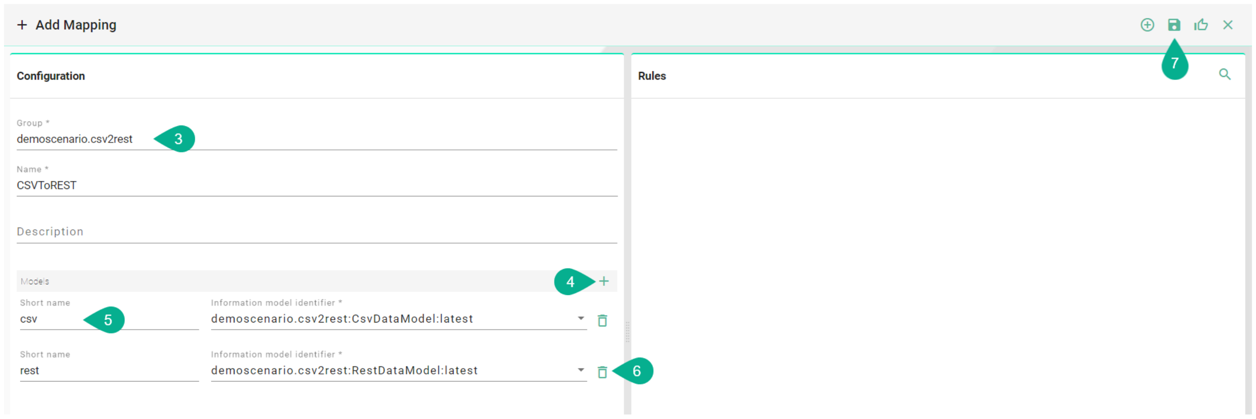
- Following screen containing a list view of existing Mappings is displayed
- In order to add a new Mapping, select the “Add Mapping” button at the top right corner **(2)**



Group ↑	Name	Version	Models	Description

- On the following screen provide the following mandatory information: Group, Name, Version and a Description which is optional **(3)**
- Click the “Add Model” button **(4)**
- Select the Information Model for this Mapping and enter a name for it **(5)**

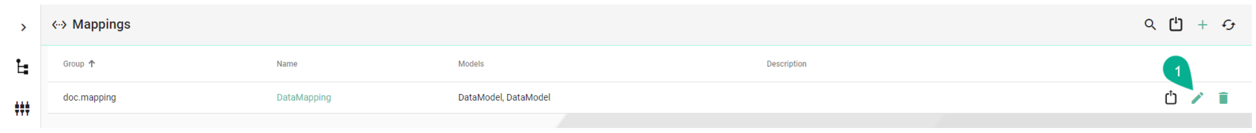
- “Remove Model” button **(6)** removes the Model
- After all mandatory fields are filled in, the “Save” button at the top right corner is enabled. Click the button to submit the new *Mapping* **(7)**
- The newly created Mapping is now visible in the list view


+ Add Mapping
Configuration
Rules
Group *
demoscenario.csv2rest
Name *
CSVToREST
Description
Models
Short name
csv
Information model identifier *
demoscenario.csv2rest:CsvDataModel:latest
Short name
rest
Information model identifier *
demoscenario.csv2rest:RestDataModel:latest
3
4
5
6
7

2.3.3 How to create Rules

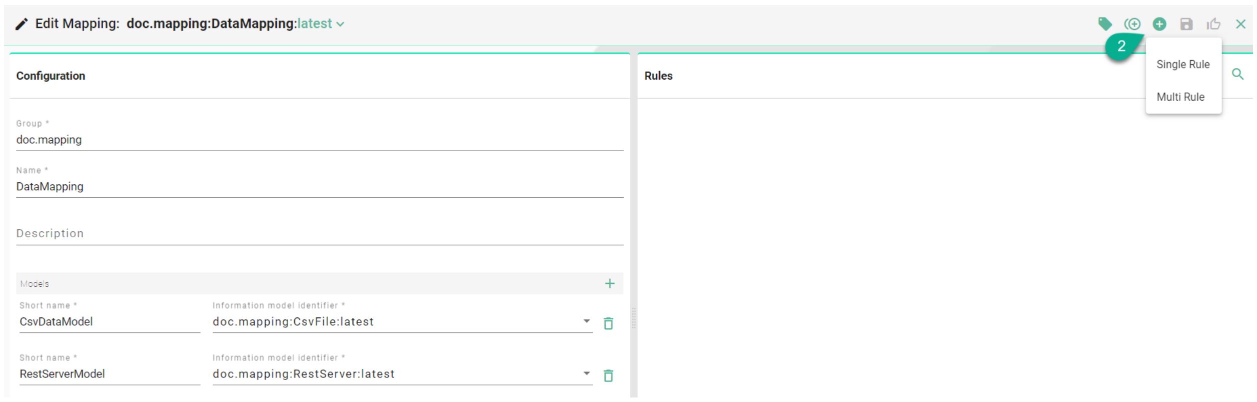
Follow the steps described below to create *Rules*:

- Select the “Edit” button **(1)**.



- Select the “Add Rule” button at the top right corner **(2)**.
- Two options are available:
  - Single Rule
  - Multi Rule

Note: Make sure to select **Single Rule** if you want to build up a rule using the code editor!


Edit Mapping: doc.mapping:DataMapping:latest
2
Single Rule
Multi Rule
Configuration
Rules
Group *
doc.mapping
Name *
DataMapping
Description
Models
Short name *
CsvDataModel
Information model identifier *
doc.mapping:CsvFile:latest
Short name *
RestServerModel
Information model identifier *
doc.mapping:RestServer:latest

Rule Naming Convention

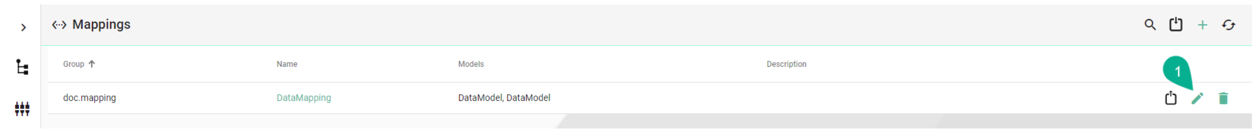
The name of the Rule should indicate what logic is executed when the Rule is triggered. Some examples are described below:

Example Scenarios	Good Naming
Inserting data into a database	Database_Insert
Executing a POST/PUT request on a REST server	Update_Data
Reacting to an input (e.g., StartOrder button on a MES)	StartOrder

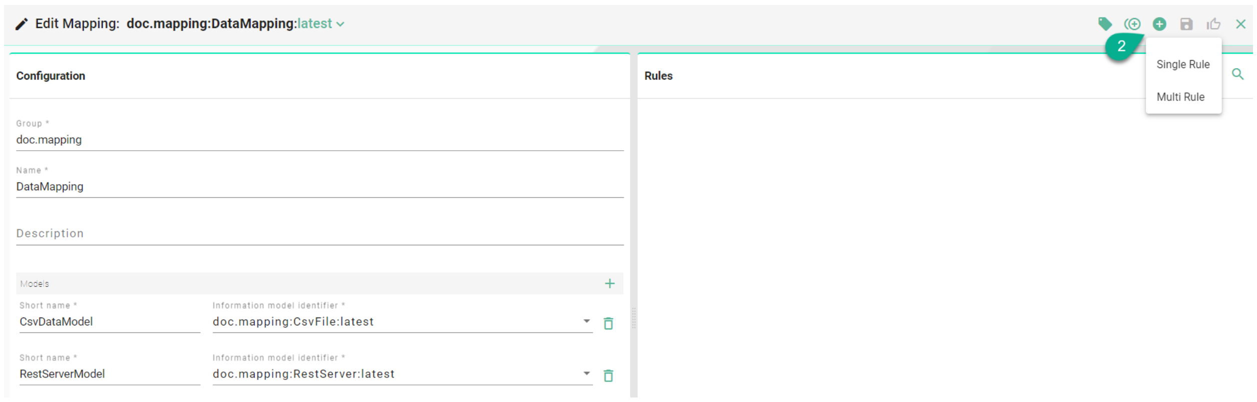
Graphical

Follow the steps described below to create Graphical-based *Rules*:

- Select the "Edit" button **(1)**.



- Select the "Add Rule" button at the top right corner **(2)**.
- Two options are available:
  - Single Rule
  - Multi Rule


Edit Mapping: doc.mapping:DataMapping:latest

Configuration

Group *
doc.mapping

Name *
DataMapping

Description

Models

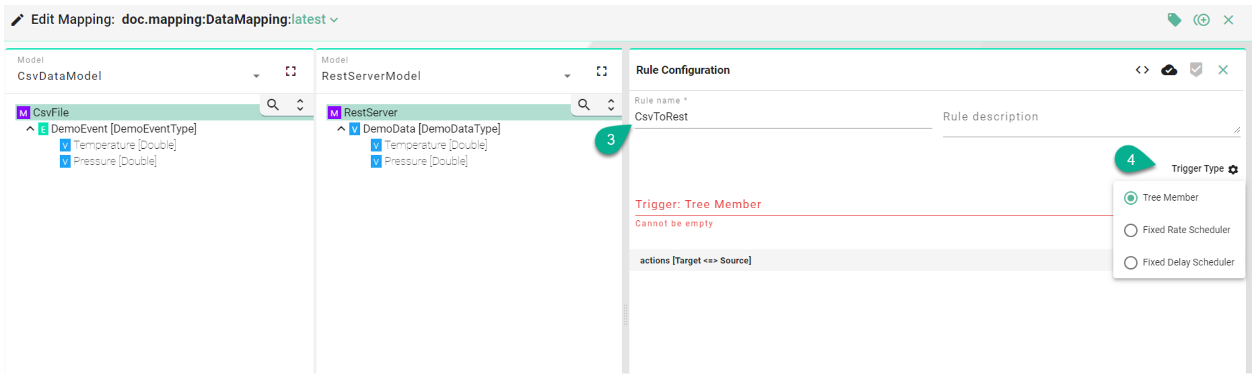
Short name *	Information model identifier *
CsvDataModel	doc.mapping:CsvFile:latest
RestServerModel	doc.mapping:RestServer:latest

Single Rule

- The following screenshot shows the Single Rule Editor. The Rule contains the following components: Name, Trigger and the Action with it's Source to Target assignments.

Note: The Single Rule includes only one Trigger.

- Enter "Rule name" **(3)**.


Edit Mapping: doc.mapping:DataMapping:latest

Model
CsvDataModel

Model
RestServerModel

Rule Configuration

Rule name *
CsvToRest

Rule description

Trigger: Tree Member
Cannot be empty

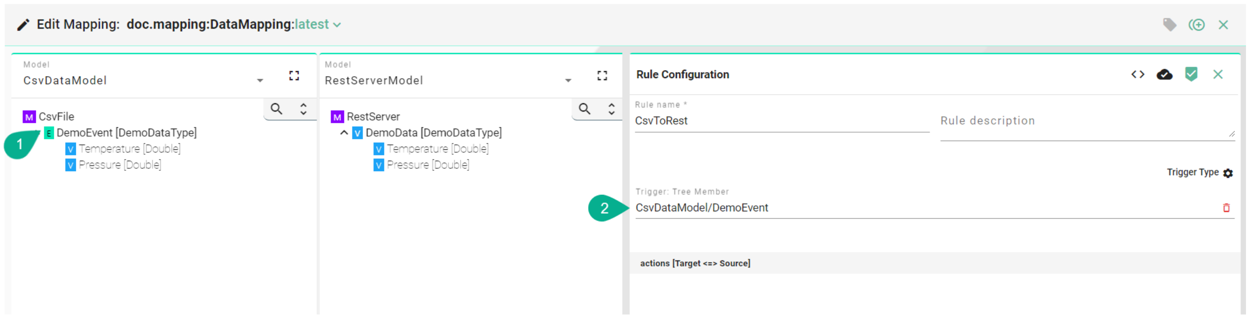
actions [Target <=> Source]

- Select the "Trigger Type" **(4)**:
  - Tree Member - rule with an Information Model tree member as trigger
  - Fixed Rate Scheduler - rule with a time based trigger, using a Cron Expression
  - Fixed Delay Scheduler - rule based on a scheduled delay

Trigger Types

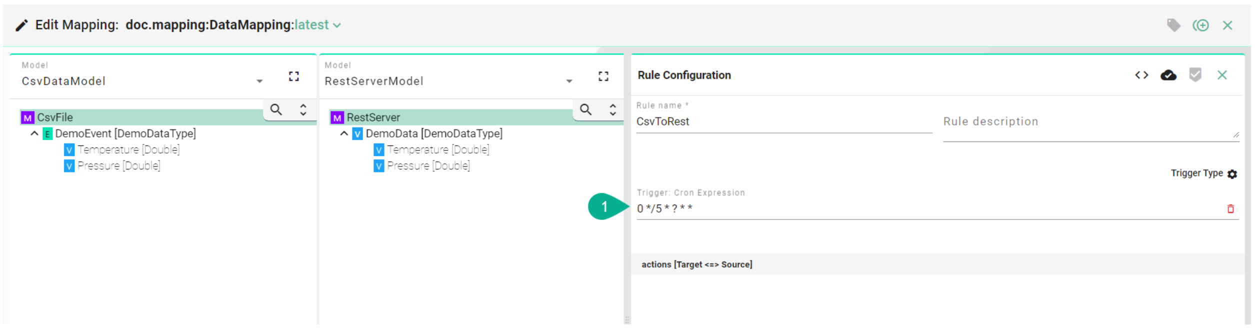
Tree Member

- Drag and drop the *Trigger* from the model panes **(1)** into the trigger field **(2)**.



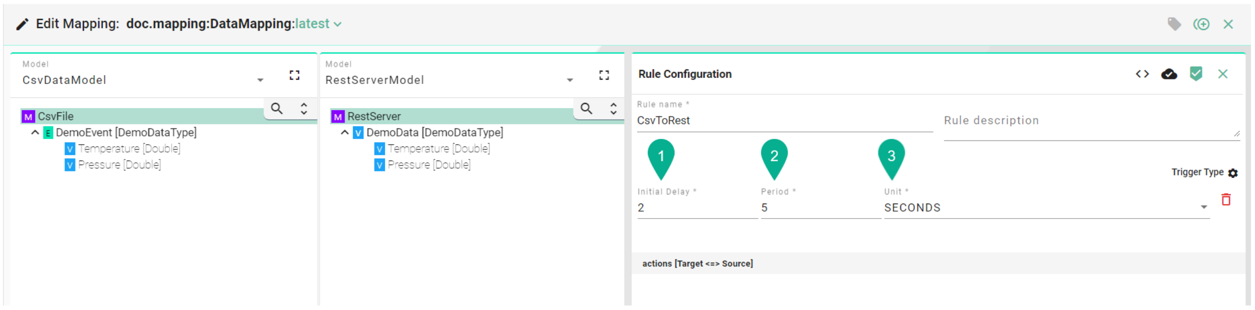
Fixed Rate Scheduler

- Input a "Cron Expression" **(1)** to set the time based trigger. (E.g., `0 */5 * ? * *` meaning the trigger is set at every 5 minutes).



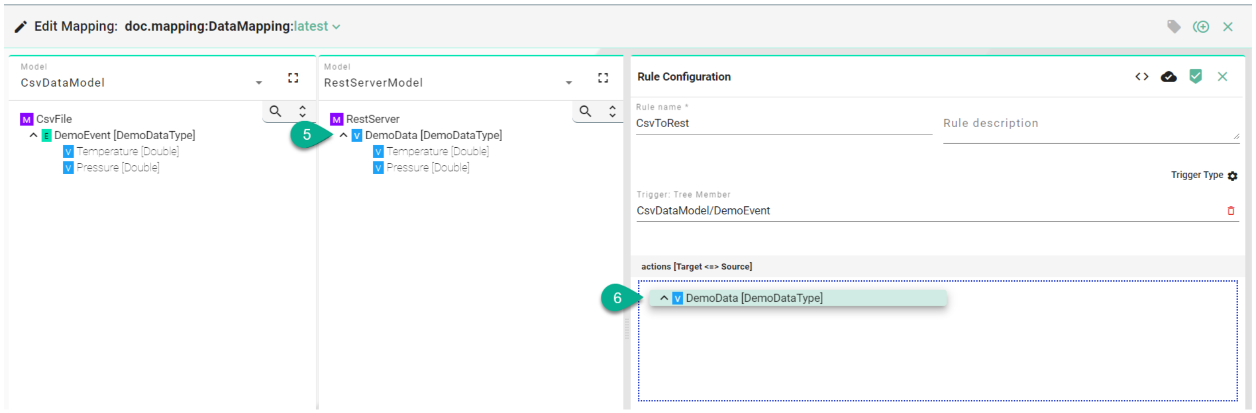
Fixed Delay Scheduler

- Input the trigger "Initial start Delay" **(1)**, the "Period" delay **(2)** and the "Unit" **(3)**.



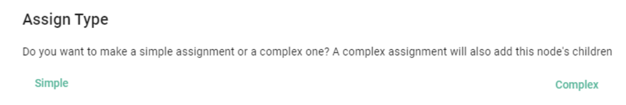
Actions

- Drag and drop the *Target* Information Model node **(5)** into the Target field **(6)**.



A popup appears to select the assignment type:

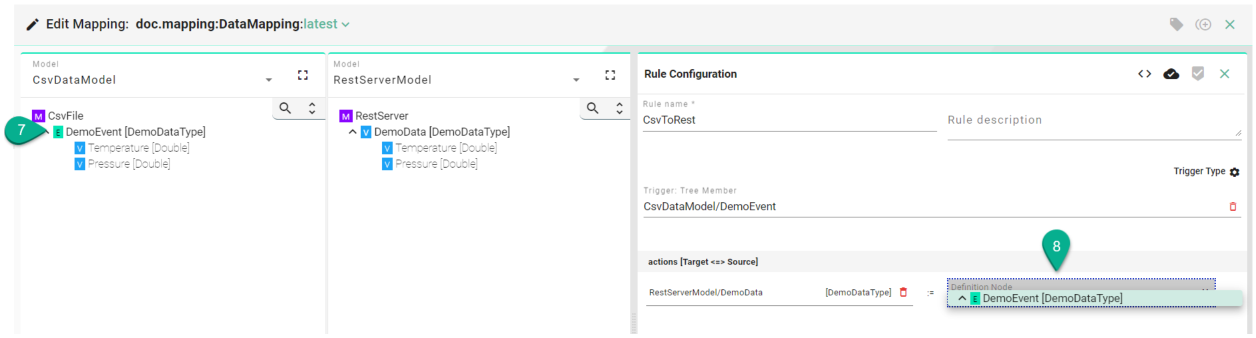
- Simple - the assignment is made at the Information Model node level
- Complex - the assignment is made at the Information Model node children's level


Assign Type

Do you want to make a simple assignment or a complex one? A complex assignment will also add this node's children

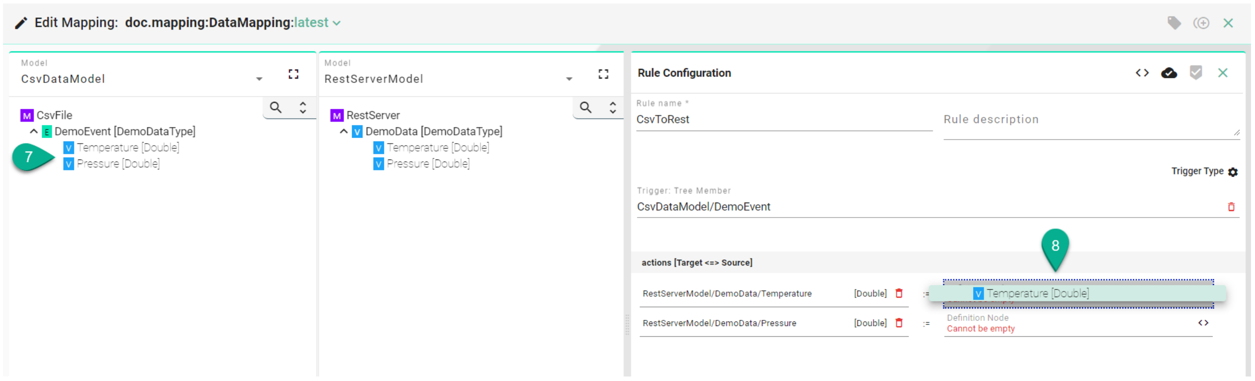
Simple Assignment

- Drag and drop the *Source* Information Model node **(7)** into the Source field **(8)**. The Source and the Target node data type must be matched one on one (e.g., DemoEventType to DemoEventType).

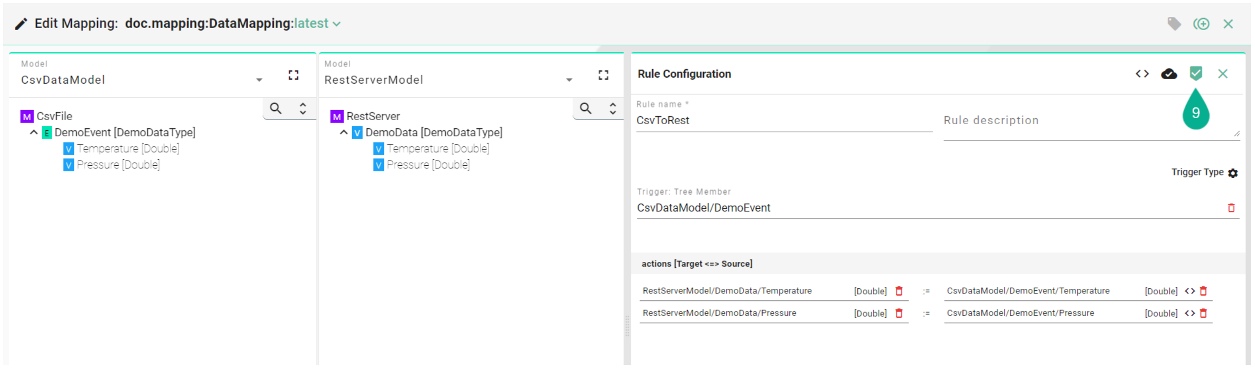


Complex Assignment

- Drag and drop the *Source* Information Model node children's **(7)** one by one into the Source field **(8)**. The Source and the Target information must be matched one on one (e.g., String to String).

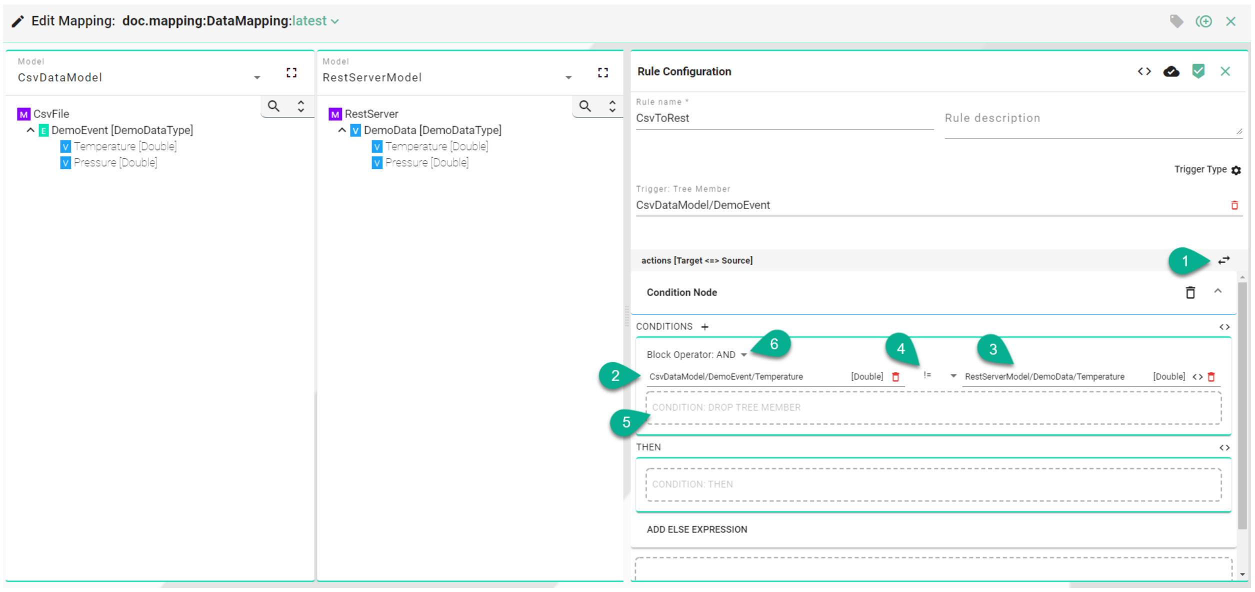


- After all mandatory fields have been filled out, select the "Apply" button **(9)** to save the newly created Rule.
- The Single Rule Editor is closed and the newly created Rule is displayed in the Rules List.
- Select the "Save" button placed in the upper right corner to save the Mapping.



Actions with Conditions

- Click on the "Add condition block" button **(1)**.
- Drag and drop a tree member **(2)** and **(3)**.
- Select the condition operator **(4)**.

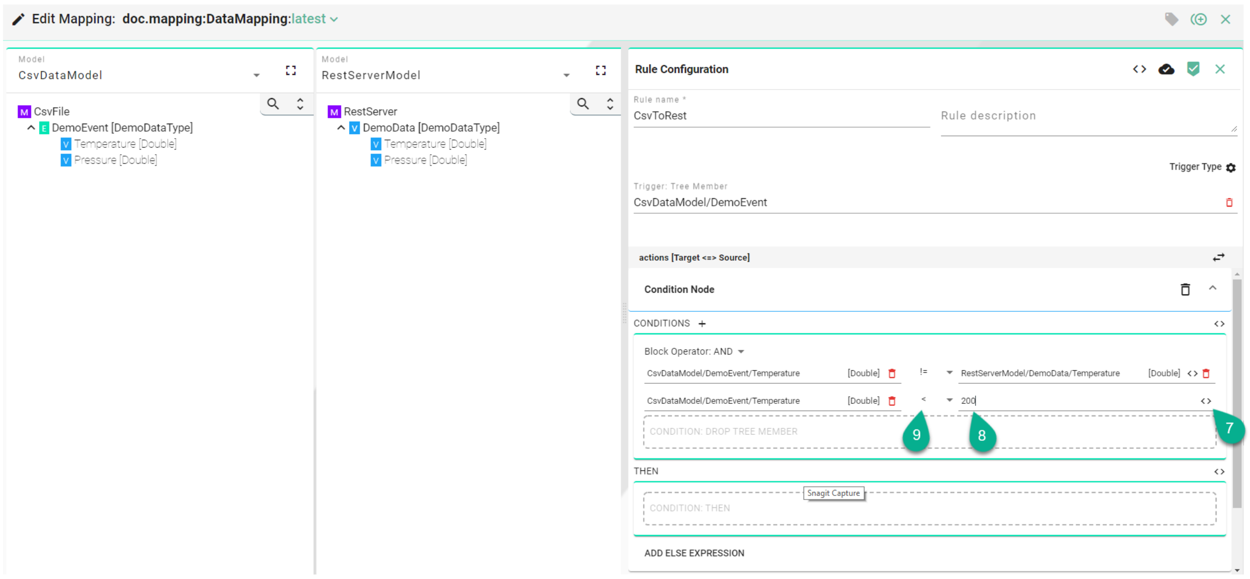


Rule name: CsvToRest

Trigger: Tree Member: CsvDataModel/DemoEvent

CsvDataModel/DemoEvent/Temperature [Double] != RestServerModel/DemoData/Temperature [Double]

- To add multiple conditions **(5)** select the block operator **(6)**.
- Click on the "Literal Node" button **(7)** to use as a definition node a value (e.g., Integer) instead of a tree member.
- Input a value **(8)**.
- Select the condition operator **(9)**.



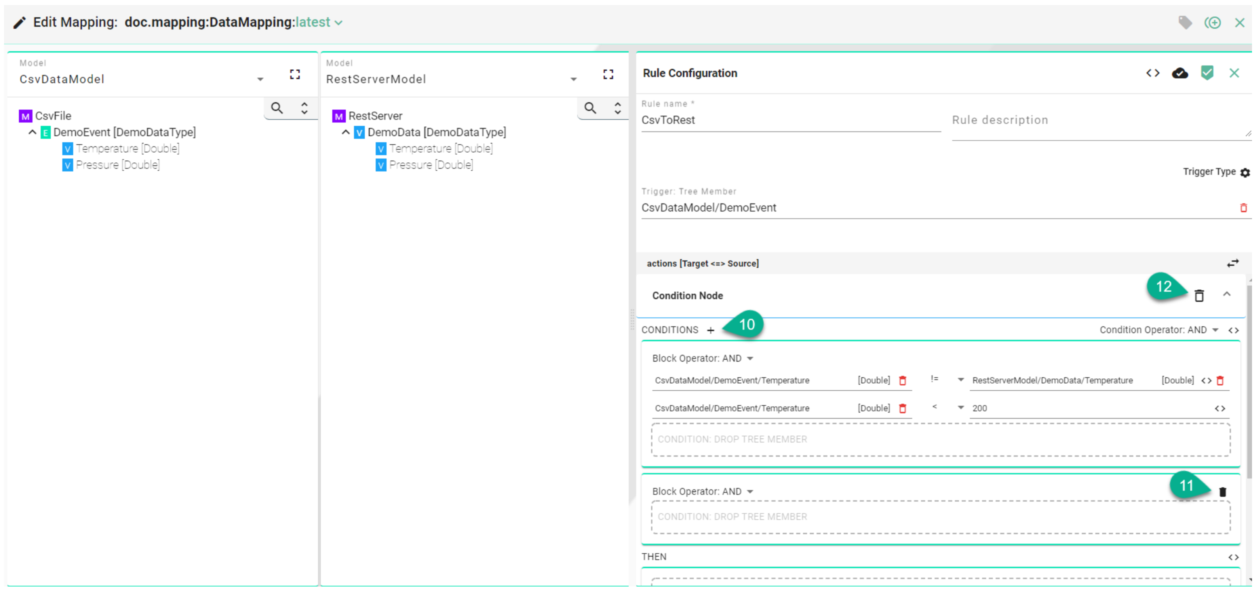
Rule name: CsvToRest

Trigger: Tree Member: CsvDataModel/DemoEvent

CsvDataModel/DemoEvent/Temperature [Double] != RestServerModel/DemoData/Temperature [Double]

CsvDataModel/DemoEvent/Temperature [Double] < 200

- Click on the "Add Condition Block" button **(10)** to add a new one.
- Click on the "Delete Condition Block" button **(11)** to remove a condition block and select the "Delete" button **(12)** to remove the condition.

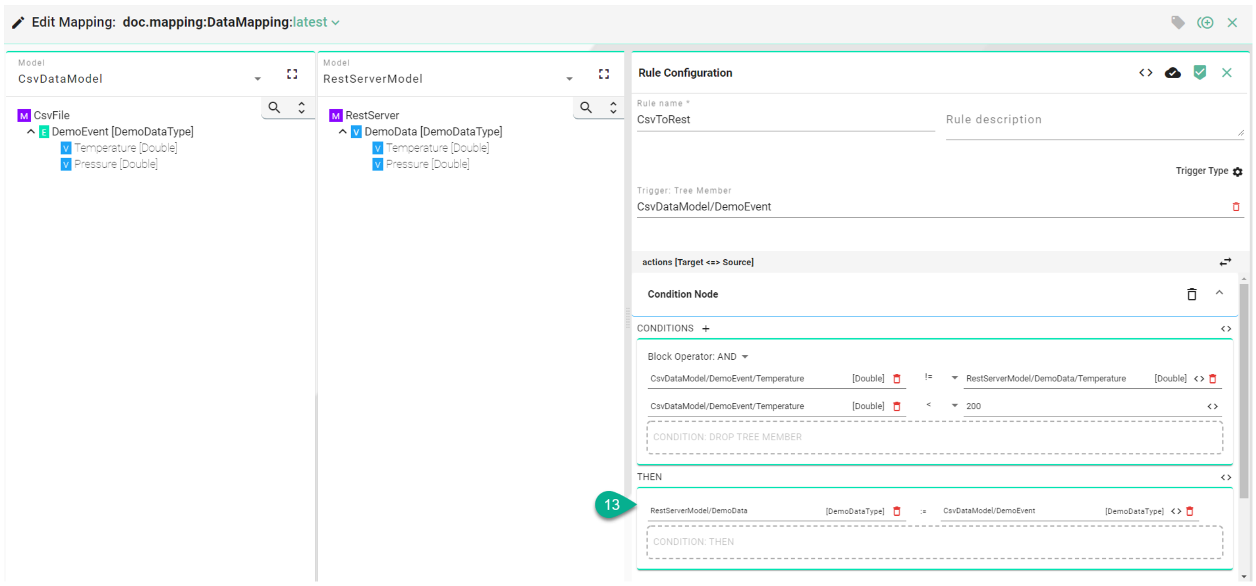

Edit Mapping: doc.mapping:DataMapping:latest

Model: CsvDataModel
- CsvFile
  - DemoEvent [DemoDataType]
    - Temperature [Double]
    - Pressure [Double]

Model: RestServerModel
- RestServer
  - DemoData [DemoDataType]
    - Temperature [Double]
    - Pressure [Double]

Rule Configuration
Rule name *: CsvToRest
Rule description
Trigger Type
Trigger: Tree Member: CsvDataModel/DemoEvent
actions [Target <=> Source]
Condition Node
CONDITIONS + — Condition Operator: AND
Block Operator: AND
CsvDataModel/DemoEvent/Temperature [Double] != RestServerModel/DemoData/Temperature [Double]
CsvDataModel/DemoEvent/Temperature [Double] < 200
CONDITION: DROP TREE MEMBER
Block Operator: AND
CONDITION: DROP TREE MEMBER
THEN

In the “THEN” **(13)** section, drag and drop the Target and Source Information Model nodes, using either the *simple* or the *complex* assignment.


Edit Mapping: doc.mapping:DataMapping:latest

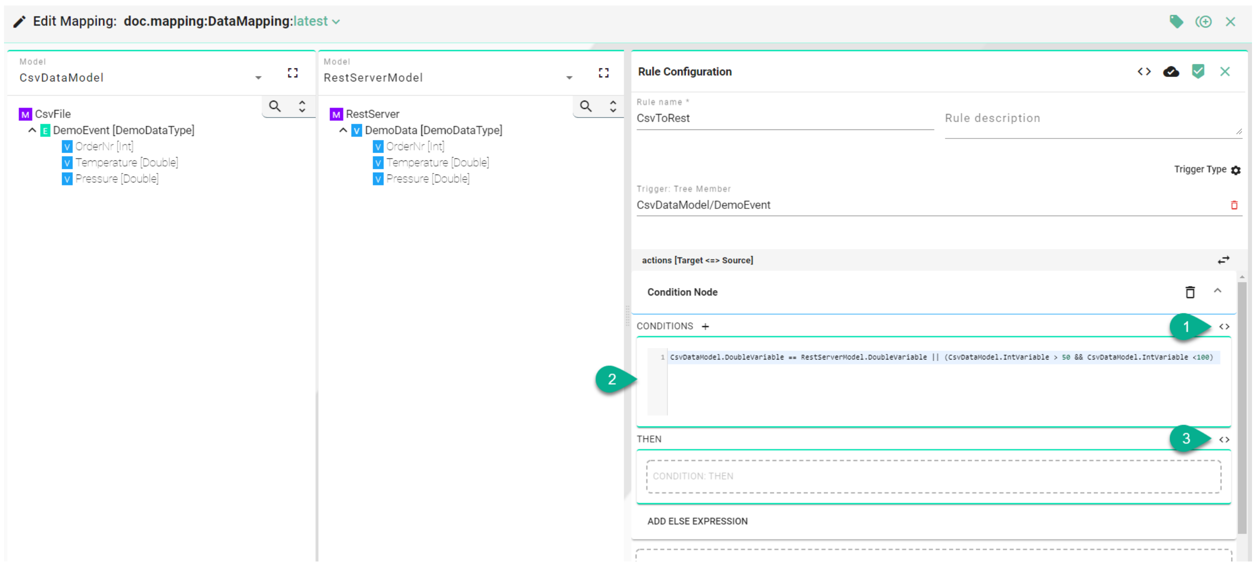
Model: CsvDataModel
- CsvFile
  - DemoEvent [DemoDataType]
    - Temperature [Double]
    - Pressure [Double]

Model: RestServerModel
- RestServer
  - DemoData [DemoDataType]
    - Temperature [Double]
    - Pressure [Double]

Rule Configuration
Rule name *: CsvToRest
Rule description
Trigger Type
Trigger: Tree Member: CsvDataModel/DemoEvent
actions [Target <=> Source]
Condition Node
CONDITIONS +
Block Operator: AND
CsvDataModel/DemoEvent/Temperature [Double] != RestServerModel/DemoData/Temperature [Double]
CsvDataModel/DemoEvent/Temperature [Double] < 200
CONDITION: DROP TREE MEMBER
THEN
RestServerModel/DemoData [DemoDataType] := CsvDataModel/DemoEvent [DemoDataType]
CONDITION: THEN

Actions with Custom Conditions

- Click on the “Source Code” button **(1)**.
- Input code for a complex condition **(2)**.



Rule name *: CsvToRest

Rule description

Trigger: Tree Member: CsvDataModel/DemoEvent

```
CsvDataModel.DoubleVariable == RestServerModel.DoubleVariable || (CsvDataModel.IntVariable > 50 && CsvDataModel.IntVariable <100)
```

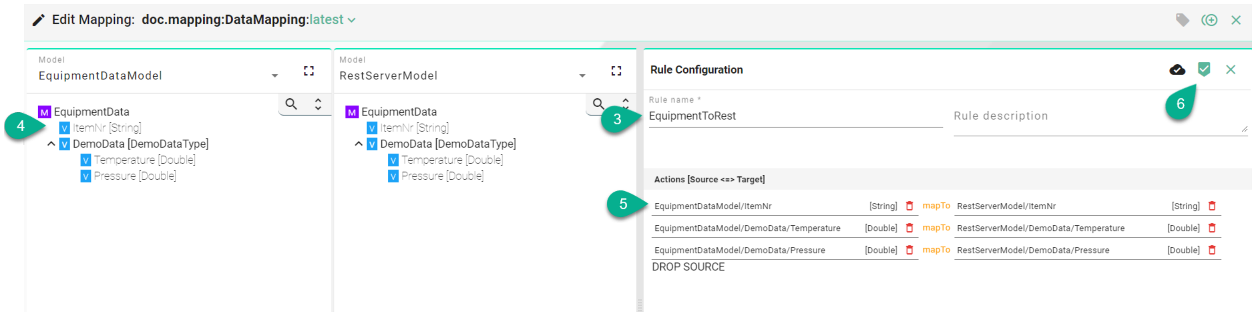
- For the “THEN” section use drag and drop or click the “Source Code” button **(3)** to input code.

Multi Rule

- The following screenshot shows the Multi Rule Editor. The Rule contains the following components: Name and the Actions with it’s Source to Target assignments.

Note: The Multi Rule configuration considers each Source as a Trigger.

- Enter “Rule name” **(3)**.
- Drag and drop the *Source* Information Model nodes **(4)** one by one into the Source field **(5)**.
- The Source and the Target information must be matched one on one (e.g., String to String). Allowed nodes for Source and Target: Simple Variables and Variables from a Complex Variable.
- After all mandatory fields have been filled out, select the “Apply” button **(6)** to save the newly created Rule.



Rule name *: EquipmentToRest

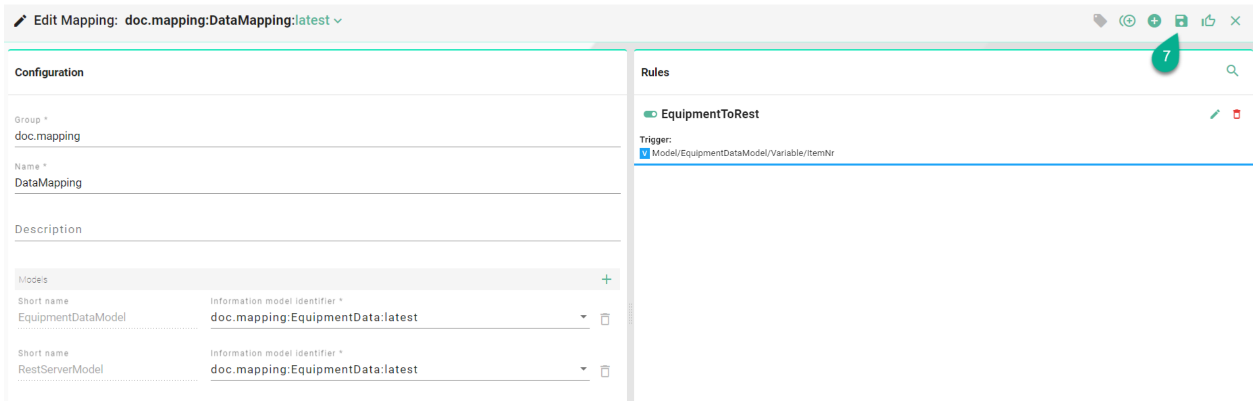
Rule description

Actions [Source <=> Target]

EquipmentDataModel/ItemNr	[String]	mapTo	RestServerModel/ItemNr	[String]
EquipmentDataModel/DemoData/Temperature	[Double]	mapTo	RestServerModel/DemoData/Temperature	[Double]
EquipmentDataModel/DemoData/Pressure	[Double]	mapTo	RestServerModel/DemoData/Pressure	[Double]

- The Multi Rule Editor is closed and the newly created Rule is displayed in the Rules List.

- Select the “Save” button **(7)** placed in the upper right corner to save the Mapping.



Edit Mapping: doc.mapping:DataMapping:latest

Configuration

Group *
doc.mapping

Name *
DataMapping

Description

Models

Short name	Information model identifier *
EquipmentDataModel	doc.mapping:EquipmentData:latest
RestServerModel	doc.mapping:EquipmentData:latest

Rules

EquipmentToRest

Trigger:
V Model/EquipmentDataModel/Variable/ItemNr

Code-based Rules

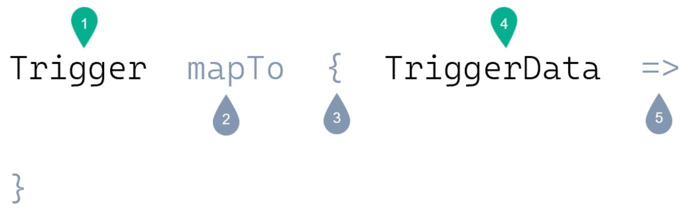
The main target of SMART**UNIFIER** is to build up the connectivity between systems. Sometimes integrations become more complex and it might require to build up Rules via the code editor using the Scala programming language. SMART**UNIFIER** extends the Scala programming language with addition operators and methods to simplify the realization of data transfer between Information Models.

Similar to Mappings via drag and drop, there is no knowledge of the underlying communication protocol (e.g., MQTT, OPCUA, etc.) needed. Protocols are hidden behind the corresponding Information Models. The parameter values of an Information Model are stored in the objects of type *VariableDefinition[T]* or *PropertyDefinition[T]*. These contain additional information and methods rather than just the parameter values. They also provide methods to listen for changes and conversion between variable types.

Note: Make sure to select **Single Sule** if you want to build up a rule using the code editor!

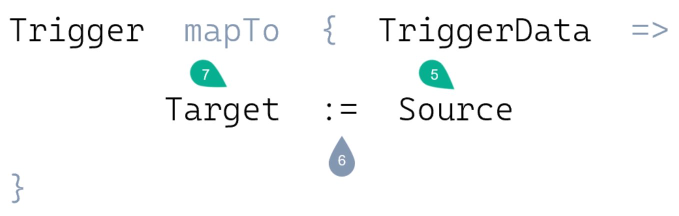
Basics - Rule construct

A Rule is always starting with a *Trigger* **(1)**. The Trigger can represent a *Variable*, an *Event* or a *Command*; within one of the selected *Information Models*. After the trigger call mapTo **(2)** and define the function body by adding curly braces **(3)**. Depending on the Trigger declare the TriggerInstance **(4)**. Depending on the type of the Trigger use the naming accordingly:



```
Trigger  mapTo  {  TriggerData  =>

}
```

The *Source* **(5)** is the content of the TriggerInstance (e.g., In case the Trigger is a Variable, then is the Source an Instance of that Variable) In order to assign the Source to the *Target*, add the := operator **(6)**. The Target can be any variable you want to map to **(7)**.



```
Trigger  mapTo  {  TriggerData  =>
         Target  :=  Source
}
```

Trigger Types

Tree Member

Represents a basic Rule that uses as Trigger an Information Model element: *Variables*, *Events* or *Commands*. Therefor define the Trigger as the following: `<Information Model>.<Element from the Information Model> mapTo { <Element type> =>` (line 1).

Listing 1: Tree Member

```
1 EquipmentDataModel.ItemNr mapTo { variable =>
2    Try {
3      EquipmentDataModel.DemoData.Temperature := RestServerModel.DemoData.Temperature
4      EquipmentDataModel.DemoData.Pressure := RestServerModel.DemoData.Pressure
5    }
6  }
```

Fixed Rate Scheduler

Rules can be scheduled to run continuously at a fixed rate. Instead of having an element of the Information Model defined as a Trigger the **fixedRateScheduler** method can be used. Therefor define the Trigger as the following: `_trigger.fixedRateScheduler(<Cron Expression>)` (line 2).

Listing 2: Fixed Rate Scheduler

```
def rule_ScheduleNode(): Unit = {
   _trigger.fixedRateScheduler("0/1 * * * * ? *") mapTo(() => {
     model1.StringVariable := model2.StringVariable
   })
 }
```

Fixed Delay Scheduler

Rules can be scheduled to run with a specific delay. Therefor define the Trigger as the following: `_trigger.fixedDelayScheduler(<Initial Delay>, <Period>, <Unit>) mapTo(() =>` (line 1).

Listing 3: Fixed Delay Scheduler

```
1 _trigger.fixedDelayScheduler(10, 60, SECONDS) mapTo(() => Try{
2    EquipmentDataModel.DemoData.Temperature := RestServerModel.DemoData.Temperature
3    EquipmentDataModel.DemoData.Pressure := RestServerModel.DemoData.Pressure
4  })
```

Target < > Source Assignment

Assignments of Same Type

When both target and source nodes are of the same data type the assignment of variables can be shorten:

Listing 4: Type Assignment (Events)

```
event1 := event2
```

Assignments of Different Type

The following examples illustrate how to implement assignments between source and target variables.

Source: Variable to Target: Event

This Mapping is used when dealing with static data that should be transformed to an Event. This might be the case when data is coming from a variable based data server (e.g., OPC UA server, Modbus, Iso-On-TCP) and needs to be mapped to an event or message-based target system (e.g., MQTT, Kafka, Databases, etc.).

The following example shows the mapping of variables from the **EnterpriseModel** and the **EquipmentModel** to an Event located in the **MesModel**:

- Trigger: **EquipmentModel.Alarm** (line 1)
- TriggerInstance of EquipmentModel.Alarm: **variable** (line 1)
- Call send method on the **EquipmentAlarm** Event (line 2) and define the TriggerInstance: **event** (line 2)
- The assignment of variables is done using the assignment operator **:=**. Both target and source are defined by entering the path of the variables in the Information Model e.g., **event.EquipmentId** and **EnterpriseModel.EquipmentName** (line 4)

Listing 5: Rule - StartOrder - Variable/Event

```
1  EquipmentModel.Alarm mapTo {variable =>
2    MesModel.EquipmentAlarm.send(event => {
3      Try {
4        event.EquipmentId := EnterpriseModel.EquipmentName
5        event.OrderNr := EquipmentModel.CurrentOrder.OrderNr
6        event.MaterialID := EquipmentModel.CurrentMaterialID
7        event.AlarmInfo := EquipmentModel.AlarmInfo
8        CommunicationLogger.log(variable, event)
9      }
10   })
11 }
```

Source: Event to Target: Variable

This Mapping is used when dealing with event driven data that should be mapped to variables. This might be the case when data is coming from an event or message-based system (e.g., MQTT, Kafka, Databases, etc.) and needs to be mapped to a variable based data server (e.g., OPC UA server, Modbus, Iso-On-TCP).

The following example describes the mapping of values inside the **TransferNewOrder** Event from the **MesModel** into variables from the **EquipmentModel**:

- The Trigger is defined by entering the path of the Event **MesModel.TransferNewOrder** (line 1). Since an Event is used as Trigger, the TriggerInstance is named accordingly **event** (line 1)
- In the function body provide the Complex Variable **NewOrder** and the Simple Variable **NewMESOrderFlag** with data from the MesModels **TransferNewOrder** Event
- Targets are defined by entering the path of the variables like **EquipmentModel.NewOrder.OrderNr** (line 3)
- In order to assign values to **OrderNr**, **MaterialNr** and **Quantity** of the Complex Variable **NewOrder**, enter the TriggerInstance event followed by the variable name of the TransferNewOrder Event **event.OrderNr** (line 3)
- In this case it is also possible to provide the variable **NewMesOrderFlag** with a Boolean like **true** (line 6)

Listing 6: Rule - TransferNewOrder - Event/Variable

```
MesModel.TransferNewOrder mapTo { event =>
  Try {
    EquipmentModel.NewOrder.OrderNr := event.OrderNr
    EquipmentModel.NewOrder.MaterialNr := event.MaterialNr
    EquipmentModel.NewOrder.Quantity := event.Quantity
    EquipmentModel.NewMESOrderFlag := true
  }
}
```

Source: Event to Target: Command

This Mapping is used when dealing with event driven data that should be mapped to a Command. This might be the case when incoming event or message driven data should be enriched with data from another system (e.g., a database, a REST server, etc.) and then further mapped to another event driven message.

The following scenario describes a Rule mapping incoming data from a file to MQTT. When the FileEvent is triggered, the rule executes first the **DatabaseCommand** to retrieve data from a database (after the request is executed, the result of the reply can be accessed directly):

- Trigger is defined by entering the path of the Event **file.FileEvent** (line 1). Since an Event is used as Trigger, the TriggerInstance should be named accordingly **event** (line 1)
- Inside the function body execute a Command. The execution of a Command is defined by entering the path of the Command. At the end of the path, call the **execute** function (line 2). The TriggerInstance is named accordingly **command** (line 4)
- The lines 4-6 show the first part of the Command. Here are assigned values from the source model to the Command Parameters
- Since every Command has a Reply, we need to define the reply section (line 8)
- In this case send out the data over MQTT after the data is retrieved from the database. In the reply function body, enter the path of the **MqttEvent**. Since this is the 2nd Event, the TriggerInstance can be named **event1** (line 1)
- Inside the function body assign values from the **FileEvent** (line 11-13) as well as from the Reply (line 14-15) to the **MqttEvent**

Listing 7: Rule - File2MqttWithDB - Event/Commands

```
1  file.FileEvent mapTo {event =>
2    database.DatabaseCommand.execute(command => {
3      Try {
4        command.orderNr := event.orderNr
5        command.materialNr := event.materialNr
6        CommunicationLogger.log(event, command)
7      }
8    }, reply => {
```

(continues on next page)

(continued from previous page)

```
     mqtt.MqttEvent.send(event1 => {
       Try {
         event1.Quality := event.quality
         event1.OrderNr := event.orderNr
         event1.MaterialNr := event.materialNr
         event1.Customer := reply.customer
         event1.Product := reply.product
         CommunicationLogger.log(reply, event1)
         }
     })
   })
 }
```

Mapping with Lists

If there are *Lists* structures within an Information Model that is going to be mapped to another Information Model, it is required to step through the list items using a foreach.

The following scenario describes a Rule that is mapping incoming data from a file to MQTT. The MQTT Model contains a List called **DataList**.

- Create a variable **listItem** that holds a reference of a **newItem** in the **DataList** (line 6)
- Call the variable from the **listItem** and assign the value from the file event (line 8)

Listing 8: Rule - FileToMQTT - Lists

```
1  csv.FileEvent mapTo { event =>
2
3       event.items.foreach { item =>
4         mqtt.MqttEvent.send(event1 => {
5           Try {
6             val listItem = event1.DataList.newItem
7
8             listItem.Timestamp := item.Timestamp
9             listItem.Pressure := item.Alarmlevel
10
11            CommunicationLogger.log(event, event1)
12          }
13        })
14      }
15  }
```

Note: Lists can only be mapped in the code view.

Logging

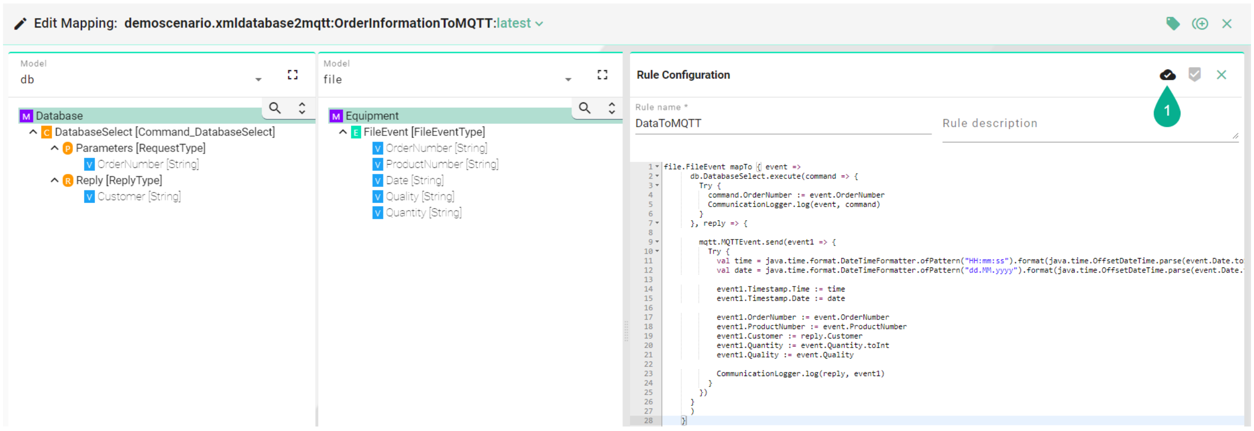
Logging can be added in the Rule implementation by calling - **CommunicationLogger.log** (line 5)

Listing 9: Rule with Logging

```
1  EquipmentModel.Alarm mapTo {variable =>
2     MesModel.EquipmentAlarm.send(event => {
3       Try {
4         event.EquipmentId := EnterpriseModel.EquipmentName
5         CommunicationLogger.log(variable, event)
6       }
7     })
8  }
```

Compiling

You can compile the code for the selected Rule by clicking the “Compile” button **(1)** and check for compilation errors before saving the Rule.



Rule name *
DataToMQTT

```
file.FileEvent mapTo { event =>
      db.DatabaseSelect.execute(command => {
        Try {
          command.OrderNumber := event.OrderNumber
          CommunicationLogger.log(event, command)
        }
      }, reply => {

        mqtt.MQTTEvent.send(event1 => {
          Try {
            val time = java.time.format.DateTimeFormatter.ofPattern("HH:mm:ss").format(java.time.OffsetDateTime.parse(event.Date.to
            val date = java.time.format.DateTimeFormatter.ofPattern("dd.MM.yyyy").format(java.time.OffsetDateTime.parse(event.Date.

            event1.Timestamp.Time := time
            event1.Timestamp.Date := date

            event1.OrderNumber := event.OrderNumber
            event1.ProductNumber := event.ProductNumber
            event1.Customer := reply.Customer
            event1.Quantity := event.Quantity.toInt
            event1.Quality := event.Quality

            CommunicationLogger.log(reply, event1)
          }
        })
      }
      )
    }
```

SMARTUNIFIER Code Constructs

Rules are written in the Scala programming language. SMART**UNIFIER** comes also with some custom code constructs that can be used within the Mapping which allow e.g., to do type conversions directly on the variable level.

Conversions

If both variables that are going to be mapped to each other are not of the same data type, use the provided type conversions. Conversions can be used on Information Model nodes such as *Variables* and on *Properties* .

Method	Description	Example
toBoolean(definition: TVariableDefinition[T])	Converts a variable to a Boolean	toBoolean(m1.IntVariable)
toBoolean(definition: TPropertyDefinition [T])	Either the literal true or the literal false	
toByte (definition: TVariableDefinition[T])	Conversion of a variable to an Byte	toByte(m1.IntVariable)
toByte (definition: TPropertyDefinition [T])	8 bit signed value. Range from -128 to 127	
toShort (definition: TVariableDefinition[T])	Conversion of a variable to a Short	toShort(m1.IntVariable)
toShort (definition: TPropertyDefinition [T])	16 bit signed value. Range -32768 to 32767	
toInt (definition: TVariableDefinition[T])	Conversion of a variable to an Integer	toInt(m1.StringVariable)
toInt (definition: TPropertyDefinition [T])	32 bit signed value. Range -2147483648 to 2147483647	
toLong (definition: TVariableDefinition[T])	Conversion of a variable to a Long	toLong(m1.IntVariable)
toLong (definition: TPropertyDefinition [T])	64 bit signed value. -9223372036854775808 to 9223372036854775807	
toFloat(definition: TVariableDefinition[T])	Conversion of a variable to a Float	toFloat(m1.IntVariable)
toFloat (definition: TPropertyDefinition [T])	32 bit IEEE 754 single-precision float	
toDouble(definition: TVariableDefinition[T])	Conversion of a variable to a Double	toDouble(m1.IntVariable)
toDouble (definition: TPropertyDefinition [T])	64 bit IEEE 754 double-precision float	
toStr(definition: TVariableDefinition[T])	Conversion of a variable to an String	toStr(m1.IntVariable)
toStr (definition: TPropertyDefinition [T])	A sequence of Chars	
Use the functional examples in the manual. e.g. toBoolean(event.MyValue)		

Operators

Operator methods can be used to implement calculations such as additions, subtractions, multiplications, and divisions. If it is required to make some calculations on the values of a variable within the mapping before sending data to the target system, the following methods can be used:

Method	Description	Example
add(Option[T],Dou	Addition of a variable with a numeric data type and a Double value	add(model.IntVariable, 2)
sub(Option[T],Dou	Subtraction of a variable with a numeric data type and a Double value	sub(model.IntVariable, 2.5)
mult(Option[T],Do	Multiplication of a variable with a numeric data type and a Double value	mult(model.IntVariable, 3)
div(Option[T],Dou	Division of a variable with a numeric data type and a Double value	div(model.IntVariable, 3.5)

Loops (foreach)

In some use cases it might be required to iterate through a collection if the Information Model contains a list or an array. In this case, call the items method on the lists element of the Information Model followed by foreach (line 13).

Listing 10: Code constructs - Loops

```
1  import java.time.LocalDateTime
2  import java.time.Instant
3  import java.time.ZoneId
4
5  equipment.FileEvent mapTo { event =>
6   val newImportDate = LocalDateTime.ofInstant(Instant.ofEpochMilli(System.
   ↪currentTimeMillis()), ZoneId.systemDefault())
7   db.MainDatabaseEvent.send(event1 => Try {
8     event1.ImportDateTime := newImportDate
9     event1.StepId := event.StepId
10
11    event.analysisData.items.foreach {
12
13      case analysisDataType: ComplexCollectionVariableDefinition[AnalysisDataType] =>
   ↪{
14
15        val analysisDataItem = event1.analysisDataTable.newItem
16
17        analysisDataItem.name := analysisDataType.name
18        analysisDataItem.length := analysisDataType.length
19      }
```

(continues on next page)

(continued from previous page)

```
    }
  })
}
```

Conditions (If - statements)

Within a Rule it is possible to implement conditions using Scala's conditional expressions. **If** statements can be used to test a condition before executing the block below. This can be used for example to check when a certain condition is met to execute an event (line 3).

Listing 11: Code constructs - Conditions

```
1  equipment.ActiveOrder.State mapTo { variable =>
2   logger.info(s"Active order state: ${variable.value} - Processing Finished")
3   if (variable.value == 3) {
4     mes.NotifyOrderFinished.send(event => {
5       Try{
6         event.EquipmentId := equipment.EquipmentInformation.EquipmentType
7         event.OrderNr := equipment.ActiveOrder.OrderInformation.OrderNo
8         event.ProductNumber := equipment.ActiveOrder.OrderInformation.ProductNo
9         event.QuantityOk := equipment.ActiveOrder.QuantityOk
10        event.QuantityNOk := equipment.ActiveOrder.QuantityNOk
11      }
12    })
13  }
14 }
```

Exception Handling (Try/Catch)

Exception Handling is an integrated part of the SMART**UNIFIER** mapping logic. The **mapTo** and **send** callbacks expect a return value of the Scala type **Try**. In case of an exception happening in one of the rules, the SMART**UNIFIER** logs the exception and shows a notification in the manager. Supported Communication Channels execute further actions once an exception occurred E.g., the File Reader Channel moves a file that initially triggered a Rule into an error folder.

In the example below, the **Try** block is placed after each command and event call (line 2 and 4).

Listing 12: Code constructs - Exception Handling

```
1  database.update mapTo { (updateCommand, reply) =>
2   Try {
3     api.AmorphAPI.send(event =>
4       Try {
5
```

(continues on next page)

(continued from previous page)

```
        event.id := updateCommand.Identifier.Id
        event.name := updateCommand.Identifier.Name

        updateCommand.Status.items.foreach(item => {
          val statusItem = event.status.newItem
          statusItem.index := item.Index
          statusItem.value := itemValue
        })
      }
    )
  }
}
```

Target Source Combinations

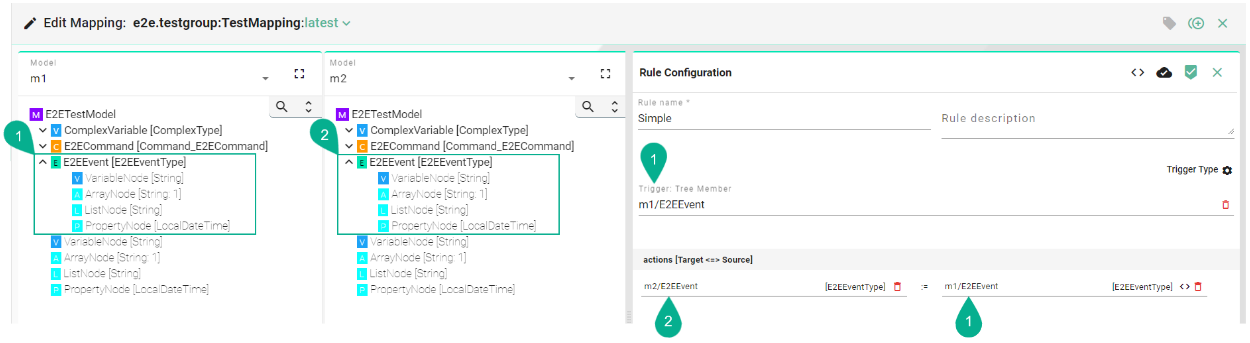
A Rule is defined by its elements: Trigger, Target and Source. Each element is a *node* assigned from an *Information Model*.

The possible combinations between Target and Source are independent of the Trigger Type. There are two kinds of assignments:

Simple

When Source and Target are of the same *data type* they can be directly assigned to one another.

Sample Assignments:



Rule name *: Simple
Rule description
Trigger: Tree Member: m1/E2EEvent
actions [Target <=> Source]
m2/E2EEvent [E2EEventType] := m1/E2EEvent [E2EEventType]

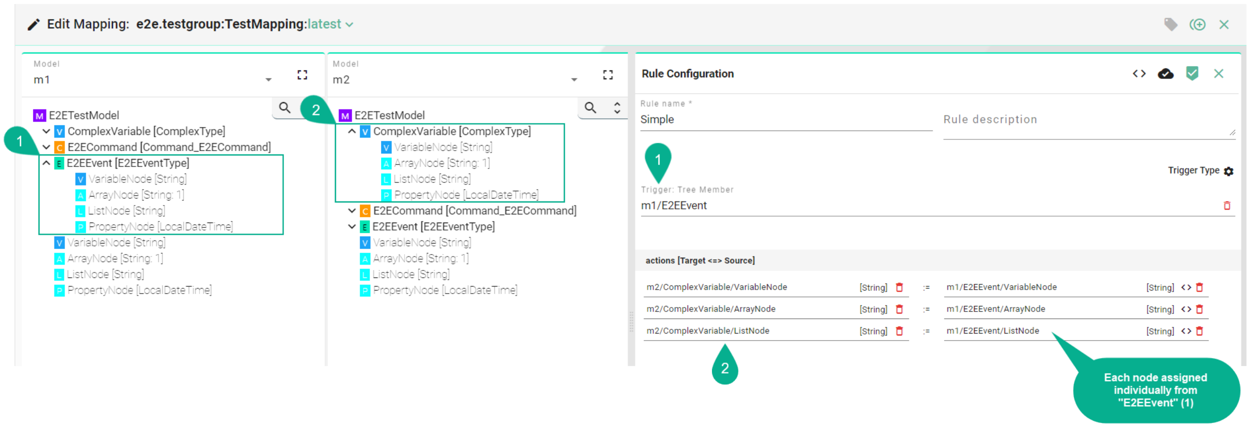
Based on the combinations of a Rule elements, all the scenarios are listed in the table below.

Trigger	Target	Source
Any Source *node*	Any Target *node*	Any Source *node*
Fixed Rate Scheduler		
Fixed Delay Scheduler		

Complex

When Source and Target differ in the *data type* their children *nodes* have to be assigned individually.

Sample Assignments:



Edit Mapping: e2e.testgroup:TestMapping:latest

Model: m1

- E2ETestModel
  - ComplexVariable [ComplexType]
  - E2ECommand [Command_E2ECommand]
  - E2EEvent [E2EEventType]
    - VariableNode [String]
    - ArrayNode [String: 1]
    - ListNode [String]
    - PropertyNode [LocalDateTime]
  - VariableNode [String]
  - ArrayNode [String: 1]
  - ListNode [String]
  - PropertyNode [LocalDateTime]

Model: m2

- E2ETestModel
  - ComplexVariable [ComplexType]
    - VariableNode [String]
    - ArrayNode [String: 1]
    - ListNode [String]
    - PropertyNode [LocalDateTime]
  - E2ECommand [Command_E2ECommand]
  - E2EEvent [E2EEventType]
  - VariableNode [String]
  - ArrayNode [String: 1]
  - ListNode [String]
  - PropertyNode [LocalDateTime]

Rule Configuration

Rule name *: Simple

Rule description

Trigger Type

Trigger: Tree Member: m1/E2EEvent

actions [Target <=> Source]

Target			Source	
m2/ComplexVariable/VariableNode	[String]	:=	m1/E2EEvent/VariableNode	[String]
m2/ComplexVariable/ArrayNode	[String]	:=	m1/E2EEvent/ArrayNode	[String]
m2/ComplexVariable/ListNode	[String]	:=	m1/E2EEvent/ListNode	[String]

Based on the combinations of a Rule elements, all the scenarios are listed in the table below.

Trigger	Target	Source
Variable of a custom type	Variable	Variable
		Variable of a custom type
		List
		Array
	Custom type Variable	Variable
		Variable of a custom type
		List
		Array
	Variable of a custom type	Variable
		Variable of a custom type
		List
		Array
	Event	Variable
		Variable of a custom type
		List
		Array
	Command	Variable
		Variable of a custom type
		List
		Array
	List	Variable
		Variable of a custom type
		List
		Array
	Array	Variable

continues on next page

Table 2 – continued from previous page

Trigger	Target	Source
		Variable of a custom type
		List
		Array
Variable	Variable	Variable
		Variable of a custom type
		List
		Array
	Custom type Variable	Variable
		Variable of a custom type
		List
		Array
	Variable of a custom type	Variable
		Variable of a custom type
		List
		Array
	Event	Variable
		Variable of a custom type
		List
		Array
	Command	Variable
		Variable of a custom type
		List
		Array
	List	Variable
		Variable of a custom type
		List
		Array
	Array	Variable
		Variable of a custom type
		List
		Array
Property of a custom type	Variable	Variable
		Variable of a custom type
		List
		Array
	Custom type Variable	Variable
		Variable of a custom type
		List
		Array
	Variable of a custom type	Variable
		Variable of a custom type
		List
		Array
	Event	Variable

continues on next page

Table 2 – continued from previous page

Trigger	Target	Source
		Variable of a custom type
		List
		Array
	Command	Variable
		Variable of a custom type
		List
		Array
	List	Variable
		Variable of a custom type
		List
		Array
	Array	Variable
		Variable of a custom type
		List
		Array
Property	Variable	Variable
		Variable of a custom type
		List
		Array
	Custom type Variable	Variable
		Variable of a custom type
		List
		Array
	Variable of a custom type	Variable
		Variable of a custom type
		List
		Array
	Event	Variable
		Variable of a custom type
		List
		Array
	Command	Variable
		Variable of a custom type
		List
		Array
	List	Variable
		Variable of a custom type
		List
		Array
	Array	Variable
		Variable of a custom type
		List
		Array
Command	Variable	Variable

continues on next page

Table 2 – continued from previous page

Trigger	Target	Source
		Variable of a custom type
		Variable of a Command
		List
		Array
	Custom type Variable	Variable
		Variable of a custom type
		Variable of a Command
		List
		Array
	Variable of a custom type	Variable
		Variable of a custom type
		Variable of a Command
		List
		Array
	Event	Variable
		Variable of a custom type
		Variable of a Command
		List
		Array
	Command	Variable
		Variable of a custom type
		Variable of a Command
		List
		Array
	List	Variable
		Variable of a custom type
		Variable of a Command
		List
		Array
	Array	Variable
		Variable of a custom type
		Variable of a Command
		List
		Array
Event	Variable	Variable of a custom type
		Variable of an Event
		Variable
		List
		Array
	Custom type Variable	Variable of a custom type
		Variable of an Event
		Variable
		List
		Array

continues on next page

Table 2 – continued from previous page

Trigger	Target	Source
	Variable of a custom type	Variable of a custom type
		Variable of an Event
		Variable
		List
		Array
	Event	Variable of a custom type
		Variable of an Event
		Variable
		List
		Array
	Command	Variable of a custom type
		Variable of an Event
		Variable
		List
		Array
	List	Variable
		Variable of a custom type
		Variable of an Event
		List
		Array
	Array	Variable
		Variable of a custom type
		Variable of an Event
		List
		Array

2.4 Device Types

2.4.1 What are Device Types

The SMART**UNIFIER** Device Type acts as a template for a Communication Instance that can be reused (i.e., one instance represents one equipment). Multiple Communication Instances, which share common configuration parameters, can be created based on one Device Type.

In the simplest integration scenario of a single equipment, one Device Type must be created to create a Communication Instance. This enables to create and spin up two Instances for test and production operation.

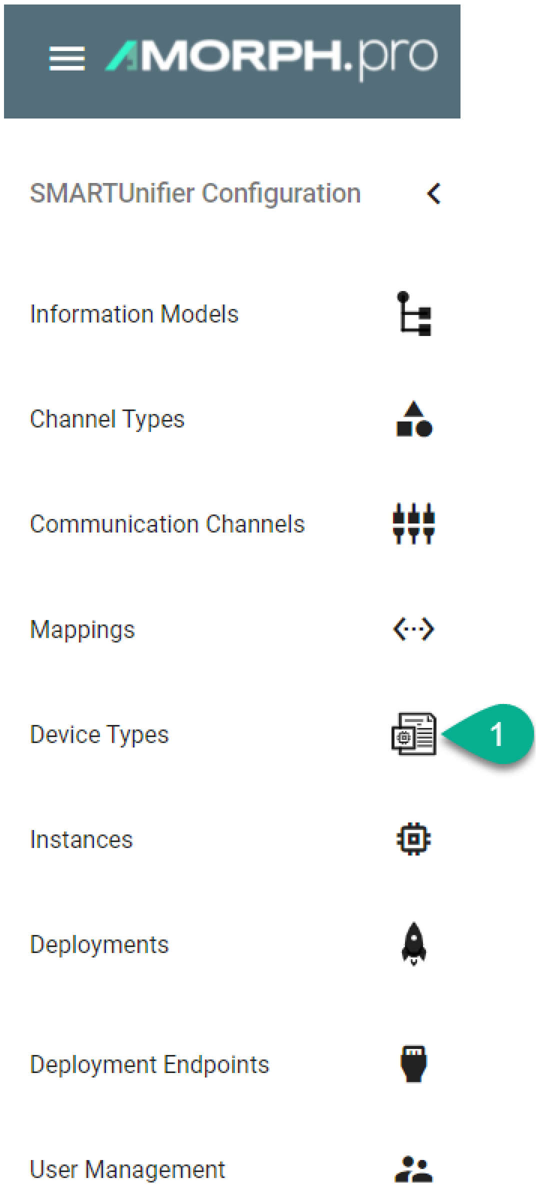
A Device Type itself contains one or multiple Mappings, which allows to build up communication flows between multiple systems (how this can be done is shown in the SMARTUNIFIER Demonstrator). Each Mapping contains one or multiple Information Models with its associated Communication Channel.

Device Types are especially important, when integrating several similar pieces of equipment or devices.

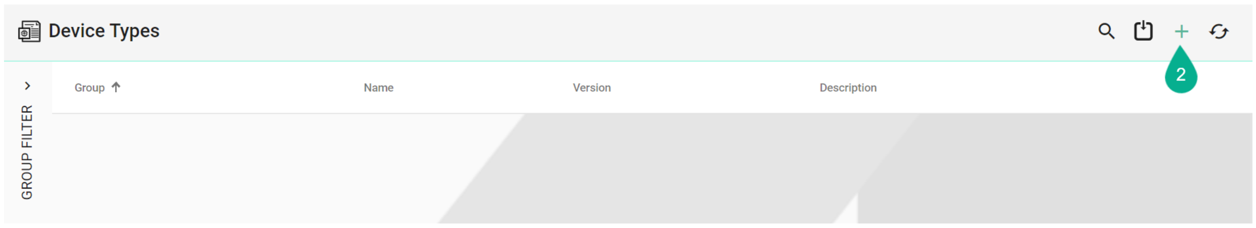
2.4.2 How to create a new Device Type

Follow the steps described below to create a SMART**UNIFIER** Device Type.

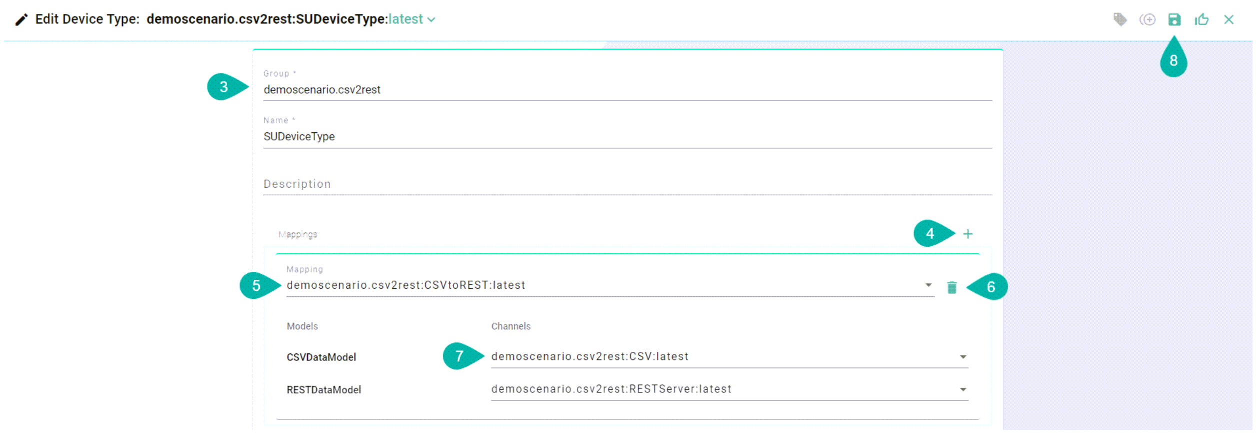
- Select the SMART**UNIFIER** Device Type perspective **(1)**.



- Click on the "Add Device Type" button in the upper right corner **(2)**.



- The creation of a Device Type is divided into two parts. Firstly, provide basic information about the Device Type, such as its Group, Name, and Version. Additionally, you can provide a short description, if desired **(3)**.
- In the next step, provide one or multiple previously created *Mappings*. To do so, click the "Add Mapping" button **(4)**. After selecting a Mapping **(5)** the associated Information Models will appear. If the wrong Mapping was selected, click the "Delete Mapping" button to remove it from the Device Type **(6)**. Finally. select a *Communication Channel* for each *Information Model* from the Drop-Down **(7)**.
- To save the new Device Type, click the "Save" button located at the top right corner of the screen **(8)**.


Edit Device Type: demoscenario.csv2rest:SUDeviceType:latest

Group *: demoscenario.csv2rest

Name *: SUDeviceType

Description

Mappings

Mapping: demoscenario.csv2rest:CSVtoREST:latest

Models	Channels
CSVDataModel	demoscenario.csv2rest:CSV:latest
RESTDataModel	demoscenario.csv2rest:RESTServer:latest

2.5 Communication Instances

2.5.1 What are Instances

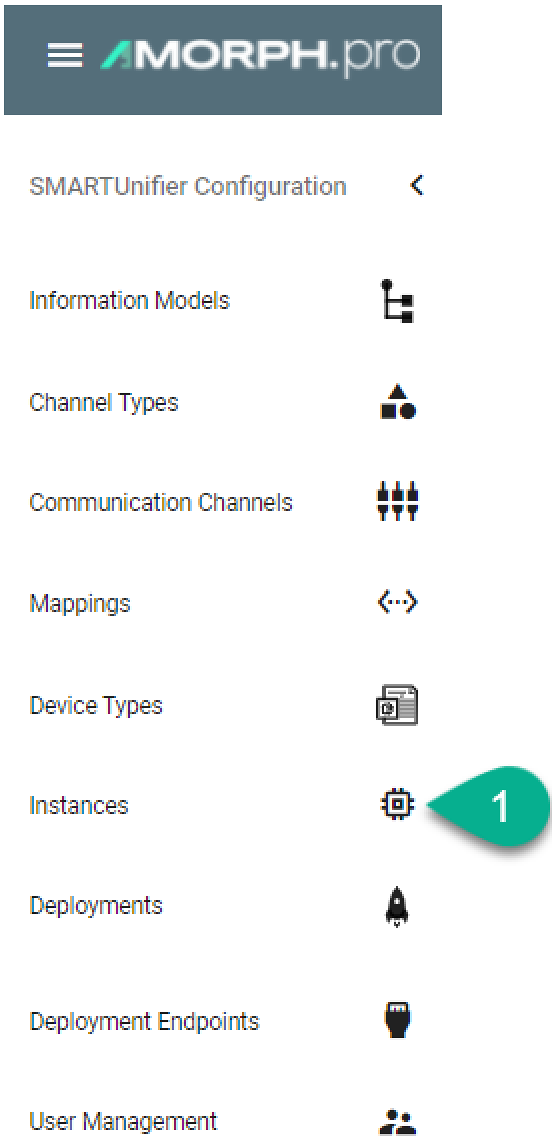
A SMART**UNIFIER** Instance is a dynamically created application that can be deployed to any suitable IT resource (e.g., Equipment PC, Server, Cloud), and which provides the connectivity functionality configured. Therefore, a SMART**UNIFIER** Instance uses one or multiple Mappings and selected Communication Channels from a previously defined *Device Type*.

The Communication Instance is the final component and when deployed it acts as a standalone application that provides the connectivity between two or multiple systems. The granular configuration of the Communication Channels can be done on this level (e.g., changing the IP address) which is useful when there are multiple Communication Instances created based on the same Device Type.

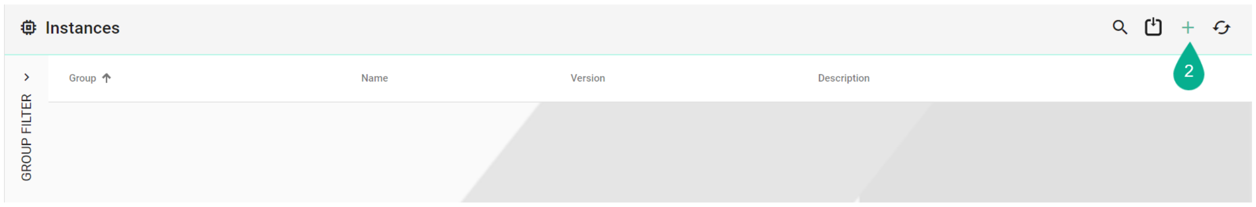
2.5.2 How to create a new Instance

Follow the steps described below to create a SMART**UNIFIER** Instance.

- Select the SMART**UNIFIER** Instances perspective **(1)**.

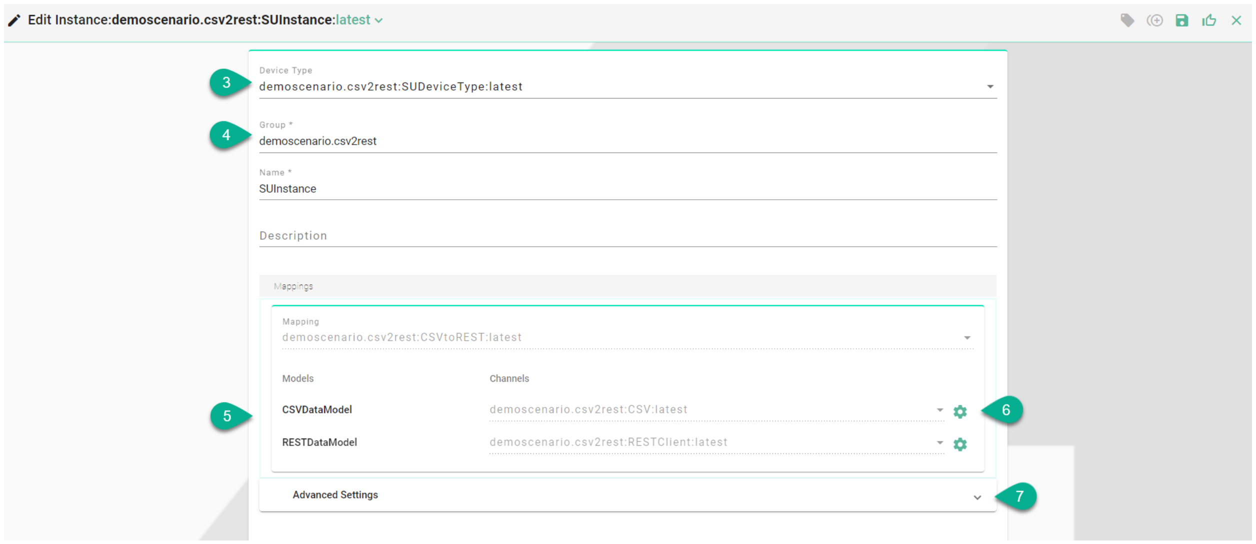


- Click on the "Add Instance" button from the upper right corner **(2)**.



- Select a Device Type from the Drop-Down **(3)**

- The Instance details are automatically populated from the Device Type **(4)** but you can still modify the Group, Name, Version, and Description fields as needed.
- Mappings that are defined in the Device Type will appear in the Mapping area **(5)**.
- To change the existing configuration or if no configuration has been made yet, click the "Configure" button **(6)**


Edit Instance:demoscenario.csv2rest:SUInstance:latest

Device Type: demoscenario.csv2rest:SUDeviceType:latest

Group *: demoscenario.csv2rest

Name *: SUInstance

Description

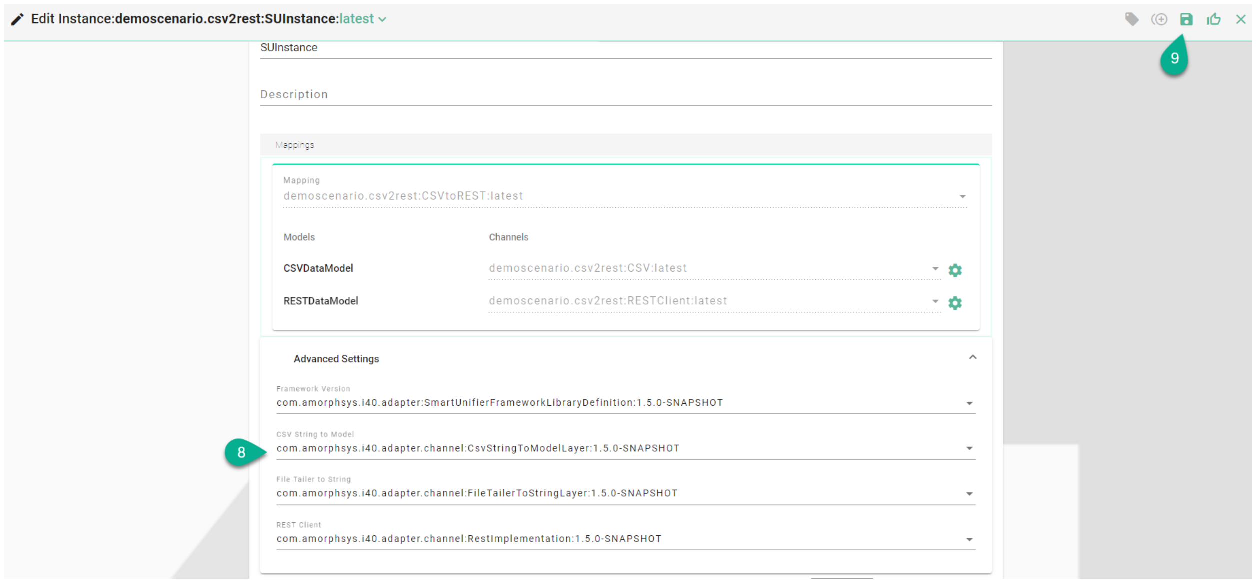
Mappings

Mapping: demoscenario.csv2rest:CSVtoREST:latest

Models	Channels
CSVDataModel	demoscenario.csv2rest:CSV:latest
RESTDataModel	demoscenario.csv2rest:RESTClient:latest

Advanced Settings

- Expand the **Advanced Settings** option **(7)** to select the framework version **(8)** for the Communication Channels. Allows backwards compatibility for Communication Instances created with previous versions of SMART**UNIFIER**.


Edit Instance:demoscenario.csv2rest:SUInstance:latest

SUInstance

Description

Mappings

Mapping: demoscenario.csv2rest:CSVtoREST:latest

Models	Channels
CSVDataModel	demoscenario.csv2rest:CSV:latest
RESTDataModel	demoscenario.csv2rest:RESTClient:latest

Advanced Settings

Framework Version: com.amorphsys.i40.adapter:SmartUnifierFrameworkLibraryDefinition:1.5.0-SNAPSHOT

CSV String to Model: com.amorphsys.i40.adapter.channel:CsvStringToModelLayer:1.5.0-SNAPSHOT

File Tailer to String: com.amorphsys.i40.adapter.channel:FileTailerToStringLayer:1.5.0-SNAPSHOT

REST Client: com.amorphsys.i40.adapter.channel:RestImplementation:1.5.0-SNAPSHOT

- Save the SMART**UNIFIER** Instance by clicking the "Save" button **(9)**
- In order to deploy, run and stop the Instance navigate to the *Deployment* perspective.

CHAPTER THREE

CONFIGURATION COMPONENT MANAGEMENT

SMART**UNIFIER** provides a comprehensive management of the configuration components:

- *Group Filter*
- *Component Version Control*
- *Operations*

In order to keep the SMART**UNIFIER** configuration components organized take a look on *how to name the configuration components*.

3.1 Naming Convention

The configuration process of a SMART**UNIFIER** Instance involves the creation of *configuration components* such as Information Models, Communication Channels, Mappings, Device Types and the Communication Instances itself. Each configuration component needs to have a group and a name as an Identifier. The Identifier helps to organize configuration components and to build up a hierarchical structure.

Group

The group can be used to leverage the concept of paths like they are used in file systems by concatenating logical entities together. In most cases there is already a naming convention or a certain style of naming for equipment's in place. This can be taken over when naming and organizing the configuration components.

Name

The Name should indicate the entity that is going to be integrated. Below some good and bad name examples are listed:

Configuration Component	Good Naming	Bad Naming
Information Model	Screwdriver1	ScrewdriverNr7Floor2SectionA
Communication Channel	Screwdriver1	OpcUaClient
Mapping	Screwdriver1ToOee	OpcUaClientToDatabase
Device Type	Screwdriver1OeeIntegration	ScrewdriverNr7Floor2SectionAIntegraton
Instance	Screwdriver1OeeIntegration	OpcUaClientDatabaseIntegration

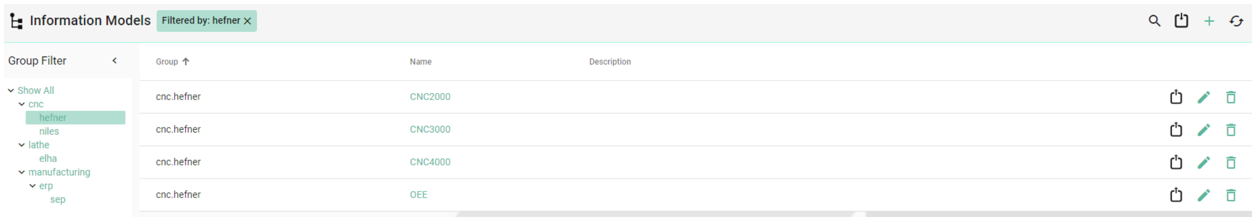
Naming Examples

We recommend the following naming convention for better comprehensibility.

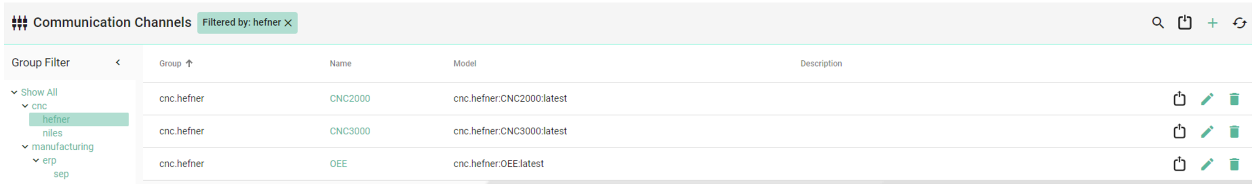
Information Models & Communication Channels

Information Models and its associated Communication Channels should both have the same naming.

System	Group Format	Name Format	Example Group	Example Name
Equipment / Machines	EquipmentType.Manufacturer	EquipmentName	cnc.hefner	CNC3000
IT-system	SystemType.Manufacturer	SystemName	erp.sep	SEP PRO 5000



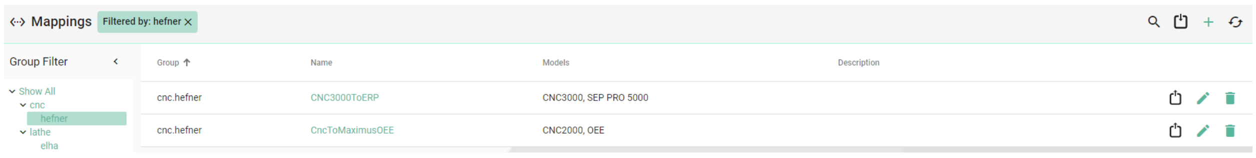
Group	Name	Description
cnc.hefner	CNC2000	
cnc.hefner	CNC3000	
cnc.hefner	CNC4000	
cnc.hefner	OEE	



Group	Name	Model	Description
cnc.hefner	CNC2000	cnc.hefner:CNC2000:latest	
cnc.hefner	CNC3000	cnc.hefner:CNC3000:latest	
cnc.hefner	OEE	cnc.hefner:OEE:latest	

Mappings

Group Format	Name Format	Example Group	Example Name
EquipmentType.Manufacturer	EquipmentToSystem	cnc.hefner	CNC3000ToERP

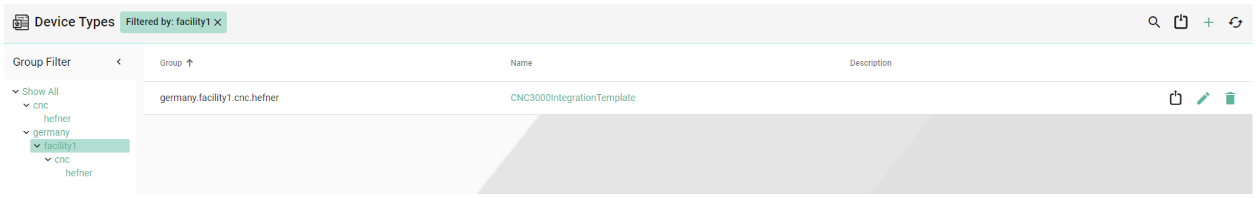

Mappings Filtered by: hefner

Group Filter
- Show All
  - cnc
    - hefner
  - lathe
    - elha

Group	Name	Models	Description
cnc.hefner	CNC3000ToERP	CNC3000, SEP PRO 5000	
cnc.hefner	CncToMaximusOEE	CNC2000, OEE	

Device Types

Group Format	Name Format	Example Group	Example Name
Re- gion.PlantName.EquipmentM	EquipmentToSys- temIntegration	ger- many.facility1.cnc.h	CNC3000Integration mplate


Device Types Filtered by: facility1

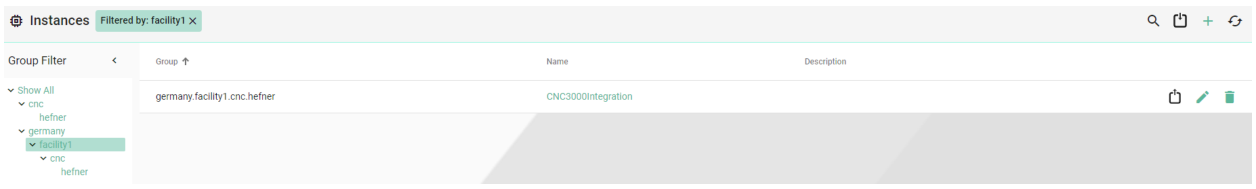
Group Filter
- Show All
  - cnc
    - hefner
  - germany
    - facility1
      - cnc
        - hefner

Group	Name	Description
germany.facility1.cnc.hefner	CNC3000IntegrationTemplate	

Communication Instances

The Communication Instance provides the connectivity between the physical equipment on the shopfloor and IT-systems. It is the most granular configuration component that is created.

Group Format	Example Group	Example Name
Region.PlantName.EquipmentManufacturer	ger- many.facility1.cnc.hefner	CNC3000Integration


Instances Filtered by: facility1

Group Filter
- Show All
  - cnc
    - hefner
  - germany
    - facility1
      - cnc
        - hefner

Group	Name	Description
germany.facility1.cnc.hefner	CNC3000Integration	

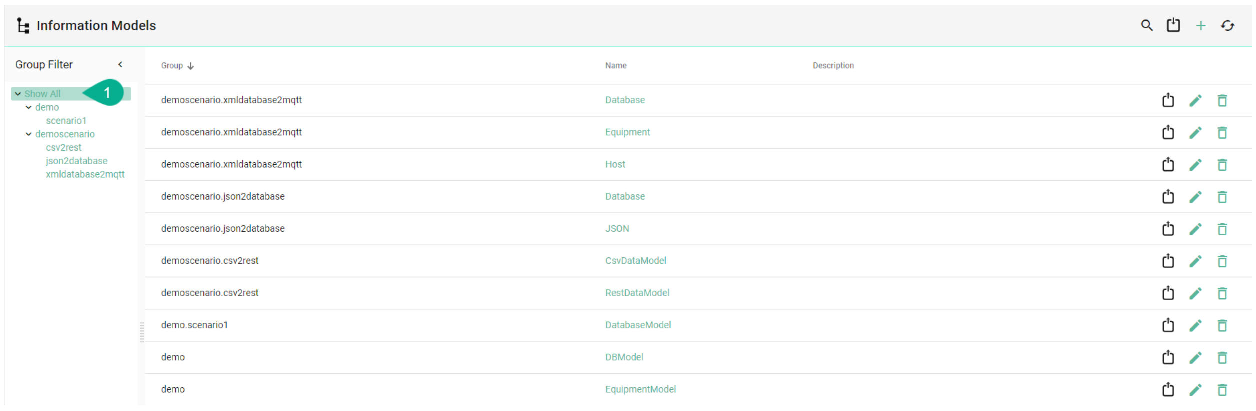
3.2 Group Filter

With Group Filter the configuration components can be visualized in a hierarchical structure, providing an upper lever configuration management.

Group Filter provides the possibility to restrict the number of components according to the substrings in the Group.

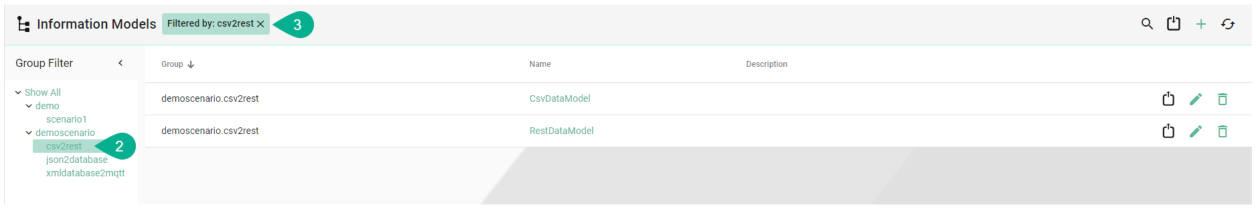
The Group Name contains substrings separated by a dot “.”. The Group Filter is then able to visualizes the Group Names in a hierarchical structure.

The *Show All* filter enables the view of all components **(1)**.



Group	Name	Description
demoscenario.xmldatabase2mqtt	Database	
demoscenario.xmldatabase2mqtt	Equipment	
demoscenario.xmldatabase2mqtt	Host	
demoscenario.json2database	Database	
demoscenario.json2database	JSON	
demoscenario.csv2rest	CsvDataModel	
demoscenario.csv2rest	RestDataModel	
demo.scenario1	DatabaseModel	
demo	DBModel	
demo	EquipmentModel	

To apply a filter, click one of the items in the Group Filter list **(2)**. At the top of the table, the selected filter is visible **(3)**.



Group	Name	Description
demoscenario.csv2rest	CsvDataModel	
demoscenario.csv2rest	RestDataModel	

Removing the filter is possible by either clicking the selected item again, selecting the *Show All* option or by clicking the cross in the filter at the top of the table.

3.3 Component Version Control

Component Version Control enables users to version SMART**UNIFIER** configuration components such as Information Models, Communication Channels, Mappings, Device Types and Communication Instances.

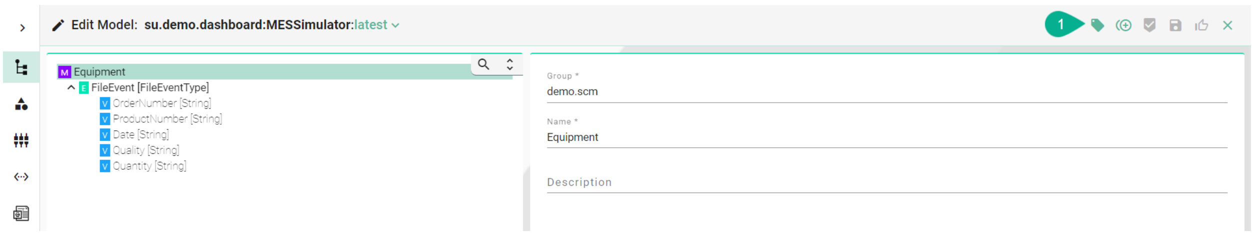
By default, SMART**UNIFIER** is using the Component Version Control internally - therefor no configuration is needed. Another option is to point to an external version control system like Gitea. In order to setup an external version control check out the SMART**UNIFIER** Installation Guide.

How it works: SMART**UNIFIER** creates a repository for each configuration component. Configuration components can be released using tags which reference a specific point in the Git history. After a tag has been created (equivalent to release of a configuration component) there will be no further history of commits/changes. This means that the configuration component can not be edited any further.

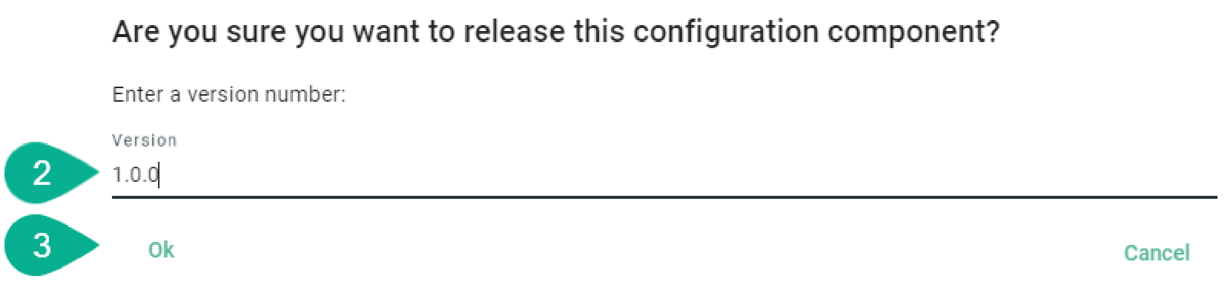
3.3.1 How to release configuration components

In order to release a configuration component follow the steps below:

1. Go to an edit page of a configuration component and click the **release** button.



1. Enter a **version number**.
2. Click **Ok** to confirm.

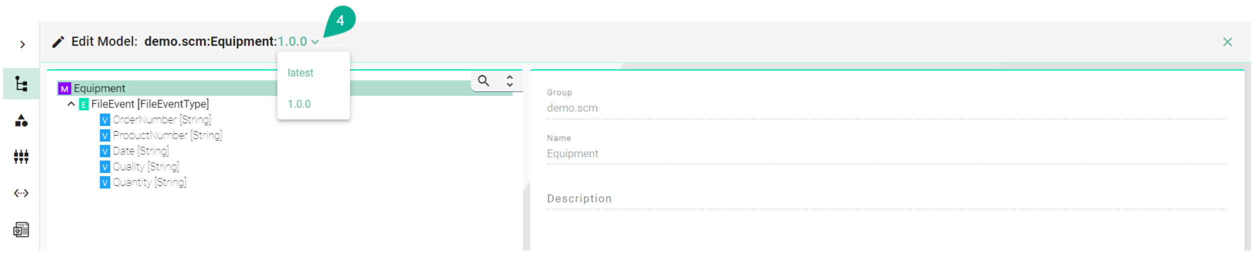

Are you sure you want to release this configuration component?

Enter a version number:

Version

1.0.0

4. Open the version drop-down to change between **latest** and other **tags**.



Note: Once a configuration component is released you can no longer edit the current tag. If changes are necessary select **latest**. Once you finished editing the final version you can repeat the release process as described above.

3.4 Operations

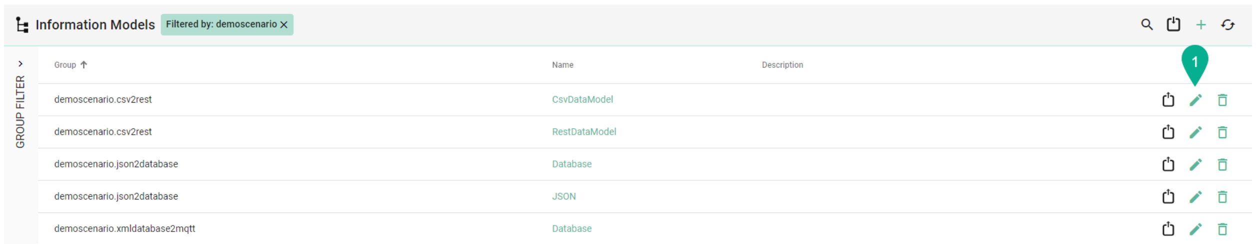
3.4.1 Add

The option to add/create a new component is described in the Instance Setup chapter, for each type:

- *Information Models*
- *Communication Channels*
- *Mappings*
- *Device Types*
- *Instances*
- *Deployments*
- *Deployment Endpoints*

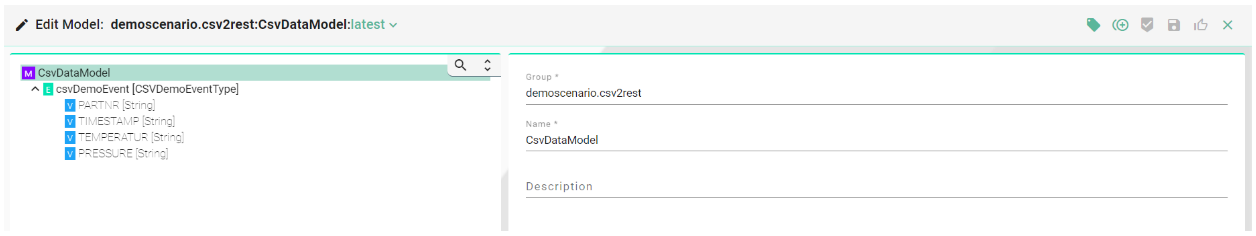
3.4.2 Edit

A component can be edited by clicking the **Edit** button **(1)**.



Group	Name	Description
demoscenario.csv2rest	CsvDataModel	
demoscenario.csv2rest	RestDataModel	
demoscenario.json2database	Database	
demoscenario.json2database	JSON	
demoscenario.xmldatabase2mqtt	Database	

The component is opened in the edit mode.



```
M CsvDataModel
  E csvDemoEvent [CSVDemoEventType]
      V PARTNR [String]
      V TIMESTAMP [String]
      V TEMPERATUR [String]
      V PRESSURE [String]
```

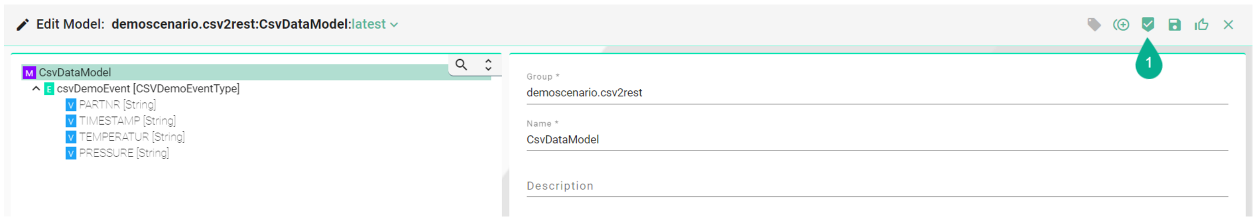
Group *: demoscenario.csv2rest
Name *: CsvDataModel
Description:

In the edit mode, the following operations are available:

- Clone
- Apply
- Save
- Save and Close
- Close

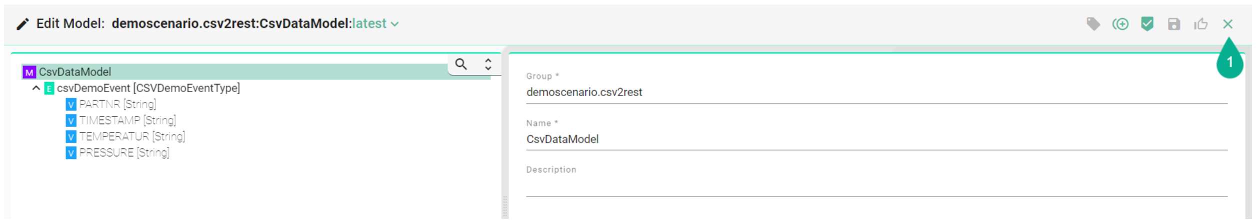
3.4.3 Apply

In the edit mode, after a new data input the **Apply** button **(1)** must be selected to validate/compile data.

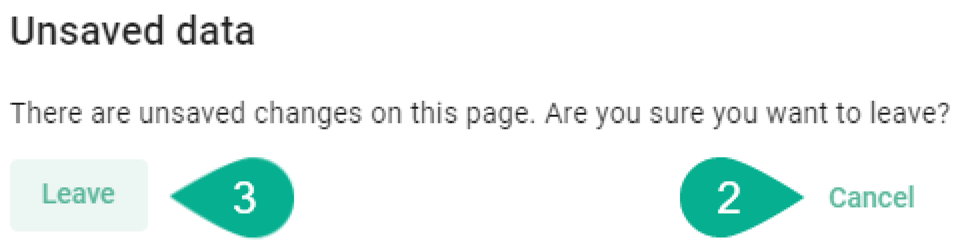


3.4.4 Exit Editing

The user can exit the edit mode by clicking on the **Close** button **(1)**.



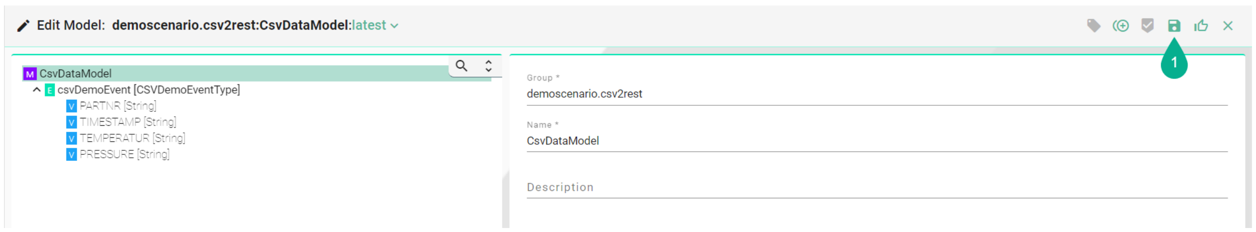
If the data is not saved, a pop-up appears and the user can select the **Cancel** button **(2)** to return to the edit mode and save the data or select the **Leave** button **(3)** to exit without saving.


Unsaved data

There are unsaved changes on this page. Are you sure you want to leave?

3.4.5 Save

In the Edit Mode, after applying the input data, the user can save the changes by clicking on the **Save** button **(1)**.



Edit Model: demoscenario.csv2rest:CsvDataModel:latest

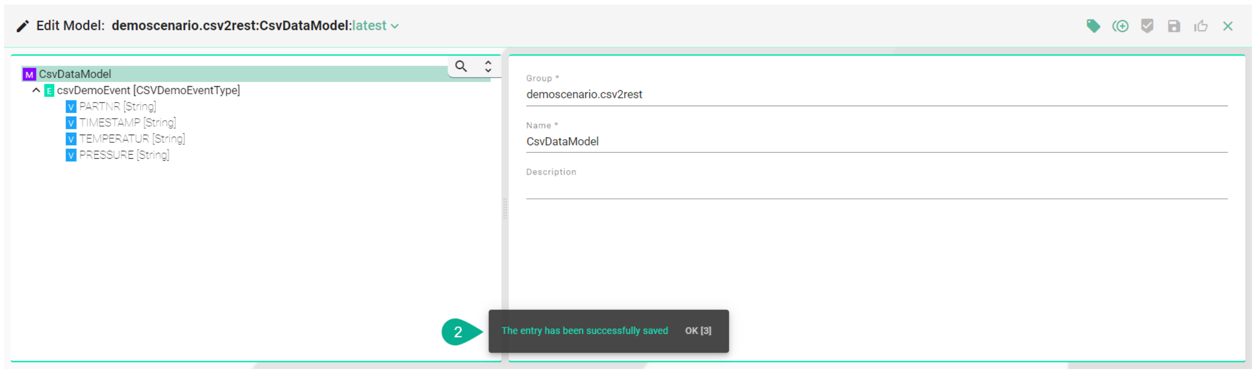
- M CsvDataModel
  - E csvDemoEvent [CSVDemoEventType]
    - V PARTNR [String]
    - V TIMESTAMP [String]
    - V TEMPERATUR [String]
    - V PRESSURE [String]

Group *: demoscenario.csv2rest

Name *: CsvDataModel

Description

A confirmation message appears **(2)**. The edit mode remains active.



Edit Model: demoscenario.csv2rest:CsvDataModel:latest

- M CsvDataModel
  - E csvDemoEvent [CSVDemoEventType]
    - V PARTNR [String]
    - V TIMESTAMP [String]
    - V TEMPERATUR [String]
    - V PRESSURE [String]

Group *: demoscenario.csv2rest

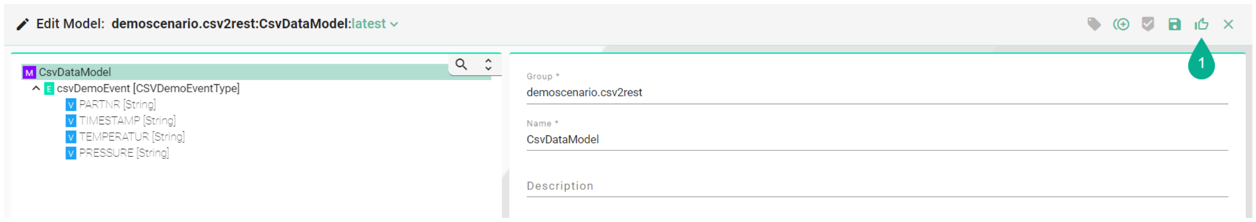
Name *: CsvDataModel

Description

The entry has been successfully saved OK [3]

3.4.6 Save and Close

When editing a component, after applying the input data, the user can save the changes and exit the edit mode by clicking on the **Save and Close** button **(1)**.



Edit Model: demoscenario.csv2rest:CsvDataModel:latest

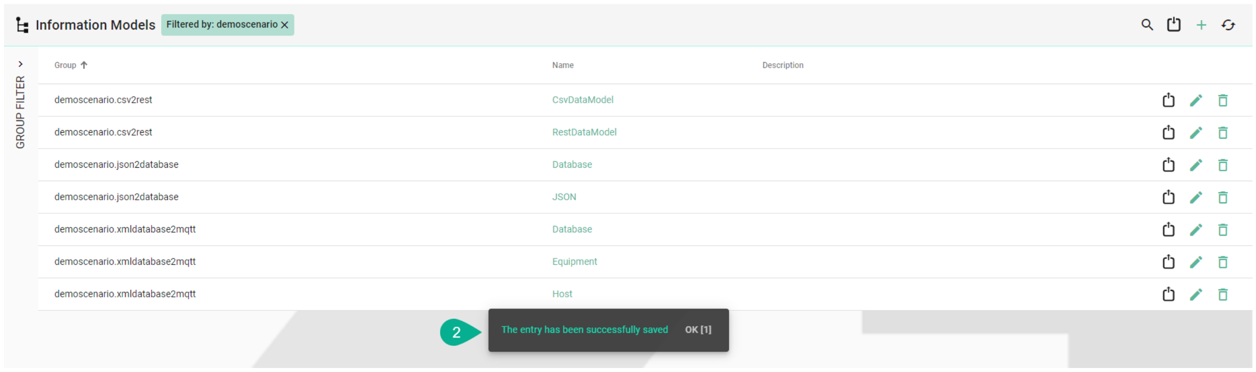
- M CsvDataModel
  - E csvDemoEvent [CSVDemoEventType]
    - V PARTNR [String]
    - V TIMESTAMP [String]
    - V TEMPERATUR [String]
    - V PRESSURE [String]

Group *: demoscenario.csv2rest

Name *: CsvDataModel

Description

A confirmation message appears **(2)**. The view mode is active.



Group	Name	Description
demoscenario.csv2rest	CsvDataModel	
demoscenario.csv2rest	RestDataModel	
demoscenario.json2database	Database	
demoscenario.json2database	JSON	
demoscenario.xmldatabase2mqtt	Database	
demoscenario.xmldatabase2mqtt	Equipment	
demoscenario.xmldatabase2mqtt	Host	

The entry has been successfully saved OK [1]

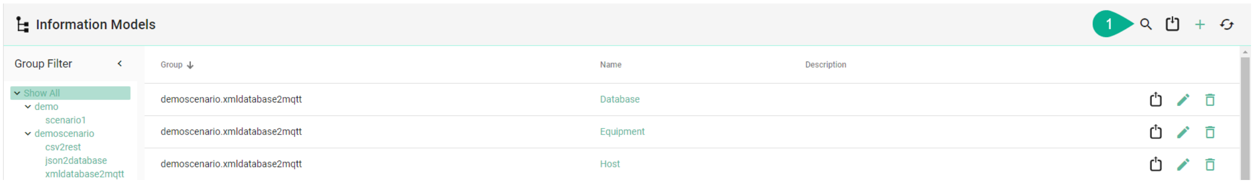
3.4.7 Search

The Search option allows the user to filter results by different criteria:

- Name
- Version
- Description

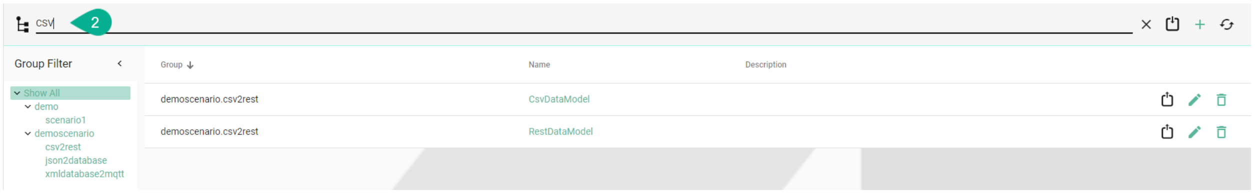
The search is not key sensitive and it works as a partial search, displaying all the results matching with the searched characters.

To search for a component, select the **Search** button **(1)** from the upper right corner.



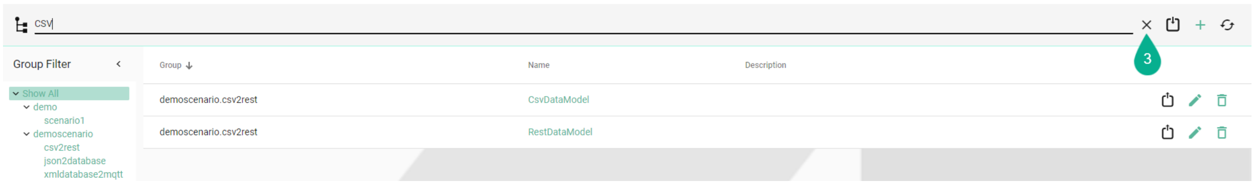
Group	Name	Description
demoscenario.xmldatabase2mqtt	Database	
demoscenario.xmldatabase2mqtt	Equipment	
demoscenario.xmldatabase2mqtt	Host	

Enter a search term **(2)**.



Group	Name	Description
demoscenario.csv2rest	CsvDataModel	
demoscenario.csv2rest	RestDataModel	

To cancel the search click on the **Close Search** button **(3)**.



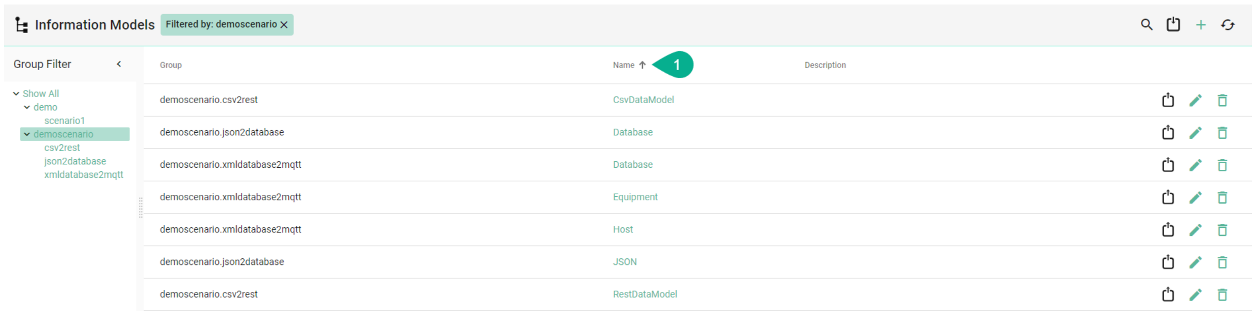
Group	Name	Description
demoscenario.csv2rest	CsvDataModel	
demoscenario.csv2rest	RestDataModel	

3.4.8 Sort

The information in the view mode can be sorted ascending or descending for each column:

- Group
- Name
- Version
- Description

To sort the information from a column click on the column header **(1)**. An arrow icon will indicate if the components are sorted ascending or descending.


Information Models Filtered by: demoscenario ×

Group Filter

- Show All
  - demo
    - scenario1
  - demoscenario
    - csv2rest
    - json2database
    - xmldatabase2mqtt

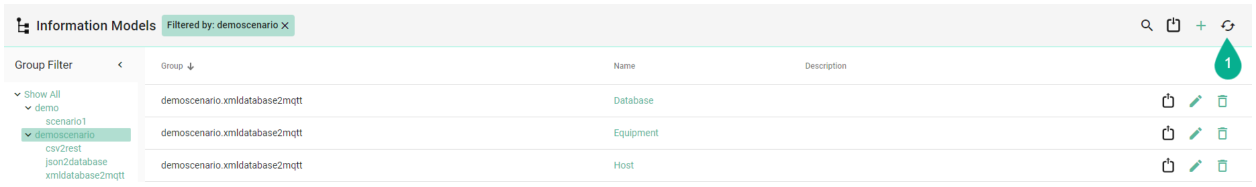
Group	Name ↑	Description
demoscenario.csv2rest	CsvDataModel	
demoscenario.json2database	Database	
demoscenario.xmldatabase2mqtt	Database	
demoscenario.xmldatabase2mqtt	Equipment	
demoscenario.xmldatabase2mqtt	Host	
demoscenario.json2database	JSON	
demoscenario.csv2rest	RestDataModel	

In the view mode on the right of each component, the following operations are available:

- Export
- Edit
- Delete

3.4.9 Reload

This option reloads the components from the repository by selecting the **Reload** button **(1)** from the upper right.


Information Models Filtered by: demoscenario ×

Group Filter

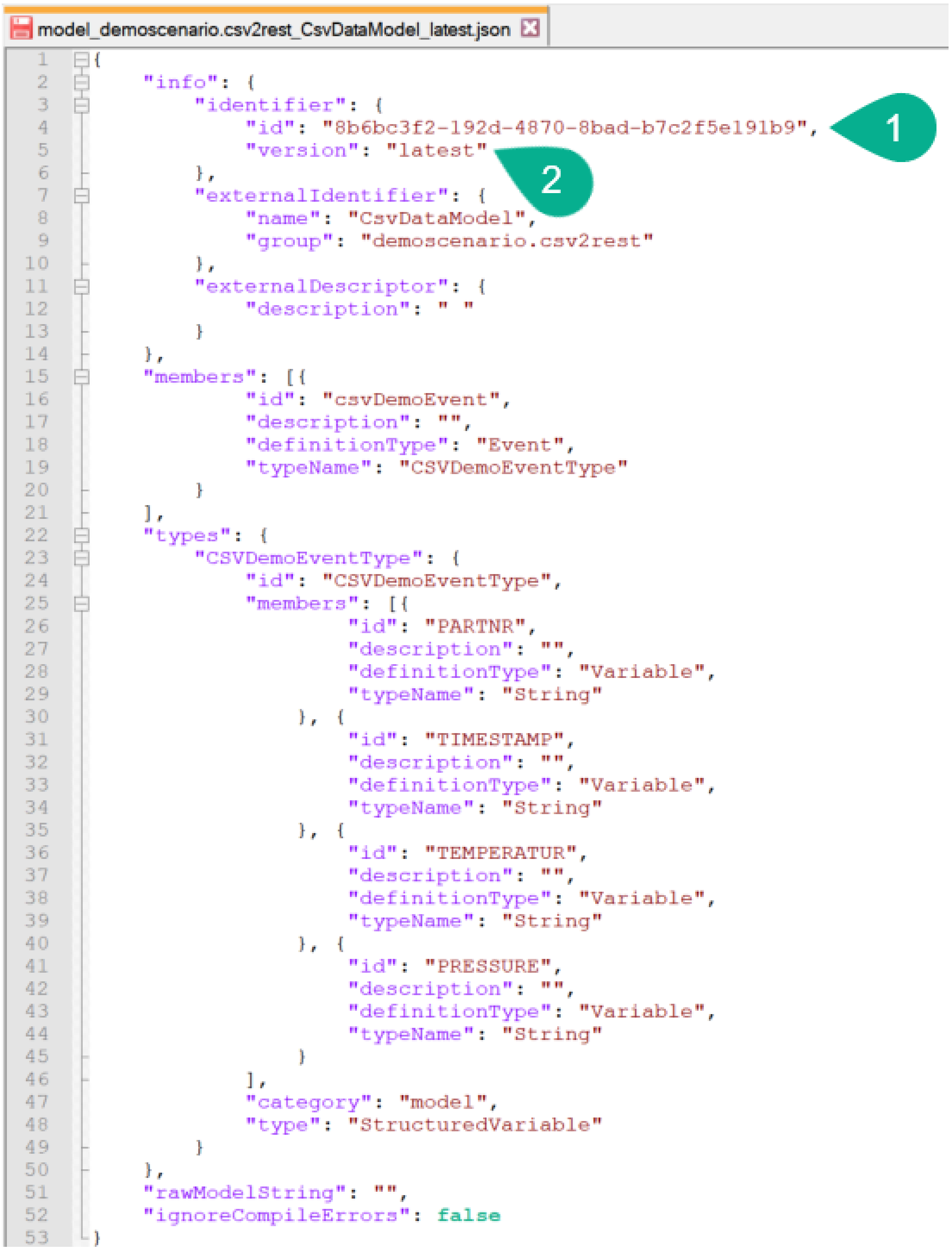
- Show All
  - demo
    - scenario1
  - demoscenario
    - csv2rest
    - json2database
    - xmldatabase2mqtt

Group ↓	Name	Description
demoscenario.xmldatabase2mqtt	Database	
demoscenario.xmldatabase2mqtt	Equipment	
demoscenario.xmldatabase2mqtt	Host	

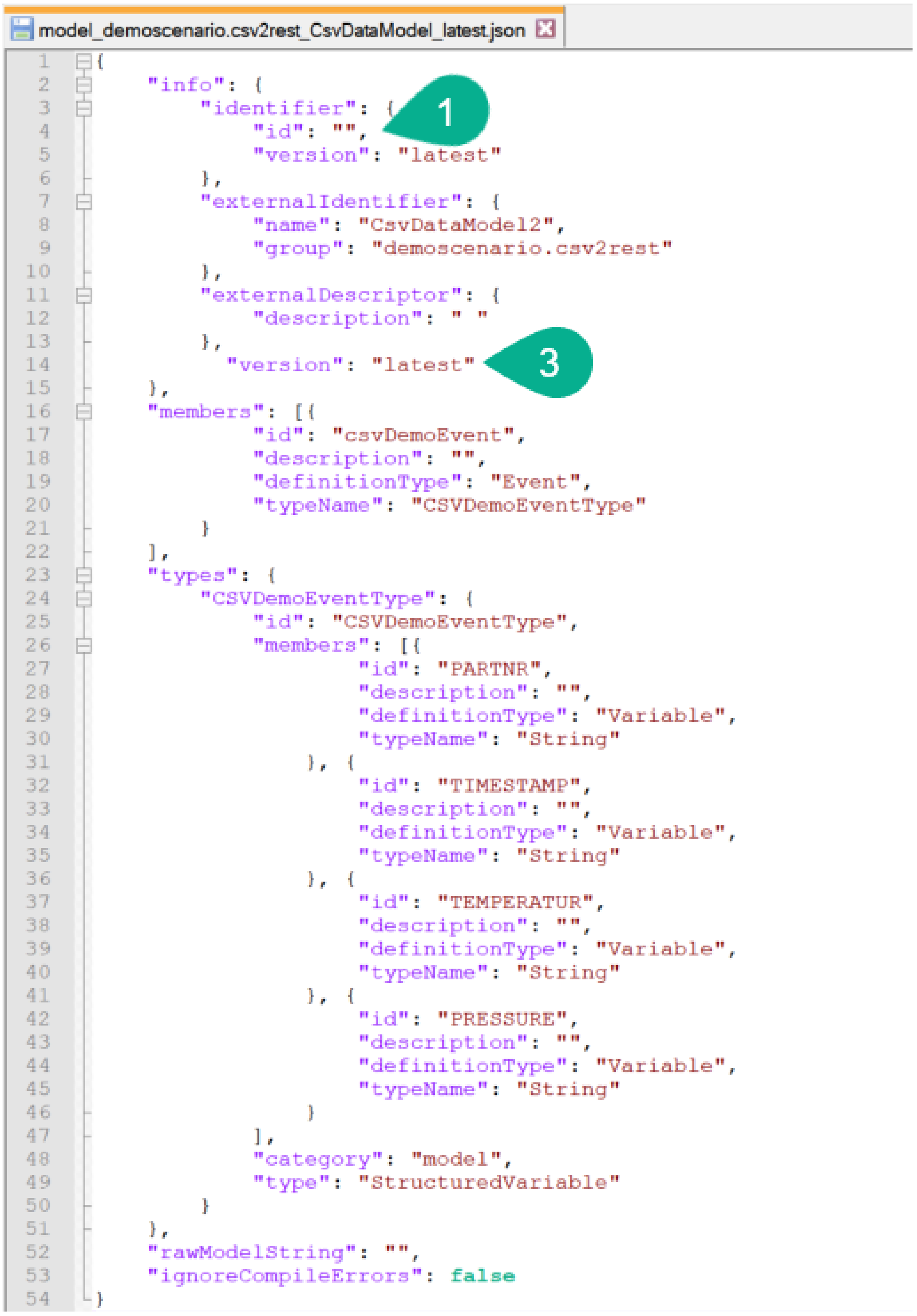
3.4.10 Import

This option allows the user to add to the scenario a new created or an exported component.

Before importing an exported component, open the JSON file and delete the component **id (1)** - when importing the database will generate a universally unique identifier (uuid). Also, copy **(2)** and paste **(3)** the **version** in the **info** section, as shown bellow.

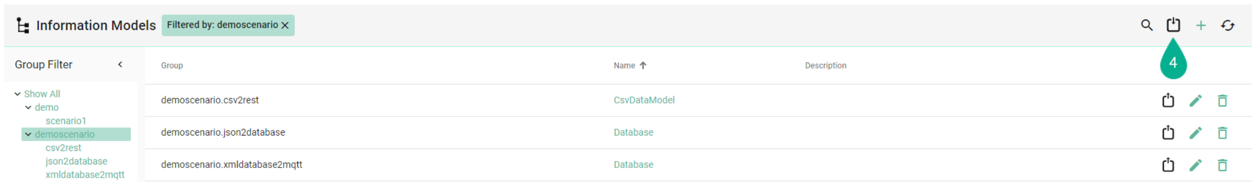


```
{
    "info": {
        "identifier": {
            "id": "8b6bc3f2-192d-4870-8bad-b7c2f5e191b9",
            "version": "latest"
        },
        "externalIdentifier": {
            "name": "CsvDataModel",
            "group": "demoscenario.csv2rest"
        },
        "externalDescriptor": {
            "description": " "
        }
    },
    "members": [{
            "id": "csvDemoEvent",
            "description": "",
            "definitionType": "Event",
            "typeName": "CSVDemoEventType"
        }
    ],
    "types": {
        "CSVDemoEventType": {
            "id": "CSVDemoEventType",
            "members": [{
                    "id": "PARTNR",
                    "description": "",
                    "definitionType": "Variable",
                    "typeName": "String"
                }, {
                    "id": "TIMESTAMP",
                    "description": "",
                    "definitionType": "Variable",
                    "typeName": "String"
                }, {
                    "id": "TEMPERATUR",
                    "description": "",
                    "definitionType": "Variable",
                    "typeName": "String"
                }, {
                    "id": "PRESSURE",
                    "description": "",
                    "definitionType": "Variable",
                    "typeName": "String"
                }
            ],
            "category": "model",
            "type": "StructuredVariable"
        }
    },
    "rawModelString": "",
    "ignoreCompileErrors": false
}
```



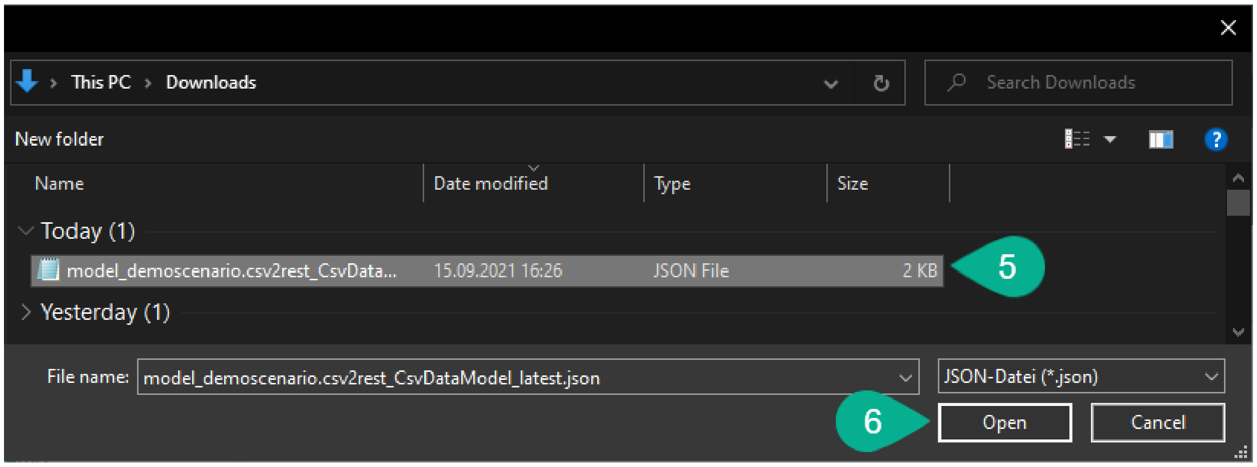
```
{
    "info": {
        "identifier": {
            "id": "",
            "version": "latest"
        },
        "externalIdentifier": {
            "name": "CsvDataModel2",
            "group": "demoscenario.csv2rest"
        },
        "externalDescriptor": {
            "description": " "
        },
          "version": "latest"
    },
    "members": [{
            "id": "csvDemoEvent",
            "description": "",
            "definitionType": "Event",
            "typeName": "CSVDemoEventType"
        }
    ],
    "types": {
        "CSVDemoEventType": {
            "id": "CSVDemoEventType",
            "members": [{
                    "id": "PARTNR",
                    "description": "",
                    "definitionType": "Variable",
                    "typeName": "String"
                }, {
                    "id": "TIMESTAMP",
                    "description": "",
                    "definitionType": "Variable",
                    "typeName": "String"
                }, {
                    "id": "TEMPERATUR",
                    "description": "",
                    "definitionType": "Variable",
                    "typeName": "String"
                }, {
                    "id": "PRESSURE",
                    "description": "",
                    "definitionType": "Variable",
                    "typeName": "String"
                }
            ],
            "category": "model",
            "type": "StructuredVariable"
        }
    },
    "rawModelString": "",
    "ignoreCompileErrors": false
}
```

To import, select the **Import** button **(4)** from the upper right.



Group	Name	Description
demoscenario.csv2rest	CsvDataModel	
demoscenario.json2database	Database	
demoscenario.xmldatabase2mqtt	Database	

A pop-up window appears. Chose the file **(5)** and select the **Open** button **(6)**.

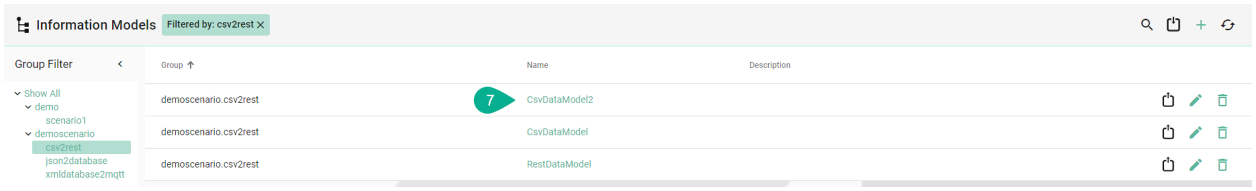


Name	Date modified	Type	Size
model_demoscenario.csv2rest_CsvData...	15.09.2021 16:26	JSON File	2 KB

File name: model_demoscenario.csv2rest_CsvDataModel_latest.json

JSON-Datei (*.json)

The imported component is now listed **(7)**.

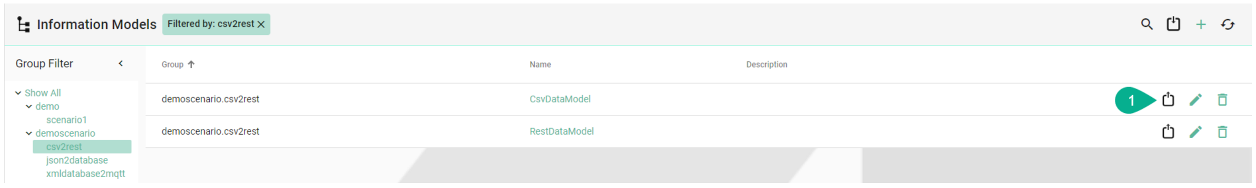


Group	Name	Description
demoscenario.csv2rest	CsvDataModel2	
demoscenario.csv2rest	CsvDataModel	
demoscenario.csv2rest	RestDataModel	

3.4.11 Export

The user has the option to export a component to the local machine.

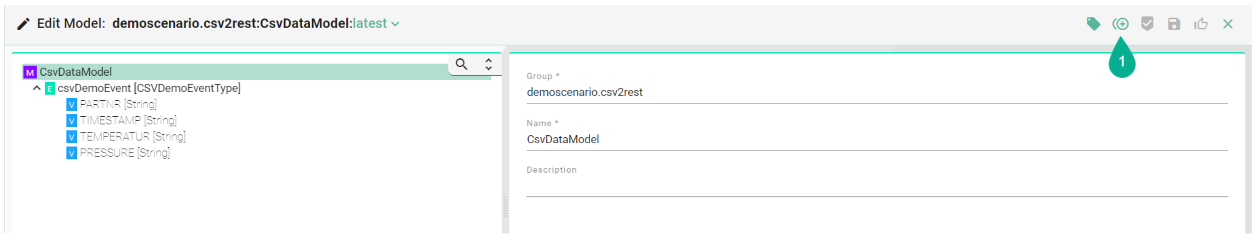
First, click on the **Export** button **(1)**.



Group	Name	Description
demoscenario.csv2rest	CsvDataModel	
demoscenario.csv2rest	RestDataModel	

3.4.12 Clone

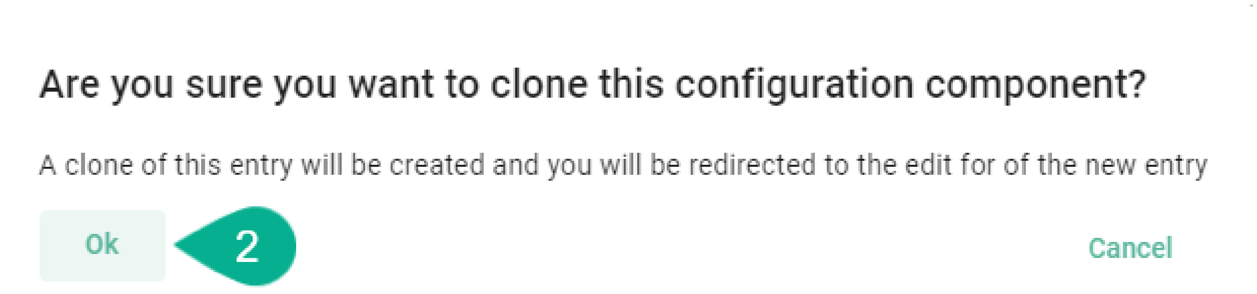
A component can be cloned from the edit mode, by selecting the **Clone** button. **(1)**.


Edit Model: demoscenario.csv2rest:CsvDataModel:latest

- M CsvDataModel
  - E csvDemoEvent [CSVDemoEventType]
    - V PARTNR [String]
    - V TIMESTAMP [String]
    - V TEMPERATUR [String]
    - V PRESSURE [String]

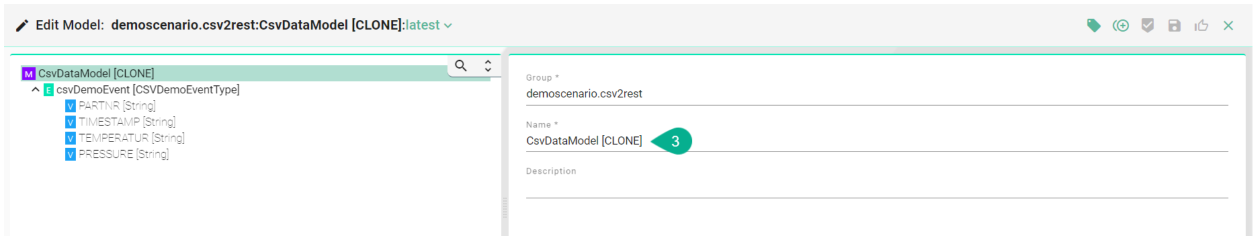
Group *: demoscenario.csv2rest
Name *: CsvDataModel
Description:

A pop-up appears, click on the **Ok** button **(2)**.


Are you sure you want to clone this configuration component?

A clone of this entry will be created and you will be redirected to the edit for of the new entry

The cloned component is visible, in edit mode, requiring to input a valid name **(4)**


Edit Model: demoscenario.csv2rest:CsvDataModel [CLONE]:latest

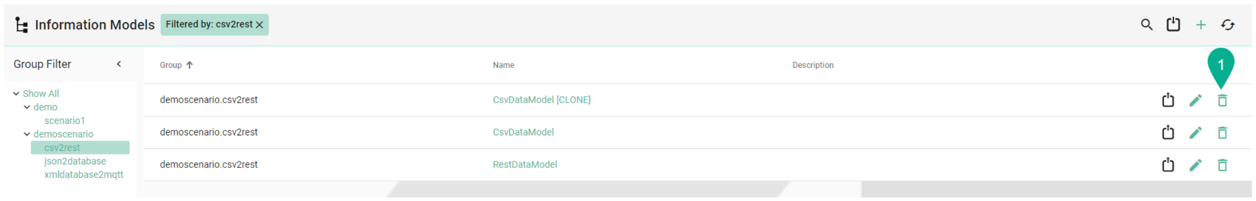
- M CsvDataModel [CLONE]
  - E csvDemoEvent [CSVDemoEventType]
    - V PARTNR [String]
    - V TIMESTAMP [String]
    - V TEMPERATUR [String]
    - V PRESSURE [String]

Group *: demoscenario.csv2rest
Name *: CsvDataModel [CLONE]
Description:

Note: The Clone operation is not available for the Deployment component.

3.4.13 Delete

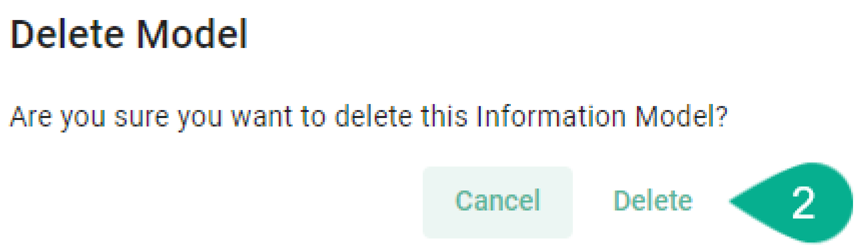
A component can be deleted by clicking the **Delete** button **(1)**.


Information Models Filtered by: csv2rest

Group Filter
- Show All
  - demo
    - scenario1
  - demoscenario
    - csv2rest
    - json2database
    - xmldatabase2mqtt

Group	Name	Description
demoscenario.csv2rest	CsvDataModel [CLONE]	
demoscenario.csv2rest	CsvDataModel	
demoscenario.csv2rest	RestDataModel	

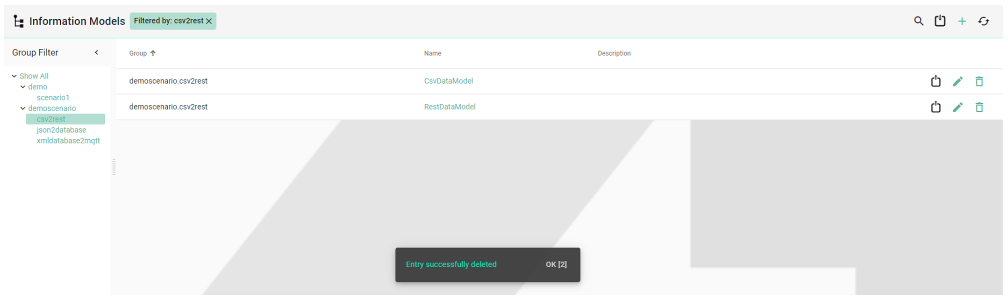
Select **Delete** on the confirmation dialog **(2)**.



Delete Model

Are you sure you want to delete this Information Model?

The component is deleted.



Group	Name	Description
demoscenario.csv2rest	CsvDataModel	
demoscenario.csv2rest	RestDataModel	

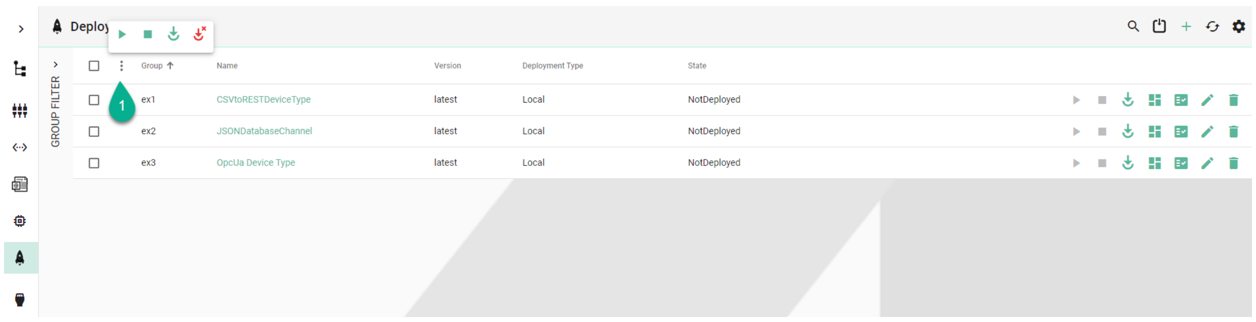
Entry successfully deleted OK [2]

3.4.14 Bulk Action

This operation is available only for the **Deployment**.

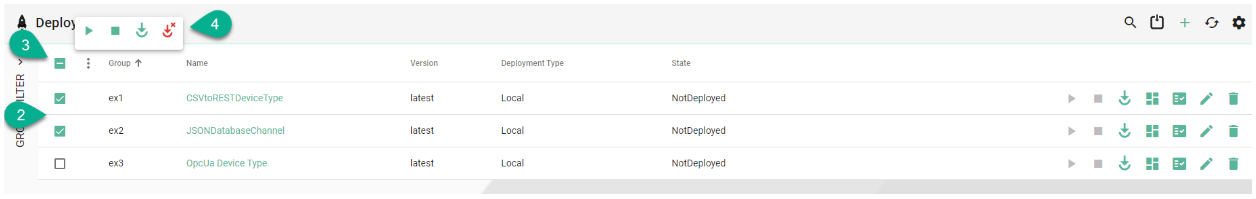
Click on the **ellipsis menu** button **(1)** to see the available bulk operations:

- Start
- Stop
- Deploy
- Undeploy



Group	Name	Version	Deployment Type	State
ex1	CSVtoRESTDeviceType	latest	Local	NotDeployed
ex2	JSONDatabaseChannel	latest	Local	NotDeployed
ex3	OpcUa Device Type	latest	Local	NotDeployed

To get started, check the boxes for specific Deployment Instances **(2)** or the box to select all **(3)**. The bulk operations popup appears **(4)**.

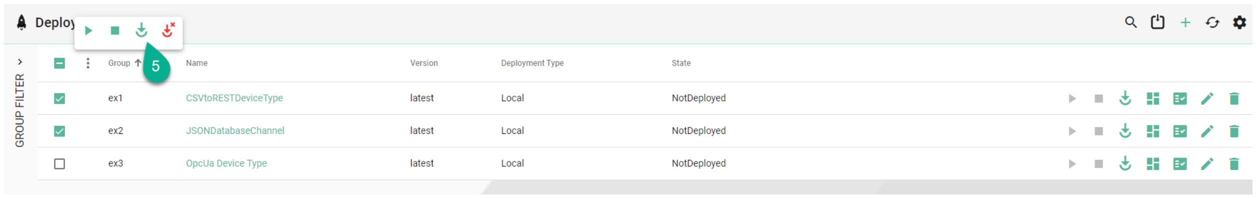


	Group	Name	Version	Deployment Type	State
✓	ex1	CSVtoRESTDeviceType	latest	Local	NotDeployed
✓	ex2	JSONDatabaseChannel	latest	Local	NotDeployed
☐	ex3	OpcUa Device Type	latest	Local	NotDeployed

For the Deployment Instance there is a defined action order:

- Deploy, Start, Stop, Start, Stop. . . Undeploy

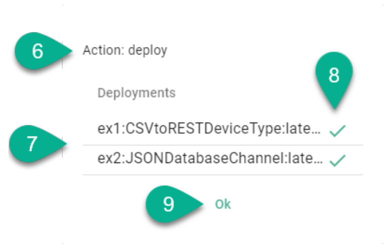
In this example the selected Instances should be deployed **(5)**.



	Group	Name	Version	Deployment Type	State
✓	ex1	CSVtoRESTDeviceType	latest	Local	NotDeployed
✓	ex2	JSONDatabaseChannel	latest	Local	NotDeployed
☐	ex3	OpcUa Device Type	latest	Local	NotDeployed

A status popup appears, displaying the following information:

- Performed action **(6)**
- The Instances included in the bulk action **(7)**
- The status of the action **(8)**



Action: deploy

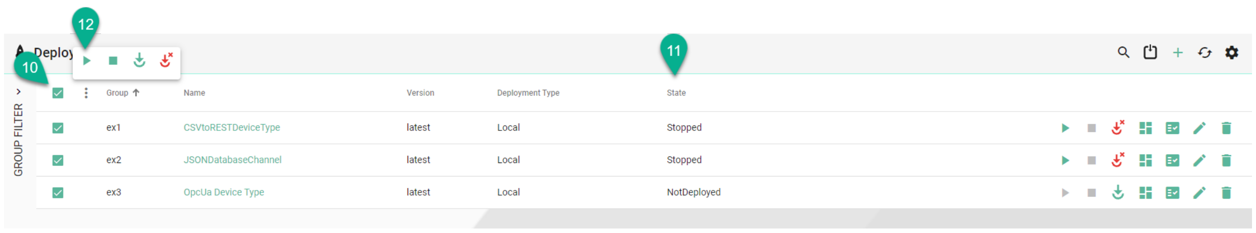
Deployments

ex1:CSVtoRESTDeviceType:late... ✓

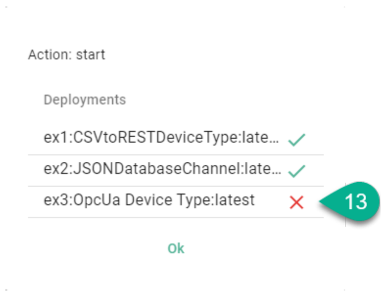
ex2:JSONDatabaseChannel:late... ✓

Click the **Ok** button **(9)** to close the popup.

When the selected Instances **(10)** are in different states **(11)**, the bulk action **(12)** will only affect Instances with the compatible state **(13)**.



	Group	Name	Version	Deployment Type	State
✓	ex1	CSVtoRESTDeviceType	latest	Local	Stopped
✓	ex2	JSONDatabaseChannel	latest	Local	Stopped
✓	ex3	OpcUa Device Type	latest	Local	NotDeployed



Action: start

Deployments

ex1:CSVtoRESTDeviceType:late... ✓

ex2:JSONDatabaseChannel:late... ✓

ex3:OpcUa Device Type:latest ✗

Note: Protected Instances will not work using bulk actions.

CHAPTER

FOUR

DEPLOYMENT

SMART**UNIFIER** supports the *deployment* of Instances on several computing environments:

- *Local* - on the same environment the SMART**UNIFIER** Manager is running on
- *Docker* - on containerized environments
- *Agent Process* - remote on any machine
- *SSH* - remote on Linux machine
- *Fargate* - on the AWS Cloud using fully managed service AWS Fargate

Learn how to *operate* and *monitor* your SMART**UNIFIER** Instances.

Learn about *notifications*.

Learn about additional *deployment options*.

4.1 What is a Deployment

With the SMART**UNIFIER** Deployment capability you can deploy your SMART**UNIFIER** *Communication Instances* to any IT resource (e.g., Equipment PC, Server, Cloud) suitable to execute SMART**UNIFIER** Instances.

Depending on the Deployment Type a Deployment Endpoint must be initially created. For deployments on a local computer, no Deployment Endpoint needs to be set.

Currently, the following Deployment Endpoints are supported:

- Local: Deployment of a SMART**UNIFIER** Communication Instance to your local computer where the SMART**UNIFIER** Manager is running on.
- *Agent*: Deployment of a SMART**UNIFIER** Communication Instance remote on any machine.
- *Docker*: Deployment of a SMART**UNIFIER** Communication Instance in containerized environments.
- *AWS*: Deployment of a SMART**UNIFIER** Communication Instance on the AWS Cloud using AWS Fargate.
- *SSH*: Deployment of a SMART**UNIFIER** Communication Instance on a Linux machine.

SMART**UNIFIER** Communication Instance can be encrypted prior the deployment by enabling the encryption option. You learn how to do so in the chapters of the specific deployment options.

Getting started:

- Select your environment and create the Deployment:
  - *Local*
  - *Agent*
  - *Docker*
  - *Fargate*
  - *SSH*
- Learn how to *operate* an Deployment.
- Learn how to *monitor* an Deployment.

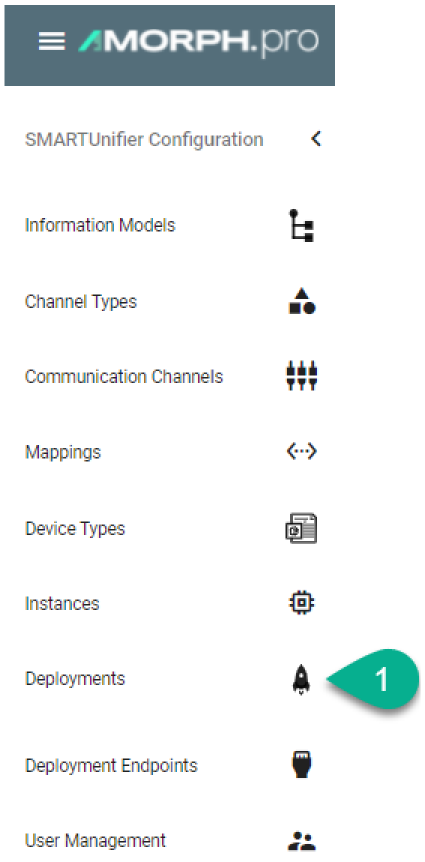
4.2 Deploy Locally

SMART**UNIFIER** Communication Instances can be deployed on the IT-resource where the SMART**UNIFIER** Manager is running on (e.g., a computer, a server or the AWS Cloud).

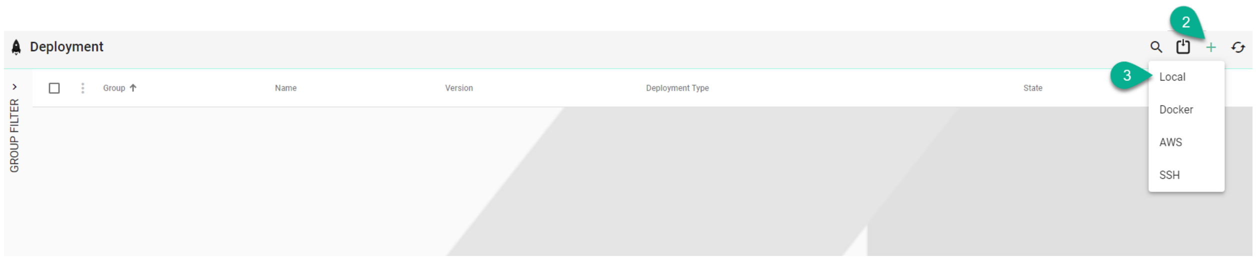
Note: Before deploying a Local Communication Instance make sure to create and **Start** a *Local Deployment Endpoint*. The Deployment Endpoint specifies the location where you want the Instance to run.

Follow the steps described below in order to deploy a Communication Instance locally:

- Select the SMART**UNIFIER** Deployment perspective **(1)**.

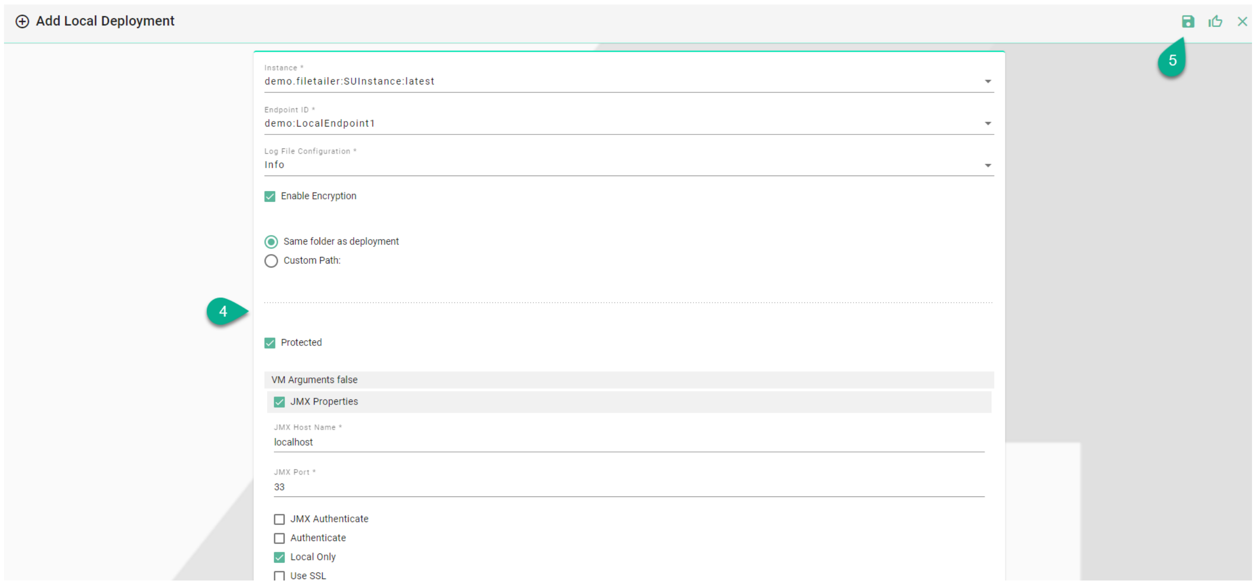


- Click on the **Add Deployment** button **(2)**.
- Select the Deployment Type **Local** from the pop-up **(3)**.



- In the Add Deployment view a set of configuration parameters is required **(4)**
  - Select the SMART**UNIFIER** Communication Instance to be used in the Deployment.
  - Select the **Local Endpoint**.
  - Select the *log file level*. We recommend the log level of type *Info* in case of a normal deployment scenario.
  - (Optional) Enable *Encryption*.
  - (Optional) Enable *Protection*.
  - (Optional) Add *VM Arguments*.

- When all mandatory fields are filled click the **Save** button **(5)**.



Instance *
demo.filetailer:SUInstance:latest

Endpoint ID *
demo:LocalEndpoint1

Log File Configuration *
Info

[✓] Enable Encryption

(●) Same folder as deployment
( ) Custom Path:

[✓] Protected

VM Arguments false

[✓] JMX Properties

JMX Host Name *
localhost

JMX Port *
33

[ ] JMX Authenticate
[ ] Authenticate
[✓] Local Only
[ ] Use SSL

When the Instance is deployed, it's configuration will be copied in the **Deployment folder** defined in the *Local Deployment Endpoint configuration*.

Note: The Instance configuration folder can be copied to another location and started, but the Instance will not be monitored by the SMART**UNIFIER** Manager.

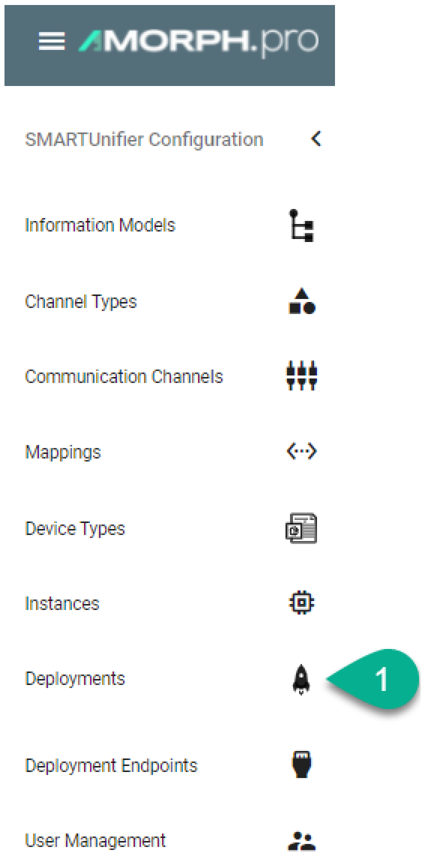
4.3 Deploy with Docker

Note: Before deploying a Communication Instance with Docker make sure to add a *Docker Java Image* and to create and **Start** a *Docker Deployment Endpoint*. The Deployment Endpoint specifies the location where you want the container to run.

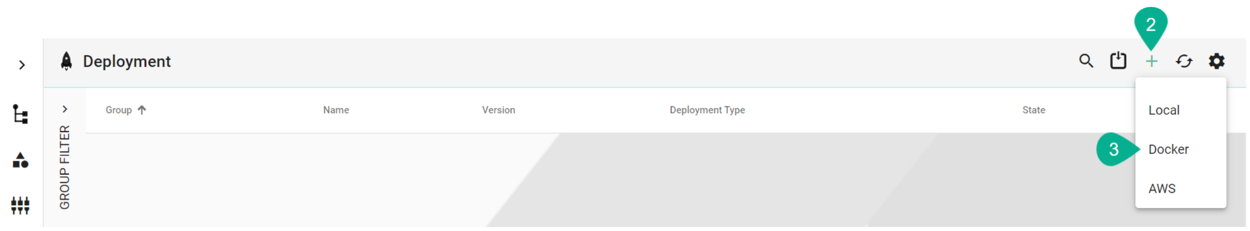
SMART**UNIFIER** Communication Instances can be deployed on any location that has an existing Docker environment in place.

Follow the steps described below to deploy a Communication Instance inside a Docker container:

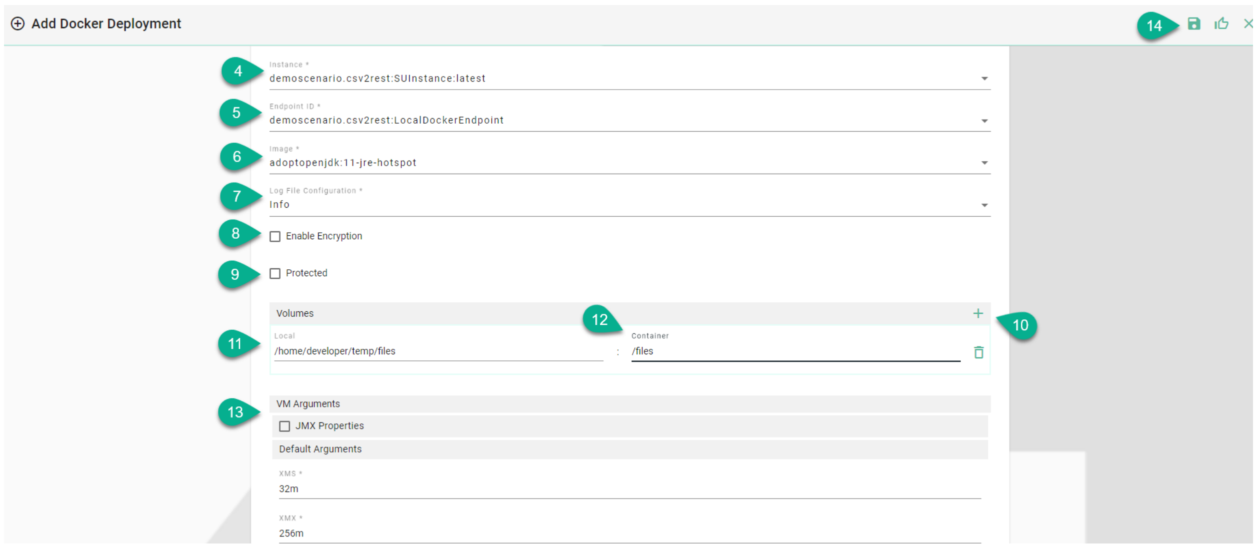
- Select the SMART**UNIFIER** Deployment perspective **(1)**.



- Click on the "Add Deployment" button **(2)**.
- Select the Deployment Type **Docker** from the pop-up **(3)**.



- Select the SMART**UNIFIER** Communication Instance to be used in the Deployment **(4)**.
- Select the Docker Endpoint ID created in the *Docker section* from the Drop-Down menu **(5)**.
- Select the Image added in the *Docker Java Image Manager* from the Drop-Down menu **(6)**.
- Select the *log file level* **(7)**. We recommend the log level of type *Info* in case of a normal deployment scenario.
- (Optional) Enable *Encryption* **(8)**
- (Optional) Enable *Protection* **(9)**



- (Optional) Add Volumes to store persisting data that can be used by the Docker container **(10)**.
  - Enter the local path of the directory or the name of an existing volume **(11)**
  - Enter the mount path inside the container **(12)**
- (Optional) Add *VM Arguments* **(13)**
- When all mandatory fields are filled click the “Save” button **(14)**.

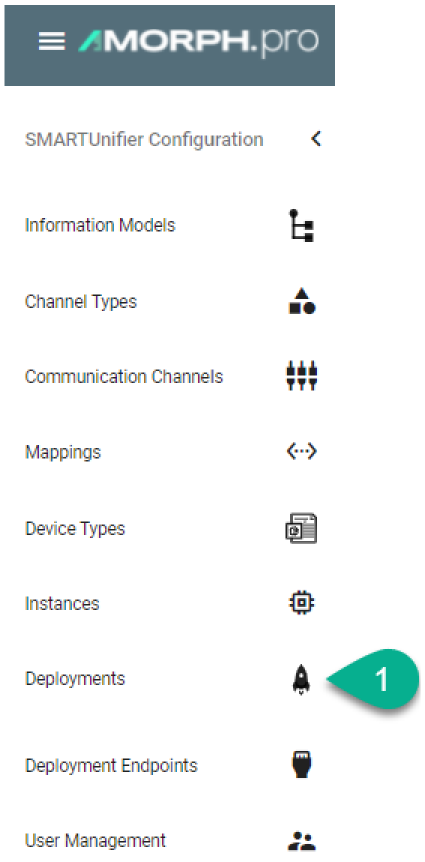
4.4 Deploy with Agent

Note: Before deploying a Communication Instance with agent, make sure to create and **start** an *Agent Deployment Endpoint*. The Deployment Endpoint specifies the location where you want the Instance to run. Make sure the Agent process is running on the remote machine.

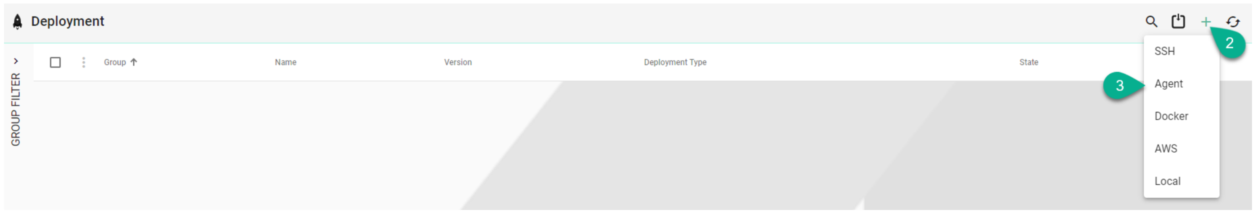
SMART**UNIFIER** Communication Instances can be deployed remotely on any machine with an active Agent process. The Agent process provides the communication between the SMART**UNIFIER** Manager and the Communication Instance.

Follow the steps described below to deploy a Communication Instance using the Agent:

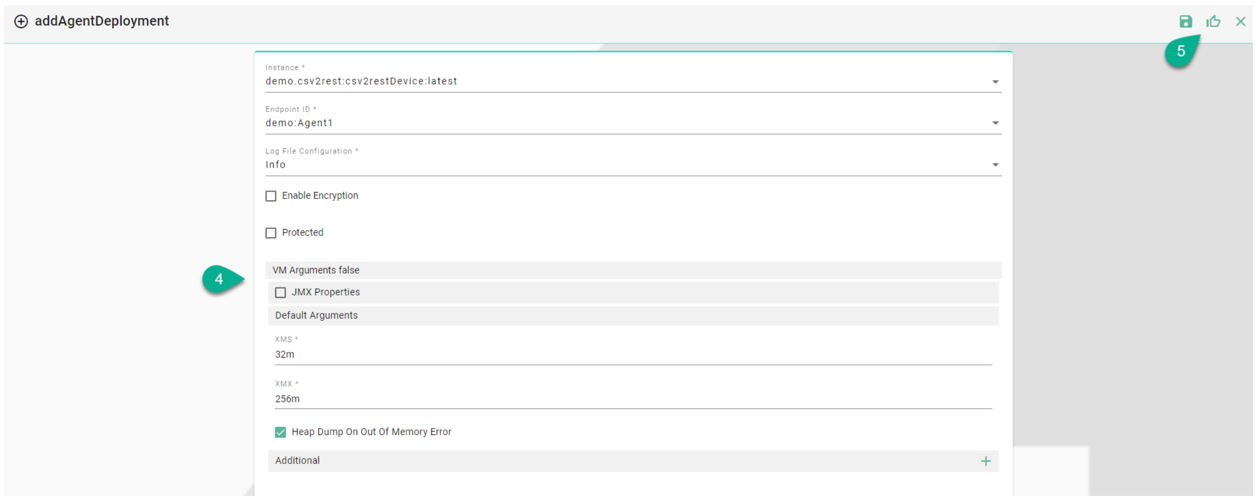
- Select the SMART**UNIFIER** Deployment perspective **(1)**.



- Click on the **Add Deployment** button **(2)**.
- Select the Deployment Type **Agent** from the pop-up **(3)**.



- In the "Add Deployment" view a set of configuration parameters is required **(4)**
- Select the SMART**UNIFIER** Communication Instance to be used in the Deployment.
- Select the agent Endpoint ID created in the *Agent section* from the Drop-Down menu.
- Select the *log file level*. We recommend the log level of type *Info* in case of a normal deployment scenario.
- (Optional) Enable *Encryption*.
- (Optional) Enable *Protection*.
- (Optional) Add *VM Arguments*.



Instance *
demo.csv2rest:csv2restDevice:latest

Endpoint ID *
demo:Agent1

Log File Configuration *
Info

☐ Enable Encryption

☐ Protected

VM Arguments false

☐ JMX Properties

Default Arguments

XMS *
32m

XMX *
256m

✓ Heap Dump On Out Of Memory Error

Additional

- When all mandatory fields are filled click on the **Save and Close** button **(5)**.

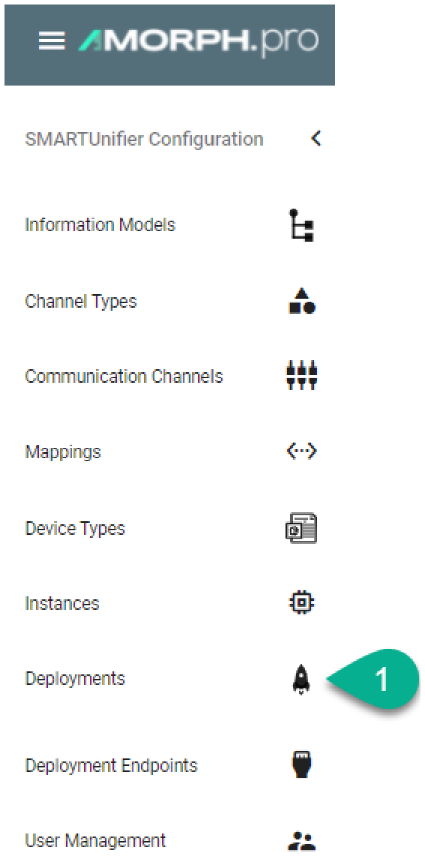
4.5 Deploy with SSH

Note: Before deploying a Communication Instance with SSH make sure to create and **Start** a *SSH Deployment Endpoint*. The Deployment Endpoint specifies the location where you want the Instance to run.

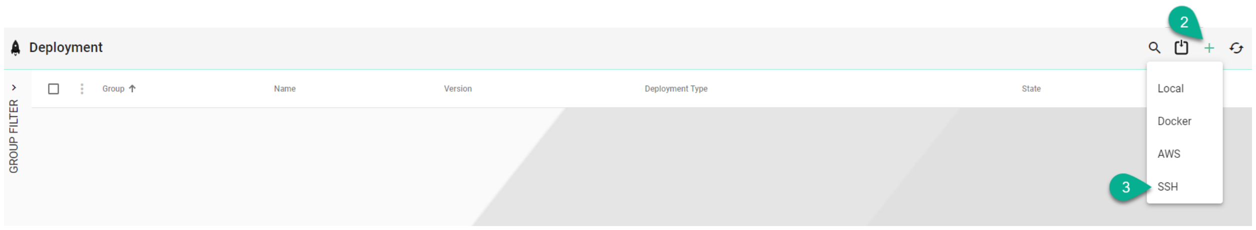
SMART**UNIFIER** Communication Instances can be remotely deployed on any Linux machine by using the Secure Socket Shell protocol.

Follow the steps described below to deploy a Communication Instance using the SSH protocol:

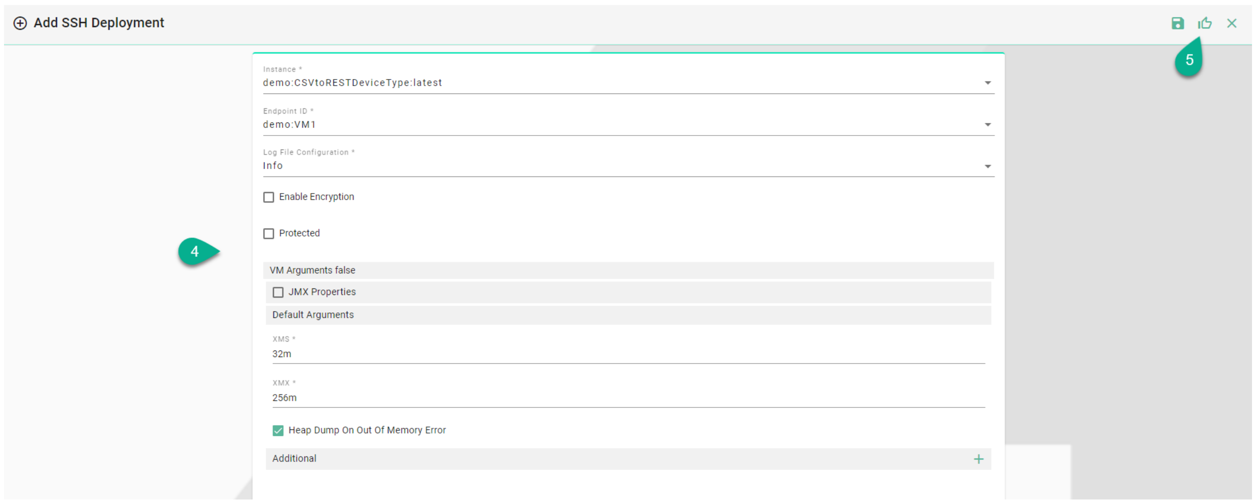
- Select the SMART**UNIFIER** Deployment perspective **(1)**.



- Click on the "Add Deployment" button **(2)**.
- Select the Deployment Type **SSH** from the pop-up **(3)**.



- In the "Add Deployment" view a set of configuration parameters is required **(4)**
- Select the SMART**UNIFIER** Communication Instance to be used in the Deployment.
- Select the SSH Endpoint ID created in the *SSH section* from the Drop-Down menu.
- Select the *log file level*. We recommend the log level of type *Info* in case of a normal deployment scenario.
- (Optional) Enable *Encryption*.
- (Optional) Enable *Protection*.
- (Optional) Add *VM Arguments*.


Add SSH Deployment

Instance *
demo:CSVtoRESTDeviceType:latest

Endpoint ID *
demo:VM1

Log File Configuration *
Info

☐ Enable Encryption

☐ Protected

VM Arguments false

☐ JMX Properties

Default Arguments

XMS *
32m

XMX *
256m

✓ Heap Dump On Out Of Memory Error

Additional

- When all mandatory fields are filled click the "Save and Close" button **(5)**.

4.6 Deploy with AWS Fargate

SMART**UNIFIER** supports the deployment of Communication Instances on Amazon Web Services (AWS) using AWS Fargate. Using AWS Fargate removes the operational overhead of managing servers by paying only for the resources actually used.

To deploy your SMART**UNIFIER** Instances using AWS Fargate an AWS Account is required.

Before deploying a SMART**UNIFIER**-Instance using AWS Fargate please refer to the *Prerequisites* section and make sure all requirements your Account needs to fulfill are met.

4.6.1 Prerequisites

Specialized Knowledge

Before deploying and operating SMART**UNIFIER** Instances using AWS Fargate, it is recommended that you become familiar with the following AWS services. (If you are new to AWS, see Getting Started with AWS)

- Amazon Elastic Container Service (ECS)
- Amazon Virtual Private Cloud (VPC)
- Amazon CloudWatch

You should also be familiar with the used Communication Channel and its capabilities of the deployed SMART**UNIFIER** Instance.

AWS Resources

For the deployment of SMART**UNIFIER** Instances on AWS Fargate the following resources are required:

Amazon S3 - Bucket

SMART**UNIFIER** is using an Amazon S3 Bucket to upload Instances in an archive file format. We recommend to create a private Bucket dedicated for the SMART**UNIFIER**.

AWS VPC and Subnets

In order for SMART**UNIFIER** to deploy Instances your AWS account a VPC and Subnets are needed. Please note that the Default VPC should not be used.

Amazon ECS - Cluster

SMART**UNIFIER** is using AWS Fargate for the deployment of Instances on the AWS Cloud. Therefor an ECS Cluster is required. We recommend to create one Cluster dedicated for SMART**UNIFIER** deployed Instances.

AWS ECR - Repository

SMART**UNIFIER** is using an AWS ECR repository in order to push Docker Images, which is created by an AWS CodeBuild project. We recommend to create one repository dedicated for SMART**UNIFIER** Instance images.

IAM - User

SMART**UNIFIER** complies with the security best practices in IAM and does not need root privileges. We recommend to create one user dedicated for SMART**UNIFIER**. The IAM user follows the general rule of least privileges and allows only policies needed for the deployment of SMART**UNIFIER** Instances.

Create the IAM user by following the steps described in the AWS IAM documentation the IAM dashboard. The IAM user for SMART**UNIFIER** must use the AWS access type **programmatic access**.

Attach the following permission:

Policy ARN	Description
arn:aws:iam::aws:policy/AmazonS3FullAccess	Provides full access to all buckets via the AWS Management Console.
arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess	Provides full access to AWS CodeBuild via the AWS Management Console. Also attach AmazonS3ReadOnlyAccess to provide access to download build artifacts, and attach IAMFullAccess to create and manage the service role for CodeBuild.
arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryFullAcces	Provides administrative access to Amazon ECR resources.
arn:aws:iam::aws:policy/AmazonECS_FullAccess	Provides administrative access to Amazon ECS resources and enables ECS features through access to other AWS service resources, including VPCs, Auto Scaling groups, and CloudFormation stacks.
arn:aws:iam::aws:policy/CloudWatchFullAccess	Provides full access to CloudWatch.

Programmatic system credentials

SMART**UNIFIER** needs the set up of a credential profile in order to deploy Instances on AWS Fargate. We recommend to create a new access key after 90 days.

Listing 1: Credentials Profile

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

IAM Role - AWS CodeBuild Service Role

CodeBuild requires a service to interact with dependent AWS services:

- Access to Amazon S3 to retrieve SMART**UNIFIER** Instance artifacts - such as libraries and configuration files.
- Access to AWS ECR to push the container image in the specified repository

Create the following IAM Role via the AWS console.

Listing 2: AWS CodeBuild Service Role

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Sid": "CloudWatchLogsPolicy",
         "Effect": "Allow",
         "Action": [
             "logs:CreateLogGroup",
             "logs:CreateLogStream",
             "logs:PutLogEvents"
         ],
         "Resource": [
             "*"
         ]
     },
     {
         "Sid": "CodeCommitPolicy",
         "Effect": "Allow",
         "Action": [
             "codecommit:GitPull"
         ],
         "Resource": [
             "*"
         ]
     },
     {
         "Sid": "S3GetObjectPolicy",
         "Effect": "Allow",
         "Action": [
             "s3:GetObject",
             "s3:GetObjectVersion"
         ],
         "Resource": [
             "*"
         ]
     },
     {
         "Sid": "S3PutObjectPolicy",
         "Effect": "Allow",
         "Action": [
             "s3:PutObject"
         ],
         "Resource": [
             "*"
```

(continues on next page)

(continued from previous page)

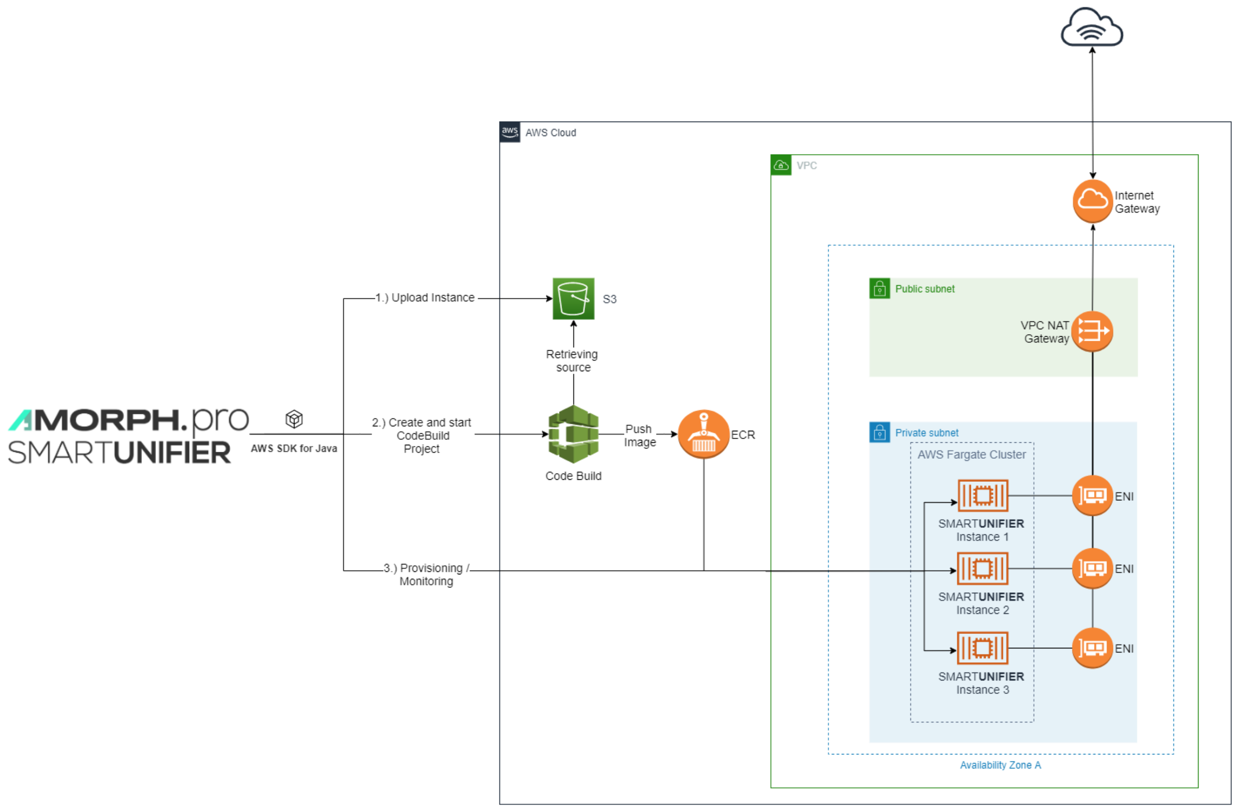
```
        ]
    },
    {
        "Sid": "ECRPullPolicy",
        "Effect": "Allow",
        "Action": [
            "ecr:BatchCheckLayerAvailability",
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage"
        ],
        "Resource": [
            "*"
        ]
    },
    {
        "Sid": "ECRAuthPolicy",
        "Effect": "Allow",
        "Action": [
            "ecr:GetAuthorizationToken"
        ],
        "Resource": [
            "*"
        ]
    },
    {
        "Sid": "S3BucketIdentity",
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketAcl",
            "s3:GetBucketLocation"
        ],
        "Resource": "*"
    }
 ]
}
```

4.6.2 Architecture

The deployment of SMART**UNIFIER**-Instances on AWS Cloud is handled by the SMART**UNIFIER** Manager. The Manager can run on any On-Premise location such as, server environments and Industrial PCs; however, in order to deploy Instances on AWS an internet connection is required. To run SMART**UNIFIER** Manager on AWS Cloud please see the SMART**UNIFIER** Installation Manual.

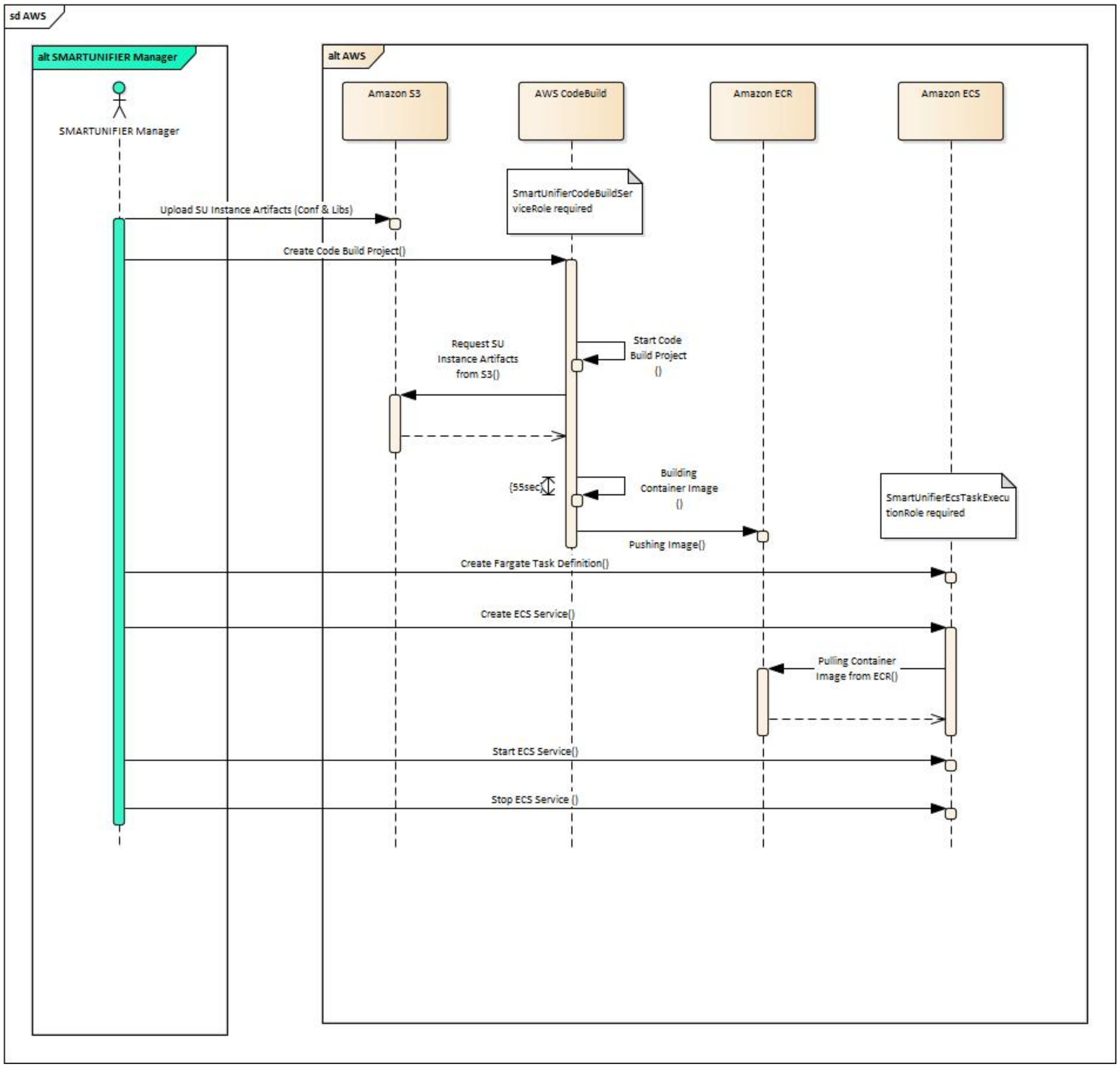
SMART**UNIFIER** is using the AWS SDK for Java to make deployments of Instances to AWS Fargate. Following AWS Services are used during the deployment process:

- AWS Simple Storage Service (Amazon S3) (Mandatory).
- AWS CodeBuild (Mandatory).
- AWS Elastic Container Registry (Mandatory).
- AWS Elastic Container Service (Mandatory).
- AWS Fargate (Mandatory).
- Amazon CloudWatch (Optional).



Sequence of events

1. Upload of the SMART**UNIFIER** Instance as an archive file format to Amazon S3.
2. Creation and automatic triggering of an AWS CodeBuild project.
3. The AWS CodeBuild project uses the archive file from the specified Amazon S3 Bucket in order to build a Docker Image for the particular SMART**UNIFIER** Instance.
4. When finished, AWS CodeBuild pushes the Image to a specified ECR Repository.
5. Is the Image available on the ECR Repository a Fargate Task Definition is created as well as an ECS Service which is using the Task Definition.
6. By default, the Task is not started directly. Starting and Stopping of tasks can be done via the SMART**UNIFIER** Manager or the AWS Console.


sd AWS
alt SMARTUNIFIER Manager
SMARTUNIFIER Manager
alt AWS
Amazon S3
AWS CodeBuild
Amazon ECR
Amazon ECS
SmartUnifierCodeBuildServiceRole required
Upload SU Instance Artifacts (Conf & Libs)
Create Code Build Project()
Start Code Build Project ()
Request SU Instance Artifacts from S3()
{55sec}
Building Container Image ()
SmartUnifierEcsTaskExecutionRole required
Pushing Image()
Create Fargate Task Definition()
Create ECS Service()
Pulling Container Image from ECR()
Start ECS Service()
Stop ECS Service ()

4.6.3 Planning the Deployment

Task Sizing

Each SMART**UNIFIER** Instance runs as java byte code, thus having a low footprint. We recommend using the following guideline for Task Sizing.

Note: Please note that AWS Fargate is pricing based on the vCPU and memory resources, which are specified during the set up.

CPU	Memory Values	Instance Workload (Number of Mappings)
0.25 vCPU	0.5GB, 1GB, and 2GB	<= 5
0.5 vCPU	Min. 1GB and Max. 4GB, in 1GB increments	> 6

4.6.4 Deployment Steps

Expected Time

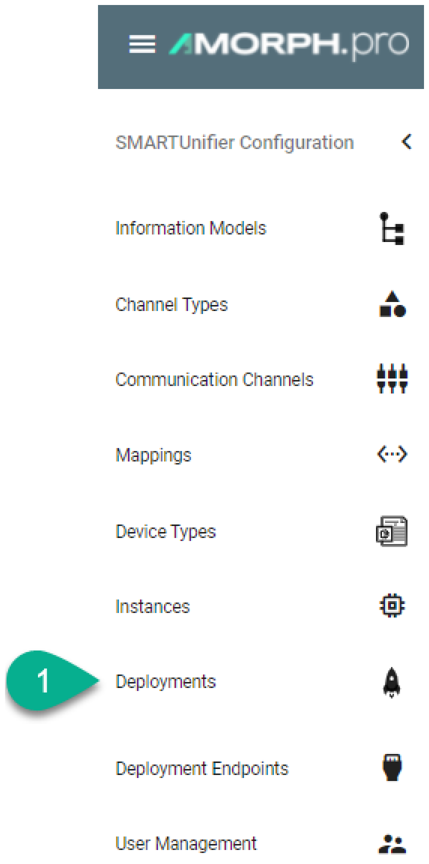
- Deployment of an SMART**UNIFIER** Instance on AWS Fargate (Existing AWS Resources) - expected deployment time: **3-5 min**
- Deployment of an SMART**UNIFIER** Instance on AWS Fargate (Creation of needed AWS Resources required) - expected deployment time: **20-30 min** (Please note that this is a one time setup of the customers AWS cloud infrastructure)

Deployment of the SMARTUNIFIER Instance

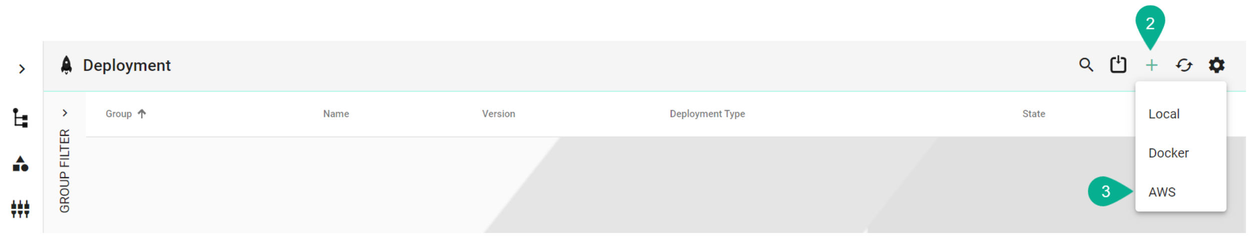
If you have not already created and **Started** an AWS Deployment Endpoint please refer to chapter: *AWS Endpoint*.

Follow the steps described below to deploy a SMART**UNIFIER** Instance on AWS Fargate:

- Select the SMART**UNIFIER** Deployment perspective **(1)**.



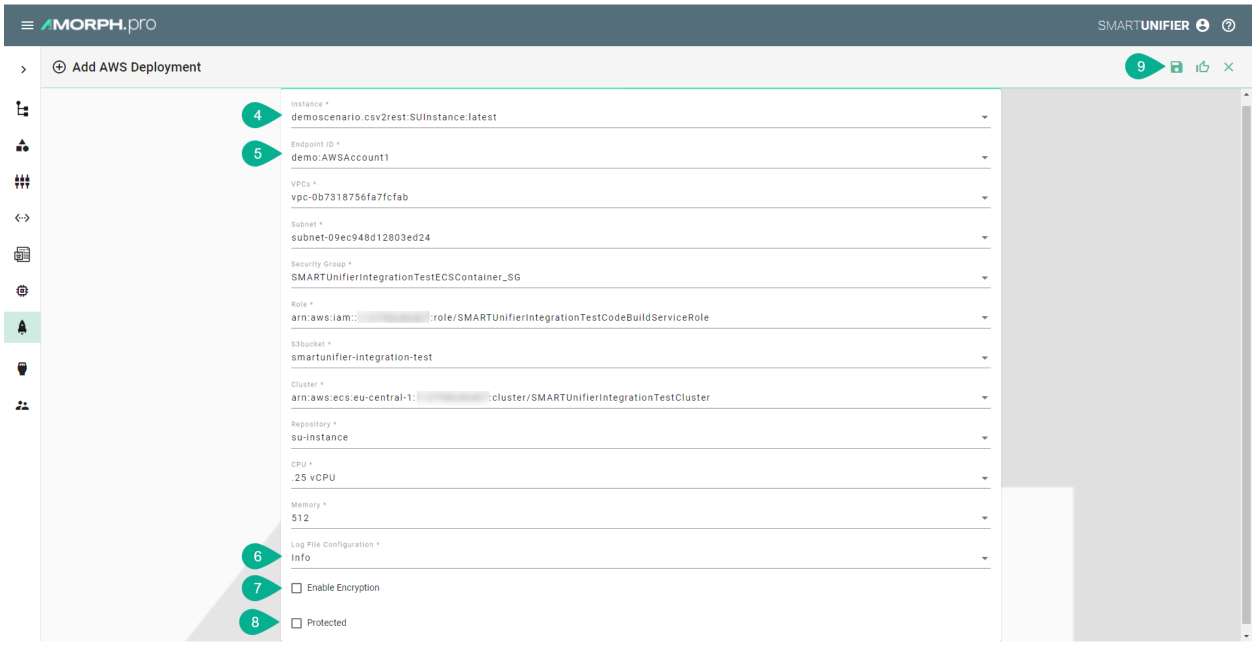
- Click the "Add" button **(2)**.
- Select AWS **(3)**.



- Select the SMART**UNIFIER** Instance you want to deploy **(4)**:
- Select your AWS account in form of a Deployment Endpoint created *previously* **(5)** and configure the following parameters:
  - Select the **VPC** in which you want to deploy the SMART**UNIFIER** Instance.
  - Select a **Subnet** within the VPC.
  - Select a **Security Group**.
  - Select a **IAM Role** for AWS CodeBuild.
    - AWS CodeBuild needs a service role so that it can interact with dependent AWS services on behalf of SMART**UNIFIER**.

- Select a **S3 Bucket**.
- Select a **ECS Cluster** in which the Instance should be deployed.
- Select an **ECR Repository**.
  - The AWS CodeBuild project, which is created and triggered by SMART**UNIFIER**, pushes an Image to the provided Amazon ECR Repository.
- Select the *Task's* - **CPU**.
- Select the *Task's* - **Memory**.

- Select the *log file level* **(6)**.
- (Optional) Enable *Encryption* **(7)**
- (Optional) Enable *Protection* **(8)**
- Save the Deployment by clicking the "Save" button **(9)**.



Instance *
demoscenario.csv2rest:SUInstance:latest

Endpoint ID *
demo:AWSAccount1

VPCs *
vpc-0b7318756fa7fcfab

Subnet *
subnet-09ec948d12803ed24

Security Group *
SMARTUnifierIntegrationTestECSContainer_SG

Role *
arn:aws:iam::[redacted]:role/SMARTUnifierIntegrationTestCodeBuildServiceRole

S3bucket *
smartunifier-integration-test

Cluster *
arn:aws:ecs:eu-central-1:[redacted]:cluster/SMARTUnifierIntegrationTestCluster

Repository *
su-instance

CPU *
.25 vCPU

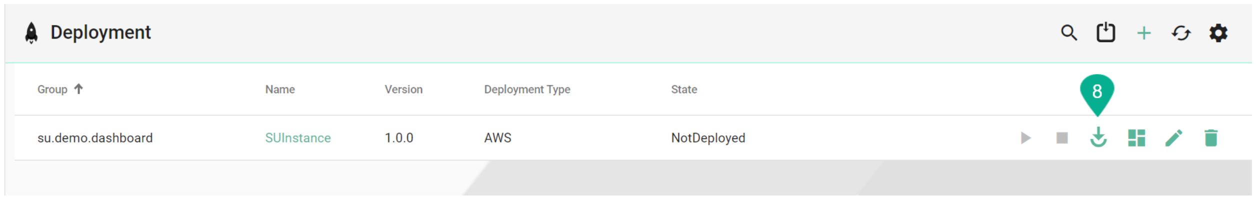
Memory *
512

Log File Configuration *
Info

☐ Enable Encryption

☐ Protected

- Go back to the list view by clicking the "Close" button and deploy your SMART**UNIFIER** Instance by clicking the "Deploy" button **(8)**.



Group ↑	Name	Version	Deployment Type	State
su.demo.dashboard	SUInstance	1.0.0	AWS	NotDeployed

- You can start and stop the Instance using SMART**UNIFIER** by clicking the "Start"/"Stop" button or using the AWS Console.

Monitoring

Once deployed and started, the SMART**UNIFIER** Instance logs can be accessed via Amazon Cloud-Watch.

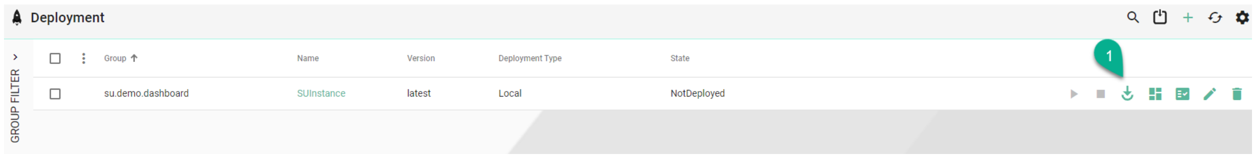
In order to access log files follow the steps below:

- Go to the Amazon CloudWatch Service via the Console.
- Select **Log groups** from the menu on the left.
- Select **awslogs-testinstance** and select a log Stream.

4.7 How to Deploy, Run and Operate a Deployed Instance

4.7.1 How to Deploy an Instance

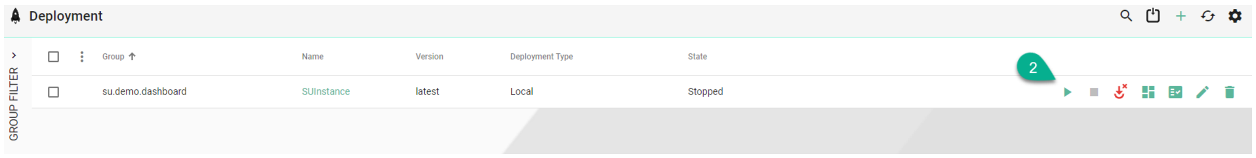
- In order to start the Instance, click first the "Deploy" button **(1)**. A message is shown, that confirms the successful deployment of the Instance.



Group	Name	Version	Deployment Type	State
su.demo.dashboard	SUInstance	latest	Local	NotDeployed

4.7.2 How to Run an Instance

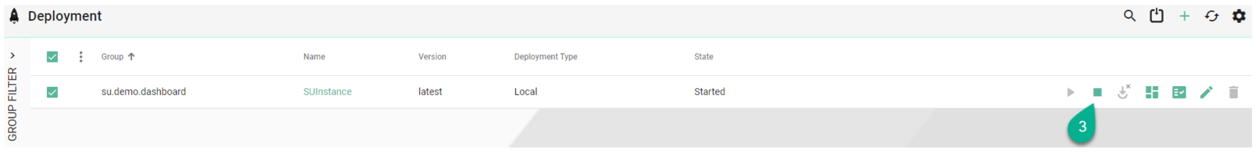
- After successfully deploying the Instance, the state changes from *NotDeployed* to *Stopped*. You can now click the enabled "Start" button **(2)**. The Instance state will change to *Started*. A message is shown, that confirms the successful start of the Instance.



Group	Name	Version	Deployment Type	State
su.demo.dashboard	SUInstance	latest	Local	Stopped

4.7.3 How to Stop an Instance

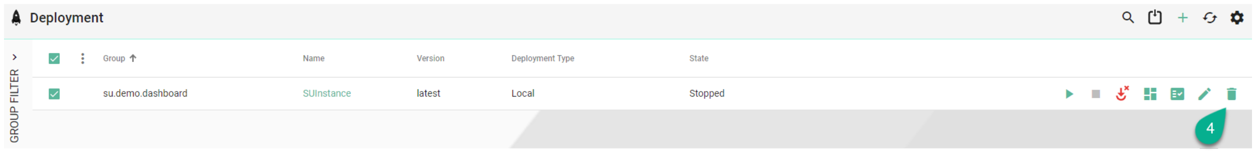
- To stop the Instance, click the "Stop" button **(3)**.



Group	Name	Version	Deployment Type	State
su.demo.dashboard	SUInstance	latest	Local	Started

4.7.4 How to Delete a Deployment of an Instance

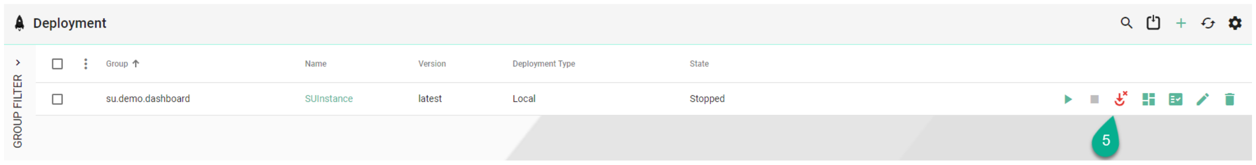
- Click on the "Delete" button to delete the Deployment for a specific Instance **(4)**. This is only possible if the Instance is in the state *Stopped*.



Group	Name	Version	Deployment Type	State
su.demo.dashboard	SUInstance	latest	Local	Stopped

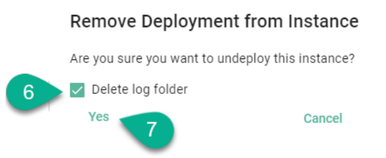
4.7.5 How to Un-deploy an Instance

- In order to un-deploy an Instance, **make sure** that the Instance is not running. If necessary *stop the Instance*.
- Click on the "Undeploy" button in the upper right corner **(5)**.



Group	Name	Version	Deployment Type	State
su.demo.dashboard	SUInstance	latest	Local	Stopped

- A popup appears, uncheck the box **(6)** to keep the log folder and click on the **Yes** button **(7)** to confirm.



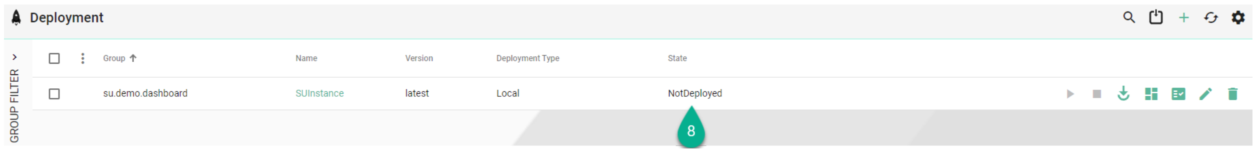
Remove Deployment from Instance

Are you sure you want to undeploy this instance?

☑ Delete log folder

Yes Cancel

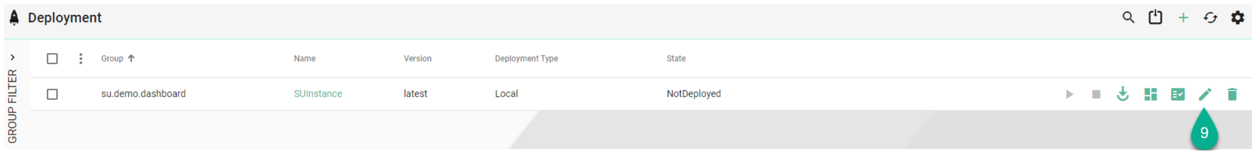
- The Instance state changes to *NotDeployed* **(8)** and the Deployment can be edited. Please **note** that the Instance associated with the Deployment cannot be changed.



Group	Name	Version	Deployment Type	State
su.demo.dashboard	SUInstance	latest	Local	NotDeployed

4.7.6 How to Edit a Deployment of an Instance

- Click on the "Edit" button to perform changes to the Deployment **(9)**. It is only possible to edit a Deployment if the Instance is not deployed. In case the Instance is deployed, only the details of the Deployment can be viewed.



Group	Name	Version	Deployment Type	State
su.demo.dashboard	SUInstance	latest	Local	NotDeployed

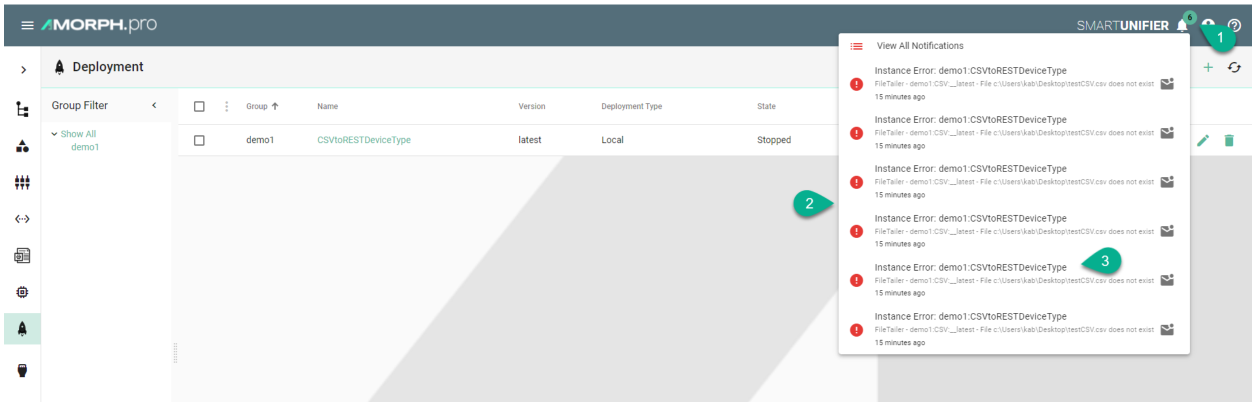
4.8 Notifications

SMART**UNIFIER** comes with an integrated notification system, which helps to gain insights when a deployed Communication Instance is started or running and errors appear.

4.8.1 How to access Notifications

When a deployed Communication Instance is started or running and errors appear, the number of errors will be displayed near the **Notifications** button **(1)**.

Click on the Notifications button and the **Notifications List (2)** will display all the Instance errors.



Group	Name	Version	Deployment Type	State
demo1	CSVtoRESTDeviceType	latest	Local	Stopped

Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
15 minutes ago

Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
15 minutes ago

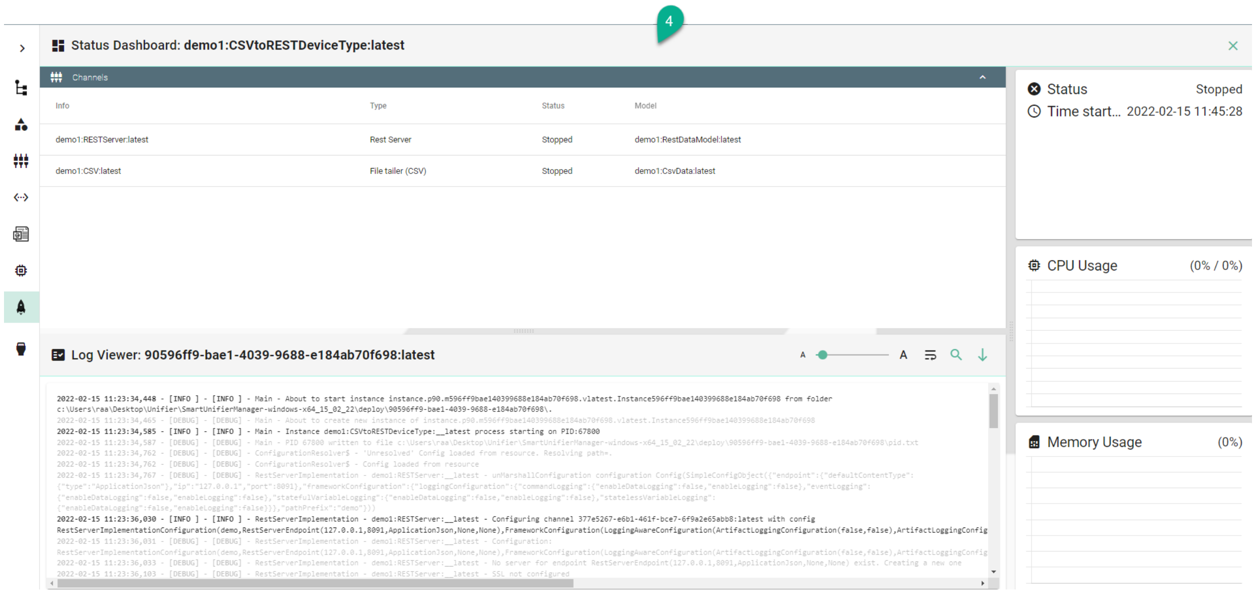
Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
15 minutes ago

Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
15 minutes ago

Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
15 minutes ago

Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
15 minutes ago

Select a notification **(3)** from the list and the **Dashboard (4)** will appear and display additional information.


4
Status Dashboard: demo1:CSVtoRESTDeviceType:latest
Channels
Status
Time start...
CPU Usage
(0% / 0%)
Memory Usage
(0%)
Log Viewer: 90596ff9-bae1-4039-9688-e184ab70f698:latest

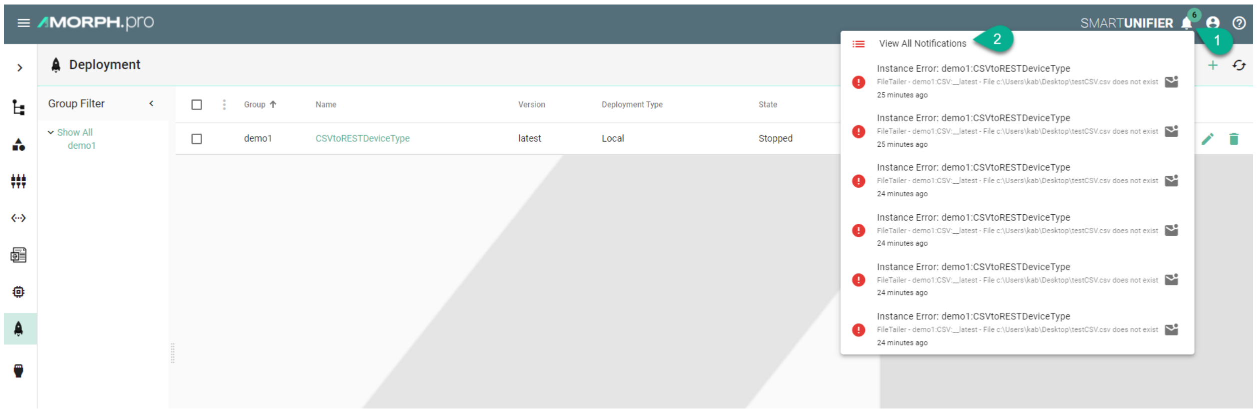
Status: Stopped
Time start...: 2022-02-15 11:45:28

Info	Type	Status	Model
demo1:RESTServer:latest	Rest Server	Stopped	demo1:RestDataModel:latest
demo1:CSV:latest	File tailer (CSV)	Stopped	demo1:CsvData:latest

```
2022-02-15 11:23:34,448 - [INFO ] - [INFO ] - Main - About to start instance instance.p90.m596ff9bae140399688e184ab70f698.vlatest.Instance596ff9bae140399688e184ab70f698 from folder
c:\Users\raa\Desktop\Unifier\SmartUnifierManager-windows-x64_15_02_22\deploy\90596ff9-bae1-4039-9688-e184ab70f698\.
2022-02-15 11:23:34,465 - [DEBUG] - [DEBUG] - Main - About to create new instance of instance.p90.m596ff9bae140399688e184ab70f698.vlatest.Instance596ff9bae140399688e184ab70f698
2022-02-15 11:23:34,585 - [INFO ] - [INFO ] - Main - Instance demo1:CSVtoRESTDeviceType:__latest process starting on PID:67800
2022-02-15 11:23:34,587 - [DEBUG] - [DEBUG] - Main - PID 67800 written to file c:\Users\raa\Desktop\Unifier\SmartUnifierManager-windows-x64_15_02_22\deploy\90596ff9-bae1-4039-9688-e184ab70f698\pid.txt
2022-02-15 11:23:34,762 - [DEBUG] - [DEBUG] - ConfigurationResolver$ - 'Unresolved' Config loaded from resource. Resolving path=.
2022-02-15 11:23:34,762 - [DEBUG] - [DEBUG] - ConfigurationResolver$ - Config loaded from resource
2022-02-15 11:23:34,767 - [DEBUG] - [DEBUG] - RestServerImplementation - demo1:RESTServer:__latest - unMarshallConfiguration configuration Config(SimpleConfigObject({"endpoint":{"defaultContentType":
{"type":"ApplicationJson"},"ip":"127.0.0.1","port":8091},"frameworkConfiguration":{"loggingConfiguration":{"commandLogging":{"enableDataLogging":false,"enableLogging":false},"eventLogging":
{"enableDataLogging":false,"enableLogging":false},"statefulVariableLogging":{"enableDataLogging":false,"enableLogging":false},"statelessVariableLogging":
{"enableDataLogging":false,"enableLogging":false}}},"pathPrefix":"demo"}))
2022-02-15 11:23:36,030 - [INFO ] - [INFO ] - RestServerImplementation - demo1:RESTServer:__latest - Configuring channel 377e5267-e6b1-461f-bce7-6f9a2e65abb8:latest with config
RestServerImplementationConfiguration(demo,RestServerEndpoint(127.0.0.1,8091,ApplicationJson,None,None),FrameworkConfiguration(LoggingAwareConfiguration(ArtifactLoggingConfiguration(false,false),ArtifactLoggingConfig
2022-02-15 11:23:36,031 - [DEBUG] - [DEBUG] - RestServerImplementation - demo1:RESTServer:__latest - Configuration:
RestServerImplementationConfiguration(demo,RestServerEndpoint(127.0.0.1,8091,ApplicationJson,None,None),FrameworkConfiguration(LoggingAwareConfiguration(ArtifactLoggingConfiguration(false,false),ArtifactLoggingConfig
2022-02-15 11:23:36,033 - [DEBUG] - [DEBUG] - RestServerImplementation - demo1:RESTServer:__latest - No server for endpoint RestServerEndpoint(127.0.0.1,8091,ApplicationJson,None,None) exist. Creating a new one
2022-02-15 11:23:36,103 - [DEBUG] - [DEBUG] - RestServerImplementation - demo1:RESTServer:__latest - SSL not configured
```

4.8.2 How to manage Notifications

In order to manage the notifications click on the **Notifications** button **(1)** and select the **View All Notifications** option **(2)**.



Group	Name	Version	Deployment Type	State
demo1	CSVtoRESTDeviceType	latest	Local	Stopped

Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
25 minutes ago

Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
25 minutes ago

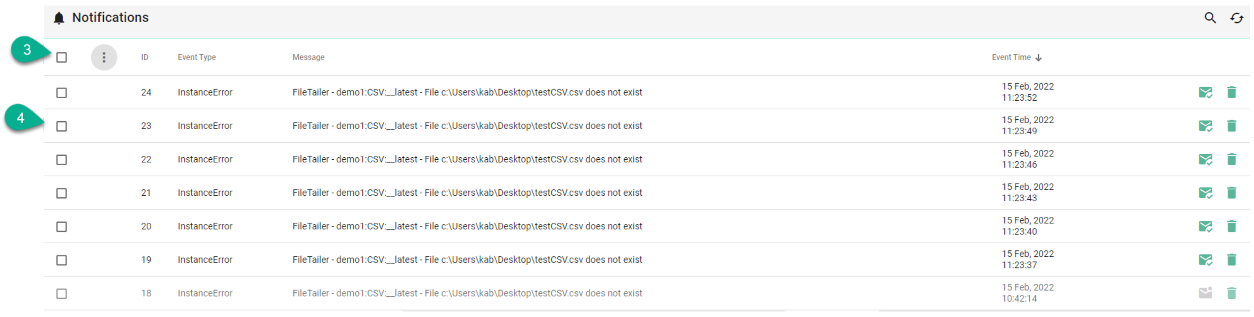
Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
24 minutes ago

Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
24 minutes ago

Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
24 minutes ago

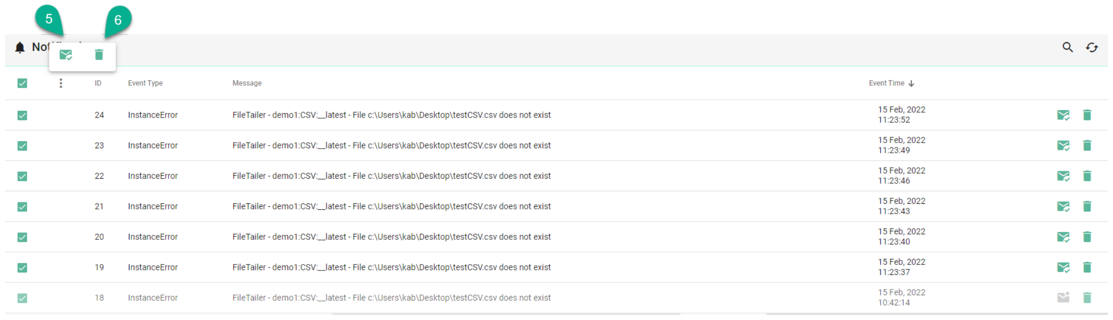
Instance Error: demo1:CSVtoRESTDeviceType
FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist
24 minutes ago

The Notifications Manager displays all the notifications. Select all **(3)** or specific notifications **(4)**.



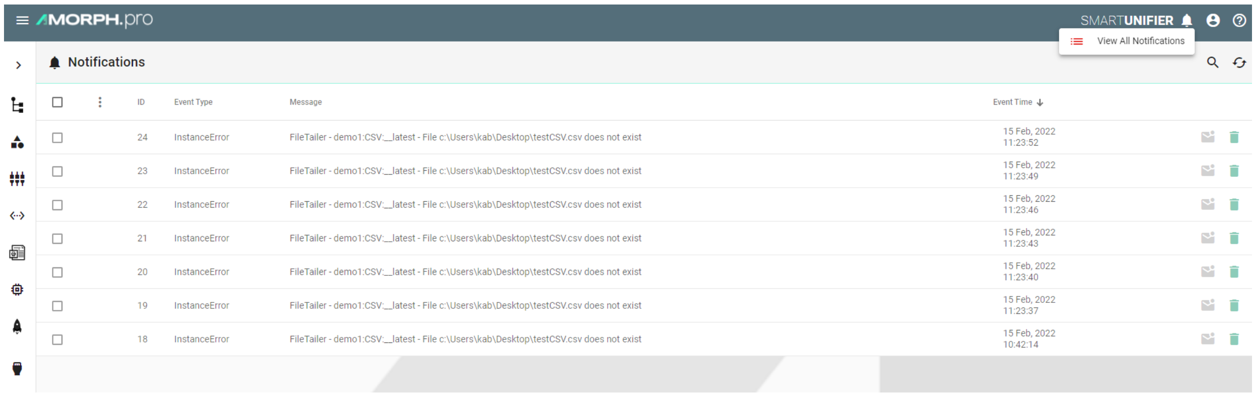
ID	Event Type	Message	Event Time
24	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:52
23	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:49
22	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:46
21	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:43
20	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:40
19	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:37
18	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 10:42:14

After selection a pop-up appears providing two options.



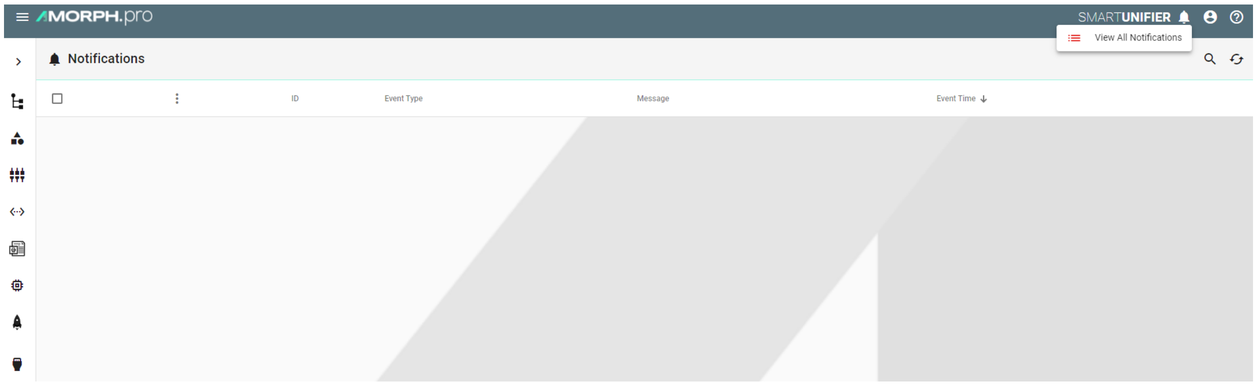
ID	Event Type	Message	Event Time
24	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:52
23	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:49
22	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:46
21	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:43
20	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:40
19	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:37
18	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 10:42:14

Click on the **Dismiss** button **(5)** to remove the selected notifications from the Notifications List. The selected notifications will still be available in the Notifications Manager.



ID	Event Type	Message	Event Time
24	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:52
23	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:49
22	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:46
21	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:43
20	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:40
19	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:37
18	InstanceError	FileTailer - demo1:CSV:__latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 10:42:14

To remove the selected notifications from the Notifications List and the Manager, click on the **Delete** button **(6)**.



4.9 How to monitor Communication Instances

Note: Monitoring of a Communication Instance is only possible after deployment.

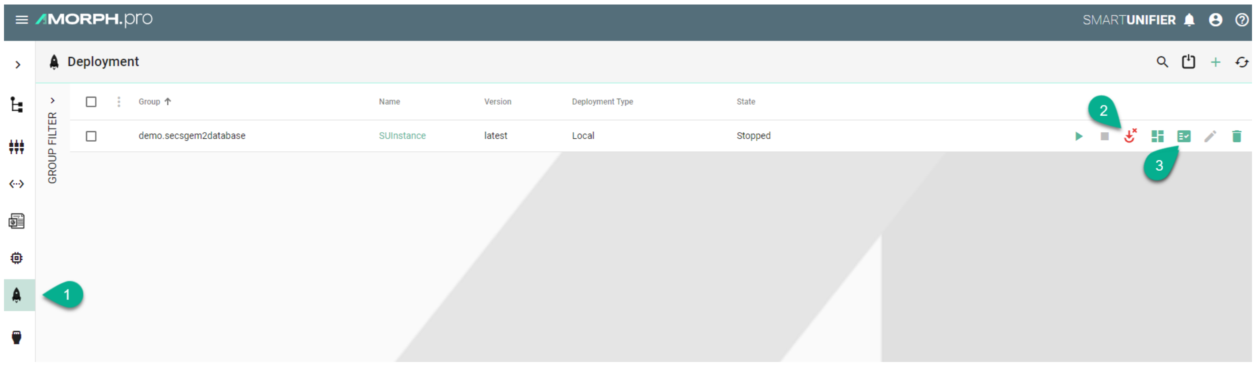
4.9.1 Log Viewer

SMART**UNIFIER** comes with an integrated Log Viewer that provides insights into deployed and running Communication Instances.

How to access

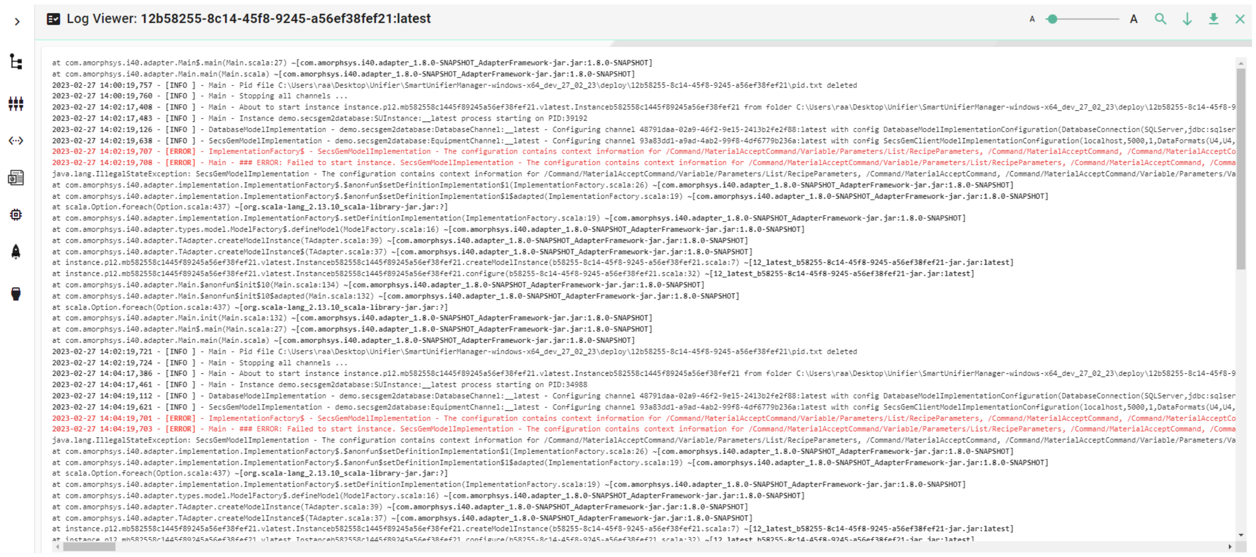
Follow the steps bellow to access the Log Viewer:

- Select the SMART**UNIFIER** Deployment perspective **(1)**
- Make sure the Instance is Deployed **(2)**
- Click on the **Log** button **(3)**



Group	Name	Version	Deployment Type	State
demo.secsgem2database	SUInstance	latest	Local	Stopped

- The Log Viewer is displaying the logs for the deployed Communication Instance



```
at com.amorphsys.i40.adapter.Main$.main(Main.scala:27) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.Main.main(Main.scala) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
2023-02-27 14:00:19,757 - [INFO ] - Main - Pid file C:\Users\raa\Desktop\Unifier\SmartUnifierManager-windows-x64_dev_27_02_23\deploy\12b58255-8c14-45f8-9245-a56ef38fef21\pid.txt deleted
2023-02-27 14:00:19,760 - [INFO ] - Main - Stopping all channels ...
2023-02-27 14:02:17,408 - [INFO ] - Main - About to start instance instance.p12.mb582558c1445f89245a56ef38fef21.vlatest.Instanceb582558c1445f89245a56ef38fef21 from folder C:\Users\raa\Desktop\Unifier\SmartUnifierManager-windows-x64_dev_27_02_23\deploy\12b58255-8c14-45f8-9
2023-02-27 14:02:17,483 - [INFO ] - Main - Instance demo.secsgem2database:SUInstance:__latest process starting on PID:39192
2023-02-27 14:02:19,126 - [INFO ] - DatabaseModelImplementation - demo.secsgem2database:DatabaseChannel:__latest - Configuring channel 48791daa-02a9-46f2-9e15-2413b2fe2f88:latest with config DatabaseModelImplementationConfiguration(DatabaseConnection(SQLServer,jdbc:sqlser
2023-02-27 14:02:19,638 - [INFO ] - SecsGemModelImplementation - demo.secsgem2database:EquipmentChannel:__latest - Configuring channel 93a83dd1-a9ad-4ab2-99f8-4df6779b236a:latest with config SecsGemClientModelImplementationConfiguration(localhost,5000,1,DataFormats(U4,U4,
2023-02-27 14:02:19,707 - [ERROR] - ImplementationFactory$ - SecsGemModelImplementation - The configuration contains context information for /Command/MaterialAcceptCommand/Variable/Parameters/List/RecipeParameters, /Command/MaterialAcceptCommand, /Command/MaterialAcceptCo
2023-02-27 14:02:19,708 - [ERROR] - Main - ### ERROR: Failed to start instance. SecsGemModelImplementation - The configuration contains context information for /Command/MaterialAcceptCommand/Variable/Parameters/List/RecipeParameters, /Command/MaterialAcceptCommand, /Comma
java.lang.IllegalStateException: SecsGemModelImplementation - The configuration contains context information for /Command/MaterialAcceptCommand/Variable/Parameters/List/RecipeParameters, /Command/MaterialAcceptCommand, /Command/MaterialAcceptCommand/Variable/Parameters/Va
at com.amorphsys.i40.adapter.implementation.ImplementationFactory$.$anonfun$setDefinitionImplementation$1(ImplementationFactory.scala:26) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.implementation.ImplementationFactory$.$anonfun$setDefinitionImplementation$1$adapted(ImplementationFactory.scala:19) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at scala.Option.foreach(Option.scala:437) ~[org.scala-lang_2.13.10_scala-library-jar.jar:?]
at com.amorphsys.i40.adapter.implementation.ImplementationFactory$.setDefinitionImplementation(ImplementationFactory.scala:19) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.types.model.ModelFactory$.defineModel(ModelFactory.scala:16) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.TAdapter.createModelInstance(TAdapter.scala:39) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.TAdapter.createModelInstance$(TAdapter.scala:37) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at instance.p12.mb582558c1445f89245a56ef38fef21.vlatest.Instanceb582558c1445f89245a56ef38fef21.createModelInstance(b58255-8c14-45f8-9245-a56ef38fef21.scala:7) ~[12_latest_b58255-8c14-45f8-9245-a56ef38fef21-jar.jar:latest]
at instance.p12.mb582558c1445f89245a56ef38fef21.vlatest.Instanceb582558c1445f89245a56ef38fef21.configure(b58255-8c14-45f8-9245-a56ef38fef21.scala:32) ~[12_latest_b58255-8c14-45f8-9245-a56ef38fef21-jar.jar:latest]
at com.amorphsys.i40.adapter.Main.$anonfun$init$10(Main.scala:134) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.Main.$anonfun$init$10$adapted(Main.scala:132) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at scala.Option.foreach(Option.scala:437) ~[org.scala-lang_2.13.10_scala-library-jar.jar:?]
at com.amorphsys.i40.adapter.Main.init(Main.scala:132) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.Main$.main(Main.scala:27) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.Main.main(Main.scala) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
2023-02-27 14:02:19,721 - [INFO ] - Main - Pid file C:\Users\raa\Desktop\Unifier\SmartUnifierManager-windows-x64_dev_27_02_23\deploy\12b58255-8c14-45f8-9245-a56ef38fef21\pid.txt deleted
2023-02-27 14:02:19,724 - [INFO ] - Main - Stopping all channels ...
2023-02-27 14:04:17,386 - [INFO ] - Main - About to start instance instance.p12.mb582558c1445f89245a56ef38fef21.vlatest.Instanceb582558c1445f89245a56ef38fef21 from folder C:\Users\raa\Desktop\Unifier\SmartUnifierManager-windows-x64_dev_27_02_23\deploy\12b58255-8c14-45f8-9
2023-02-27 14:04:17,461 - [INFO ] - Main - Instance demo.secsgem2database:SUInstance:__latest process starting on PID:34988
2023-02-27 14:04:19,112 - [INFO ] - DatabaseModelImplementation - demo.secsgem2database:DatabaseChannel:__latest - Configuring channel 48791daa-02a9-46f2-9e15-2413b2fe2f88:latest with config DatabaseModelImplementationConfiguration(DatabaseConnection(SQLServer,jdbc:sqlser
2023-02-27 14:04:19,621 - [INFO ] - SecsGemModelImplementation - demo.secsgem2database:EquipmentChannel:__latest - Configuring channel 93a83dd1-a9ad-4ab2-99f8-4df6779b236a:latest with config SecsGemClientModelImplementationConfiguration(localhost,5000,1,DataFormats(U4,U4,
2023-02-27 14:04:19,701 - [ERROR] - ImplementationFactory$ - SecsGemModelImplementation - The configuration contains context information for /Command/MaterialAcceptCommand/Variable/Parameters/List/RecipeParameters, /Command/MaterialAcceptCommand, /Command/MaterialAcceptCo
2023-02-27 14:04:19,703 - [ERROR] - Main - ### ERROR: Failed to start instance. SecsGemModelImplementation - The configuration contains context information for /Command/MaterialAcceptCommand/Variable/Parameters/List/RecipeParameters, /Command/MaterialAcceptCommand, /Comma
java.lang.IllegalStateException: SecsGemModelImplementation - The configuration contains context information for /Command/MaterialAcceptCommand/Variable/Parameters/List/RecipeParameters, /Command/MaterialAcceptCommand, /Command/MaterialAcceptCommand/Variable/Parameters/Va
at com.amorphsys.i40.adapter.implementation.ImplementationFactory$.$anonfun$setDefinitionImplementation$1(ImplementationFactory.scala:26) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.implementation.ImplementationFactory$.$anonfun$setDefinitionImplementation$1$adapted(ImplementationFactory.scala:19) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at scala.Option.foreach(Option.scala:437) ~[org.scala-lang_2.13.10_scala-library-jar.jar:?]
at com.amorphsys.i40.adapter.implementation.ImplementationFactory$.setDefinitionImplementation(ImplementationFactory.scala:19) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.types.model.ModelFactory$.defineModel(ModelFactory.scala:16) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.TAdapter.createModelInstance(TAdapter.scala:39) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.TAdapter.createModelInstance$(TAdapter.scala:37) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at instance.p12.mb582558c1445f89245a56ef38fef21.vlatest.Instanceb582558c1445f89245a56ef38fef21.createModelInstance(b58255-8c14-45f8-9245-a56ef38fef21.scala:7) ~[12_latest_b58255-8c14-45f8-9245-a56ef38fef21-jar.jar:latest]
at instance.p12.mb582558c1445f89245a56ef38fef21.vlatest.Instanceb582558c1445f89245a56ef38fef21.configure(b58255-8c14-45f8-9245-a56ef38fef21.scala:32) ~[12_latest_b58255-8c14-45f8-9245-a56ef38fef21-jar.jar:latest]
```

Log Levels

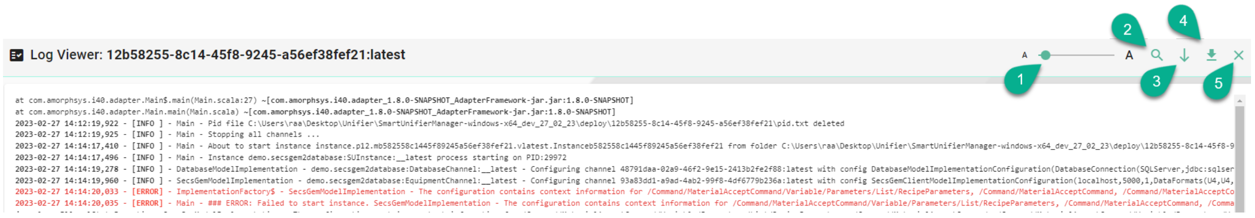
The log viewer will show the details of logs based on the level defined throughout the creation of the deployment:

- TRACE - The most fine-grained information only used in rare cases where full visibility of what is happening inside a Communication Instance.
- DEBUG - Less granular compared to the TRACE level, but more than needed in a production environment. The DEBUG log level should be used for troubleshooting a faulty Communication Instance or when running a Communication Instance inside a test environment.
- INFO - Is the standard log level used for a standard deployment of a Communication Instance.
- WARNING - Log level that indicates that something unexpected happened inside a Communication Instance that might cause problems for the course of communication.

Log Viewer Operations

Log Viewer provides the following features:

- Adjust Font - Using the slider, the log font can be adjusted **(1)**
- Filter - Searching based on a regular expression (Regex) **(2)**
- Scroll to End - Follow Tail to make the latest log lines visible **(3)**
- Download Logs - Save the logs into a ZIP file on the machine for analysis and sharing **(4)**
- Close - To exit the Log Viewer and return to Deployment perspective **(5)**



```
at com.amorphsys.i40.adapter.Main$.main(Main.scala:27) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
at com.amorphsys.i40.adapter.Main.main(Main.scala) ~[com.amorphsys.i40.adapter_1.8.0-SNAPSHOT_AdapterFramework-jar.jar:1.8.0-SNAPSHOT]
2023-02-27 14:12:19,922 - [INFO ] - Main - Pid file C:\Users\raa\Desktop\Unifier\SmartUnifierManager-windows-x64_dev_27_02_23\deploy\12b58255-8c14-45f8-9245-a56ef38fef21\pid.txt deleted
2023-02-27 14:12:19,925 - [INFO ] - Main - Stopping all channels ...
2023-02-27 14:14:17,410 - [INFO ] - Main - About to start instance instance.p12.mb582558c1445f89245a56ef38fef21.vlatest.Instanceb582558c1445f89245a56ef38fef21 from folder C:\Users\raa\Desktop\Unifier\SmartUnifierManager-windows-x64_dev_27_02_23\deploy\12b58255-8c14-45f8-9
2023-02-27 14:14:17,496 - [INFO ] - Main - Instance demo.secsgem2database:SUInstance:__latest process starting on PID:29972
2023-02-27 14:14:19,278 - [INFO ] - DatabaseModelImplementation - demo.secsgem2database:DatabaseChannel:__latest - Configuring channel 48791daa-02a9-46f2-9e15-2413b2fe2f88:latest with config DatabaseModelImplementationConfiguration(DatabaseConnection(SQLServer,jdbc:sqlser
2023-02-27 14:14:19,960 - [INFO ] - SecsGemModelImplementation - demo.secsgem2database:EquipmentChannel:__latest - Configuring channel 93a83dd1-a9ad-4ab2-99f8-4df6779b236a:latest with config SecsGemClientModelImplementationConfiguration(localhost,5000,1,DataFormats(U4,U4,
2023-02-27 14:14:20,033 - [ERROR] - ImplementationFactory$ - SecsGemModelImplementation - The configuration contains context information for /Command/MaterialAcceptCommand/Variable/Parameters/List/RecipeParameters, /Command/MaterialAcceptCommand, /Command/MaterialAcceptCo
2023-02-27 14:14:20,035 - [ERROR] - Main - ### ERROR: Failed to start instance. SecsGemModelImplementation - The configuration contains context information for /Command/MaterialAcceptCommand/Variable/Parameters/List/RecipeParameters, /Command/MaterialAcceptCommand, /Comma
```

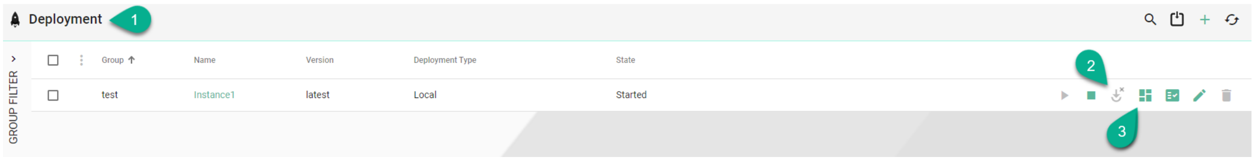
4.9.2 Dashboard

SMART**UNIFIER** provides a Dashboard with an integrated Log Viewer, which helps to gain insights in running Communication Instance performance.

How to access

Follow the steps bellow to access the Dashboard:

- Select the SMARTUNIFIER Deployment perspective **(1)**
- Make sure the Instance is Deployed **(2)**
- Click on the **Dashboard** button **(3)**

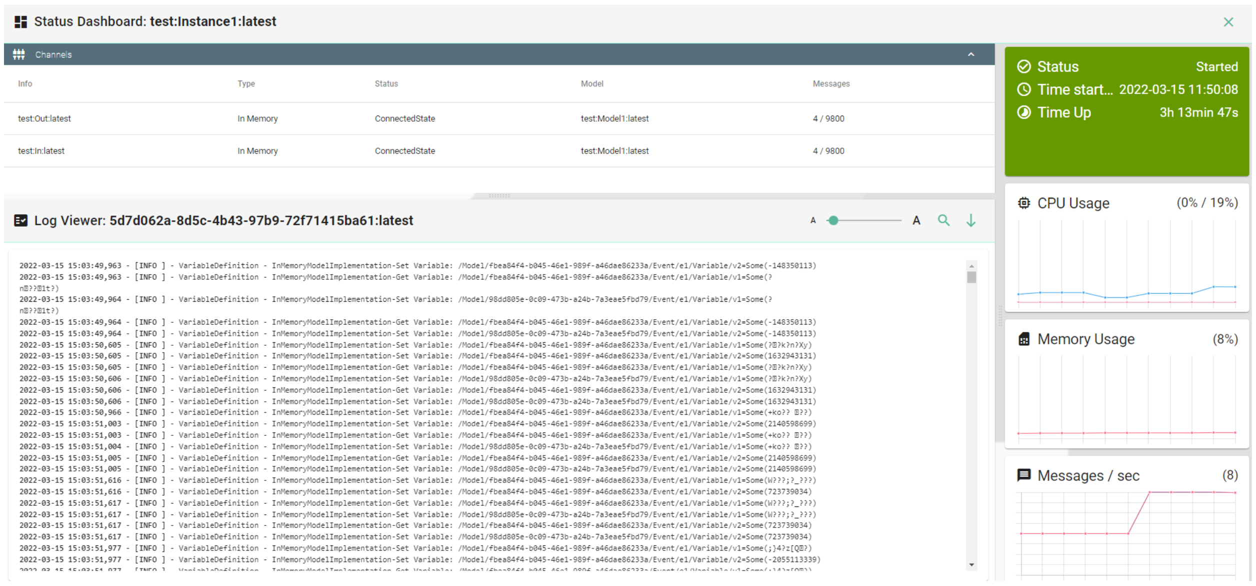


Group	Name	Version	Deployment Type	State
test	Instance1	latest	Local	Started

Dashboard's Data

The Dashboard provides the following information:

- *Channels* associated with the Instance
- *Log Viewer*
- Status of the Instance
- Start time of the Instance
- Instance Time Up
- CPU Usage of the Instance
- Memory Usage of the Instance
- Sent and received messages



Info	Type	Status	Model	Messages
test:Out:latest	In Memory	ConnectedState	test:Model1:latest	4 / 9800
test:In:latest	In Memory	ConnectedState	test:Model1:latest	4 / 9800

Status	Started
Time start...	2022-03-15 11:50:08
Time Up	3h 13min 47s

Log Viewer: 5d7d062a-8d5c-4b43-97b9-72f71415ba61:latest

```
2022-03-15 15:03:49,963 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v2=Some(-148350113)
2022-03-15 15:03:49,963 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Get Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v1=Some(?
n?1t?)
2022-03-15 15:03:49,964 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/98dd805e-0c09-473b-a24b-7a3eae5fbd79/Event/e1/Variable/v1=Some(?
n?1t?)
2022-03-15 15:03:49,964 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Get Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v2=Some(-148350113)
2022-03-15 15:03:49,964 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/98dd805e-0c09-473b-a24b-7a3eae5fbd79/Event/e1/Variable/v2=Some(-148350113)
2022-03-15 15:03:50,605 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v1=Some(??k?n?Xy)
2022-03-15 15:03:50,605 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v2=Some(1632943131)
2022-03-15 15:03:50,605 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Get Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v1=Some(??k?n?Xy)
2022-03-15 15:03:50,606 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/98dd805e-0c09-473b-a24b-7a3eae5fbd79/Event/e1/Variable/v1=Some(??k?n?Xy)
2022-03-15 15:03:50,606 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Get Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v2=Some(1632943131)
2022-03-15 15:03:50,606 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/98dd805e-0c09-473b-a24b-7a3eae5fbd79/Event/e1/Variable/v2=Some(1632943131)
2022-03-15 15:03:50,966 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v1=Some(+ko?? ??)
2022-03-15 15:03:51,003 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v2=Some(2140598699)
2022-03-15 15:03:51,003 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Get Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v1=Some(+ko?? ??)
2022-03-15 15:03:51,004 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/98dd805e-0c09-473b-a24b-7a3eae5fbd79/Event/e1/Variable/v1=Some(+ko?? ??)
2022-03-15 15:03:51,005 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Get Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v2=Some(2140598699)
2022-03-15 15:03:51,005 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/98dd805e-0c09-473b-a24b-7a3eae5fbd79/Event/e1/Variable/v2=Some(2140598699)
2022-03-15 15:03:51,616 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v1=Some(W???;?_???)
2022-03-15 15:03:51,616 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v2=Some(723739034)
2022-03-15 15:03:51,617 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Get Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v1=Some(W???;?_???)
2022-03-15 15:03:51,617 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/98dd805e-0c09-473b-a24b-7a3eae5fbd79/Event/e1/Variable/v1=Some(W???;?_???)
2022-03-15 15:03:51,617 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Get Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v2=Some(723739034)
2022-03-15 15:03:51,617 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/98dd805e-0c09-473b-a24b-7a3eae5fbd79/Event/e1/Variable/v2=Some(723739034)
2022-03-15 15:03:51,977 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v1=Some(;}4?z[Q?)
2022-03-15 15:03:51,977 - [INFO ] - VariableDefinition - InMemoryModelImplementation-Set Variable: /Model/fbea84f4-b045-46e1-989f-a46dae86233a/Event/e1/Variable/v2=Some(-2055113339)
```

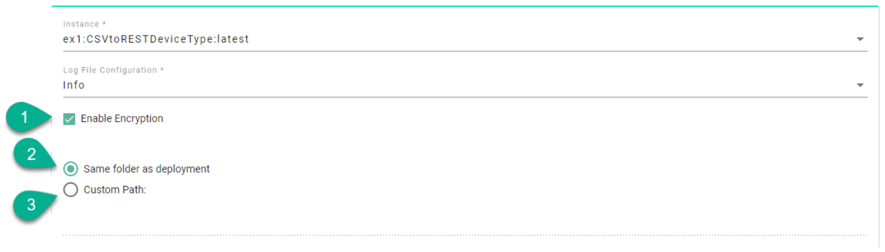
4.10 Additional Options

4.10.1 Encryption of Communication Instances

This feature provides the possibility to encrypt the configuration files of Communication Channels used by the Instance, which may contain credentials to access a database or external services. The encryption method used is Advanced Encryption Standard (AES).

The encryption is available for all deployment options, by following the steps bellow:

- Check the **Enable Encryption** box **(1)**.
- A symmetrical key (cfg.key) is generated and can be saved in the same folder as the deployment **(2)** or check the **Custom Path** option **(3)** to save the key into a secured location.

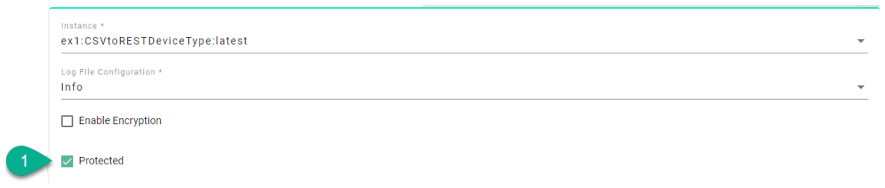


Instance *
ex1:CSVtoRESTDeviceType:latest
Log File Configuration *
Info
☑ Enable Encryption
● Same folder as deployment
○ Custom Path:

4.10.2 Protect Communication Instances

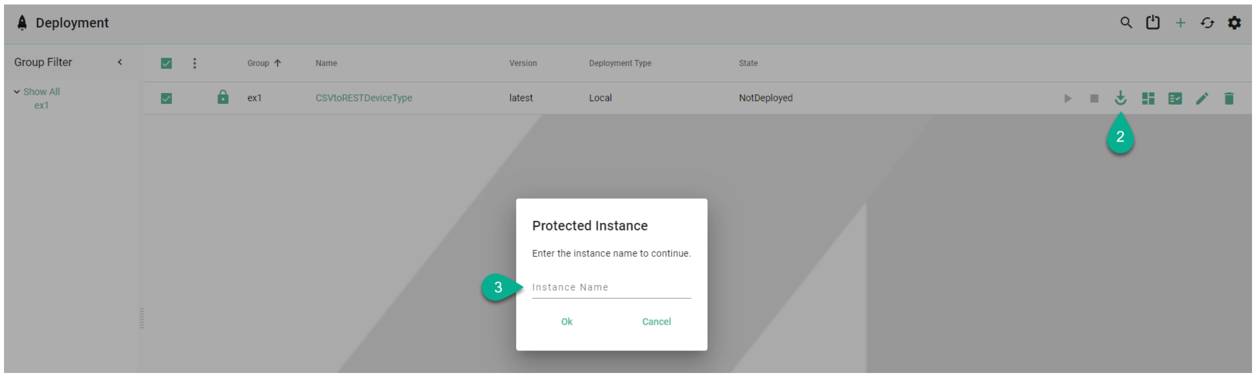
This feature provides an additional protection when performing an Instance action (e.g., deploy, undeploy, start, stop).

The protection is available for all deployment options, by checking the **Protected** box **(1)**.



Instance *
ex1:CSVtoRESTDeviceType:latest
Log File Configuration *
Info
☐ Enable Encryption
☑ Protected

Now the Instance is protected, meaning that when the user performs an action like Deploy **(2)**, a popup appears requiring to input the Instance name **(3)**.



	Group	Name	Version	Deployment Type	State
✓	ex1	CSVtoRESTDeviceType	latest	Local	NotDeployed

Protected Instance

Enter the instance name to continue.

Instance Name

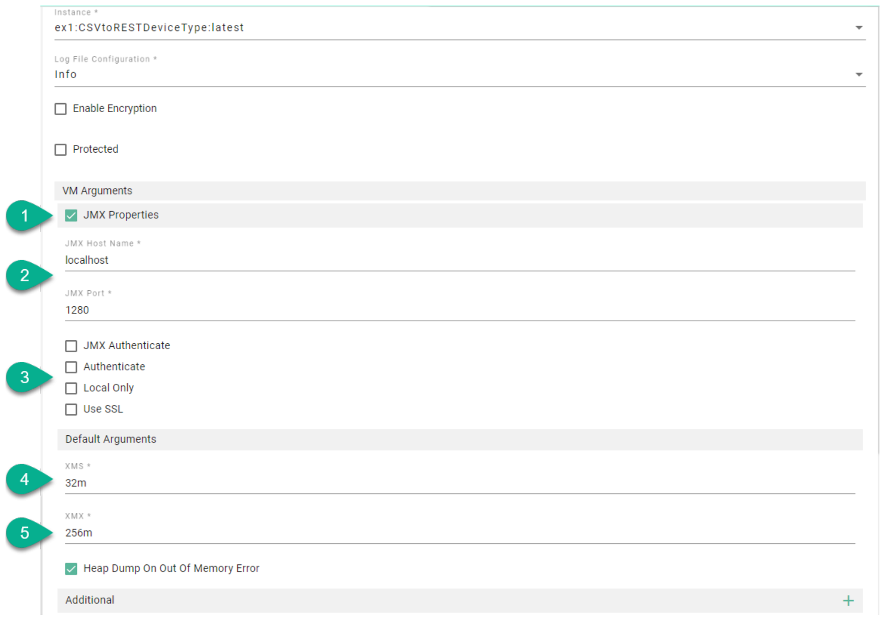
Note: Protected Instances will not work with Bulk actions.

4.10.3 VM Arguments

This feature provides the possibility to configure the Java Virtual Machine (JVM). In some cases, when dealing with larger files when using the File Reader Communication Channel (large XML file), it might be necessary to increase the **XMX** in order to avoid running into a *java.lang.OutOfMemoryError* - exception.

VM Arguments can be configured when deploying an Communication Instance locally or on Docker, by following the steps below:

- Check the **JMX Properties** box **(1)** to expand the Java Management Extensions parameters and input the **JMX Host Name** and **Port (2)**.
- Check the authentication method **(3)**.
- Update the **XMS** value **(4)**, minimal heap size, representing the amount of memory used by the JVM to start with.
- Update the **XMX** value **(5)**, maximal heap size, representing the maximum amount of memory that JVM will be able to use.



Instance *
ex1:CSVtoRESTDeviceType:latest

Log File Configuration *
Info

☐ Enable Encryption

☐ Protected

VM Arguments

✓ JMX Properties

JMX Host Name *
localhost

JMX Port *
1280

☐ JMX Authenticate
☐ Authenticate
☐ Local Only
☐ Use SSL

Default Arguments

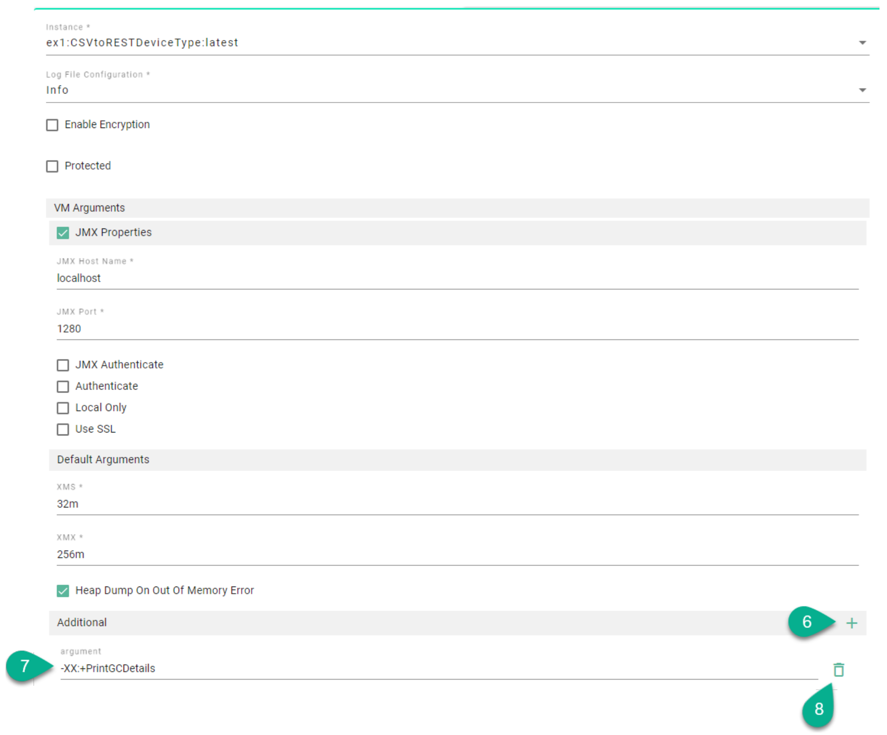
XMS *
32m

XMX *
256m

✓ Heap Dump On Out Of Memory Error

Additional +

- By default, the **Heap Dump On Out Of Memory Error** option is checked, providing an analysis file for debugging.
- Additional JVM arguments can be added by selecting the **add Arg** button **(6)** and input the argument **(7)**. For example, to debug memory issues or application performance, the Garbage Collection logging can be enabled in JVM, as seen below.



Instance *
ex1:CSVtoRESTDeviceType:latest

Log File Configuration *
Info

- [ ] Enable Encryption
- [ ] Protected

VM Arguments

- [x] JMX Properties

JMX Host Name *
localhost

JMX Port *
1280

- [ ] JMX Authenticate
- [ ] Authenticate
- [ ] Local Only
- [ ] Use SSL

Default Arguments

XMS *
32m

XMX *
256m

- [x] Heap Dump On Out Of Memory Error

Additional

argument
-XX:+PrintGCDetails

- An additional argument can be deleted by clicking on the **delete Arg** button **(8)**.

CHAPTER FIVE

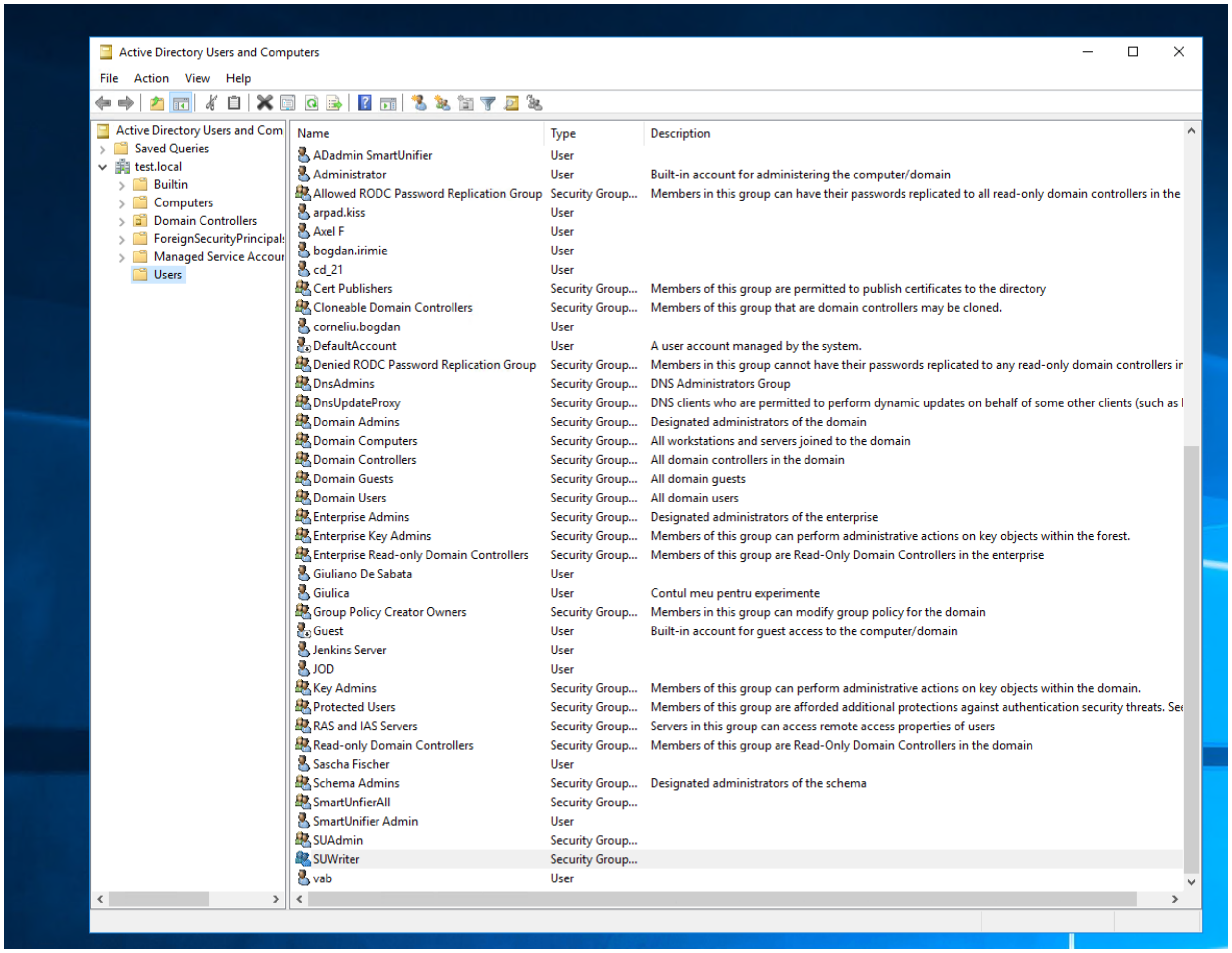
ADMINISTRATION

Learn how to:

- Integrate an *Active Directory*
- *Backup* and *Restore* the Repository
- Manage *Communication Channel Types*
- Manage *Docker Java Images*
- Create *Deployment Endpoints*
- Manage *Credentials*
- Manage *User Accounts*
- Manage *Logging Configurations*
- Create *Alerts*
- Manage *Alert Channels*
- Use *Extensions*
- Validate the *Configuration Components*

5.1 Active Directory Integration (ADI)

SMART**UNIFIER** supports Windows Active Directory (AD). System administrators can use the Active Directory to add/remove users, groups, and resources quickly and efficiently through one dashboard.

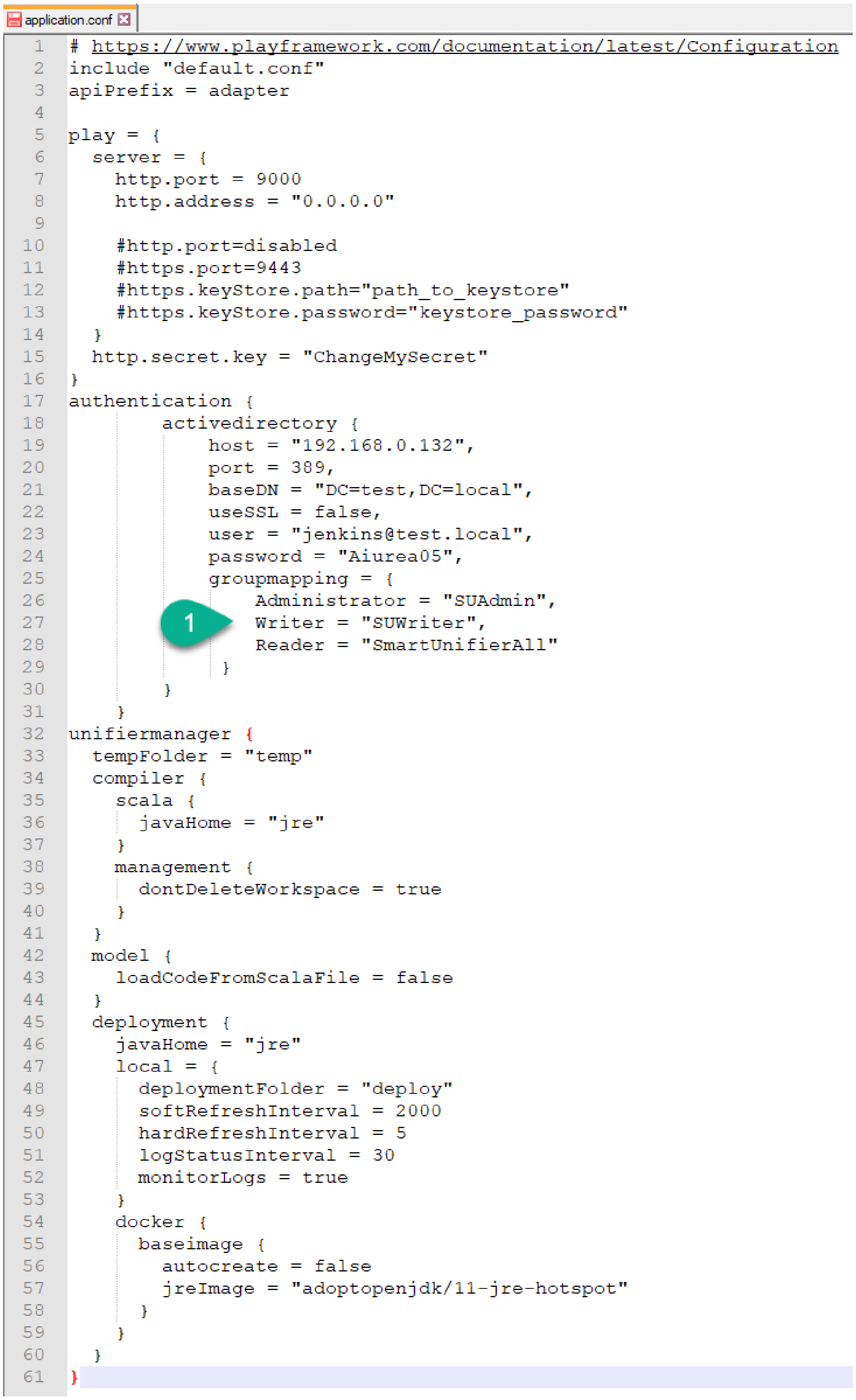


Name	Type	Description
ADadmin SmartUnifier	User	
Administrator	User	Built-in account for administering the computer/domain
Allowed RODC Password Replication Group	Security Group...	Members in this group can have their passwords replicated to all read-only domain controllers in the
arpad.kiss	User	
Axel F	User	
bogdan.irimie	User	
cd_21	User	
Cert Publishers	Security Group...	Members of this group are permitted to publish certificates to the directory
Cloneable Domain Controllers	Security Group...	Members of this group that are domain controllers may be cloned.
corneliu.bogdan	User	
DefaultAccount	User	A user account managed by the system.
Denied RODC Password Replication Group	Security Group...	Members in this group cannot have their passwords replicated to any read-only domain controllers ir
DnsAdmins	Security Group...	DNS Administrators Group
DnsUpdateProxy	Security Group...	DNS clients who are permitted to perform dynamic updates on behalf of some other clients (such as I
Domain Admins	Security Group...	Designated administrators of the domain
Domain Computers	Security Group...	All workstations and servers joined to the domain
Domain Controllers	Security Group...	All domain controllers in the domain
Domain Guests	Security Group...	All domain guests
Domain Users	Security Group...	All domain users
Enterprise Admins	Security Group...	Designated administrators of the enterprise
Enterprise Key Admins	Security Group...	Members of this group can perform administrative actions on key objects within the forest.
Enterprise Read-only Domain Controllers	Security Group...	Members of this group are Read-Only Domain Controllers in the enterprise
Giuliano De Sabata	User	
Giulica	User	Contul meu pentru experimente
Group Policy Creator Owners	Security Group...	Members in this group can modify group policy for the domain
Guest	User	Built-in account for guest access to the computer/domain
Jenkins Server	User	
JOD	User	
Key Admins	Security Group...	Members of this group can perform administrative actions on key objects within the domain.
Protected Users	Security Group...	Members of this group are afforded additional protections against authentication security threats. See
RAS and IAS Servers	Security Group...	Servers in this group can access remote access properties of users
Read-only Domain Controllers	Security Group...	Members of this group are Read-Only Domain Controllers in the domain
Sascha Fischer	User	
Schema Admins	Security Group...	Designated administrators of the schema
SmartUnfierAll	Security Group...	
SmartUnifier Admin	User	
SUAdmin	Security Group...	
SUWriter	Security Group...	
vab	User	

5.1.1 AD Group Mapping

An user from AD must be added to a group that acts as a role. The role determines what permissions are assigned to the user.

The mapping between the AD groups and the SMART**UNIFIER** roles is defined in the **application.conf** file from the **conf** folder.



```
# https://www.playframework.com/documentation/latest/Configuration
include "default.conf"
apiPrefix = adapter

play = {
  server = {
    http.port = 9000
    http.address = "0.0.0.0"

    #http.port=disabled
    #https.port=9443
    #https.keyStore.path="path_to_keystore"
    #https.keyStore.password="keystore_password"
  }
  http.secret.key = "ChangeMySecret"
}
authentication {
        activedirectory {
            host = "192.168.0.132",
            port = 389,
            baseDN = "DC=test,DC=local",
            useSSL = false,
            user = "jenkins@test.local",
            password = "Aiurea05",
            groupmapping = {
                Administrator = "SUAdmin",
                Writer = "SUWriter",
                Reader = "SmartUnifierAll"
             }
        }
    }
unifiermanager {
  tempFolder = "temp"
  compiler {
    scala {
      javaHome = "jre"
    }
    management {
      dontDeleteWorkspace = true
    }
  }
  model {
    loadCodeFromScalaFile = false
  }
  deployment {
    javaHome = "jre"
    local = {
      deploymentFolder = "deploy"
      softRefreshInterval = 2000
      hardRefreshInterval = 5
      logStatusInterval = 30
      monitorLogs = true
    }
    docker {
      baseimage {
        autocreate = false
        jreImage = "adoptopenjdk/11-jre-hotspot"
      }
    }
  }
}
```

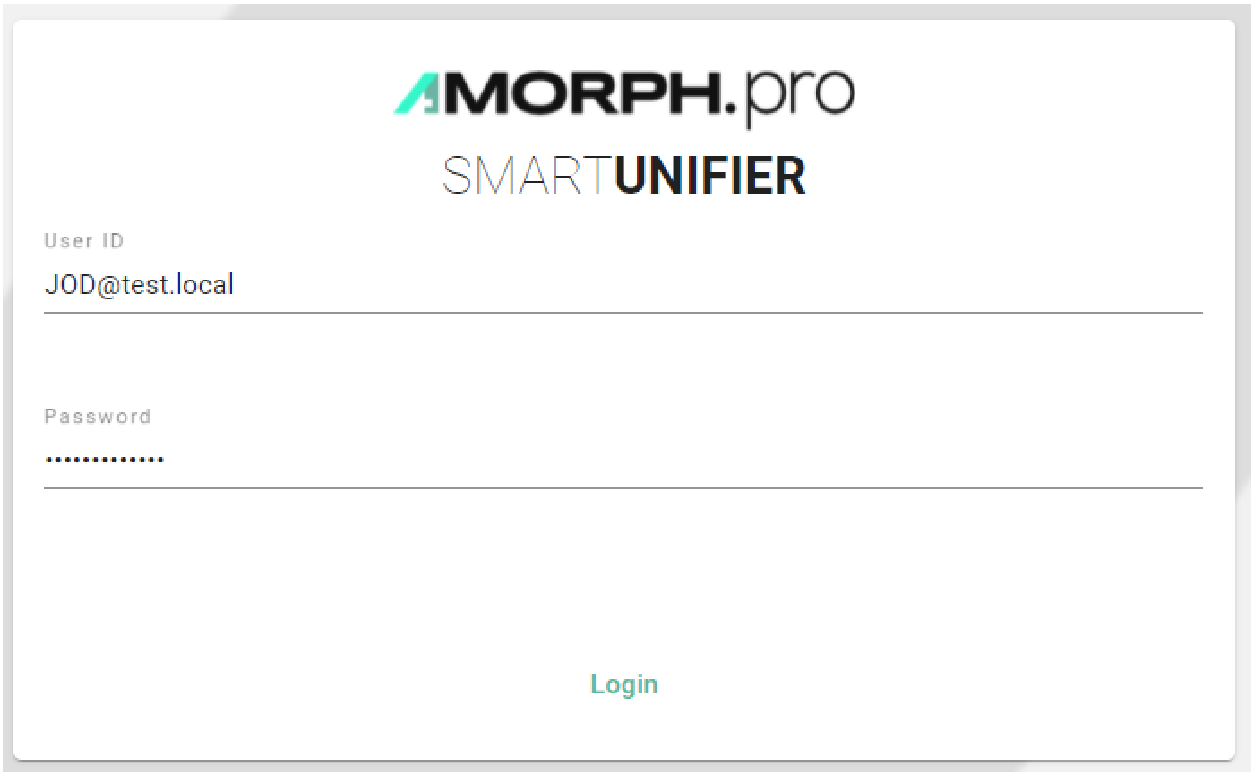
As seen above **(1)** in the left side are the SMART**UNIFIER** roles and in the right side, between the quotation marks are the AD groups.

The SMARTUNIFIER roles are predefined:

- Administrator - global permission
- Writer - limited permission, write and read access
- Reader - limited permission, read access

A user from an AD group will have permission based on the mapping of the AD group to a predefined SMART**UNIFIER** role.

After all the above configuration is done, the user can login to the SMART**UNIFIER** with the **User logon name** and the **Password** defined in AD.



JOD@test.local

5.2 Backup and Restore

SMART**UNIFIER** provides the possibility to manually backup and restore the repository and the internal database.

The **repository** represents a central location in which all the configuration components are stored:

- Information Models
- Communication Channels
- Mappings

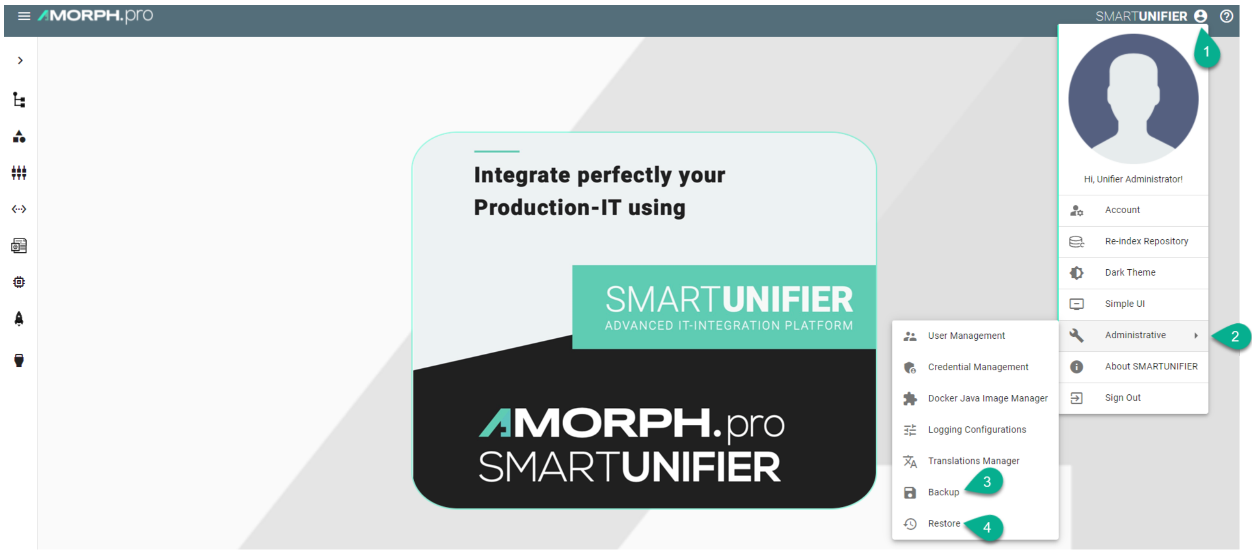
- Device Types
- Communication Instances

The **internal database** is used for the operation of the SMART**UNIFIER** Manager and stores information like:

- User Accounts and Credentials
- Deployment Endpoints
- Base Images
- Channel Types
- Alert Configurations and Channels

5.2.1 How to access

To access the Backup or the Restore option, click on the **Account** icon **(1)**, go to the **Administrative** option **(2)** and select **Backup (3)** or **Restore (4)**.



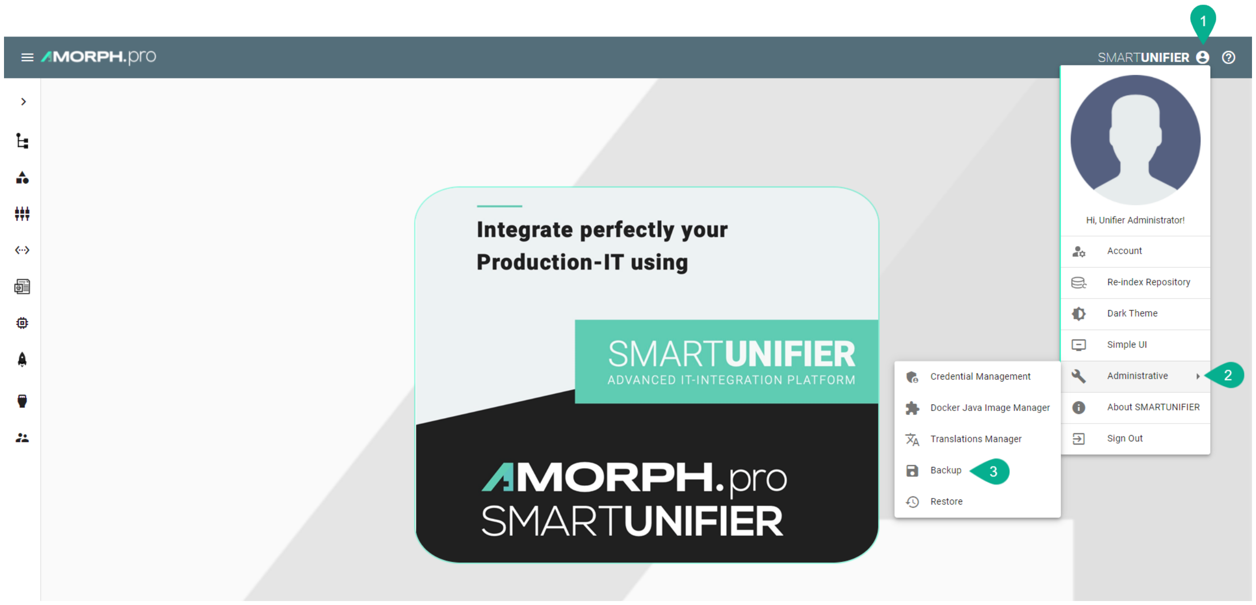
Note: The Backup and the Restore features can only be accessed by user accounts with an administrator role assigned. Also keep in mind that the same SMART**UNIFIER** Manager version must be used.

5.2.2 Backup

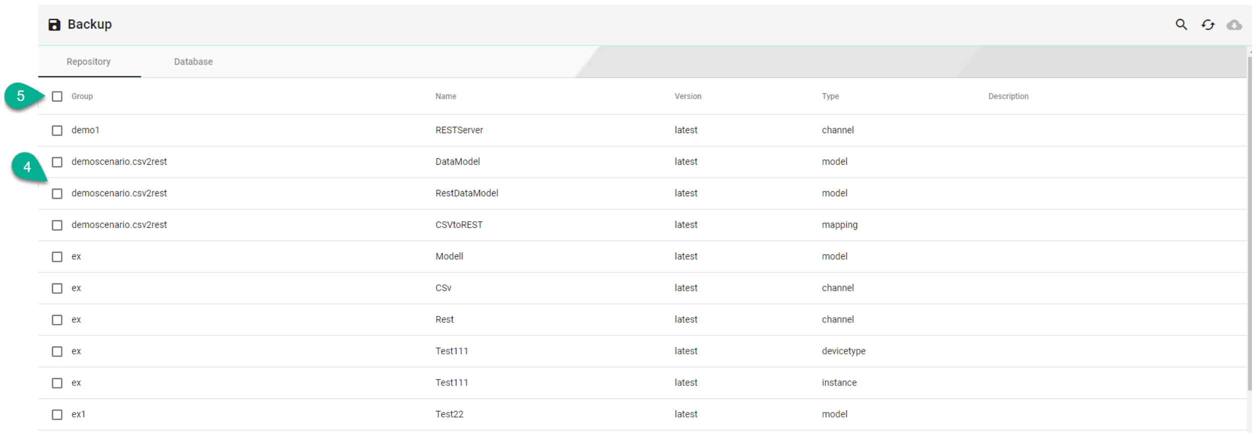
The Backup feature provides the possibility to create a copy of the configuration components to store elsewhere, so that it can be used to restore the last used after a data loss event occurs.

Follow the steps described below to create a backup of the repository:

- Select the **Account** icon **(1)**, go to the **Administrative** section **(2)** and select the **Backup** option **(3)**.



- The configuration components (Repository) are visible. Check the boxes **(4)** to select what to backup or check the top box **(5)** to select all.

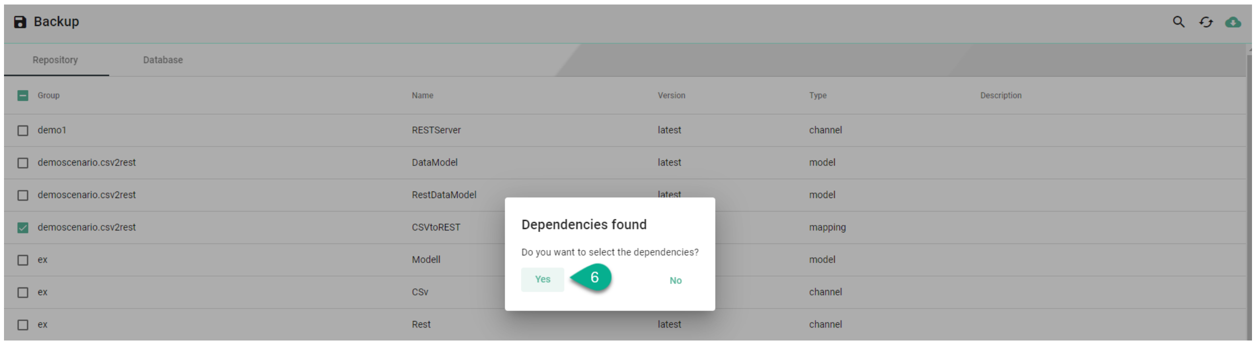


Backup

Repository | Database

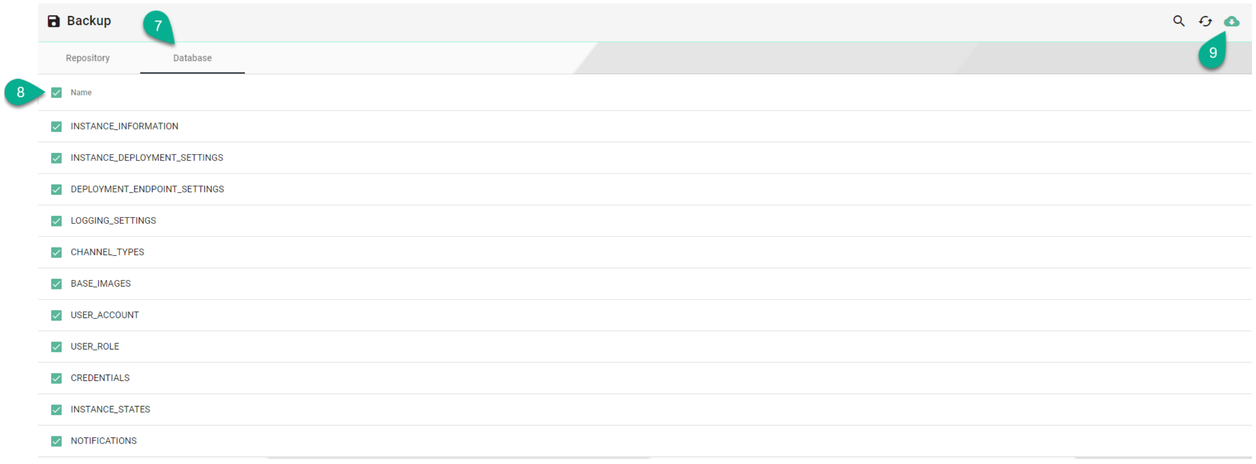
Group	Name	Version	Type	Description
demo1	RESTServer	latest	channel	
demoscenario.csv2rest	DataModel	latest	model	
demoscenario.csv2rest	RestDataModel	latest	model	
demoscenario.csv2rest	CSVtoREST	latest	mapping	
ex	Modell	latest	model	
ex	CSv	latest	channel	
ex	Rest	latest	channel	
ex	Test111	latest	devicetype	
ex	Test111	latest	instance	
ex1	Test22	latest	model	

- When the selected component has dependencies, a pop-up will appear and click on the **Yes** button **(6)** to select all the dependencies.



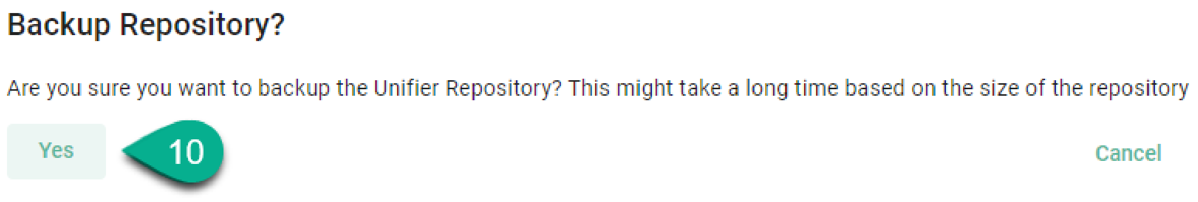
Group	Name	Version	Type	Description
☐ demo1	RESTServer	latest	channel	
☐ demoscenario.csv2rest	DataModel	latest	model	
☐ demoscenario.csv2rest	RestDataModel	latest	model	
✓ demoscenario.csv2rest	CSVtoREST		mapping	
☐ ex	Modell		model	
☐ ex	CSV		channel	
☐ ex	Rest	latest	channel	

- Click on the **Database** tab **(7)** and select the desired tables **(8)** for backup.



Name
✓ INSTANCE_INFORMATION
✓ INSTANCE_DEPLOYMENT_SETTINGS
✓ DEPLOYMENT_ENDPOINT_SETTINGS
✓ LOGGING_SETTINGS
✓ CHANNEL_TYPES
✓ BASE_IMAGES
✓ USER_ACCOUNT
✓ USER_ROLE
✓ CREDENTIALS
✓ INSTANCE_STATES
✓ NOTIFICATIONS

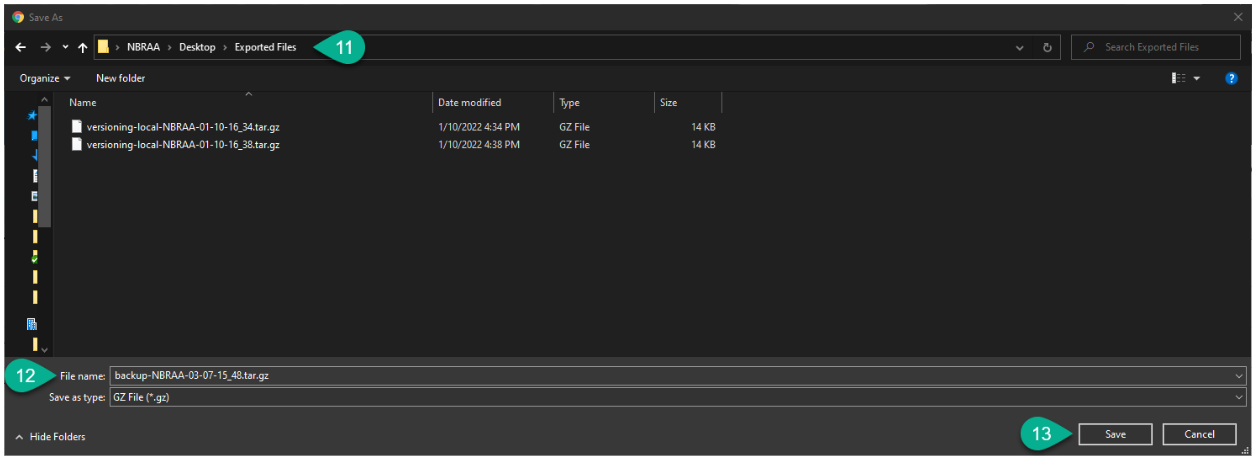
- After the desired components are selected, click on the **Backup** button **(9)**. Click on the **Yes** button **(10)** to confirm.



Backup Repository?

Are you sure you want to backup the Unifier Repository? This might take a long time based on the size of the repository

- Choose the path **(11)** and the name **(12)** to save the repository **TAR** file then click on the **Save** button **(13)** to finish.



Name	Date modified	Type	Size
versioning-local-NBRAA-01-10-16_34.tar.gz	1/10/2022 4:34 PM	GZ File	14 KB
versioning-local-NBRAA-01-10-16_38.tar.gz	1/10/2022 4:38 PM	GZ File	14 KB

File name: backup-NBRAA-03-07-15_48.tar.gz

Save as type: GZ File (*.gz)

5.2.3 Restore

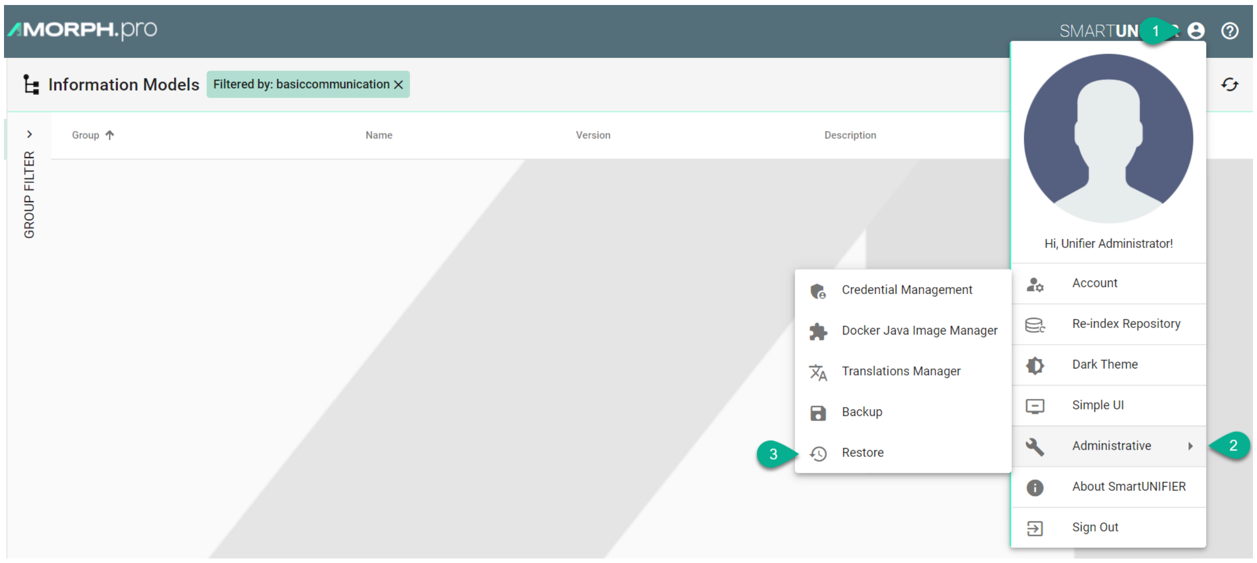
The Restore feature allows you to recover SMART**UNIFIER** configuration components from a backup and apply them to your current SMART**UNIFIER** Manager. This feature is particularly useful to share configuration components across several SMART**UNIFIER** Manager installations or when you encounter unexpected issues and need to revert back to a previous state.

Note: When restoring, the existing configuration components will be overwritten by with the selected configuration components from the backup if the name match!

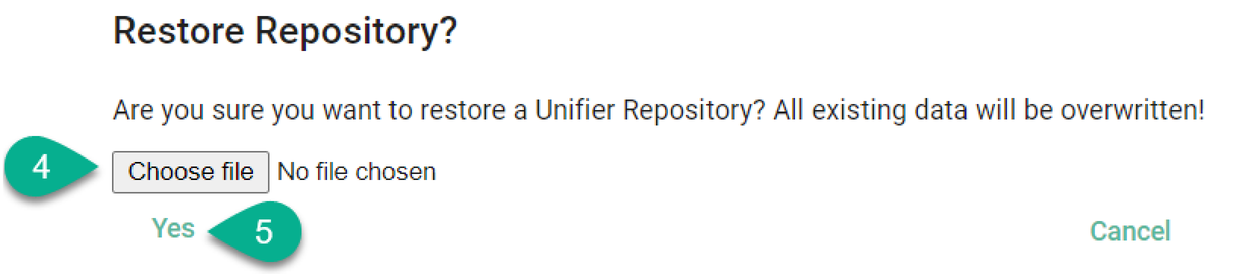
Note: Before restoring, ensure that you undeploy all communication instances.

Follow the steps described below to restore the SMART**UNIFIER** repository:

- Select the **Account** icon **(1)**, go to the **Administrative** section **(2)** and select the **Restore** option **(3)**.



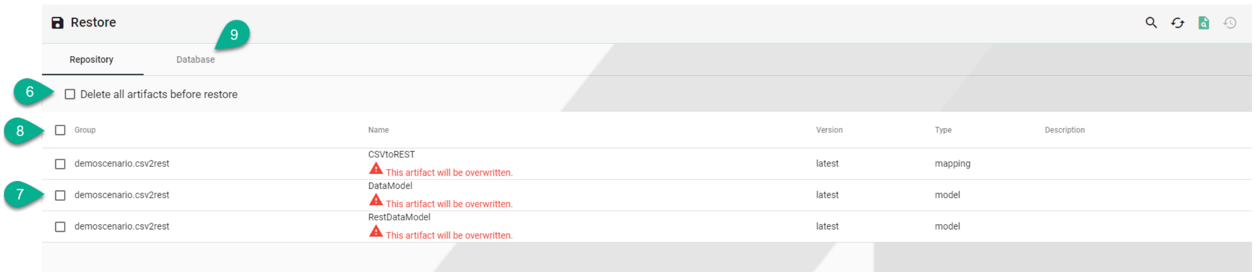
- A pop-up appears, choose the **TAR** file to restore **(4)** and select the **Yes** button **(5)** to confirm.



Restore Repository?

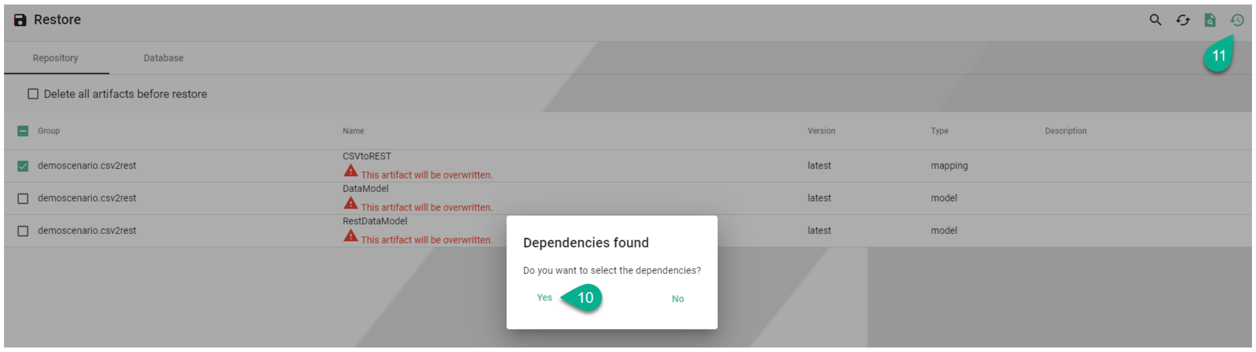
Are you sure you want to restore a Unifier Repository? All existing data will be overwritten!

- The backup configuration components (Repository) are visible.
- If needed, check the box **(6)** to delete all existing components, before restoring.
- Check the boxes **(7)** to select what to restore or check the top box **(8)** to select all. Do the same for **Database** tab **(9)** if needed.



Group	Name	Version	Type	Description
demoscenario.csv2rest	CSVtoREST ⚠ This artifact will be overwritten.	latest	mapping	
demoscenario.csv2rest	DataModel ⚠ This artifact will be overwritten.	latest	model	
demoscenario.csv2rest	RestDataModel ⚠ This artifact will be overwritten.	latest	model	

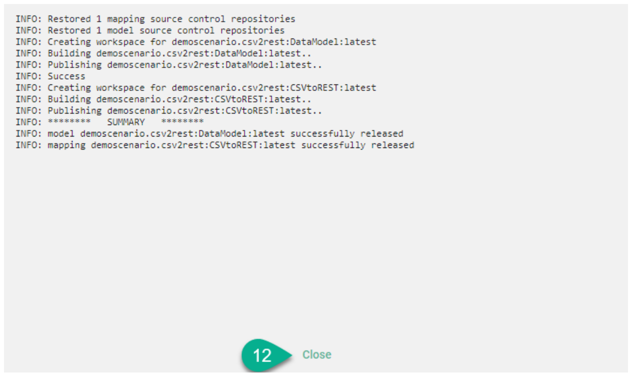
- If a component from the current configuration (if any) has the same name as one from the backup, it will be overwritten.
- When the selected component has dependencies, a pop-up will appear and click on the **Yes** button **(10)** to select all the dependencies.



☐ Delete all artifacts before restore

	Group	Name	Version	Type	Description
✓	demoscenario.csv2rest	CSVtoREST This artifact will be overwritten.	latest	mapping	
☐	demoscenario.csv2rest	DataModel This artifact will be overwritten.	latest	model	
☐	demoscenario.csv2rest	RestDataModel This artifact will be overwritten.	latest	model	

- After the desired components are selected, click on the **Restore** button **(11)**.
- The configuration components are uploading and all existing data will be overwritten!
- The uploading progress is displayed, including errors, if any.



```
INFO: Restored 1 mapping source control repositories
INFO: Restored 1 model source control repositories
INFO: Creating workspace for demoscenario.csv2rest:DataModel:latest
INFO: Building demoscenario.csv2rest:DataModel:latest..
INFO: Publishing demoscenario.csv2rest:DataModel:latest..
INFO: Success
INFO: Creating workspace for demoscenario.csv2rest:CSVtoREST:latest
INFO: Building demoscenario.csv2rest:CSVtoREST:latest..
INFO: Publishing demoscenario.csv2rest:CSVtoREST:latest..
INFO: ********   SUMMARY   ********
INFO: model demoscenario.csv2rest:DataModel:latest successfully released
INFO: mapping demoscenario.csv2rest:CSVtoREST:latest successfully released
```

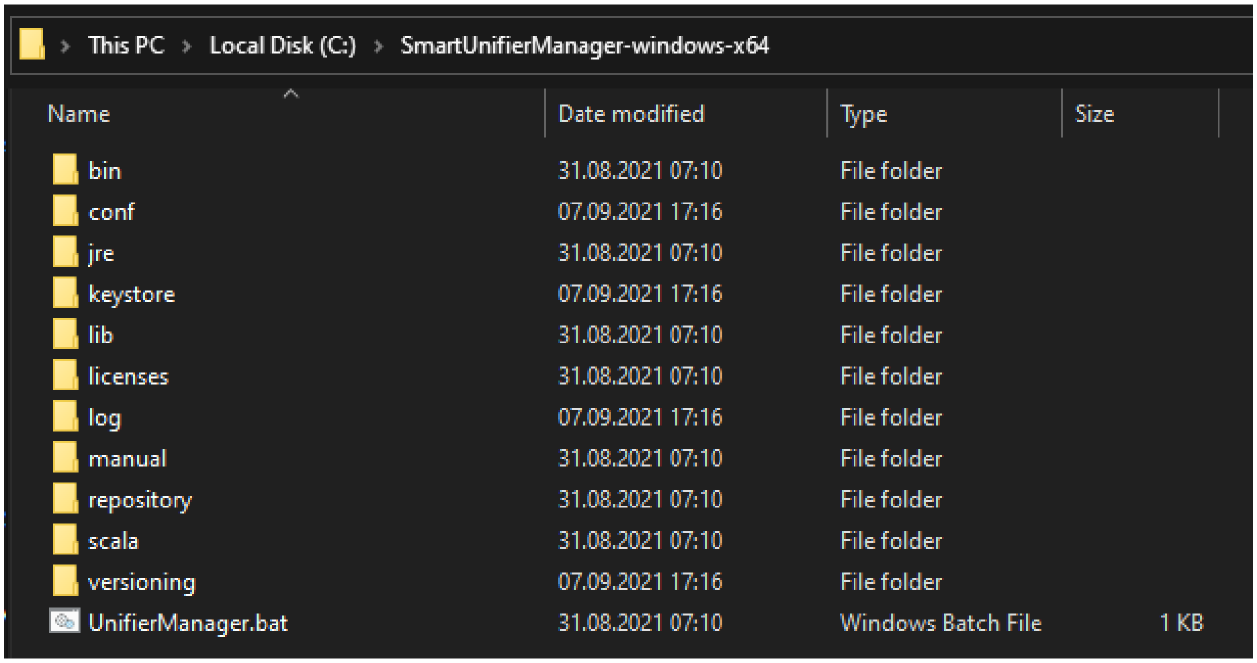
- Click on the **Close** button to finish **(12)**.

5.2.4 Manager Backup

In order to backup SMART**UNIFIER** Manager make a copy of the SMART**UNIFIER** installation package.

Before the backup make sure to remove the following directories:

- temp
- workspace
- log
- deploy



This PC › Local Disk (C:) › SmartUnifierManager-windows-x64

Name	Date modified	Type	Size
bin	31.08.2021 07:10	File folder	
conf	07.09.2021 17:16	File folder	
jre	31.08.2021 07:10	File folder	
keystore	07.09.2021 17:16	File folder	
lib	31.08.2021 07:10	File folder	
licenses	31.08.2021 07:10	File folder	
log	07.09.2021 17:16	File folder	
manual	31.08.2021 07:10	File folder	
repository	31.08.2021 07:10	File folder	
scala	31.08.2021 07:10	File folder	
versioning	07.09.2021 17:16	File folder	
UnifierManager.bat	31.08.2021 07:10	Windows Batch File	1 KB

5.3 Channel Types Manager

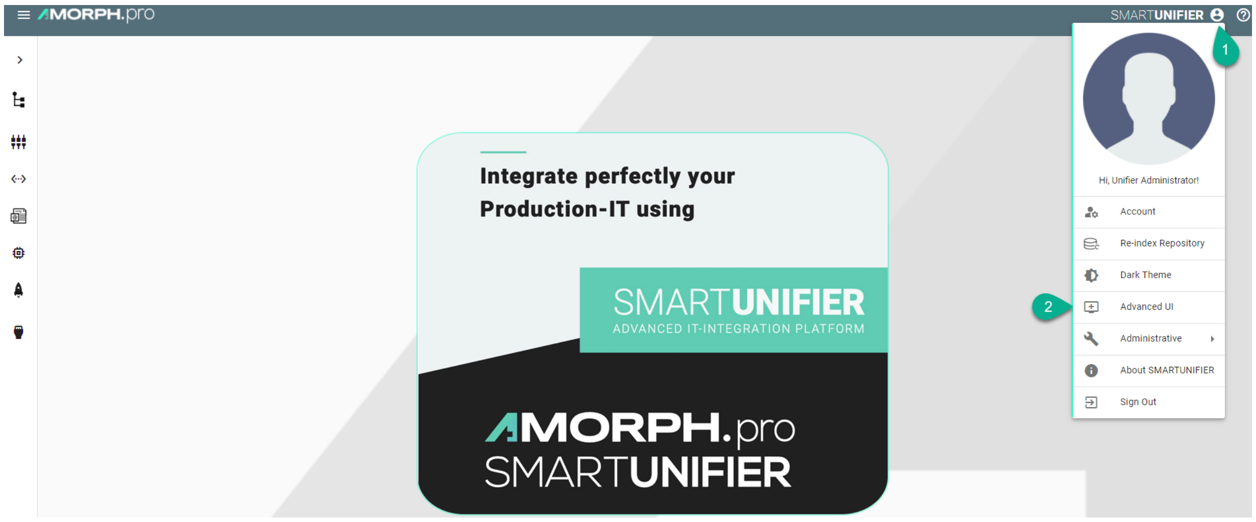
By default, the Channel Types Manager displays all *Channels* included in your current version of SMART**UNIFIER**.

Communication Channels that should be used within the configuration of a SMART**UNIFIER** Communication Instance have to exist in the Channel Types Manager. How to add new Channel Types is shown in the *section below*.

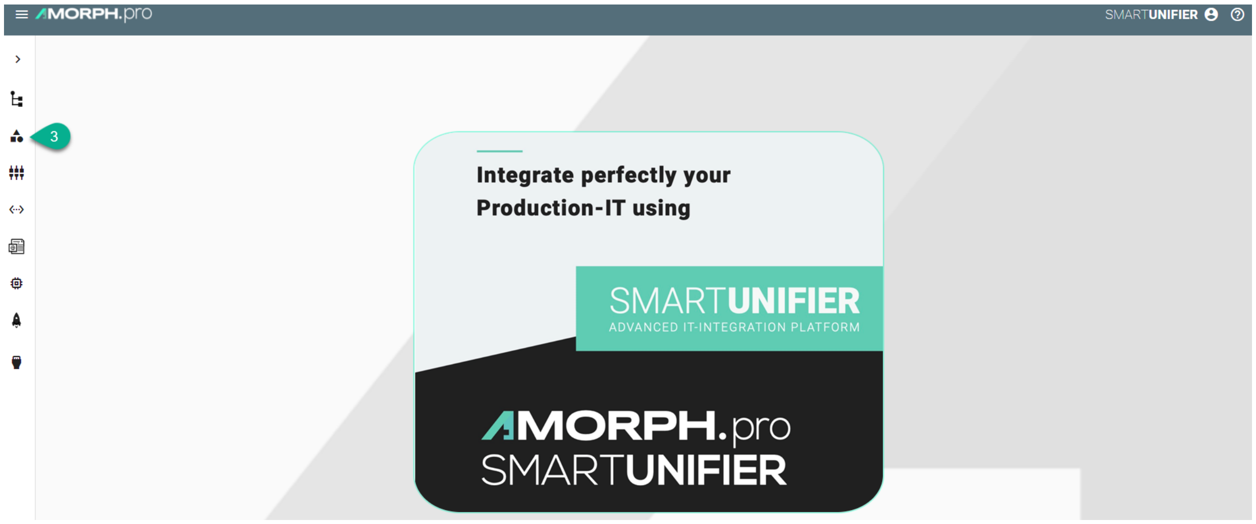
5.3.1 How to access

Follow the steps bellow to access the Channel Types Manager:

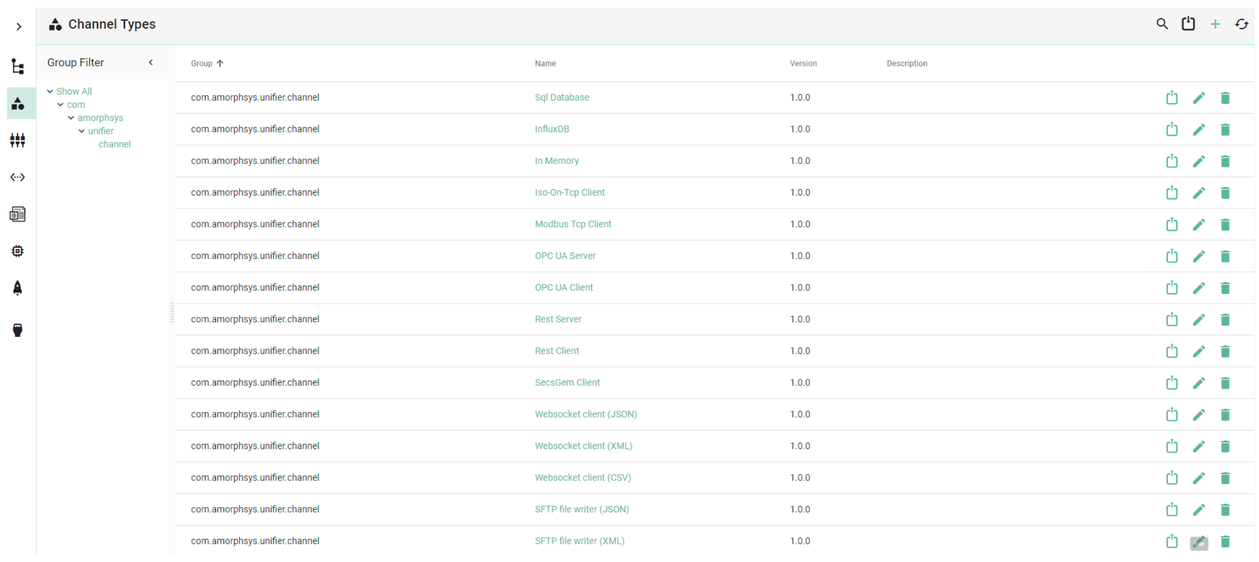
- Click on the **Account** icon **(1)** and select the **Advanced UI (2)**.



- Click on the **Channel Types** button **(3)** to open the Channel Types perspective.


MORPH.pro
SMARTUNIFIER
3
Integrate perfectly your Production-IT using
SMARTUNIFIER
ADVANCED IT-INTEGRATION PLATFORM
MORPH.pro
SMARTUNIFIER

- The main view of the Channel Types is visible.



Group	Name	Version	Description
com.amorphsys.unifier.channel	Sql Database	1.0.0	
com.amorphsys.unifier.channel	InfluxDB	1.0.0	
com.amorphsys.unifier.channel	In Memory	1.0.0	
com.amorphsys.unifier.channel	Iso-On-Tcp Client	1.0.0	
com.amorphsys.unifier.channel	Modbus Tcp Client	1.0.0	
com.amorphsys.unifier.channel	OPC UA Server	1.0.0	
com.amorphsys.unifier.channel	OPC UA Client	1.0.0	
com.amorphsys.unifier.channel	Rest Server	1.0.0	
com.amorphsys.unifier.channel	Rest Client	1.0.0	
com.amorphsys.unifier.channel	SecsGem Client	1.0.0	
com.amorphsys.unifier.channel	Websocket client (JSON)	1.0.0	
com.amorphsys.unifier.channel	Websocket client (XML)	1.0.0	
com.amorphsys.unifier.channel	Websocket client (CSV)	1.0.0	
com.amorphsys.unifier.channel	SFTP file writer (JSON)	1.0.0	
com.amorphsys.unifier.channel	SFTP file writer (XML)	1.0.0	

Note: The Channel Types Manager can only be accessed by user accounts with an administrator role assigned.

5.3.2 About Layers

Implementations of SMART**UNIFIER** Communication Channels consist of one and up to three so-called layers.

The target of layers is to transform data from Information Models into the respective data format of the specific protocol used in case the data traffic is outgoing from a SMART**UNIFIER** Communication Instance. The same principle applies when data is incoming.

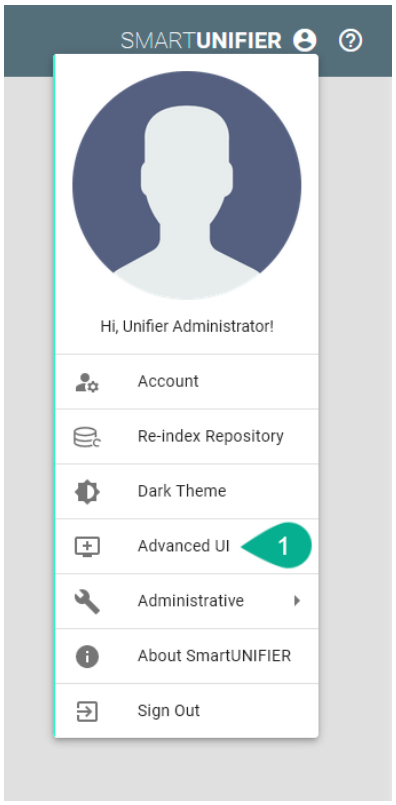
As an example for such a layer stack you can see below the layer stack for the *MQTT Communication Channel*:

- Data that is incoming from a Device is transformed into a String behind the scene.
- The String is then converted into a JSON Object.
- Finally, the JSON is used to assign data to the assigned Information Model.

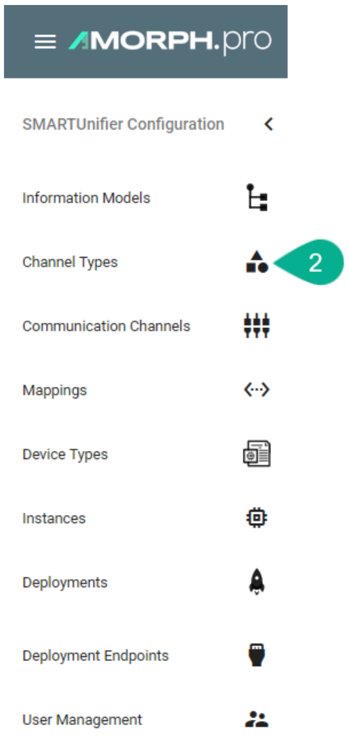
5.3.3 How to create a new Channel Type

Follow the steps below to create a new Channel Type:

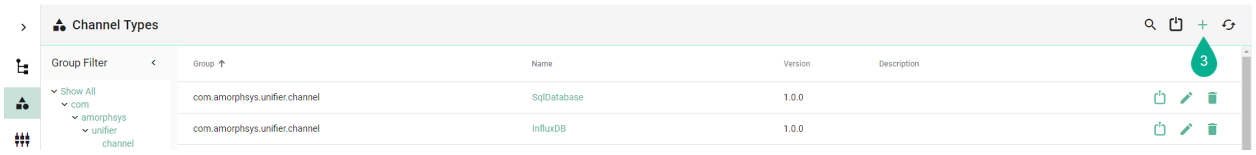
1. Open the SMART**UNIFIER** menu and select **Advanced UI**.



2. Go to the Channel Types perspective by clicking the **Channel Types** button.

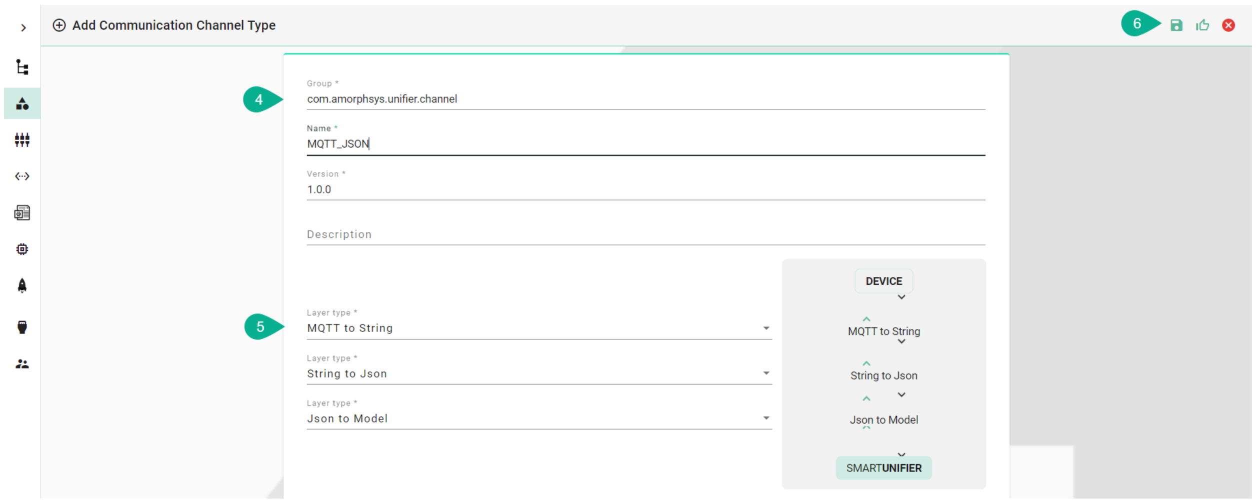


3. Click on the **Add** button in the upper right corner.



Group	Name	Version	Description
com.amorphsys.unifier.channel	SqlDatabase	1.0.0	
com.amorphsys.unifier.channel	InfluxDB	1.0.0	

4. Enter some descriptive information:
   - Enter a **group**
   - Enter the **name** of the Channel
   - Enter a **version**
5. Next, define the layer stack of the new Channel Type:
   - Select a layer with the **Layer type** drop-down menu.
   - In case the selected layer has more layers dependent on itself, select again another layer with the **Layer type** drop-down menu showing up below.
6. To save the Communication Channel Type select the **Save** button.



5.4 Docker Java Image Manager

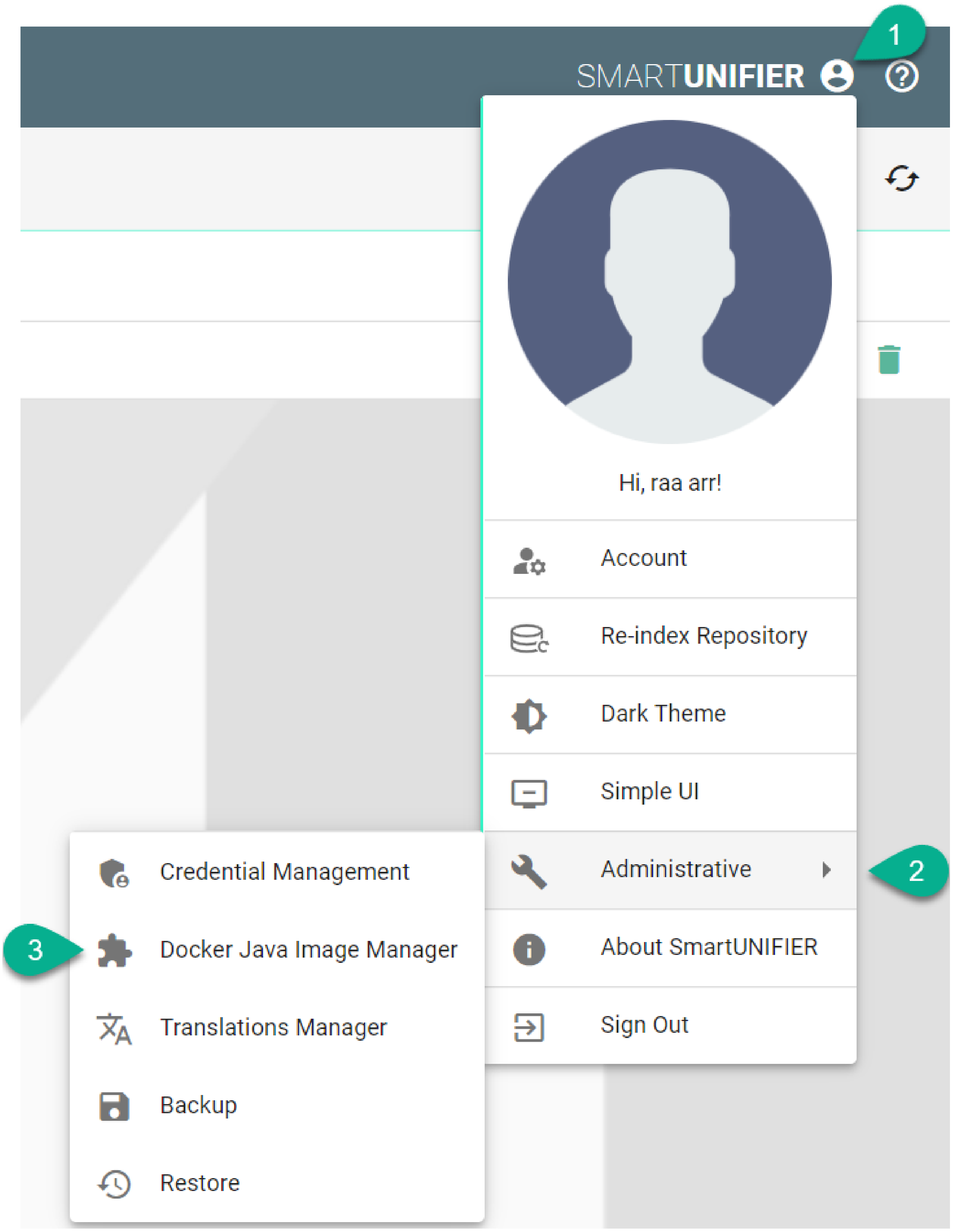
SMART**UNIFIER** supports the Deployment of Instances using Docker Containers using different Java base images. With the Docker Java Images Manager, the user can create and maintain different versions of Docker Java images.

This feature can only be accessed by a user with the administrator role.

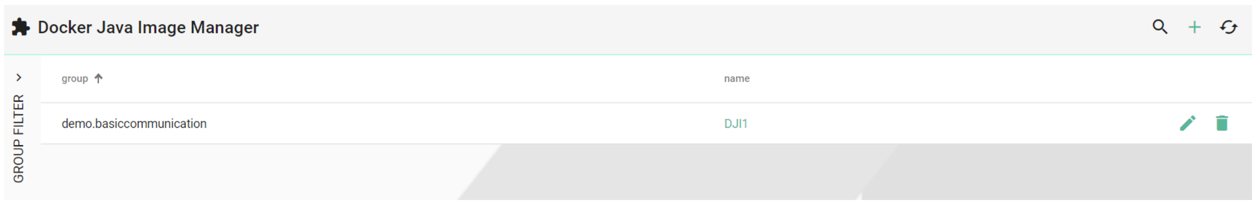
5.4.1 How to access

Follow the steps bellow to access the Docker Java Image Manager:

- Click on the **Account** icon **(1)**, go to **Administrative** section **(2)** and select the **Docker Java Image Manager** option **(3)**.



- The Docker Java Image Manager is visible.



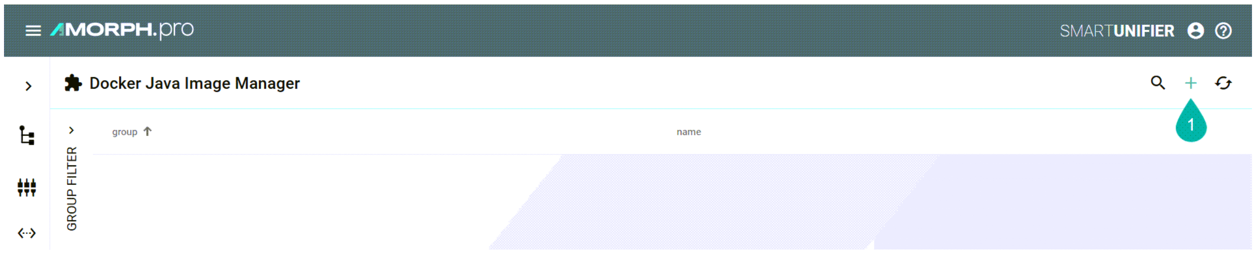
group	name
demo.basiccommunication	DJI1

Note: The Docker Java Image Manager can only be accessed by user accounts with an administrator role assigned.

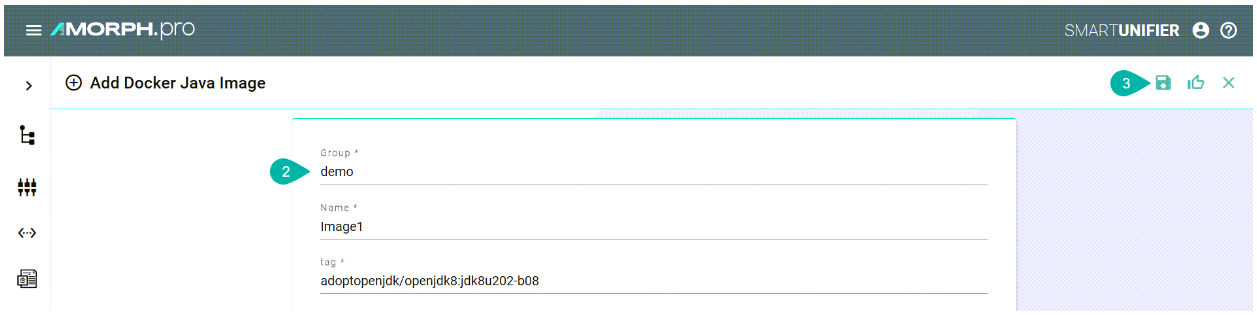
5.4.2 Add a New Docker Java Image

Follow the steps described below to add a new Docker Java image:

- Click on the **Add** button **(1)**.



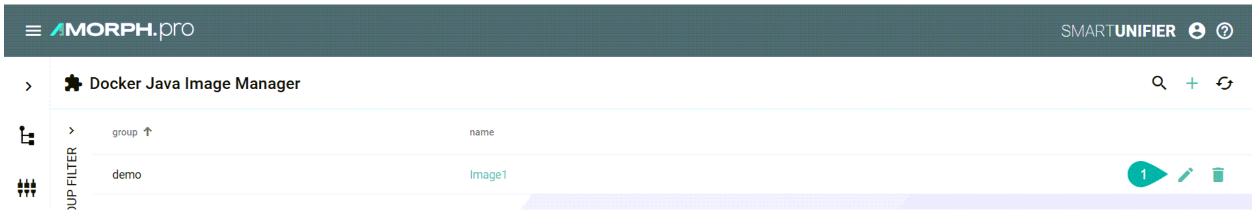
- In the *Add Docker Java Image* view, a set of configuration parameters is required **(2)**: * Provide a **Group** and a **Name** * Provide a **tag** e.g., `adoptopenjdk/openjdk8:jdk8u202-b08`
- After all mandatory fields are filled in, click the **Save** button **(3)**.



Group *
demo
Name *
Image1
tag *
adoptopenjdk/openjdk8:jdk8u202-b08

5.4.3 Edit a Docker Java Image

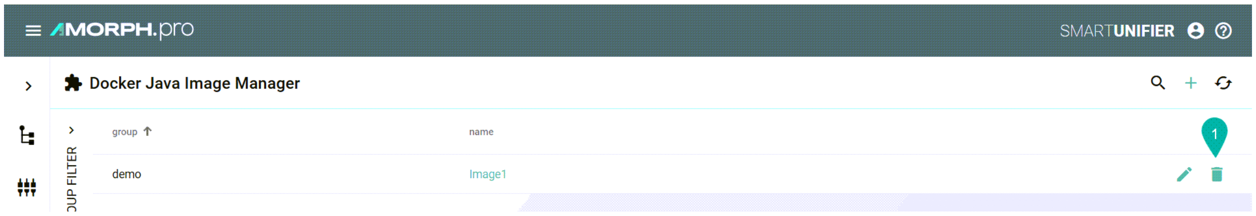
To edit a Docker Java image, select the **Edit** button **(1)**.



The Docker Java image is in the Edit Mode, the configuration parameters can be edited and then save the session by selecting the **Save** button.

5.4.4 Delete a Docker Java Image

To delete a Docker Java image, select the **Delete** button **(1)**.



A pop-up confirmation appears, select the **Delete** button.

5.5 Deployment Endpoints

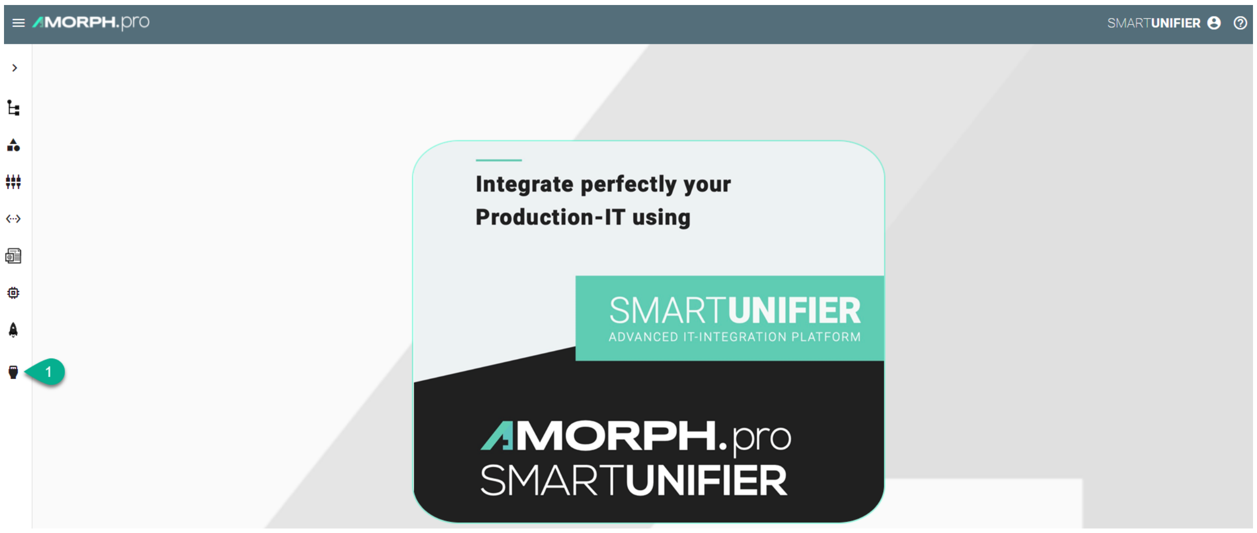
5.5.1 What are Deployment Endpoints

Deployment Endpoints are used to identify the location of a Deployment (i.e., the definition where an Instance is executed). With the Deployment Endpoints, you can create and maintain those locations. This feature can only be accessed by a user with the administrator role.

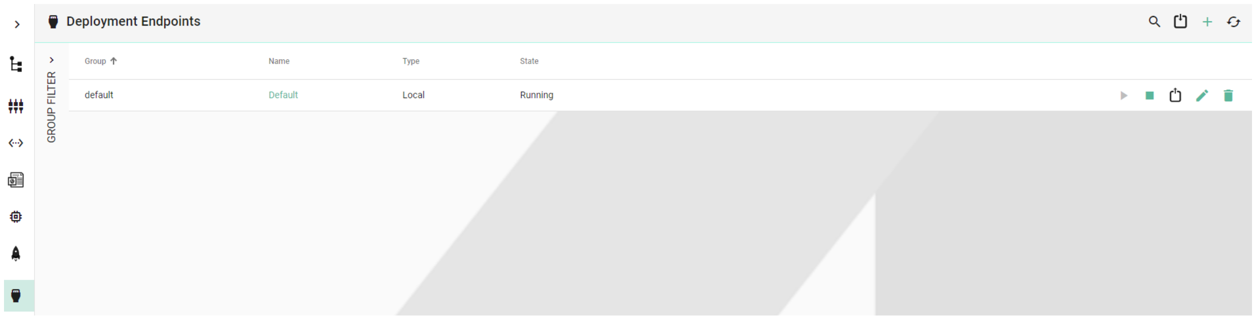
5.5.2 How to access

Follow the steps bellow to access the Deployment Endpoints:

- Click on the **Deployment Endpoints** button **(1)** to open the Deployment Endpoints perspective.


MORPH.pro
SMARTUNIFIER
1
Integrate perfectly your Production-IT using
SMARTUNIFIER
ADVANCED IT-INTEGRATION PLATFORM
AMORPH.pro SMARTUNIFIER

- The main view of the Deployment Endpoints is visible.



Group ↑	Name	Type	State
default	Default	Local	Running

Note: The Deployment Endpoints can only be accessed by user accounts with an administrator role assigned.

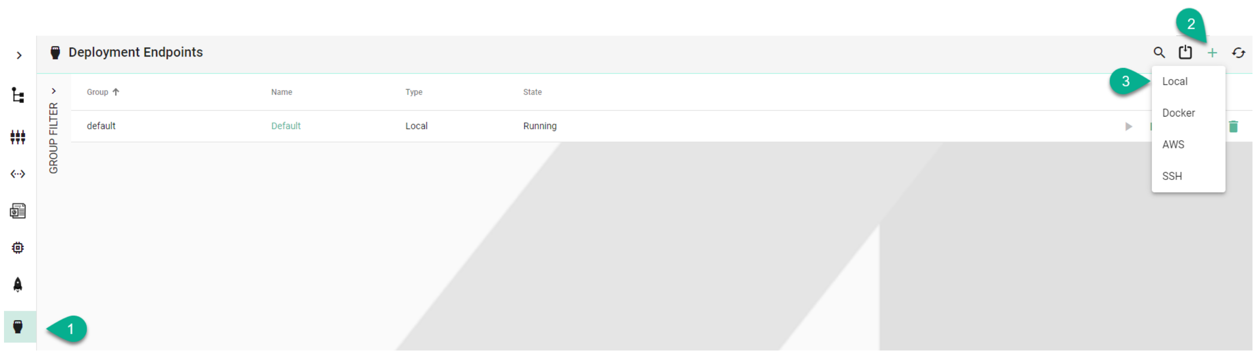
5.5.3 Deployment Endpoints Types

Local

SMART**UNIFIER** supports Endpoint for Local Deployment. A **Default Local Endpoint** is preconfigured.

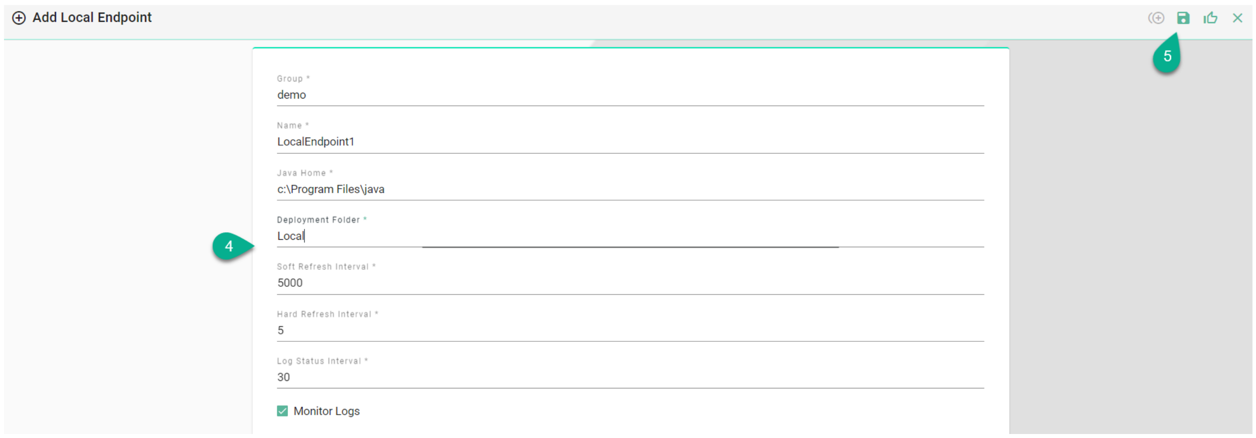
Follow the steps described below to create a Local Deployment Endpoint:

- Navigate to the SMART**UNIFIER** Deployment Endpoints perspective **(1)**.
- Click on the **Add Endpoint** button **(2)**.
- Select the Deployment Type **Local** from the pop-up **(3)**.



Group	Name	Type	State
default	Default	Local	Running

- In the **Add Endpoint** view a set of configuration parameters is required **(4)**
  - Provide a **Group** and a **Name**
  - Input the path for **Java**
  - Provide the **Deployment Folder**
  - Configure the **Soft/Hard Refresh Interval** and the **Log Status Interval** (in milliseconds)
  - Enable **Monitor Logs** (optionally)



Group *
demo

Name *
LocalEndpoint1

Java Home *
c:\Program Files\java

Deployment Folder *
Local

Soft Refresh Interval *
5000

Hard Refresh Interval *
5

Log Status Interval *
30

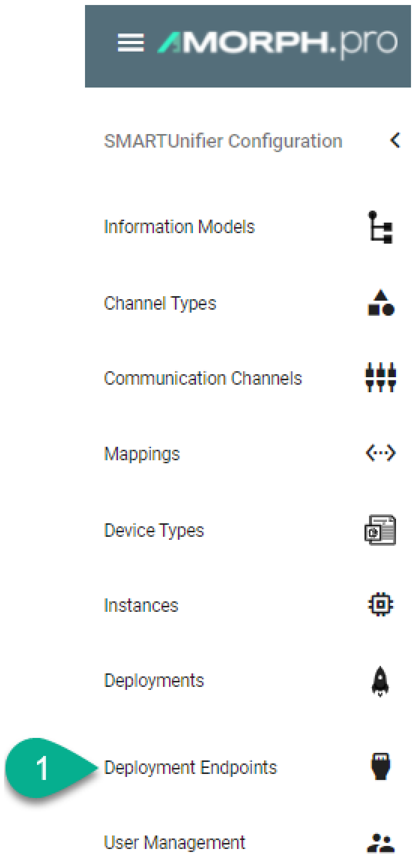
✓ Monitor Logs

- After all mandatory fields are filled in, click the **Save** button **(5)**.

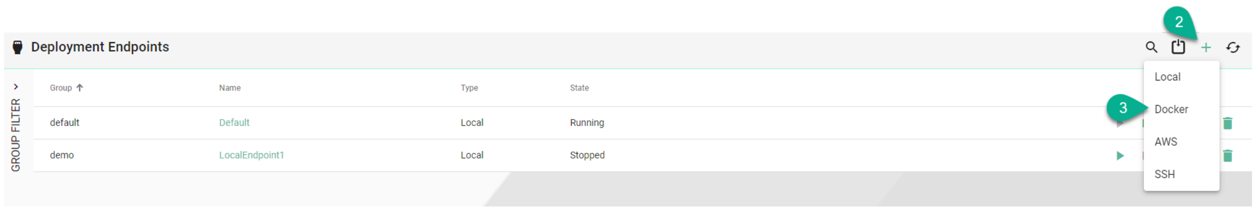
Docker

SMART**UNIFIER** supports the Deployment of Instances using Docker Containers. Before creating a new Deployment for an Instance using Docker, install Docker on your device and open up the Docker Remote API Interface. If you want to learn more about Docker and how to install it, visit the Docker Website. When your Docker Daemon is up and running you must provide a Docker endpoint.

- Navigate to the SMART**UNIFIER** Deployment Endpoints perspective **(1)**.

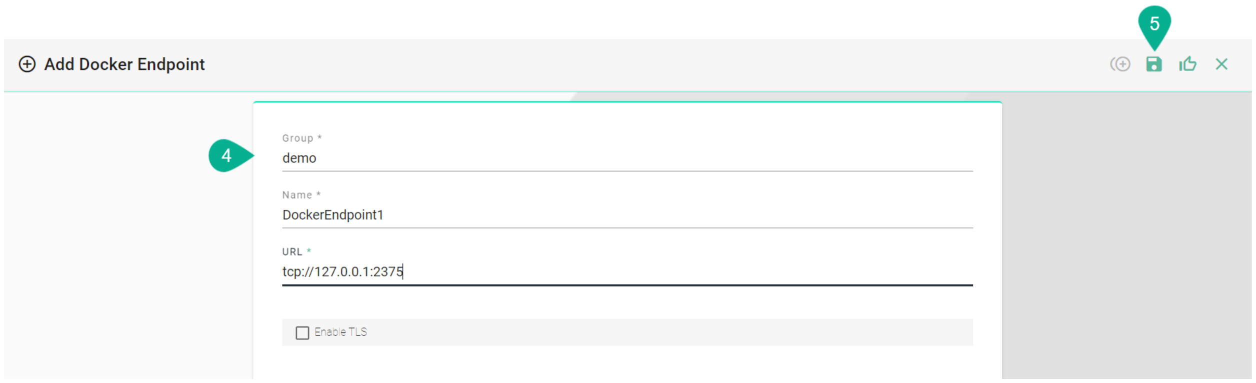


- Click on the **Add Endpoint** button **(2)**.
- Select the Deployment Type **Docker** from the pop-up **(3)**.



Group	Name	Type	State
default	Default	Local	Running
demo	LocalEndpoint1	Local	Stopped

- In the **Add Endpoint** view a set of configuration parameters is required **(4)**
  - Provide a **Group** and a **Name**
  - Provide **URL**. Depending on your use case choose between the **unix** e.g., `unix:///var/run/docker.sock` or the **tcp** e.g., `tcp://127.0.0.1:2375` **protocol**.
  - If needed, enable **TLS** by enabling the checkbox
- After all mandatory fields are filled in, click the **Save** button **(5)**.



Group *
demo

Name *
DockerEndpoint1

URL *
tcp://127.0.0.1:2375

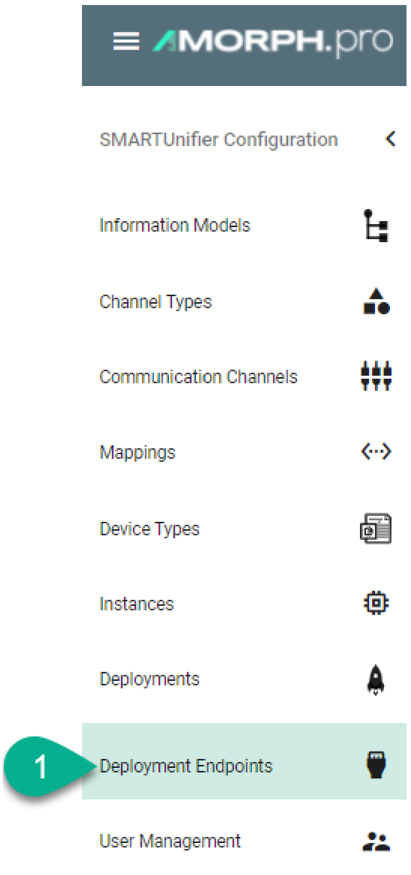
Enable TLS

AWS

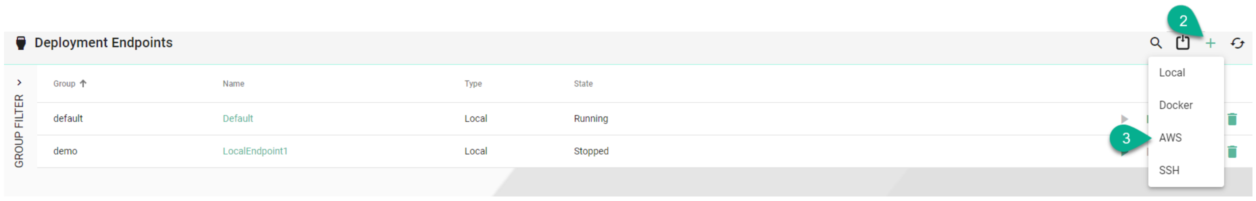
Before deploying a SMART**UNIFIER** Instance on AWS Fargate you need to create an AWS Deployment Endpoint. The AWS Deployment Endpoint specifies, which AWS account should be used for the deployment.

Follow the steps described below to create an AWS Deployment Endpoint:

- Select the SMART**UNIFIER** Deployment Endpoints perspective **(1)**.

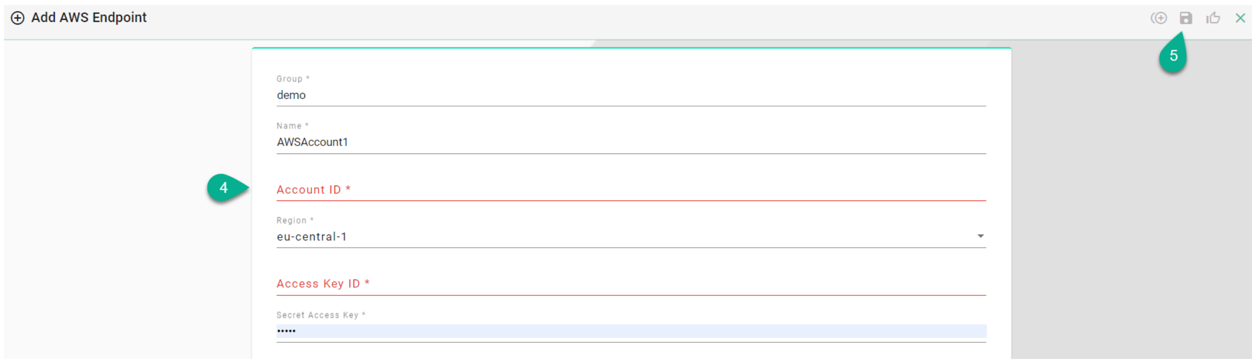


- Click the **Add** button **(2)**.
- Select the Deployment Type **AWS** from the pop-up **(3)**.



Group	Name	Type	State
default	Default	Local	Running
demo	LocalEndpoint1	Local	Stopped

- Configure your AWS account by entering the following parameters **(4)**:
  - Enter a **Group** and a **Name**.
  - Enter your **AWS Account ID**.
  - Select the Region.
  - Enter the **Access Key ID** and the **Secret Access Key** that allows SMART**UNIFIER** to connect to your AWS account.
- Save the new Endpoint by clicking the **Save** button **(5)**:

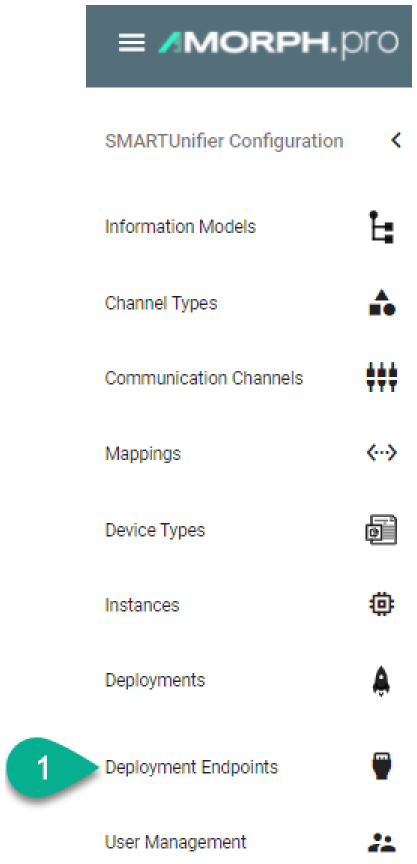


Group *
demo
Name *
AWSAccount1
Account ID *
Region *
eu-central-1
Access Key ID *
Secret Access Key *
•••••

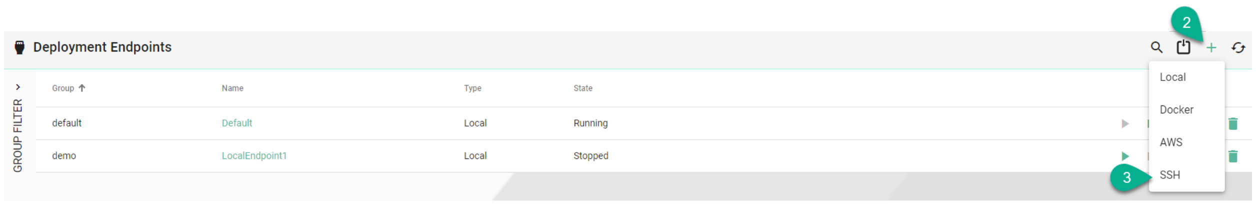
SSH

SMART**UNIFIER** supports the Deployment of Instances using SSH protocol.

- Navigate to the SMART**UNIFIER** Deployment Endpoints perspective **(1)**.

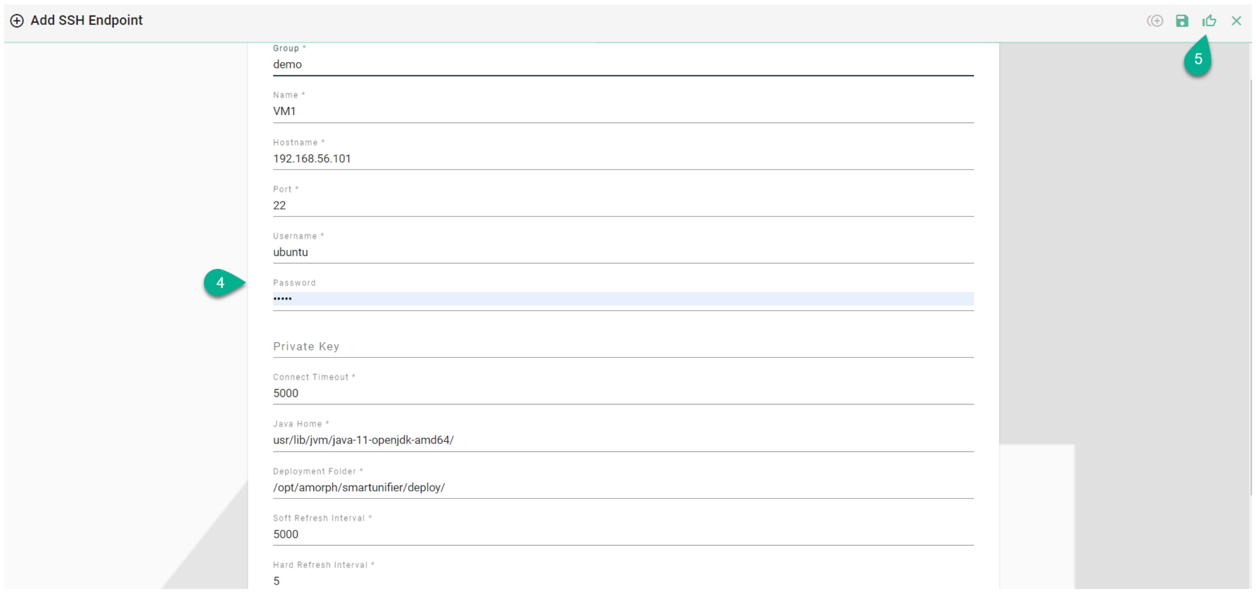


- Click on the **Add Endpoint** button **(2)**.
- Select the Deployment Type **SSH** from the pop-up **(3)**.



Group	Name	Type	State
default	Default	Local	Running
demo	LocalEndpoint1	Local	Stopped

- In the **Add Endpoint** view a set of configuration parameters is required **(4)**
  - Provide a **Group** and a **Name** .
  - Provide the VM **Hostname**. The default used port is **22**.
  - Provide the **Username** and the **Password** .
  - If needed, input **Private Key** for secured connections.
  - Provide the **Java Home** path.
  - Provide the **Deployment Folder** path.
- After all mandatory fields are filled in, click the **Save and Close** button **(5)**.



Group *
demo

Name *
VM1

Hostname *
192.168.56.101

Port *
22

Username *
ubuntu

Password
•••••

Private Key

Connect Timeout *
5000

Java Home *
usr/lib/jvm/java-11-openjdk-amd64/

Deployment Folder *
/opt/amorph/smartunifier/deploy/

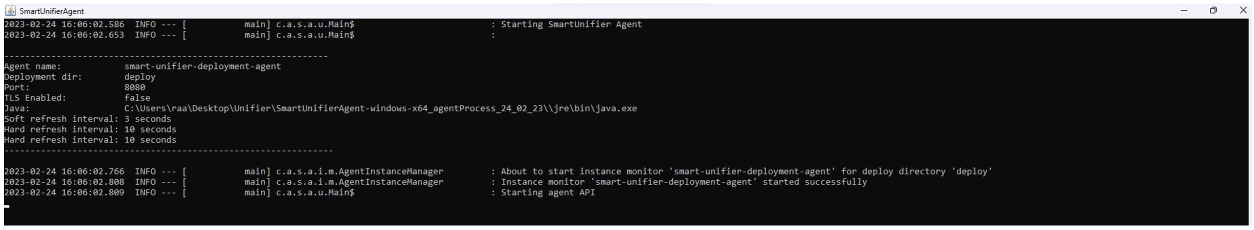
Soft Refresh Interval *
5000

Hard Refresh Interval *
5

Agent

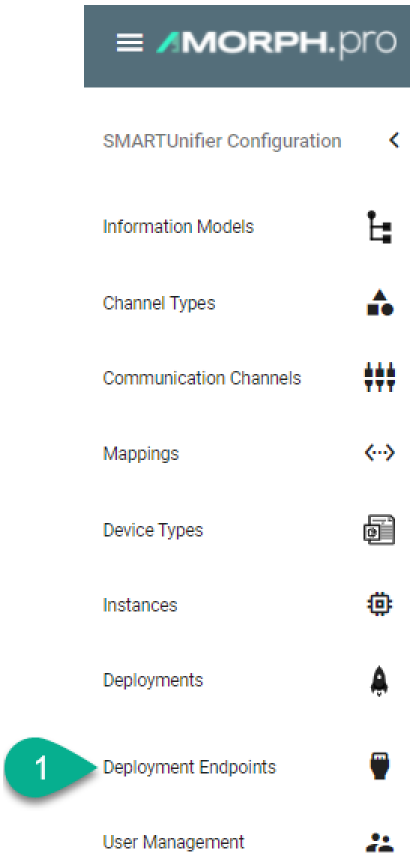
SMART**UNIFIER** allows Communication Instances to be deployed on any machine with an active Agent process. The Agent enables communication between the SMART**UNIFIER** Manager and the Communication Instance.

Example of an Agent running on a remote machine:

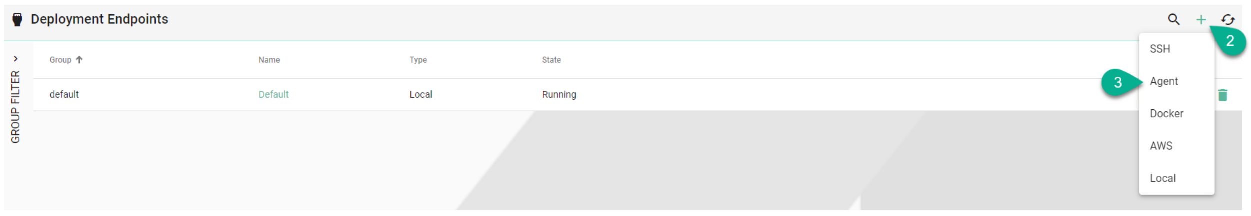


```
2023-02-24 16:06:02.586  INFO --- [           main] c.a.s.a.u.Main$                          : Starting SmartUnifier Agent
2023-02-24 16:06:02.653  INFO --- [           main] c.a.s.a.u.Main$                          :

--------------------------------------------------------------
Agent name:             smart-unifier-deployment-agent
Deployment dir:         deploy
Port:                   8080
TLS Enabled:            false
Java:                   C:\Users\raa\Desktop\Unifier\SmartUnifierAgent-windows-x64_agentProcess_24_02_23\\jre\bin\java.exe
Soft refresh interval: 3 seconds
Hard refresh interval: 10 seconds
Hard refresh interval: 10 seconds
---------------------------------------------------------------

2023-02-24 16:06:02.766  INFO --- [           main] c.a.s.a.i.m.AgentInstanceManager         : About to start instance monitor 'smart-unifier-deployment-agent' for deploy directory 'deploy'
2023-02-24 16:06:02.808  INFO --- [           main] c.a.s.a.i.m.AgentInstanceManager         : Instance monitor 'smart-unifier-deployment-agent' started successfully
2023-02-24 16:06:02.809  INFO --- [           main] c.a.s.a.u.Main$                          : Starting agent API
```

- Navigate to the SMART**UNIFIER** Deployment Endpoints perspective **(1)**.

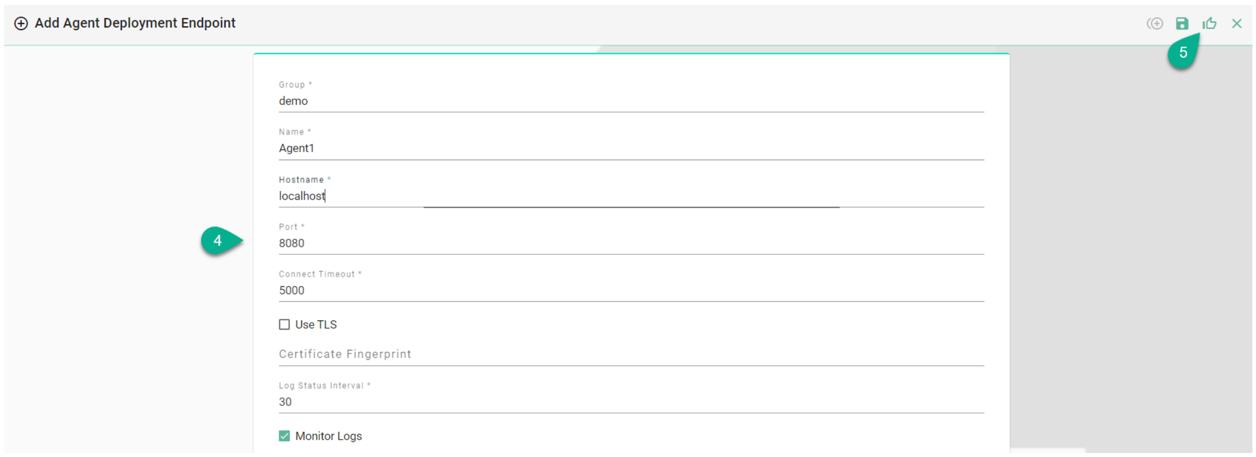


- Click on the **Add Endpoint** button **(2)**.
- Select the Deployment Type **Agent** from the pop-up **(3)**.



Group	Name	Type	State
default	Default	Local	Running

- In the **Add Endpoint** view a set of configuration parameters is required **(4)**
  - Provide a **Group** and a **Name**
  - Provide the VM **Hostname** (Default port is: **8080**)
  - Set the **Connection Timeout**
  - If needed, check the **useTls** box and input certificates for secured connections
  - Set the **Log Status Interval**
  - Enable the **Monitor Logs**
- After all mandatory fields are filled in, click the **Save and Close** button **(5)**.

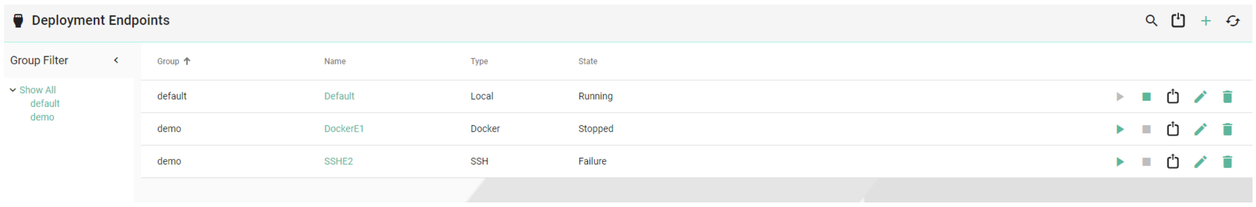


Group *	demo
Name *	Agent1
Hostname *	localhost
Port *	8080
Connect Timeout *	5000
☐ Use TLS	
Certificate Fingerprint	
Log Status Interval *	30
☑ Monitor Logs	

5.5.4 Deployment Endpoints States

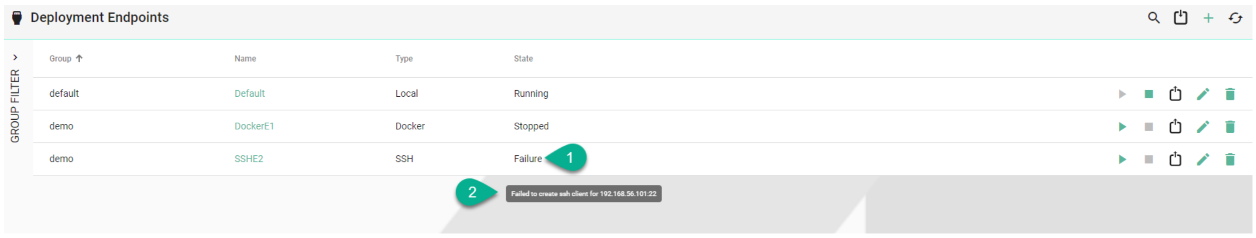
A Deployment Endpoint can have the following states:

- **Stopped** - The Stop command has been sent and the Deployment Endpoint is stopped
- **Starting** - The Start command has been sent
- **Running** - Deployment Endpoint is up and running
- **Failure** - The Start command has been sent and the Deployment Endpoint has failed to start



Group	Name	Type	State
default	Default	Local	Running
demo	DockerE1	Docker	Stopped
demo	SSHE2	SSH	Failure

For the **Failure** state, hover over it **(1)** and a pop-up will display the error **(2)**.



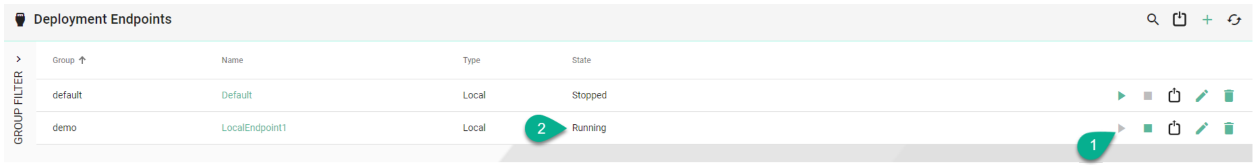
Group	Name	Type	State
default	Default	Local	Running
demo	DockerE1	Docker	Stopped
demo	SSHE2	SSH	Failure

Failed to create ssh client for 192.168.56.101:22

5.5.5 Deployment Endpoints Operations

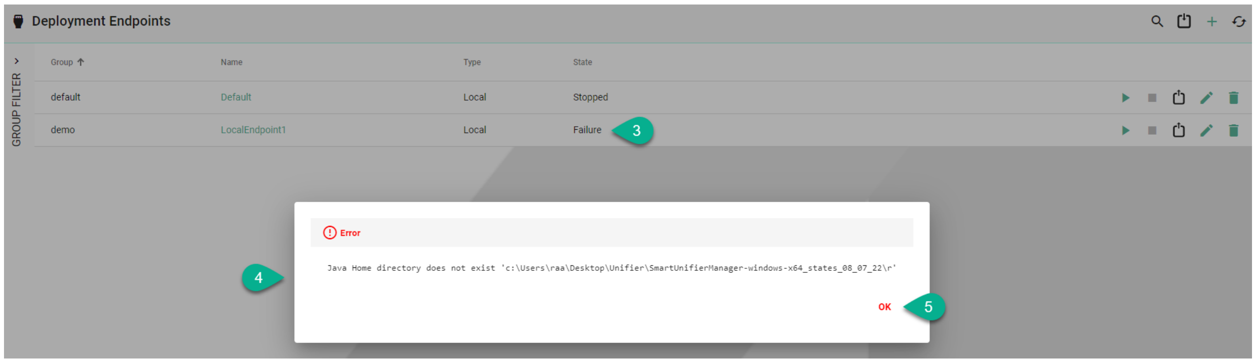
Start Endpoint

After a Deployment Endpoint is created, its default state is Stopped. To start it, click on the **Start** button **(1)**. The state will change into **Starting** and if it succeeds, the state becomes **Running (2)**.



Group	Name	Type	State
default	Default	Local	Stopped
demo	LocalEndpoint1	Local	Running

If the Deployment Endpoint fails to start, the state changes into **Failure (3)** and an error message will be displayed **(4)**.



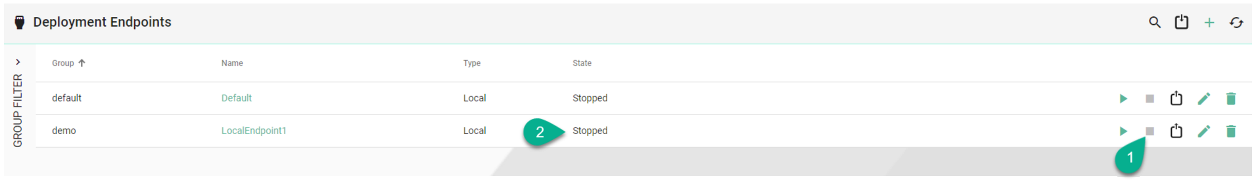
Group	Name	Type	State
default	Default	Local	Stopped
demo	LocalEndpoint1	Local	Failure

```
Java Home directory does not exist 'c:\Users\raa\Desktop\Unifier\SmartUnifierManager-windows-x64_states_08_07_22\r'
```

Click on the **OK** button **(5)** to close the error message.

Stop Endpoint

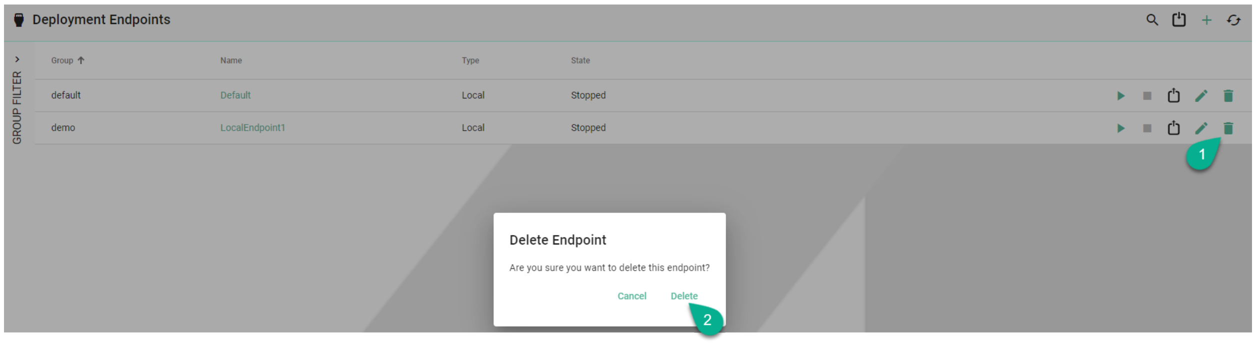
To stop a Deployment Endpoint, click on the **Stop** button **(1)** and the state will change accordingly **(2)**.



Group	Name	Type	State
default	Default	Local	Stopped
demo	LocalEndpoint1	Local	Stopped

Delete Endpoint

To remove a Deployment Endpoint, click on the **Delete** button **(1)** and confirm the action **(2)**.

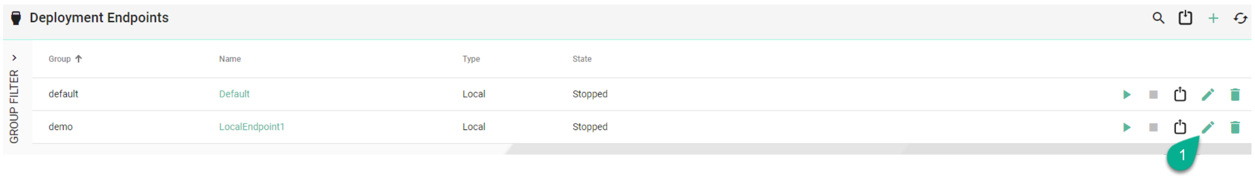


Group	Name	Type	State
default	Default	Local	Stopped
demo	LocalEndpoint1	Local	Stopped

Are you sure you want to delete this endpoint?

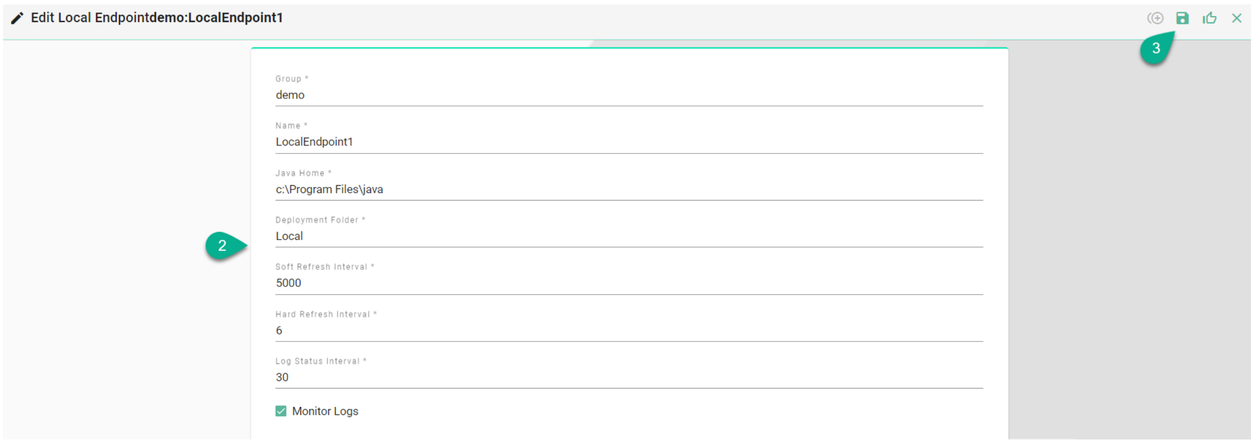
Edit Endpoint

To edit a Deployment Endpoint, click on the **Edit** button **(1)**.



Group	Name	Type	State
default	Default	Local	Stopped
demo	LocalEndpoint1	Local	Stopped

In the Deployment Endpoint edit view update the configuration **(2)** and click on the **Save** button **(3)**.



Group *
demo

Name *
LocalEndpoint1

Java Home *
c:\Program Files\java

Deployment Folder *
Local

Soft Refresh Interval *
5000

Hard Refresh Interval *
6

Log Status Interval *
30

✓ Monitor Logs

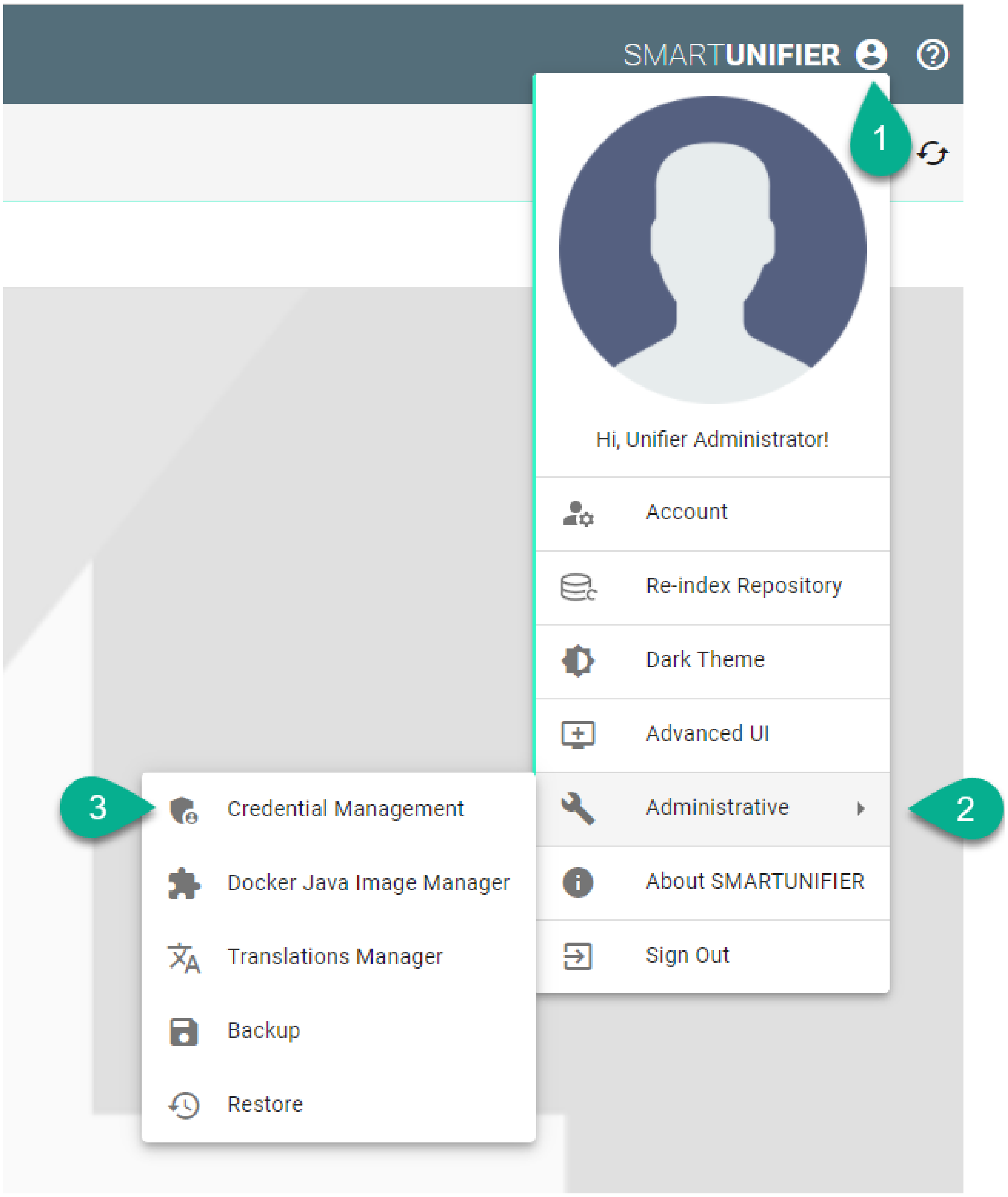
5.6 Credential Management

Within the Credential Manager the user can store and manage the credentials needed for the Communication Channel configuration (e.g., password for certificates, username and password for SQL Server).

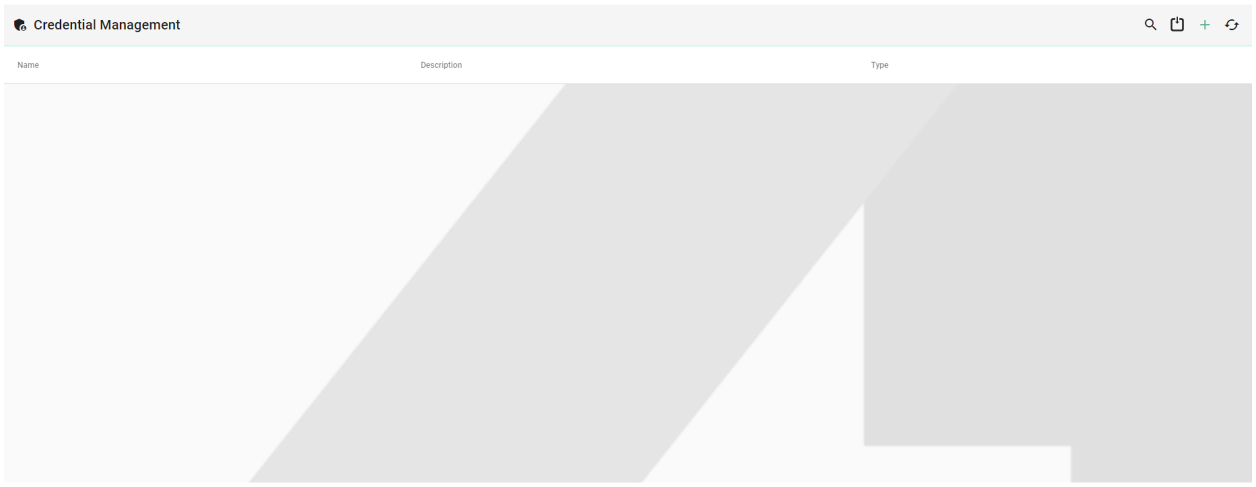
5.6.1 How to access

Follow the steps bellow to access the Credential Management:

- Click on the **Account** icon **(1)**, go to **Administrative** section **(2)** and select the **Credential Management** option **(3)**.



- The Credential Management is visible.

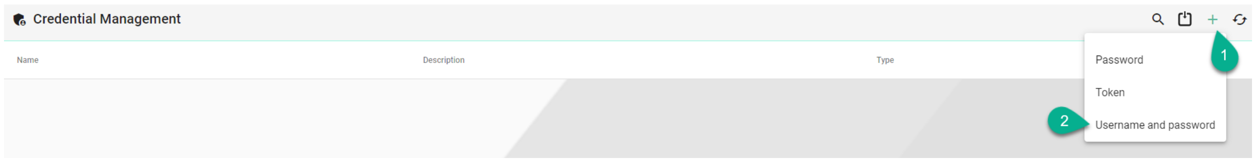


Note: The Credential Management can only be accessed by user accounts with an administrator role assigned.

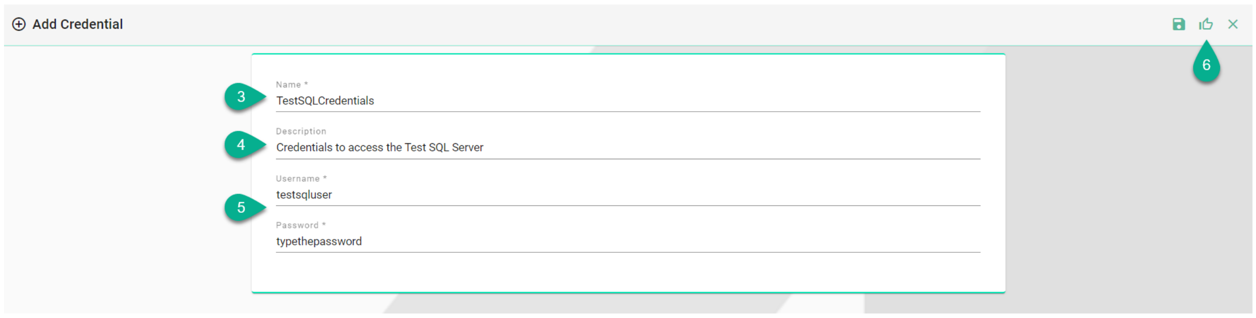
5.6.2 Add Credentials

Follow the steps described below to add credentials:

- Click on the **Add** button **(1)**.
- Select an option **(2)** Password or Username and Password.

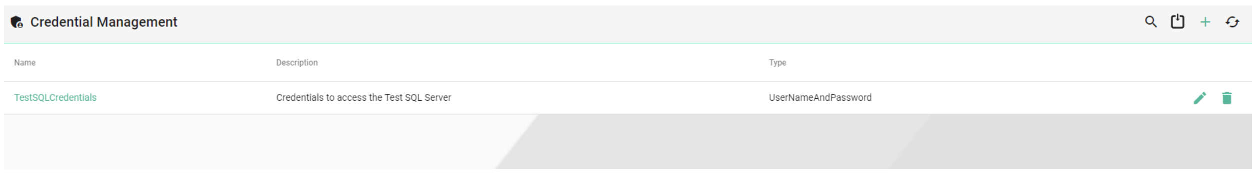


- Type a name for Credentials **(3)**.
- Add description **(4)** (optional).
- Input the Username and Password **(5)**.



Name *
TestSQLCredentials
Description
Credentials to access the Test SQL Server
Username *
testsqluser
Password *
typethepassword

- Click on the **Save and Close** button **(6)**.

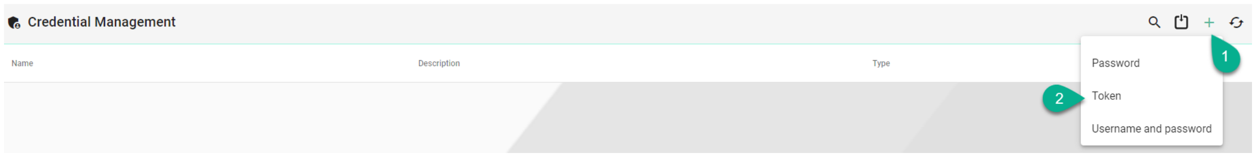

Credential Management

Name	Description	Type
TestSQLCredentials	Credentials to access the Test SQL Server	UserNameAndPassword

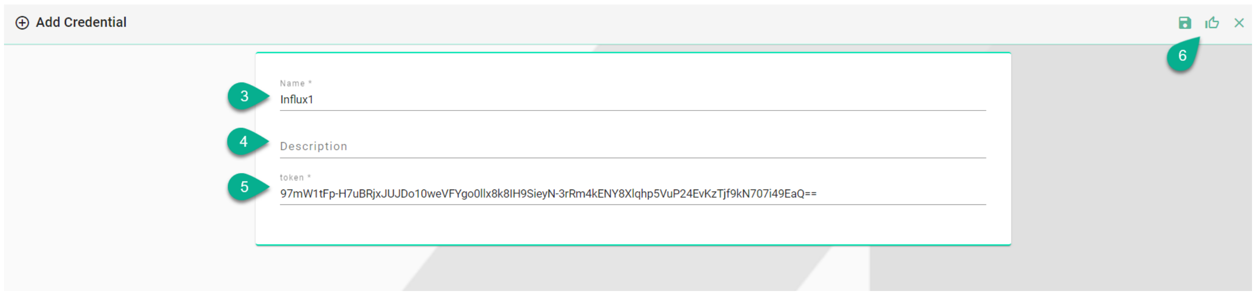
5.6.3 Add Token

Follow the steps described below to add a token:

- Click on the **Add** button **(1)**.
- Select the **Token** option **(2)**.



- Type a name for the Token **(3)**.
- Add description **(4)** (optional).
- Input the Token **(5)**.

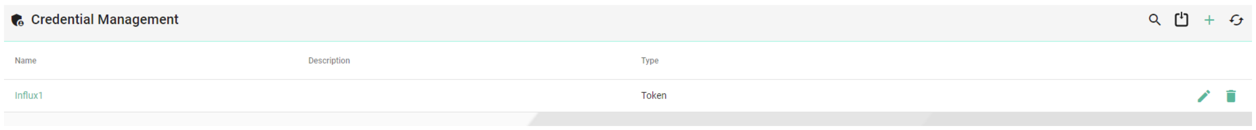

Add Credential

Name *
Influx1

Description

token *
97mW1tFp-H7uBRjxJUJDo10weVFYgo0llx8k8IH9SieyN-3rRm4kENY8Xlqhp5VuP24EvKzTjf9kN707i49EaQ==

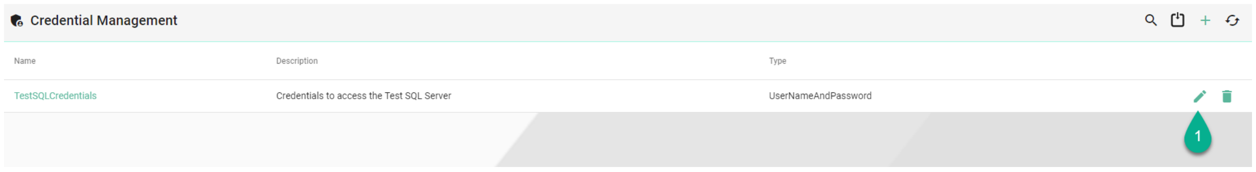
- Click on the **Save and Close** button **(6)**.


Credential Management

Name	Description	Type
Influx1		Token

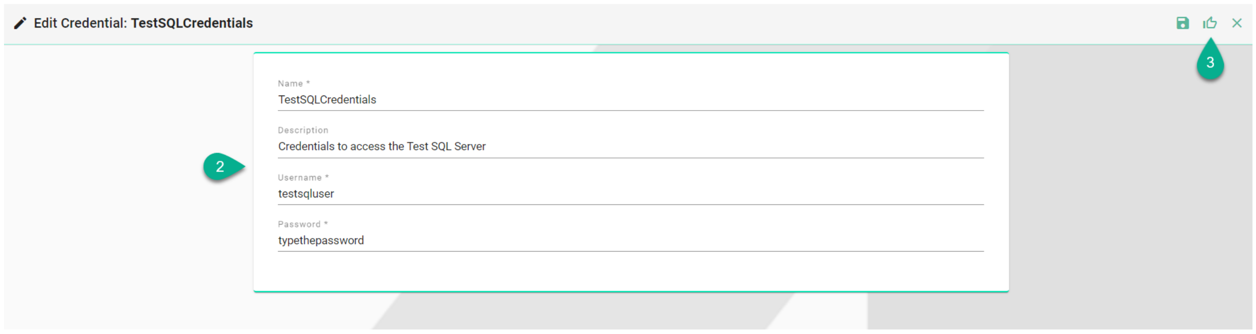
5.6.4 Edit Credentials

To edit the credentials, select the **Edit** button **(1)**.



Name	Description	Type
TestSQLCredentials	Credentials to access the Test SQL Server	UserNameAndPassword

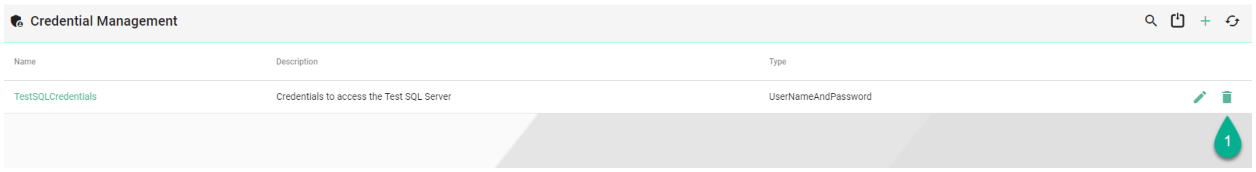
The Edit Mode is visible, the configuration can be edited **(2)** and then save the session by selecting the **Save and Close** button **(3)**.



Name *
TestSQLCredentials
Description
Credentials to access the Test SQL Server
Username *
testsqluser
Password *
typethepassword

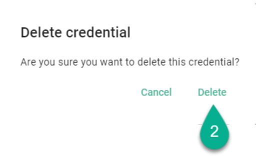
5.6.5 Delete Credentials

To delete credentials, select the **Delete** button **(1)**.



Name	Description	Type
TestSQLCredentials	Credentials to access the Test SQL Server	UserNameAndPassword

A pop-up confirmation appears, select the **Delete** button **(2)**.



Delete credential

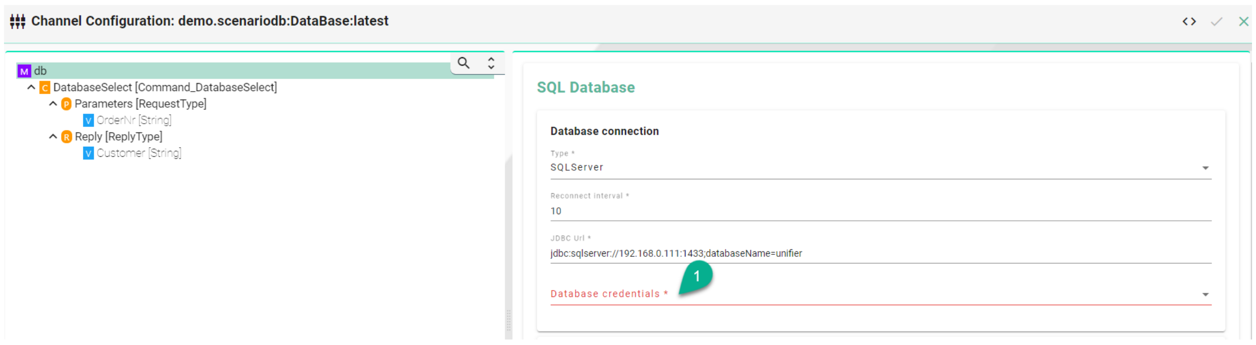
Are you sure you want to delete this credential?

5.6.6 Using Credential Manager when configuring the Communication Channels

When configuring the Communication Channels, the user has the option to manually input the credentials or to select one from the Credential Manager.

Example of SQL Database Communication Channel configuration:

- Click on the **Database credentials** field **(1)**.



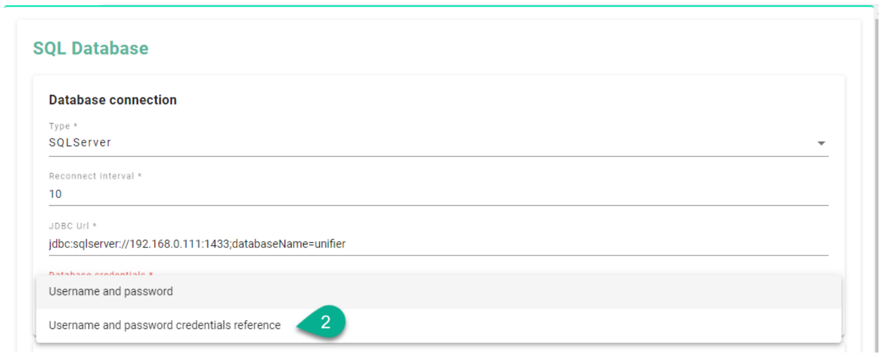
Type *
SQLServer

Reconnect interval *
10

JDBC Url *
jdbc:sqlserver://192.168.0.111:1433;databaseName=unifier

Database credentials *

- Select the **Username and password credentials reference** option **(2)**.

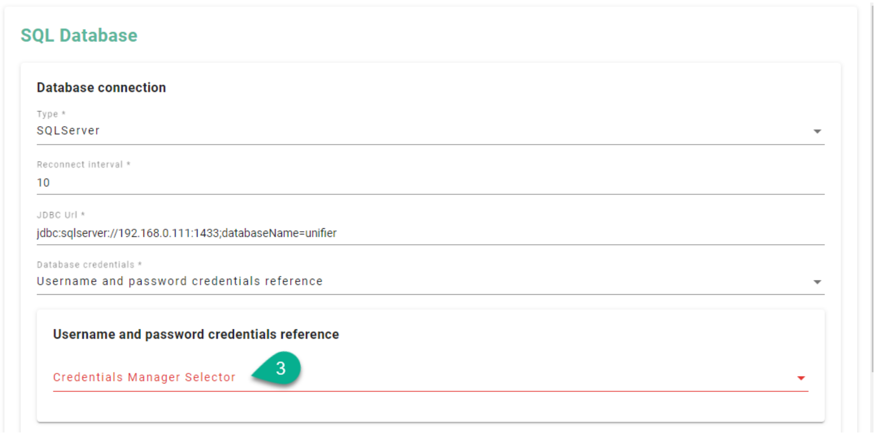


Type *
SQLServer

Reconnect interval *
10

JDBC Url *
jdbc:sqlserver://192.168.0.111:1433;databaseName=unifier

- Click on the **Credentials Manager Selector** option **(3)**.


SQL Database
Database connection
Username and password credentials reference
Credentials Manager Selector
3

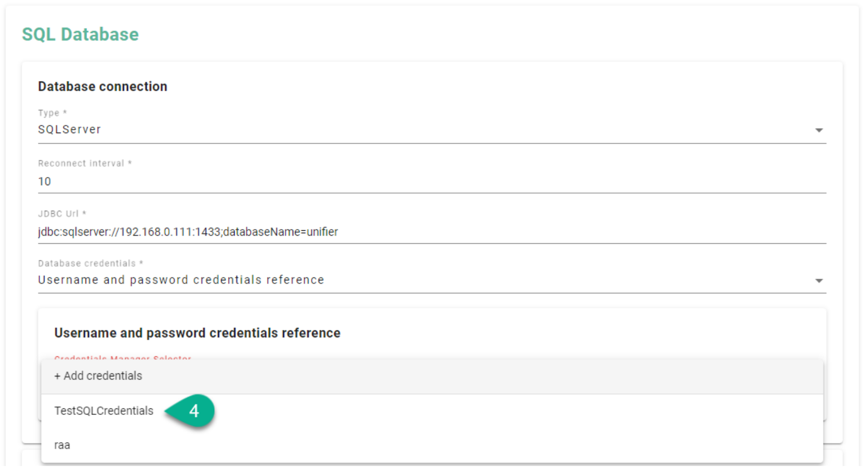
Type *
SQLServer

Reconnect interval *
10

JDBC Url *
jdbc:sqlserver://192.168.0.111:1433;databaseName=unifier

Database credentials *
Username and password credentials reference

- Select one of the credentials from the list **(4)**.


SQL Database

Database connection

Type *
SQLServer

Reconnect interval *
10

JDBC Url *
jdbc:sqlserver://192.168.0.111:1433;databaseName=unifier

Database credentials *
Username and password credentials reference

Username and password credentials reference

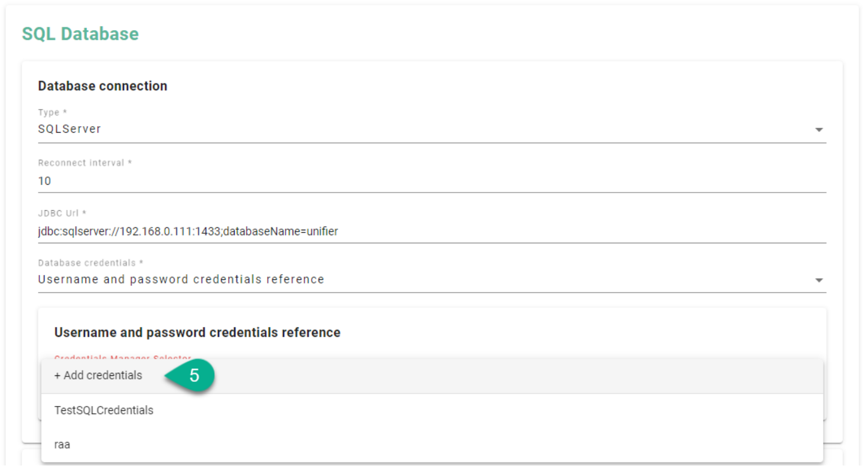
Credentials Manager Selector

+ Add credentials

TestSQLCredentials

raa

- If the credentials are not saved in the Credentials Manager, click on the **Add credentials** option **(5)**.


SQL Database

Database connection

Type *
SQLServer

Reconnect interval *
10

JDBC Url *
jdbc:sqlserver://192.168.0.111:1433;databaseName=unifier

Database credentials *
Username and password credentials reference

Username and password credentials reference

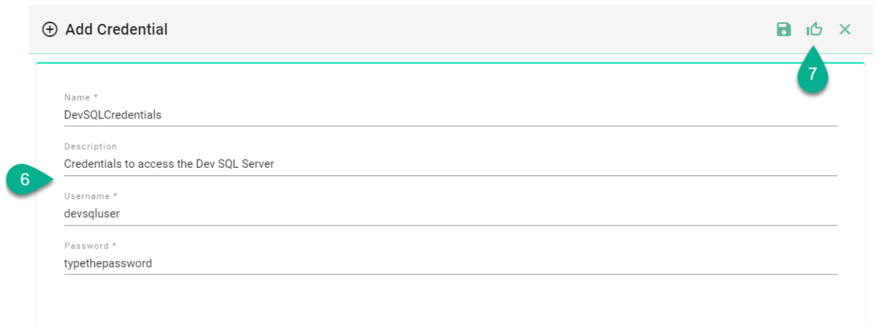
Credentials Manager Selector

+ Add credentials

TestSQLCredentials

raa

- Input the credentials details **(6)** and click on the **Save and Close** button **(7)**.


Add Credential

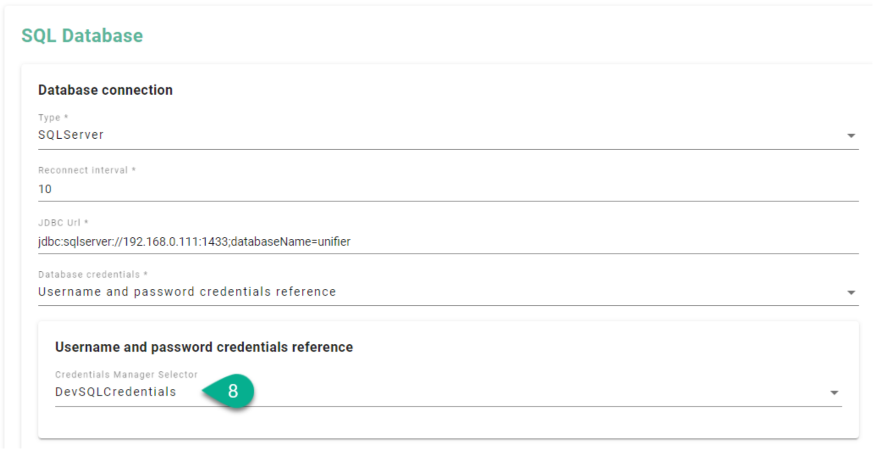
Name *
DevSQLCredentials

Description
Credentials to access the Dev SQL Server

Username *
devsqluser

Password *
typethepassword

- The new credentials are saved and added into the configuration **(8)**.


SQL Database

Database connection

Type *
SQLServer

Reconnect interval *
10

JDBC Url *
jdbc:sqlserver://192.168.0.111:1433;databaseName=unifier

Database credentials *
Username and password credentials reference

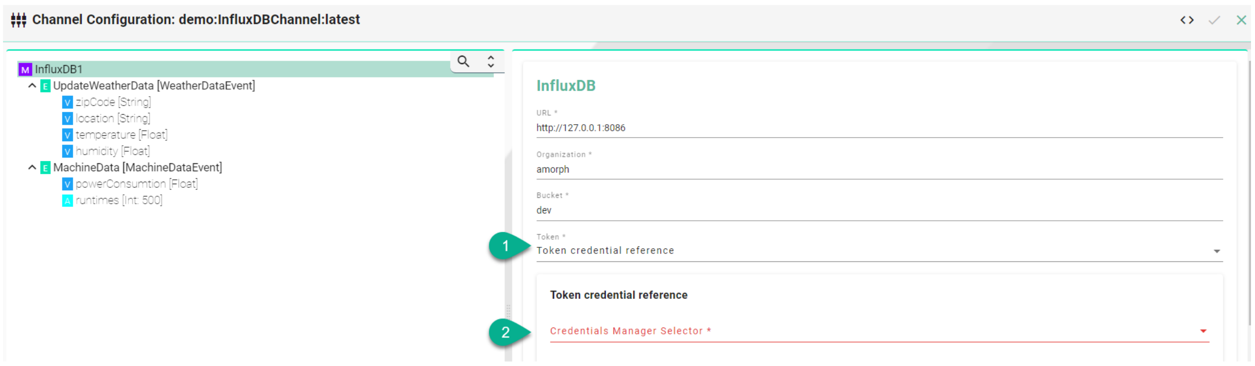
Username and password credentials reference

Credentials Manager Selector
DevSQLCredentials

When configuring the Communication Channels, the user has the option to manually input a token or to select one from the Credential Manager.

Example of InfluxDB V2 Communication Channel configuration:

- For the **Token** field, select the **Token credentials reference** option **(1)**.


Channel Configuration: demo:InfluxDBChannel:latest

InfluxDB1
UpdateWeatherData [WeatherDataEvent]
zipCode [String]
location [String]
temperature [Float]
humidity [Float]
MachineData [MachineDataEvent]
powerConsumtion [Float]
runtimes [Int: 500]

InfluxDB

URL *
http://127.0.0.1:8086

Organization *
amorph

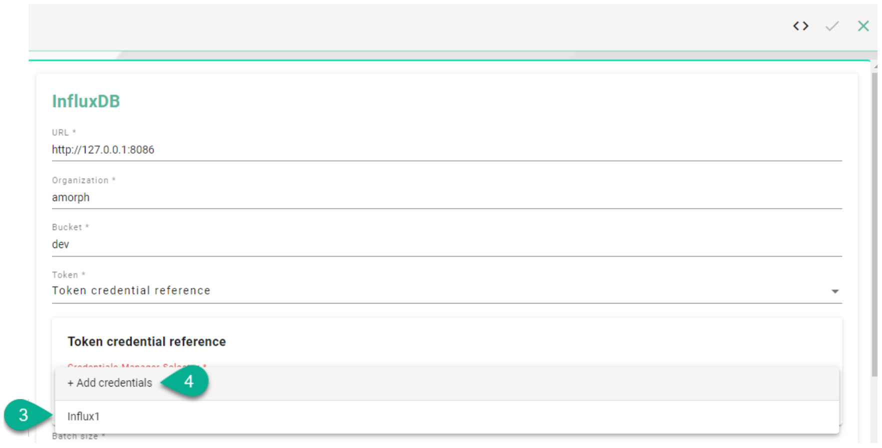
Bucket *
dev

Token *
Token credential reference

Token credential reference

Credentials Manager Selector *

- Click on the **Credential Manager Selector** field **(2)**.
- Select a Token from the list **(3)**.


InfluxDB

URL *
http://127.0.0.1:8086

Organization *
amorph

Bucket *
dev

Token *
Token credential reference

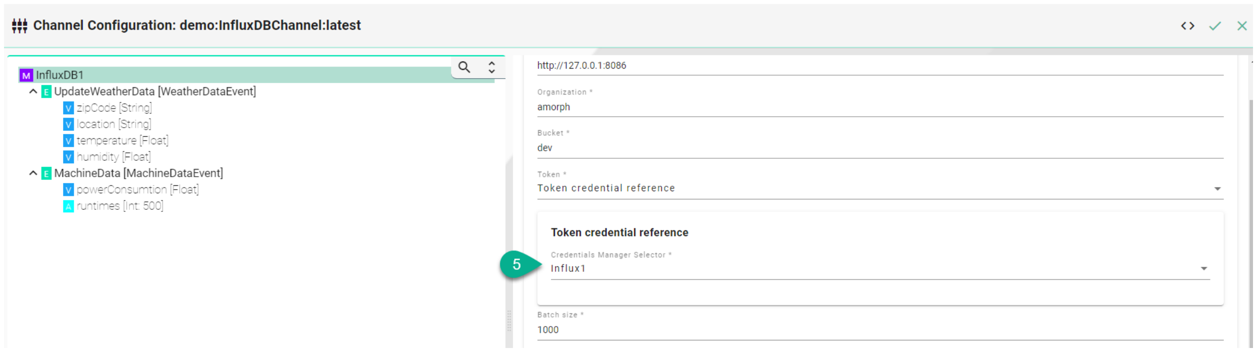
Token credential reference

+ Add credentials

Influx1

Batch size *

- If the token is not saved in the Credentials Manager, click on the **Add credentials** option **(4)**, input the token details and save it.
- The new token is saved and added into the configuration **(5)**.


Channel Configuration: demo:InfluxDBChannel:latest

InfluxDB1
UpdateWeatherData [WeatherDataEvent]
zipCode [String]
location [String]
temperature [Float]
humidity [Float]
MachineData [MachineDataEvent]
powerConsumtion [Float]
runtimes [Int: 500]

http://127.0.0.1:8086
Organization *: amorph
Bucket *: dev
Token *: Token credential reference

Token credential reference
Credentials Manager Selector *: Influx1

Batch size *: 1000

5.7 User Management

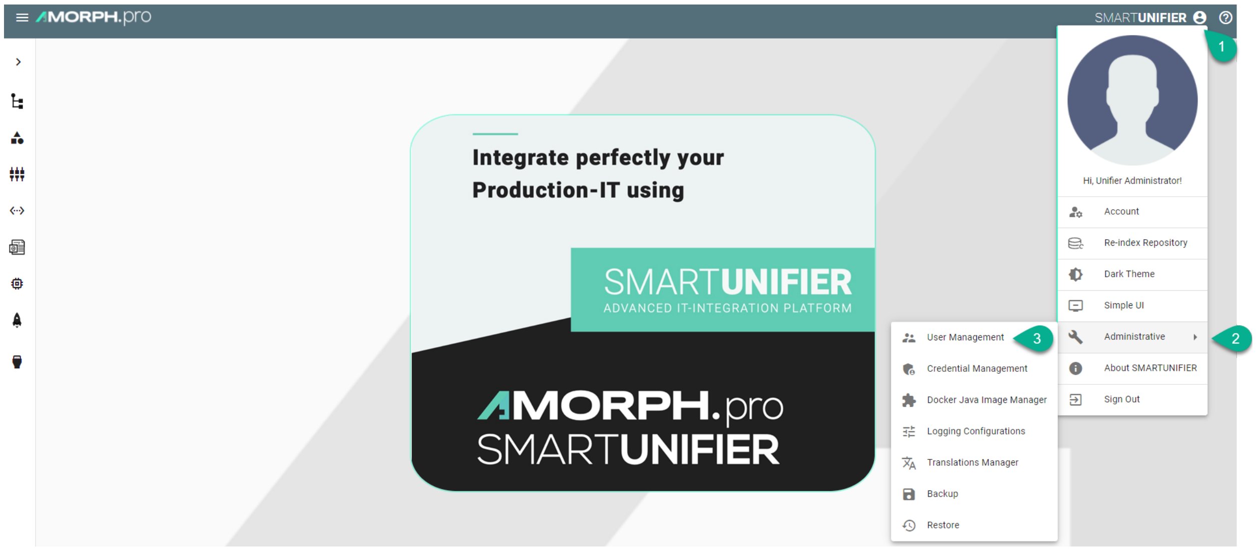
5.7.1 About User Management

Within the User Management the administrator can create users accounts, assign permissions as well as activate or deactivate user accounts.

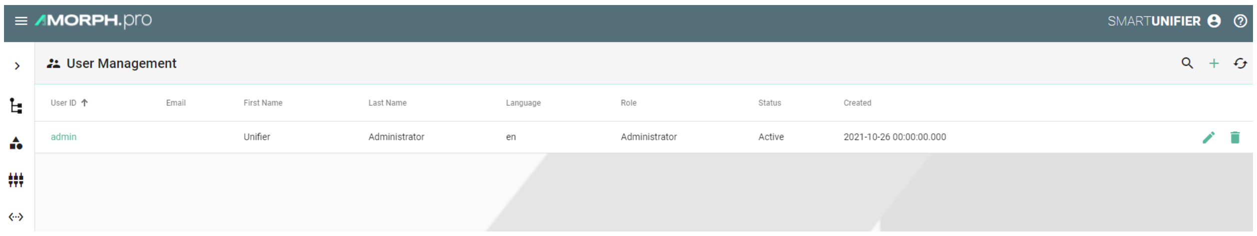
5.7.2 How to access

Follow the steps bellow to access the User Management:

- Click on the **Account** icon **(1)**, go to the **Administrative** option **(2)** and select the **User Management** perspective **(3)**.



- The User Management main view is visible.



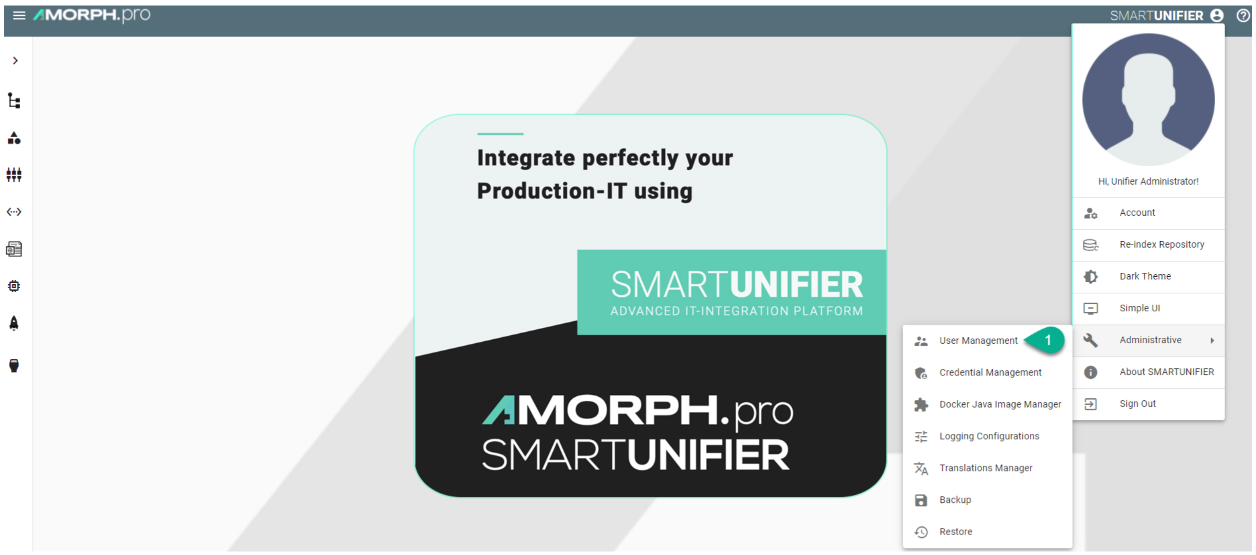
User ID	Email	First Name	Last Name	Language	Role	Status	Created
admin		Unifier	Administrator	en	Administrator	Active	2021-10-26 00:00:00.000

Note: The User Management can only be accessed by user accounts with an administrator role assigned.

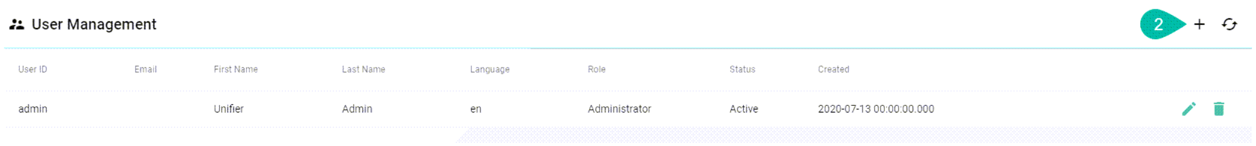
5.7.3 Add a new user

This procedure describes how to create a new user account.

- Select the SMART**UNIFIER** User Management perspective **(1)**.



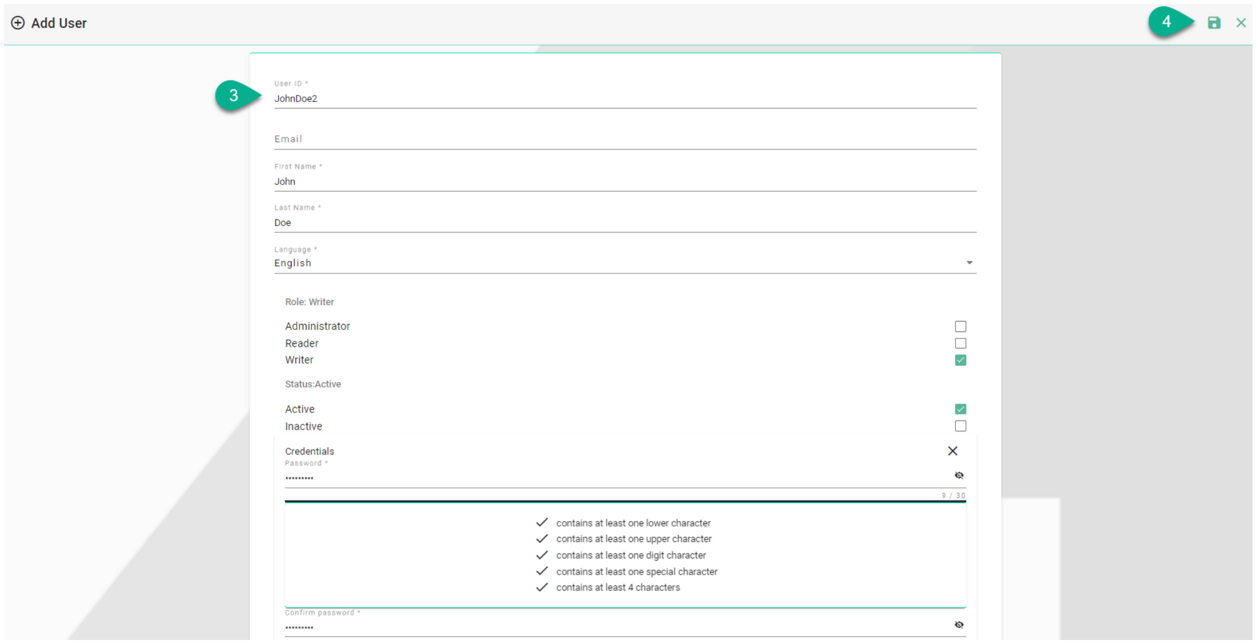
- Click the “Add User” button **(2)**.



User ID	Email	First Name	Last Name	Language	Role	Status	Created
admin		Unifier	Admin	en	Administrator	Active	2020-07-13 00:00:00.000

- In the “Add User” view provide the following information **(3)**:
  - Provide a **user id**, **first** and **last name**
  - Optionally, provide an e-mail address
  - Set a preferred language for the *SMARTUNIFIER Manager*.
- The role defines the permission of the user. It is mandatory to assign a role for the user. The following roles are available for use in the SMART**UNIFIER**.

   - **Administrator**: Full read and write access for the SMART**UNIFIER** Configuration and Administration.
   - **Reader**: Only read access for the SMART**UNIFIER** Configuration
   - **Writer**: Read and write access for the SMART**UNIFIER** Configuration
- Choose the account status: Active or Inactive.
   - **Active**: User account is activated and ready to use.
   - **Inactive**: User account is deactivated and cannot be used until it is activated again.
- Set an initial password for the first login of the new user.
- After all mandatory fields are filled in, click the "Save" button **(4)**.

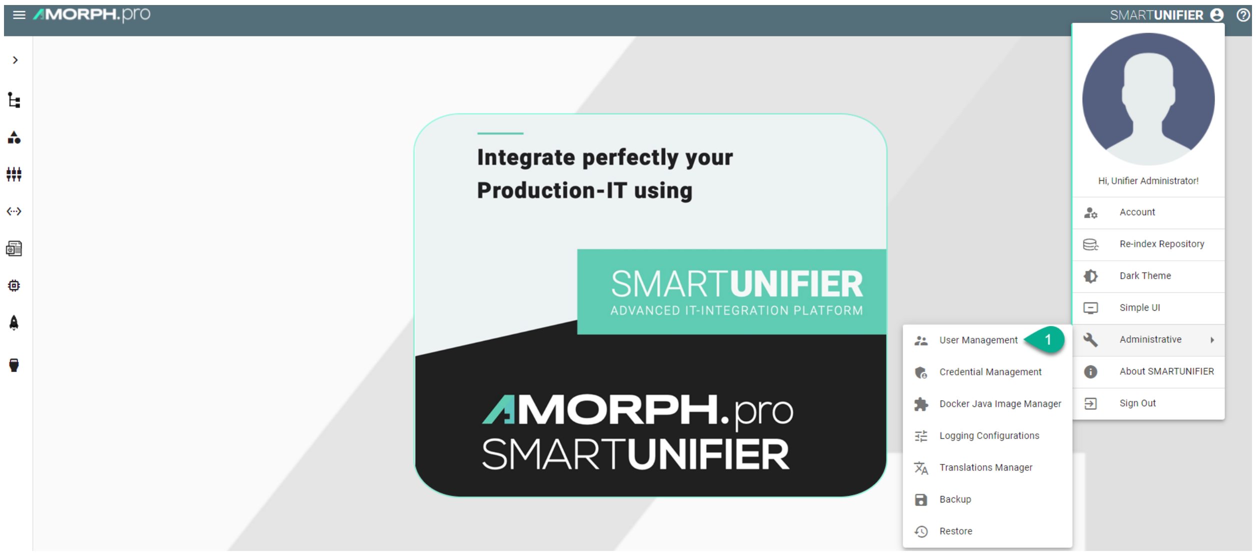


User ID *
JohnDoe2
Email
First Name *
John
Last Name *
Doe
Language *
English
Role: Writer
Administrator ☐
Reader ☐
Writer ✓
Status:Active
Active ✓
Inactive ☐
Credentials
Password *
....
9 / 30
✓ contains at least one lower character
✓ contains at least one upper character
✓ contains at least one digit character
✓ contains at least one special character
✓ contains at least 4 characters
Confirm password *
....

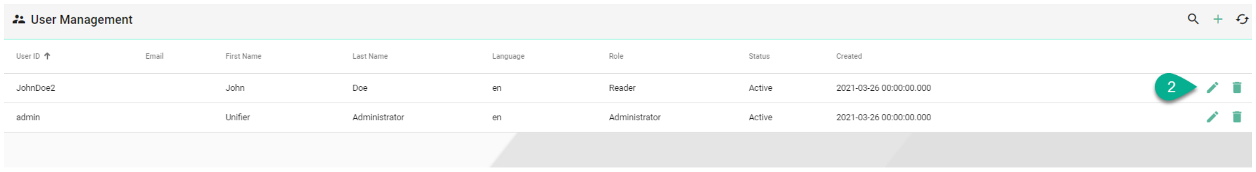
5.7.4 Edit a user

This procedure describes how to edit an existing user account.

- Select the SMART**UNIFIER** User Management perspective **(1)**.



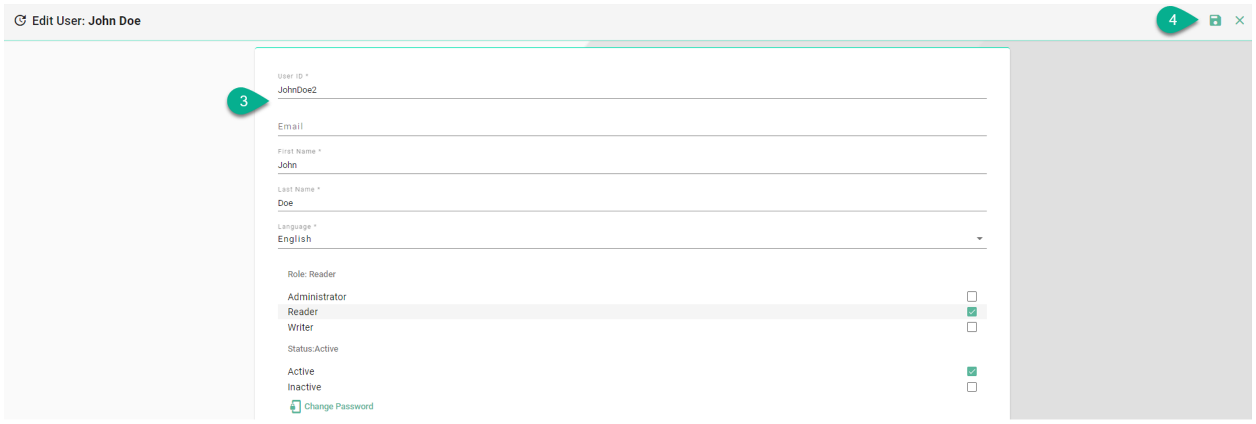
- Click the "Edit" button **(2)**.



User ID	Email	First Name	Last Name	Language	Role	Status	Created
JohnDoe2		John	Doe	en	Reader	Active	2021-03-26 00:00:00.000
admin		Unifier	Administrator	en	Administrator	Active	2021-03-26 00:00:00.000

In the "Edit" view the user account can be redefined **(3)**.

- update the user details: user id, first and last name, email address
- change the language
- edit the user permission: Administrator, Writer or Reader
- **activate** or **inactivate** the user account
- change the password


Edit User: John Doe

User ID *: JohnDoe2
Email:
First Name *: John
Last Name *: Doe
Language *: English

Role: Reader

Administrator	☐
Reader	✓
Writer	☐

Status:Active

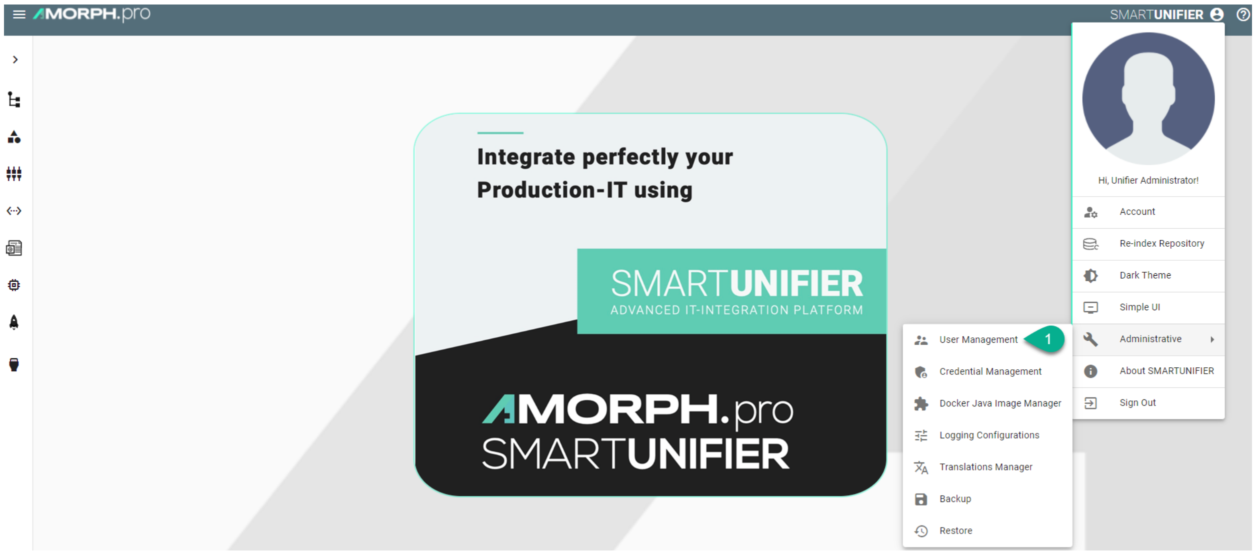
Active	✓
Inactive	☐

- After editing, click the "Save" button **(4)**.

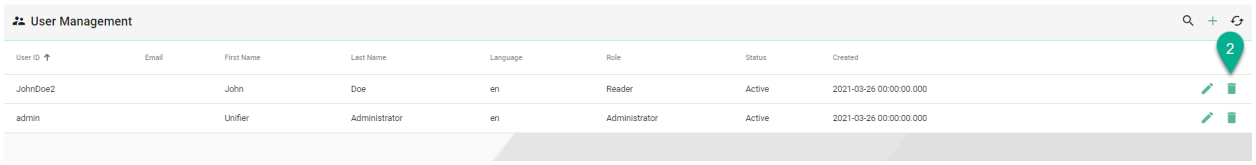
5.7.5 Delete a user

This procedure describes how to delete a user account.

- Select the SMART**UNIFIER** User Management perspective **(1)**.

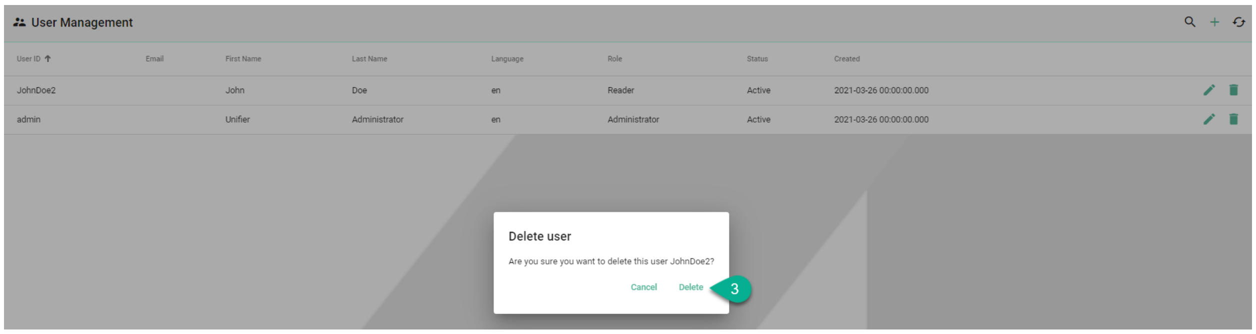


- Click the "Delete" button **(2)**.


User Management

User ID	Email	First Name	Last Name	Language	Role	Status	Created
JohnDoe2		John	Doe	en	Reader	Active	2021-03-26 00:00:00.000
admin		Unifier	Administrator	en	Administrator	Active	2021-03-26 00:00:00.000

Confirm by selecting the "Delete" button **(3)**.


User Management

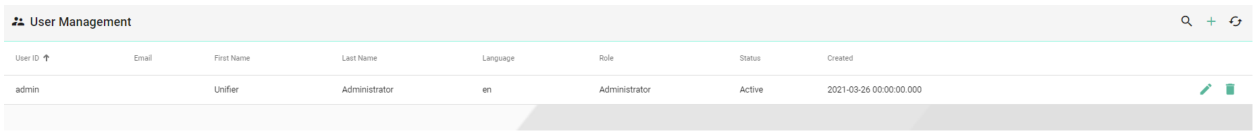
User ID	Email	First Name	Last Name	Language	Role	Status	Created
JohnDoe2		John	Doe	en	Reader	Active	2021-03-26 00:00:00.000
admin		Unifier	Administrator	en	Administrator	Active	2021-03-26 00:00:00.000

Delete user

Are you sure you want to delete this user JohnDoe2?

Cancel Delete

The user account is deleted and no more visible in the SMART**UNIFIER** User Management perspective.


User Management

User ID	Email	First Name	Last Name	Language	Role	Status	Created
admin		Unifier	Administrator	en	Administrator	Active	2021-03-26 00:00:00.000

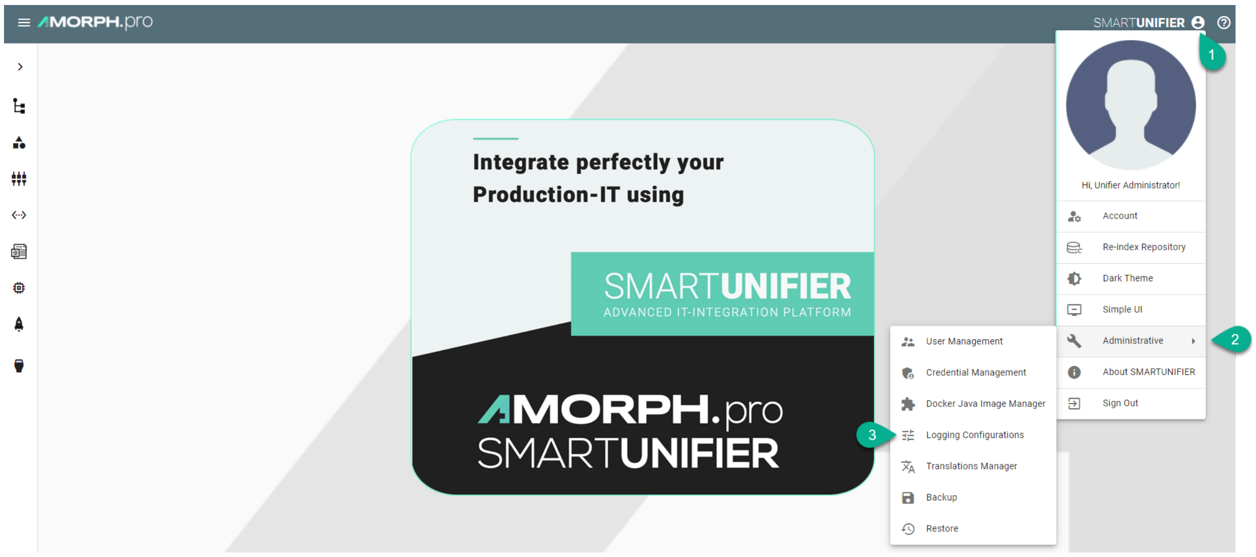
5.8 Logging Configurations

Log files in SMART**UNIFIER** are generated using the log4j framework. The Logging Configuration features enables to create new *log Levels* configurations that can be selected when deploying a Communication Instance.

5.8.1 How to access

Follow the steps below to access the feature:

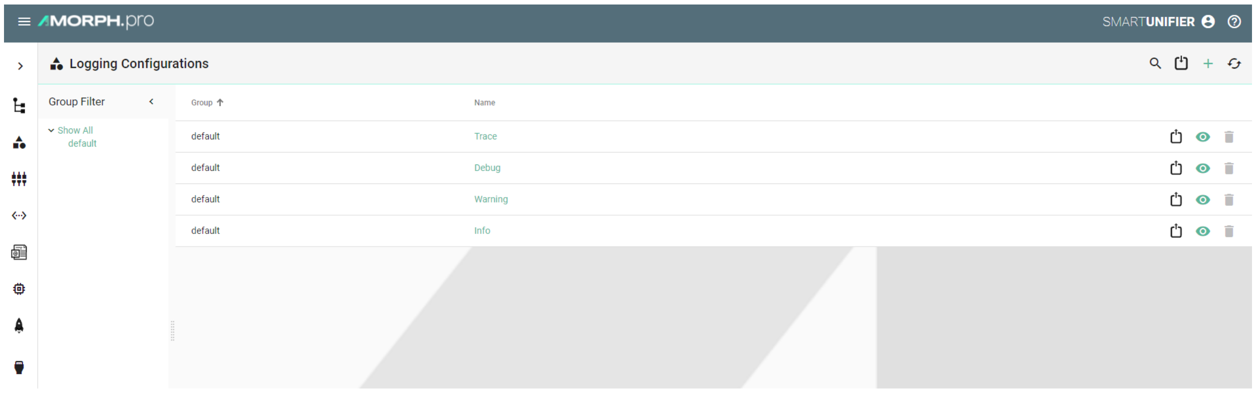
- Click on the **Account** icon **(1)**, go to the **Administrative** option **(2)** and select the **Logging Configurations** perspective **(3)**.



- Logging Configurations main view is visible, as seen below.

- There are four predefined log4j configurations that can be used as template when creating a new log level.

Note: The predefined log4j configurations can not be edited or deleted.



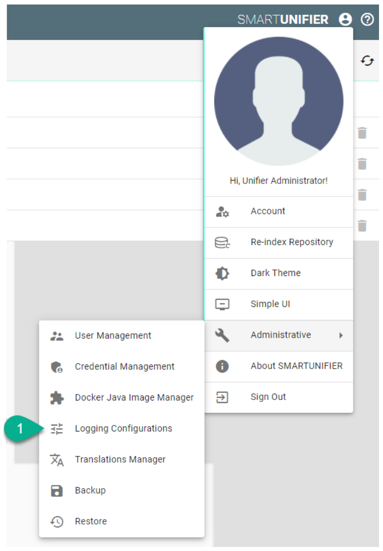
Group	Name
default	Trace
default	Debug
default	Warning
default	Info

Note: This feature can be only used by users with the administration role.

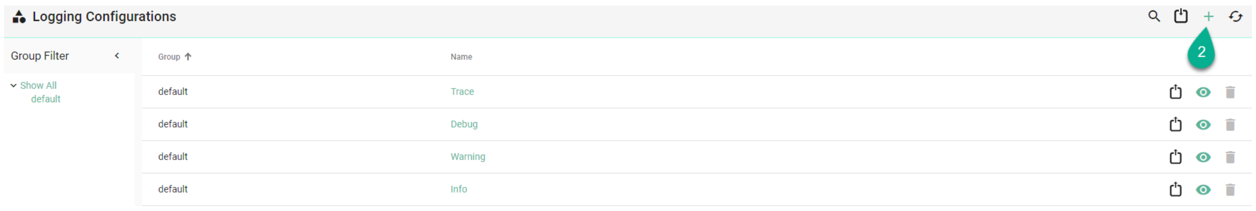
5.8.2 Add a new logging file

Follow the steps below to add a new log4j configuration file:

- Select the **Logging Configurations** perspective **(1)**.

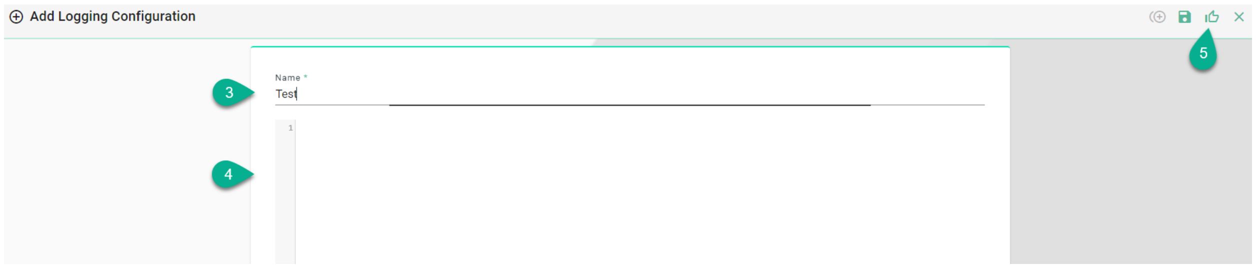


- Click on the “Add” button **(2)**.



Group	Name
default	Trace
default	Debug
default	Warning
default	Info

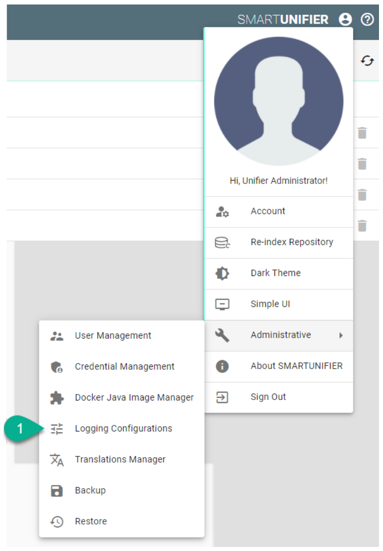
- Input the file **Name (3)** and the configuration **(4)**.
- Click on the **Save and Close** button to exit **(5)**.



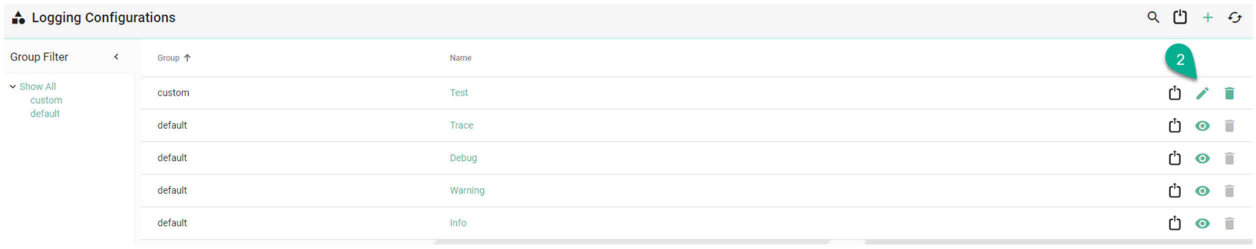
5.8.3 Edit a logging file

Follow the steps below to edit a log4j configuration file:

- Select the **Logging Configurations** perspective **(1)**.

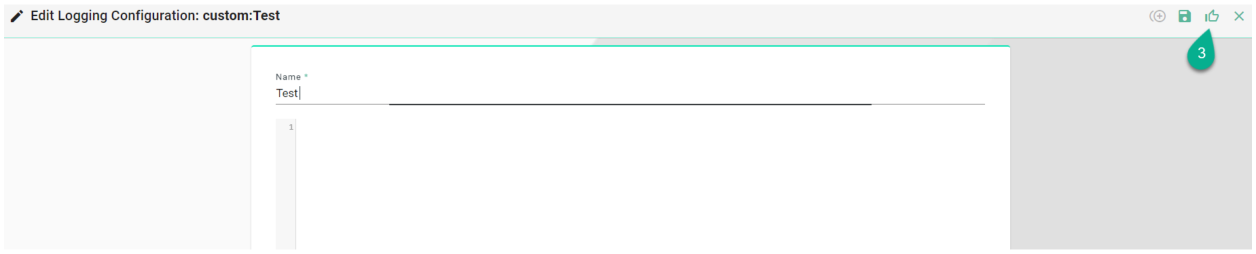


- Click on the "Edit" button **(2)**.



Group	Name
custom	Test
default	Trace
default	Debug
default	Warning
default	Info

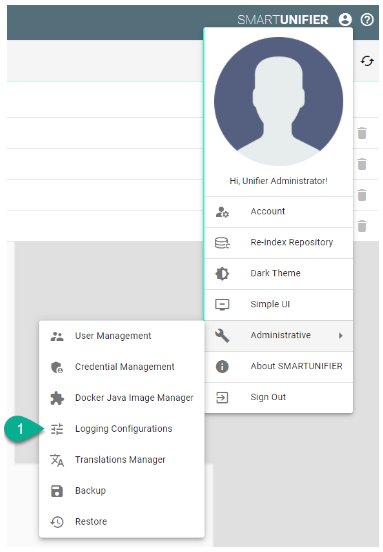
- Edit and click on the **Save and Close** button to exit **(3)**.



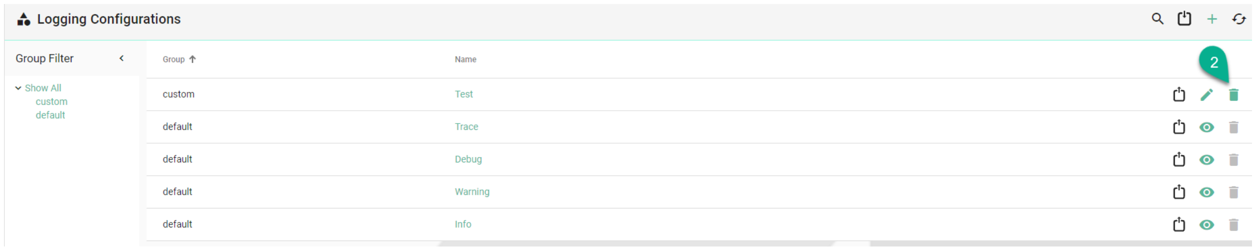
5.8.4 Delete a logging file

Follow the steps below to delete a log4j configuration file:

- Select the **Logging Configurations** perspective **(1)**.

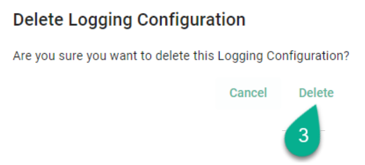


- Click on the "Delete" button **(2)**.



Group	Name
custom	Test
default	Trace
default	Debug
default	Warning
default	Info

- To confirm, click on the **Delete** button **(3)**.


Delete Logging Configuration
Are you sure you want to delete this Logging Configuration?
Cancel
Delete
3

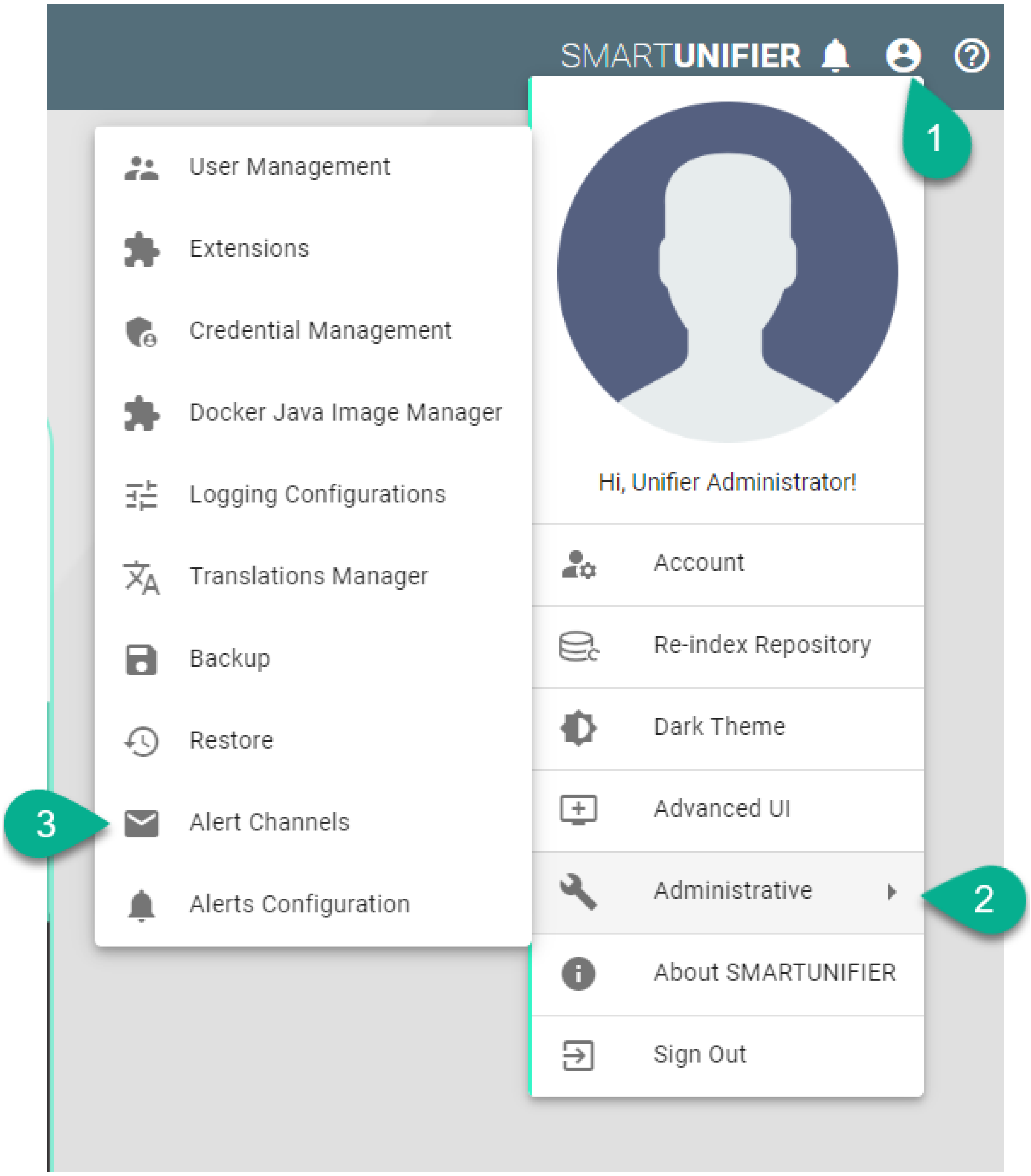
5.9 Alert Channels

Within the Alert Channels, the user can configure and manage the channels for sending alert notifications (e.g., send via email an alert for Instance errors).

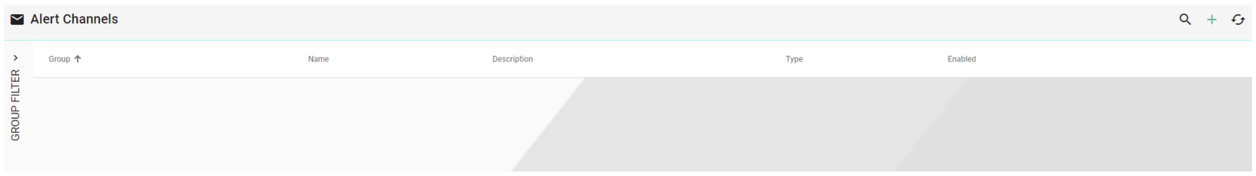
5.9.1 How to access

Follow the steps bellow to access the Alert Channels:

- Click on the **Account** icon **(1)**, go to **Administrative** section **(2)** and select the **Alert Channels** option **(3)**.



- The Alert Channels section is visible.

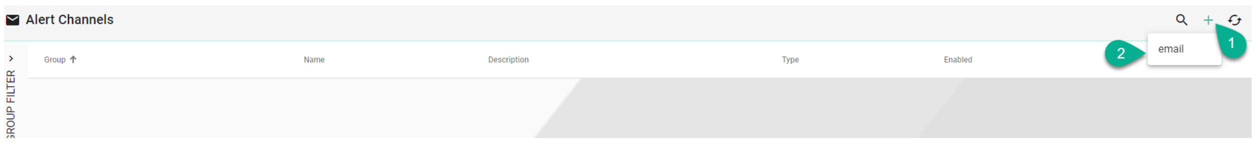


Note: The Alert Channels can only be accessed by user accounts with an administrator role assigned.

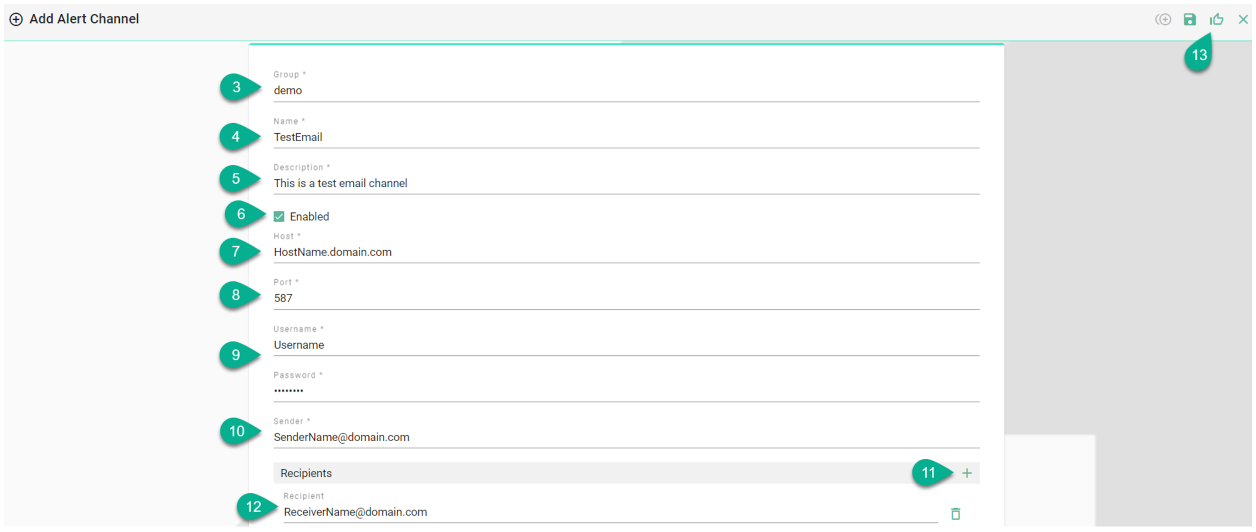
5.9.2 Add an Email Channel

Follow the steps described below to add an Email channel:

- Click on the **Add** button **(1)**.
- Select the **Email** option **(2)**.



- Type a name for the Group **(3)**.
- Input the email channel name **(4)**.
- Add description **(5)**.
- Click to check the **Enabled** box **(6)**.
- Provide the Host name **(7)**.
- Type the host Port **(8)**.
- Input the Username and Password **(9)**.
- Provide the Sender email address **(10)**.
- Click on the **Add** button **(11)** to input the Recipients email addresses **(12)**.



Group *
demo

Name *
TestEmail

Description *
This is a test email channel

Enabled

Host *
HostName.domain.com

Port *
587

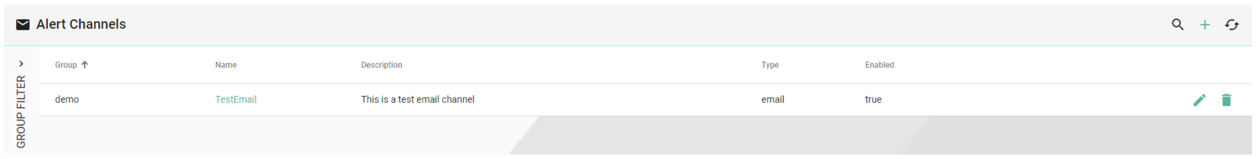
Username *
Username

Password *
········

Sender *
SenderName@domain.com

Recipient
ReceiverName@domain.com

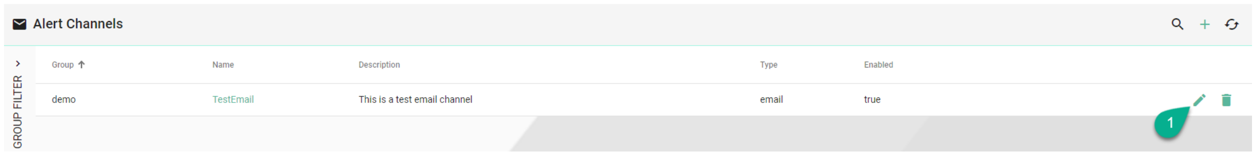
- Click on the **Save and Close** button **(13)**.



Group	Name	Description	Type	Enabled
demo	TestEmail	This is a test email channel	email	true

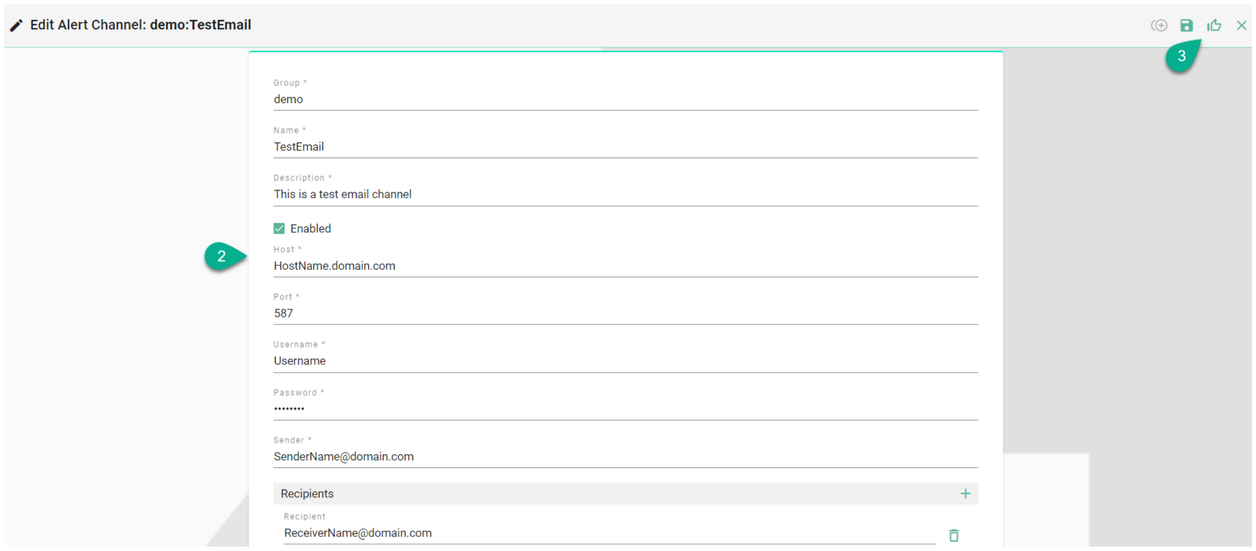
5.9.3 Edit Alert Channels

To edit an alert channel, select the **Edit** button **(1)**.



Group	Name	Description	Type	Enabled
demo	TestEmail	This is a test email channel	email	true

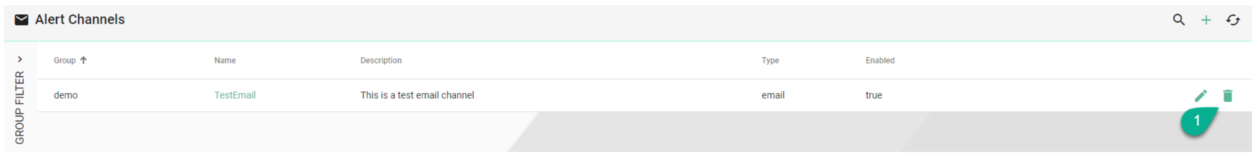
The Edit Mode is visible, the configuration can be edited **(2)** and then save the session by selecting the **Save and Close** button **(3)**.



Group *	demo
Name *	TestEmail
Description *	This is a test email channel
Enabled	✓
Host *	HostName.domain.com
Port *	587
Username *	Username
Password *	••••••••
Sender *	SenderName@domain.com
Recipient	ReceiverName@domain.com

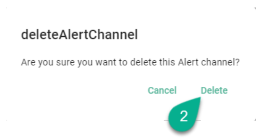
5.9.4 Delete Alert Channels

To delete an alert channel, select the **Delete** button **(1)**.



Group	Name	Description	Type	Enabled
demo	TestEmail	This is a test email channel	email	true

A pop-up confirmation appears, select the **Delete** button **(2)**.


deleteAlertChannel

Are you sure you want to delete this Alert channel?

5.10 Alerts Configuration

Within this section, the user can configure and manage alerts.

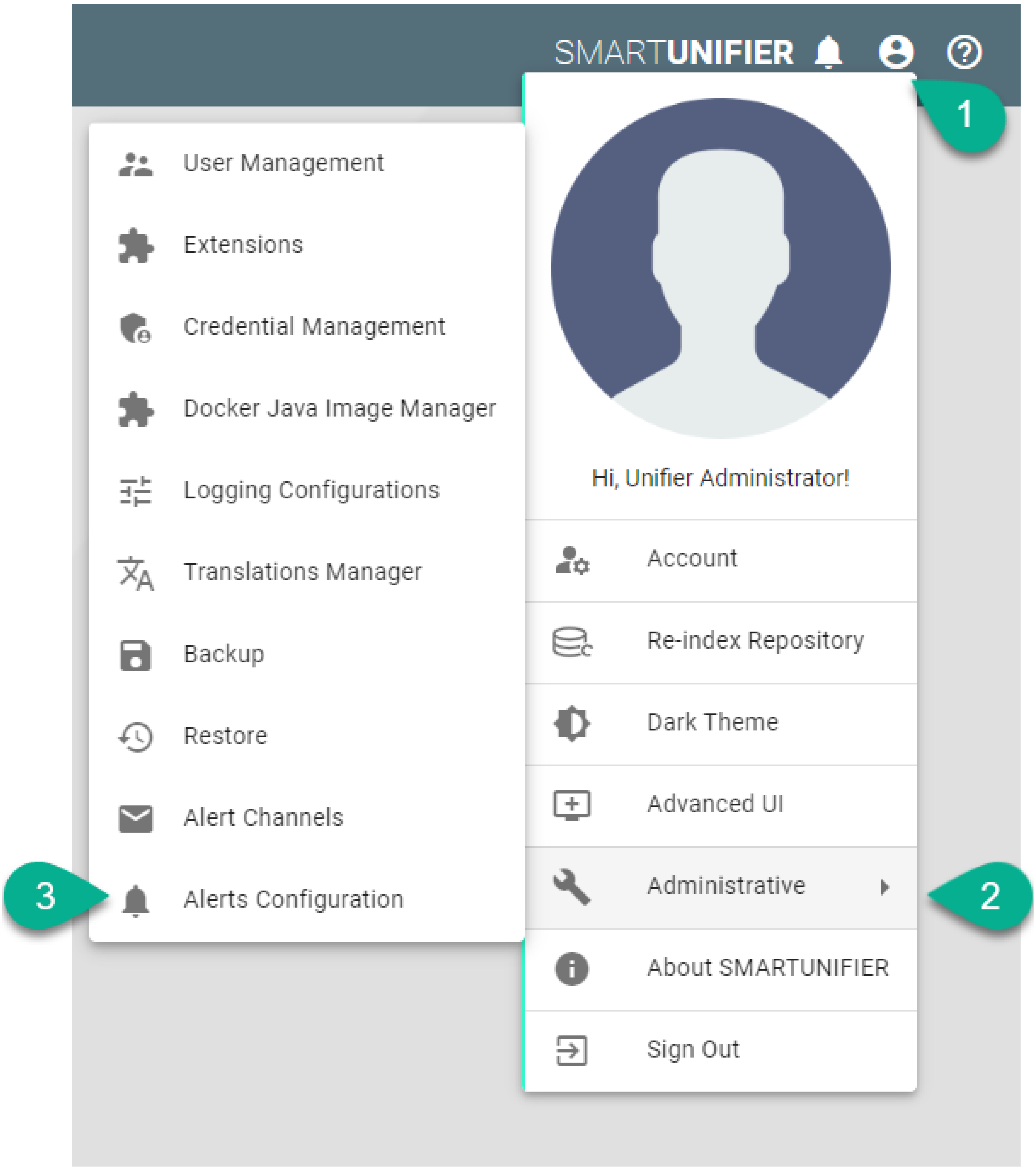
Alerts can be sent for:

- Instance errors
- Instance Deployment status changed
- Endpoint Client status changed

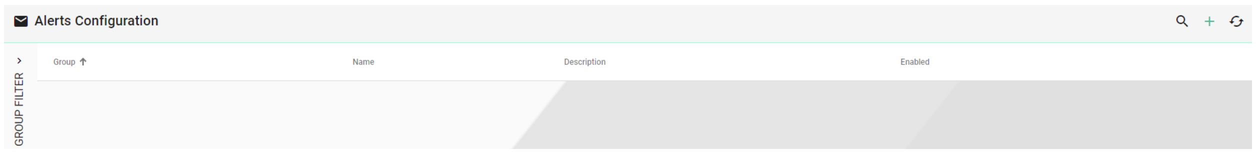
5.10.1 How to access

Follow the steps bellow to access the Alerts Configuration:

- Click on the **Account** icon **(1)**, go to **Administrative** section **(2)** and select the **Alerts Configuration** option **(3)**.



- The Alerts Configuration is visible.



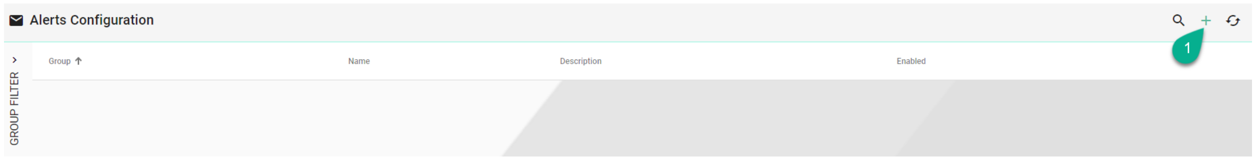
Group	Name	Description	Enabled

Note: The Alerts Configuration can only be accessed by user accounts with an administrator role assigned.

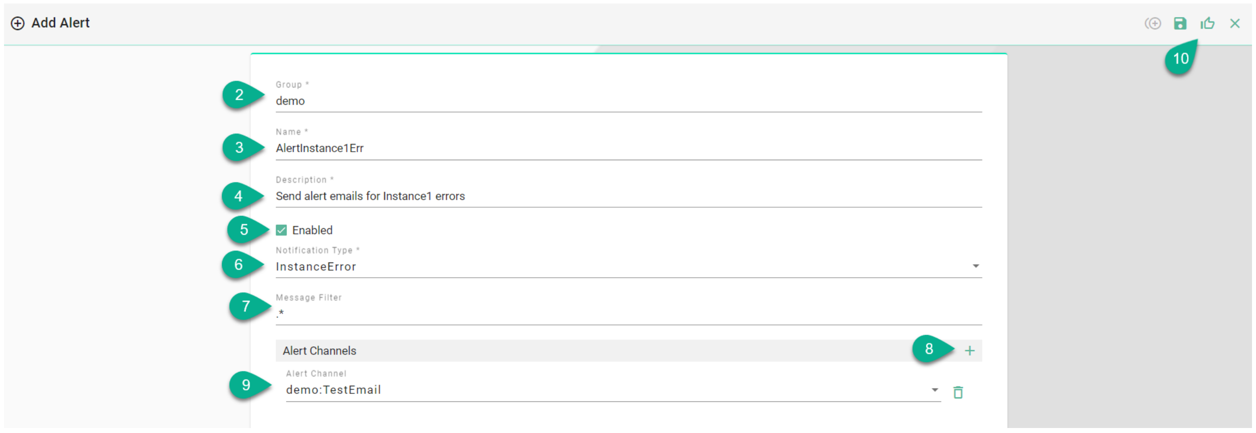
5.10.2 Add Alerts

Follow the steps described below to add an alert:

- Click on the **Add** button **(1)**.

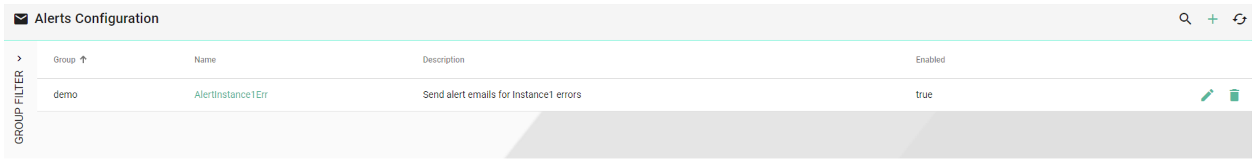


- Type a name for Group **(2)**.
- Input the alert name **(3)**.
- Add description **(4)**.
- Check the box for Enabled **(5)**.
- Select the Notification Type **(6)**.
- Input the regex filter **(7)**.
- Click on the **Add** button **(8)** to select the alert channel **(9)**.



Group *
demo
Name *
AlertInstance1Err
Description *
Send alert emails for Instance1 errors
✓ Enabled
Notification Type *
InstanceError
Message Filter
.*
Alert Channel
demo:TestEmail

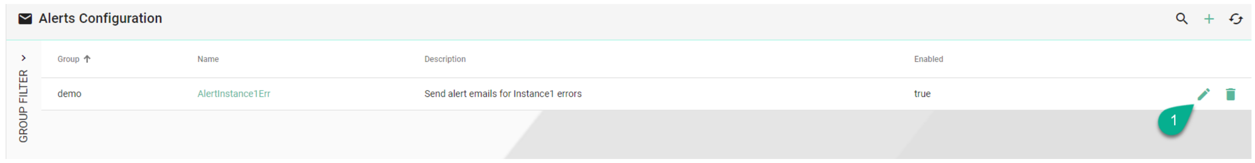
- Click on the **Save and Close** button **(10)**.



Group	Name	Description	Enabled
demo	AlertInstance1Err	Send alert emails for Instance1 errors	true

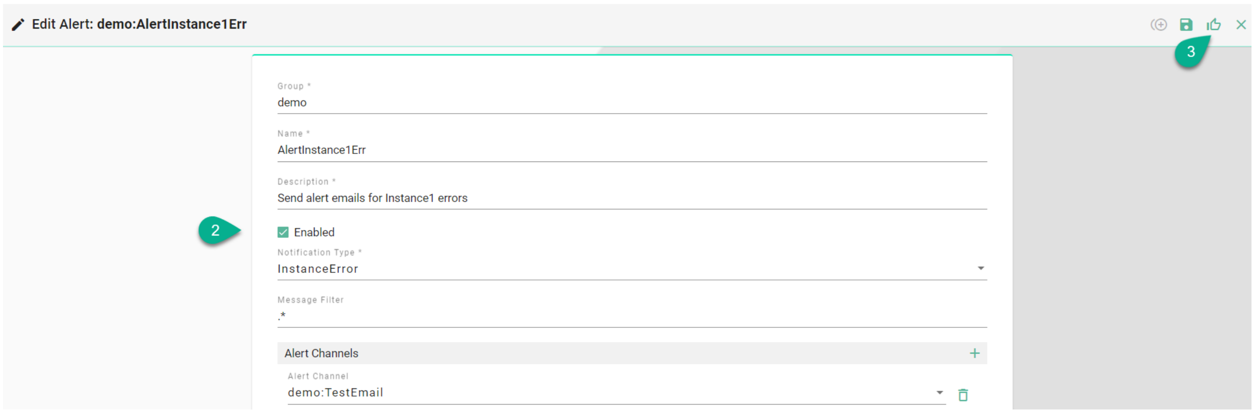
5.10.3 Edit Alerts

To edit an alert, select the **Edit** button **(1)**.



Group	Name	Description	Enabled
demo	AlertInstance1Err	Send alert emails for Instance1 errors	true

The Edit Mode is visible, the configuration can be edited **(2)** and then save the session by selecting the **Save and Close** button **(3)**.



Group *
demo

Name *
AlertInstance1Err

Description *
Send alert emails for Instance1 errors

Enabled

Notification Type *
InstanceError

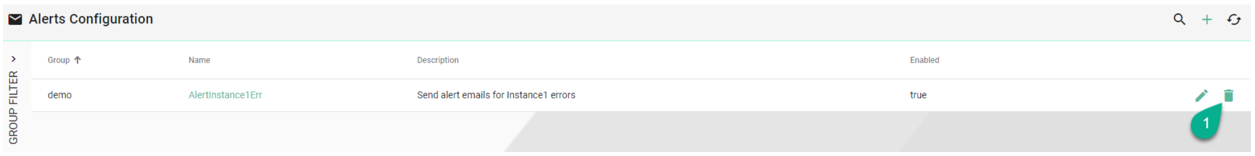
Message Filter
.*

Alert Channels

Alert Channel
demo:TestEmail

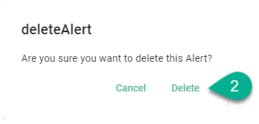
5.10.4 Delete Alerts

To delete an alert, select the **Delete** button **(1)**.



Group	Name	Description	Enabled
demo	AlertInstance1Err	Send alert emails for Instance1 errors	true

A pop-up confirmation appears, select the **Delete** button **(2)**.



deleteAlert

Are you sure you want to delete this Alert?

5.11 Extensions

Note: Please contact Amorph Systems for guidance on how to enable and use extensions.

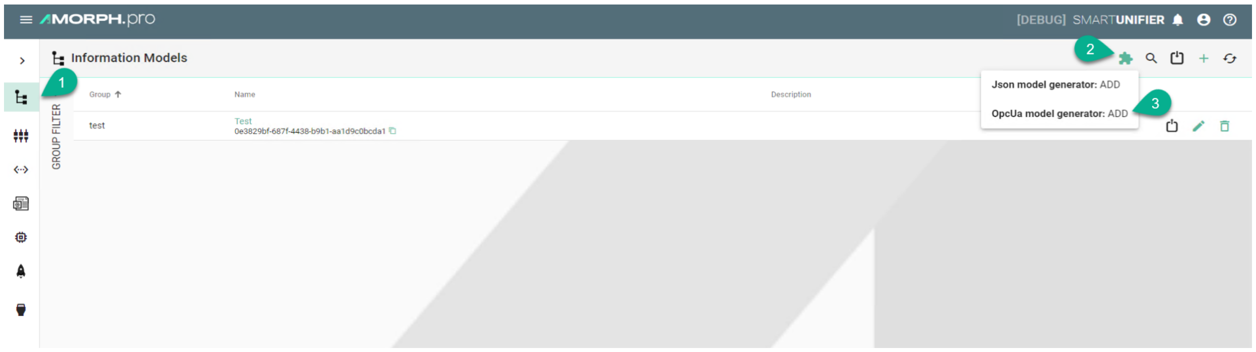
5.11.1 OpcUa Model Import

SMART**UNIFIER** provides the possibility to generate an OpcUa Information Model using a XML-file or connecting to the OpcUa server.

Create a new Information Model (OPCUA)

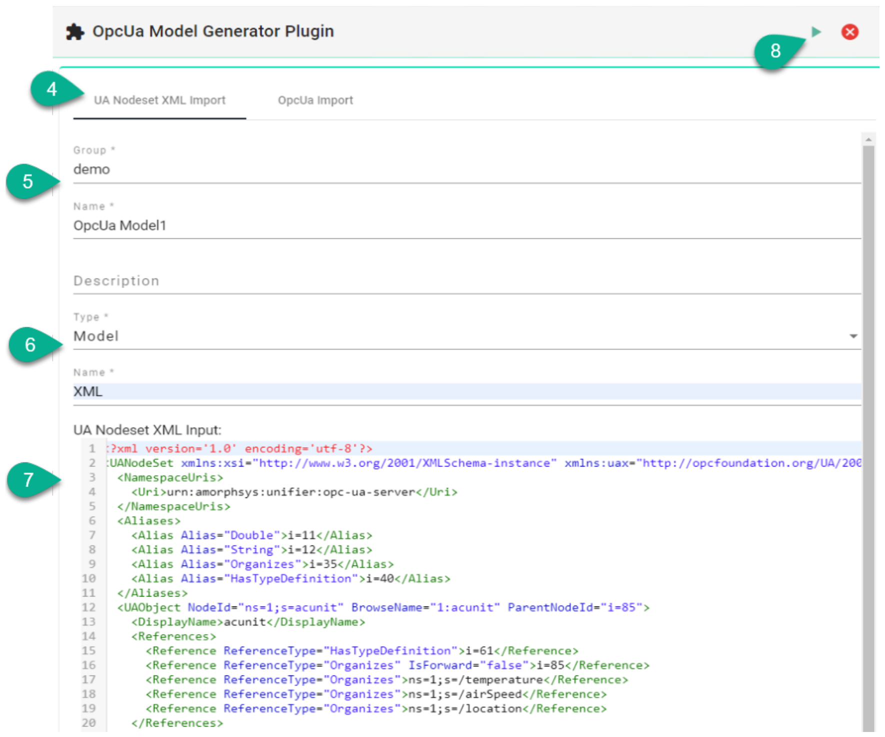
Follow the steps described below to generate an Information Model:

- Select the SMART**UNIFIER** Information Model Perspective **(1)**.
- Click on the **Extensions** button **(2)**.
- Select the **OpcUa model generator: ADD** option **(3)**.



OpcUa Nodeset XML Import

- Select the **UA Nodeset XML Import** option **(4)**.
- Provide the following mandatory information: **Group** and **Name (5)**.
- Select the type of the **Information Model Node** and provide a **Name (6)** :
  - **Model** - the OpcUa data is converted inside the root model node
  - **Event** - the OpcUa data is converted inside an Event node type
  - **Variable** - the OpcUa data is converted inside a Variable node type
- Paste the content from the XML file **(7)**.
- To finish, click on the **Save** Button **(8)**.



Group *
demo

Name *
OpcUa Model1

Description

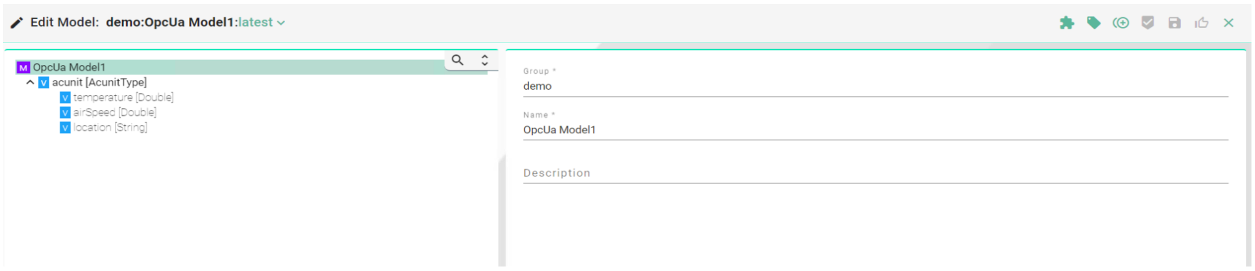
Type *
Model

Name *
XML

UA Nodeset XML Input:

```
:?xml version='1.0' encoding='utf-8'?>
:UANodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/200
  <NamespaceUris>
    <Uri>urn:amorphsys:unifier:opc-ua-server</Uri>
  </NamespaceUris>
  <Aliases>
    <Alias Alias="Double">i=11</Alias>
    <Alias Alias="String">i=12</Alias>
    <Alias Alias="Organizes">i=35</Alias>
    <Alias Alias="HasTypeDefinition">i=40</Alias>
  </Aliases>
  <UAObject NodeId="ns=1;s=acunit" BrowseName="1:acunit" ParentNodeId="i=85">
    <DisplayName>acunit</DisplayName>
    <References>
      <Reference ReferenceType="HasTypeDefinition">i=61</Reference>
      <Reference ReferenceType="Organizes" IsForward="false">i=85</Reference>
      <Reference ReferenceType="Organizes">ns=1;s=/temperature</Reference>
      <Reference ReferenceType="Organizes">ns=1;s=/airSpeed</Reference>
      <Reference ReferenceType="Organizes">ns=1;s=/location</Reference>
    </References>
```

- The Information Model is generated.



Edit Model: demo:OpcUa Model1:latest

- M OpcUa Model1
  - V acunit [AcunitType]
    - V temperature [Double]
    - V airSpeed [Double]
    - V location [String]

Group *
demo

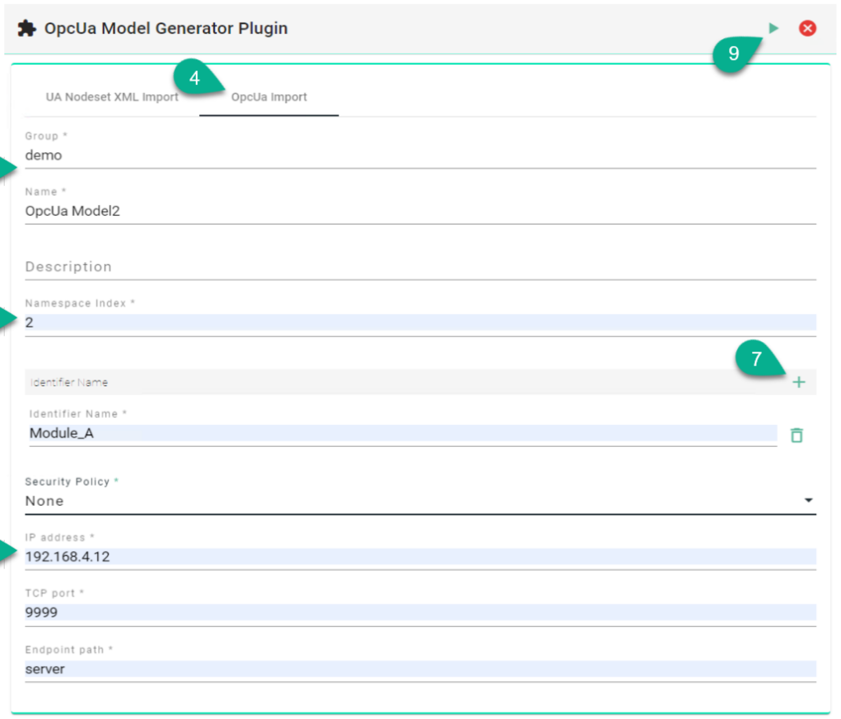
Name *
OpcUa Model1

Description

OpcUa Direct Import

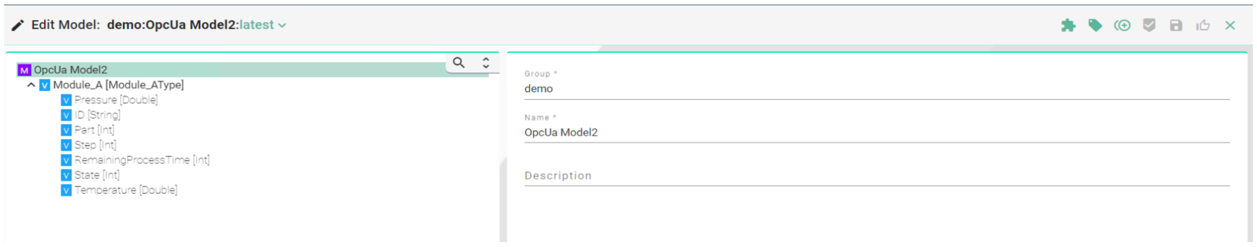
- Select the **OpcUa Import** option **(4)**.
- Provide the following mandatory information: **Group** and **Name (5)**.
- Input the **Namespace Index (6)**.
- Click on the **Add identifier name** button and provide an **Identifier Name (7)**.
- Provide the server details **(8)**:
  - **Security Policy**
  - **IP address**
  - **TCP port**
  - **Endpoint path**

- To finish, click on the **Save** Button **(9)**.



Group *
demo
Name *
OpcUa Model2
Description
Namespace Index *
2
Identifier Name
Identifier Name *
Module_A
Security Policy *
None
IP address *
192.168.4.12
TCP port *
9999
Endpoint path *
server

- The Information Model is generated.



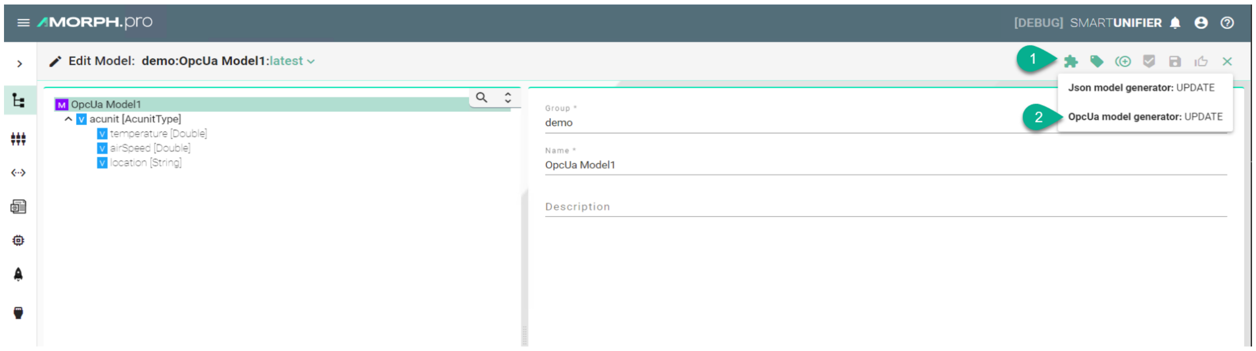
OpcUa Model2
Module_A [Module_AType]
Pressure [Double]
ID [String]
Part [Int]
Step [Int]
RemainingProcessTime [Int]
State [Int]
Temperature [Double]

Group *
demo
Name *
OpcUa Model2
Description

Update an existing Information Model (OPCUA)

Follow the steps described below to update an Information Model:

- Open an Information Model to edit and click on the **Extensions** button **(1)**.
- Select the **OpcUa model generator: UPDATE** option **(2)**.



Edit Model: demo:OpcUa Model1:latest

OpcUa Model1
acunit [AcunitType]
temperature [Double]
airSpeed [Double]
location [String]

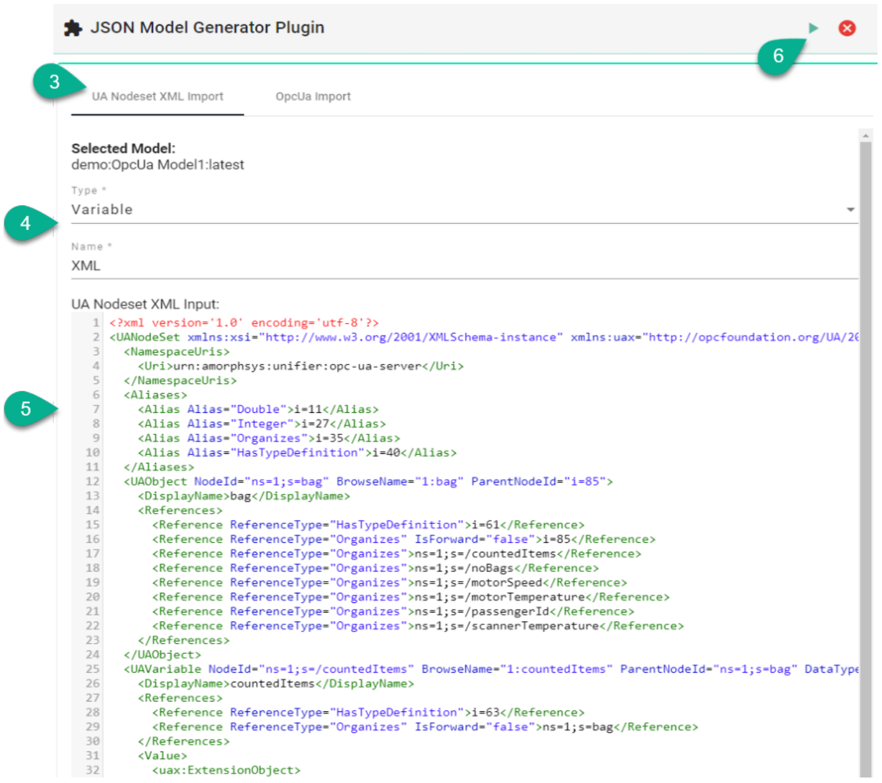
Group *
demo

Name *
OpcUa Model1

Description

OpcUa Nodeset XML Import (Update)

- Select the **UA Nodeset XML Import** option **(3)**.
- Update the **Type** and the **Name (4)**.
- Paste the updated content from a XML-file **(5)**.
- To finish, click on the **Save** button **(6)**.



Selected Model:
demo:OpcUa Model1:latest

Type *
Variable

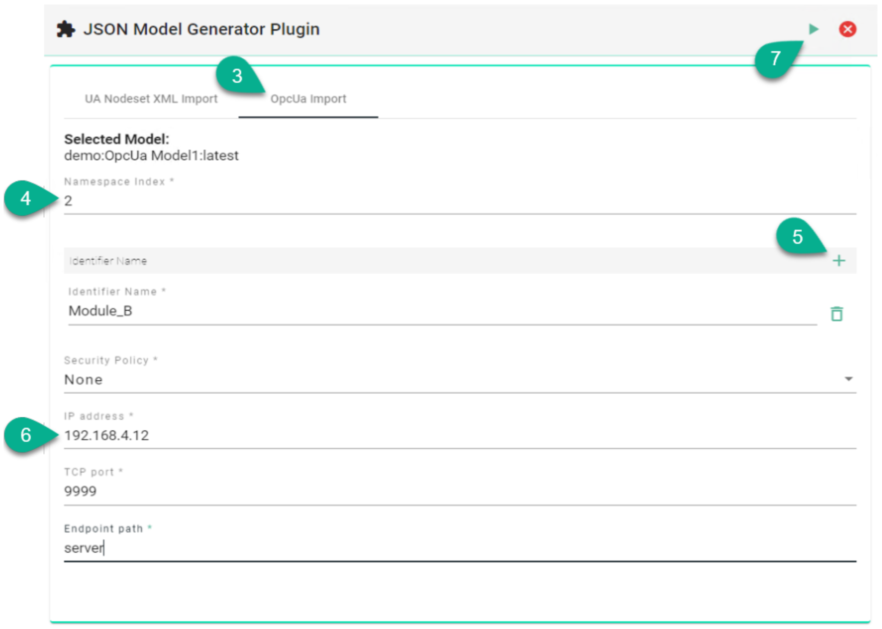
Name *
XML

UA Nodeset XML Input:

```
<?xml version='1.0' encoding='utf-8'?>
<UANodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/20
  <NamespaceUris>
    <Uri>urn:amorphsys:unifier:opc-ua-server</Uri>
  </NamespaceUris>
  <Aliases>
    <Alias Alias="Double">i=11</Alias>
    <Alias Alias="Integer">i=27</Alias>
    <Alias Alias="Organizes">i=35</Alias>
    <Alias Alias="HasTypeDefinition">i=40</Alias>
  </Aliases>
  <UAObject NodeId="ns=1;s=bag" BrowseName="1:bag" ParentNodeId="i=85">
    <DisplayName>bag</DisplayName>
    <References>
      <Reference ReferenceType="HasTypeDefinition">i=61</Reference>
      <Reference ReferenceType="Organizes" IsForward="false">i=85</Reference>
      <Reference ReferenceType="Organizes">ns=1;s=/countedItems</Reference>
      <Reference ReferenceType="Organizes">ns=1;s=/noBags</Reference>
      <Reference ReferenceType="Organizes">ns=1;s=/motorSpeed</Reference>
      <Reference ReferenceType="Organizes">ns=1;s=/motorTemperature</Reference>
      <Reference ReferenceType="Organizes">ns=1;s=/passengerId</Reference>
      <Reference ReferenceType="Organizes">ns=1;s=/scannerTemperature</Reference>
    </References>
  </UAObject>
  <UAVariable NodeId="ns=1;s=/countedItems" BrowseName="1:countedItems" ParentNodeId="ns=1;s=bag" DataType
    <DisplayName>countedItems</DisplayName>
    <References>
      <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
      <Reference ReferenceType="Organizes" IsForward="false">ns=1;s=bag</Reference>
    </References>
    <Value>
      <uax:ExtensionObject>
```

OpcUa Direct Import (Update)

- Select the **OpcUa Import** option **(3)**.
- Input the **Namespace Index (4)**.
- Click on the **Add identifier name** button and provide an **Identifier Name (5)**.
- Provide the server details **(6)**:
  - **Security Policy**
  - **IP address**
  - **TCP port**
  - **Endpoint path**
- To finish, click on the **Save** Button **(7)**.



Selected Model:
demo:OpcUa Model1:latest

Namespace Index *
2

Identifier Name *
Module_B

Security Policy *
None

IP address *
192.168.4.12

TCP port *
9999

Endpoint path *
server

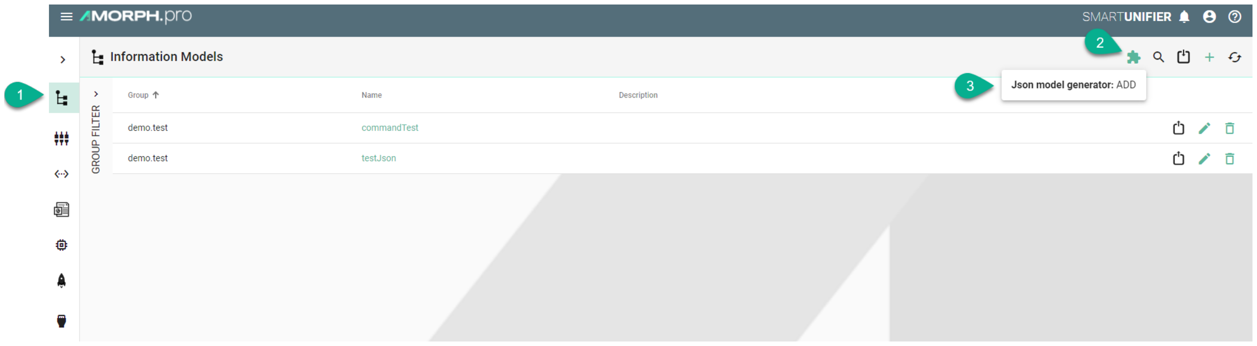
5.11.2 JSON Model Import

SMART**UNIFIER** provides the possibility to generate an Information Model using a JSON-file.

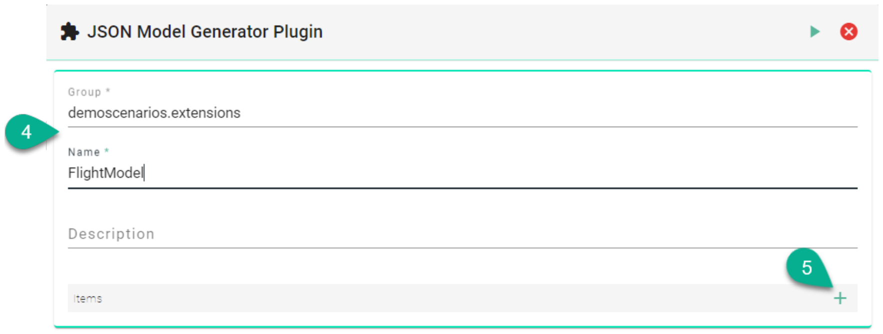
Create a new Information Model (JSON)

Follow the steps described below to generate an Information Model:

- Select the SMART**UNIFIER** Information Model Perspective **(1)**.
- Click on the **Extensions** button **(2)**.
- Select the **Json model generator: ADD** option **(3)**.



- Provide the following mandatory information: **Group** and **Name (4)**.
- Click on the **Add item** button **(5)**.

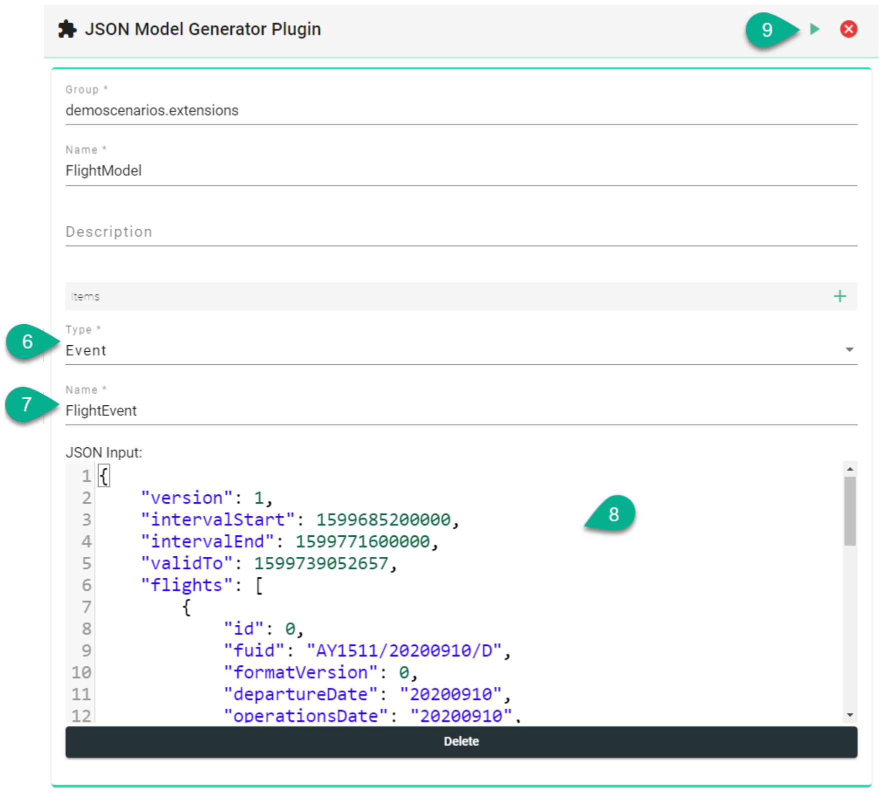


- Select the type of the **Information Model Node (6)**:
  - **Model** - the Json data is converted inside the root model node

  - **Event** - the Json data is converted inside an Event node type
  - **Variable** - the Json data is converted inside a Variable node type
  - **Command** - the Json data is converted inside a Command node type
- Enter a **Name (7)**.
- Paste the content from a Json file **(8)**.

Note: Make sure to copy the JSON object { }.

- To finish, click on the **Save** Button **(9)**.



Group *
demoscenarios.extensions

Name *
FlightModel

Description

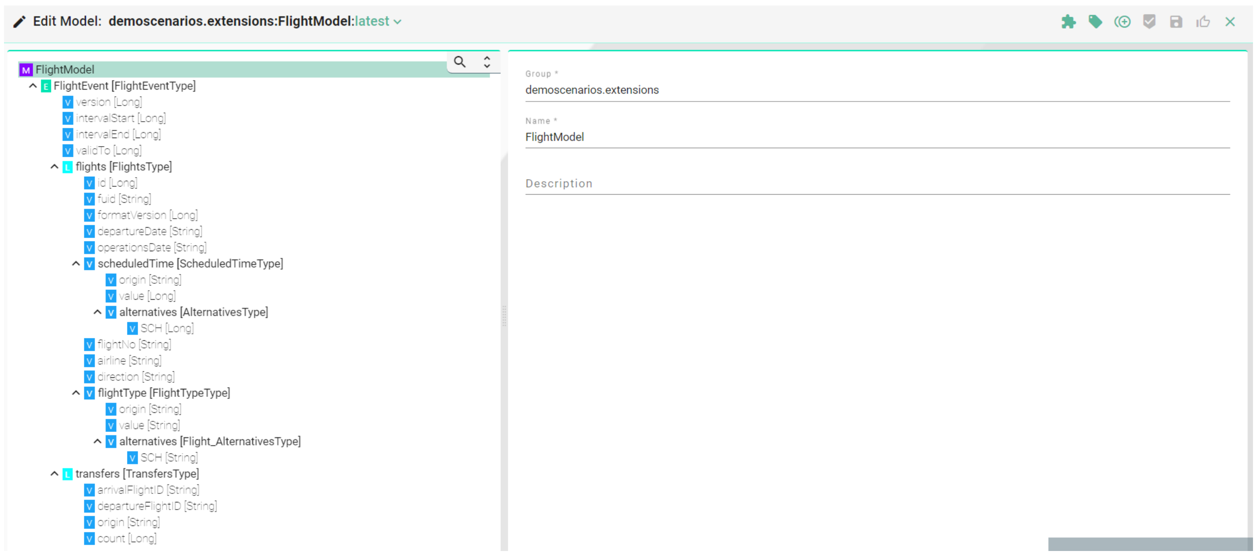
Type *
Event

Name *
FlightEvent

JSON Input:

```
{
    "version": 1,
    "intervalStart": 1599685200000,
    "intervalEnd": 1599771600000,
    "validTo": 1599739052657,
    "flights": [
        {
            "id": 0,
            "fuid": "AY1511/20200910/D",
            "formatVersion": 0,
            "departureDate": "20200910",
            "operationsDate": "20200910",
```

- The Information Model is generated.



Edit Model: **demoscenarios.extensions:FlightModel:**latest

```
FlightModel
  FlightEvent [FlightEventType]
      version [Long]
      intervalStart [Long]
      intervalEnd [Long]
      validTo [Long]
    flights [FlightsType]
        id [Long]
        fuid [String]
        formatVersion [Long]
        departureDate [String]
        operationsDate [String]
        scheduledTime [ScheduledTimeType]
          origin [String]
          value [Long]
          alternatives [AlternativesType]
            SCH [Long]
        flightNo [String]
        airline [String]
        direction [String]
        flightType [FlightTypeType]
          origin [String]
          value [String]
          alternatives [Flight_AlternativesType]
            SCH [String]
    transfers [TransfersType]
        arrivalFlightID [String]
        departureFlightID [String]
        origin [String]
        count [Long]
```

Group *
demoscenarios.extensions

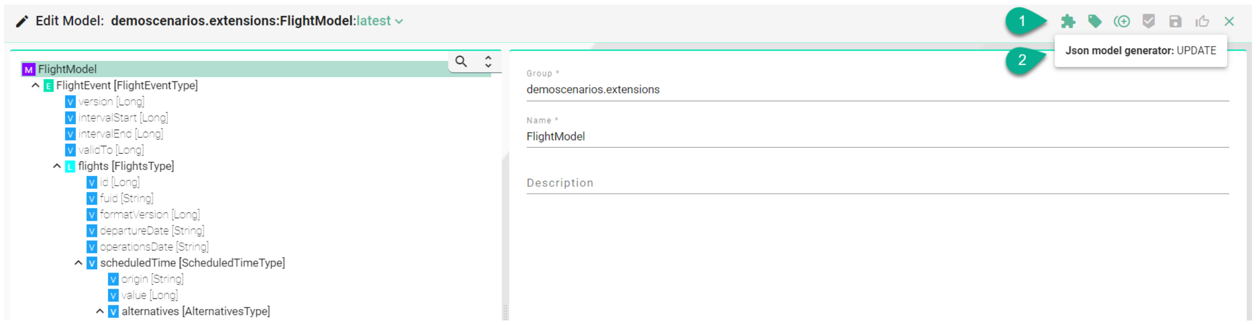
Name *
FlightModel

Description

Update an existing Information Model (JSON)

Follow the steps described below to update an Information Model:

- Open an Information Model to edit and click on the **Extensions** button **(1)**.
- Select the **Json model generator: UPDATE** option **(2)**.



Edit Model: **demoscenarios.extensions:FlightModel:**latest

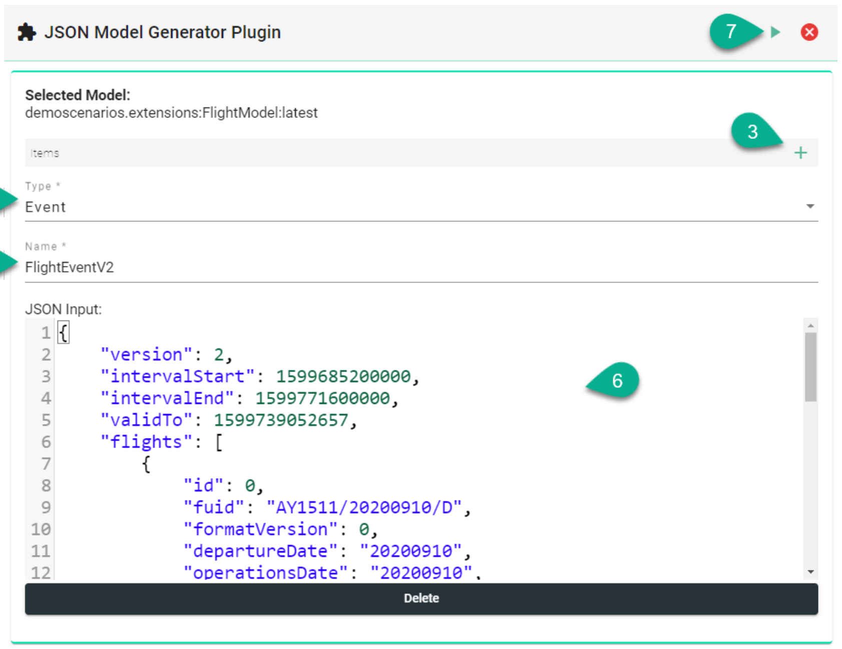
```
FlightModel
  FlightEvent [FlightEventType]
      version [Long]
      intervalStart [Long]
      intervalEnd [Long]
      validTo [Long]
    flights [FlightsType]
        id [Long]
        fuid [String]
        formatVersion [Long]
        departureDate [String]
        operationsDate [String]
        scheduledTime [ScheduledTimeType]
          origin [String]
          value [Long]
          alternatives [AlternativesType]
```

Group *
demoscenarios.extensions

Name *
FlightModel

Description

- Click on the **Add item** button **(3)**.
- Update the **Type (4)** and the **Name (5)**.
- Paste the updated content from a Json-file **(6)**.
- To finish, click on the **Save** button **(7)**.



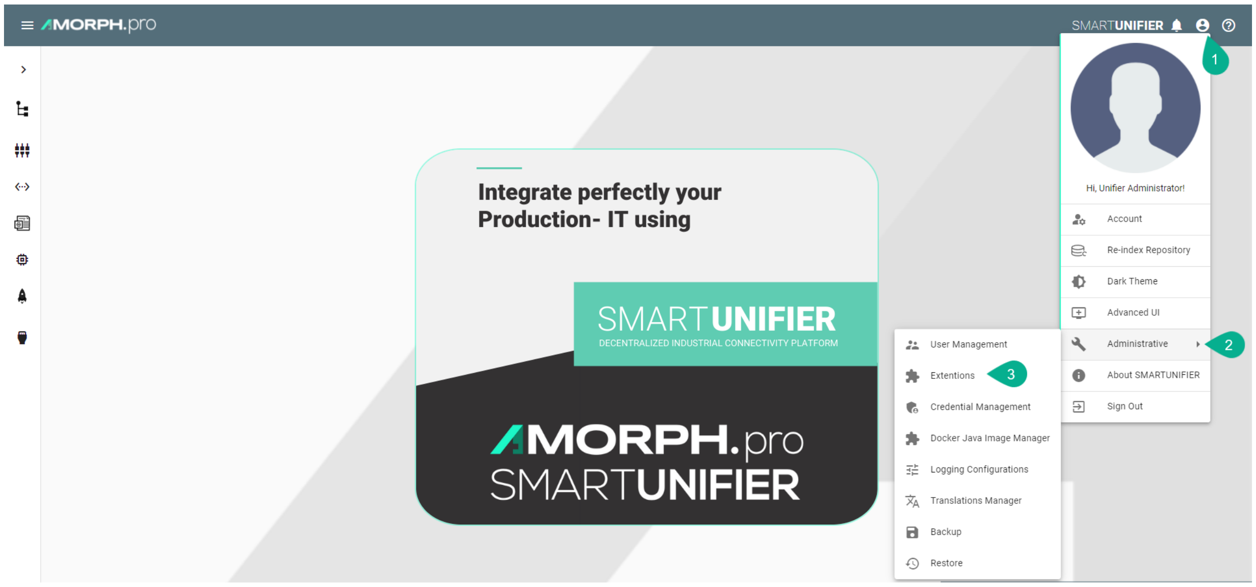
```
{
    "version": 2,
    "intervalStart": 1599685200000,
    "intervalEnd": 1599771600000,
    "validTo": 1599739052657,
    "flights": [
        {
            "id": 0,
            "fuid": "AY1511/20200910/D",
            "formatVersion": 0,
            "departureDate": "20200910",
            "operationsDate": "20200910",
```

5.11.3 AWS IoT SiteWise Model Export

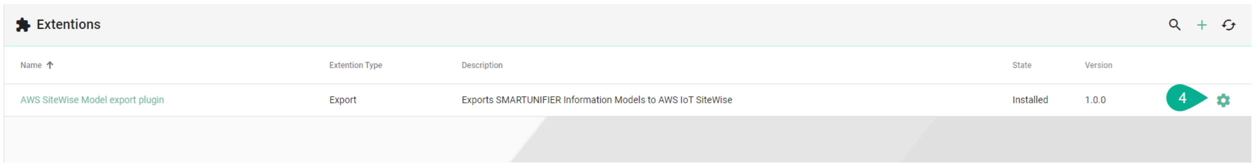
This extension allows you to export an SMART**UNIFIER** Information Model to AWS IoT SiteWise.

How to access

To access the AWS IoT SiteWise extension, click on the Account icon (1), go to the Administrative option (2) and select the Extensions (3).



Then select the **configuration** button of the (4)



Name	Extention Type	Description	State	Version
AWS SiteWise Model export plugin	Export	Exports SMARTUNIFIER Information Models to AWS IoT SiteWise	Installed	1.0.0

How to export Information to AWS IoT SiteWise

We recommend to have one user dedicated for SMART**UNIFIER**.

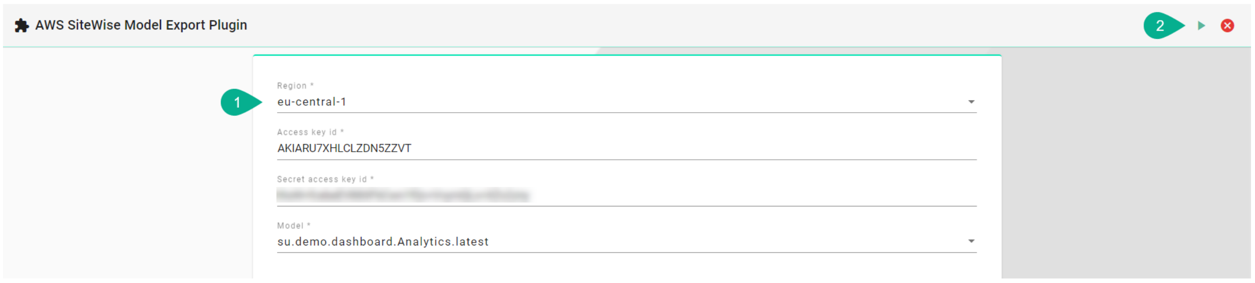
Attach the following permission:

Policy ARN	Description
arn:aws:iam::aws:policy/AWSIoTSiteWiseFullAccess	Provides full access to IoT Site-Wise.

If you do not have already an access key available you have to create a new access key. We recommend to create a new access key after 90 days.

Follow the steps described below to export a the SMART**UNIFIER** Information Model:

- Configuration of the extension (1):
  - Select the region of the AWS Iot SiteWise service you are using
  - Enter the **access key id** and the **secret access key id**
  - Select the **Information Model** you want to export
- Click on the **Run** button to execute the export (2)



Region *
eu-central-1

Access key id *
AKIARU7XHLCLZDN5ZZVT

Secret access key id *
[redacted]

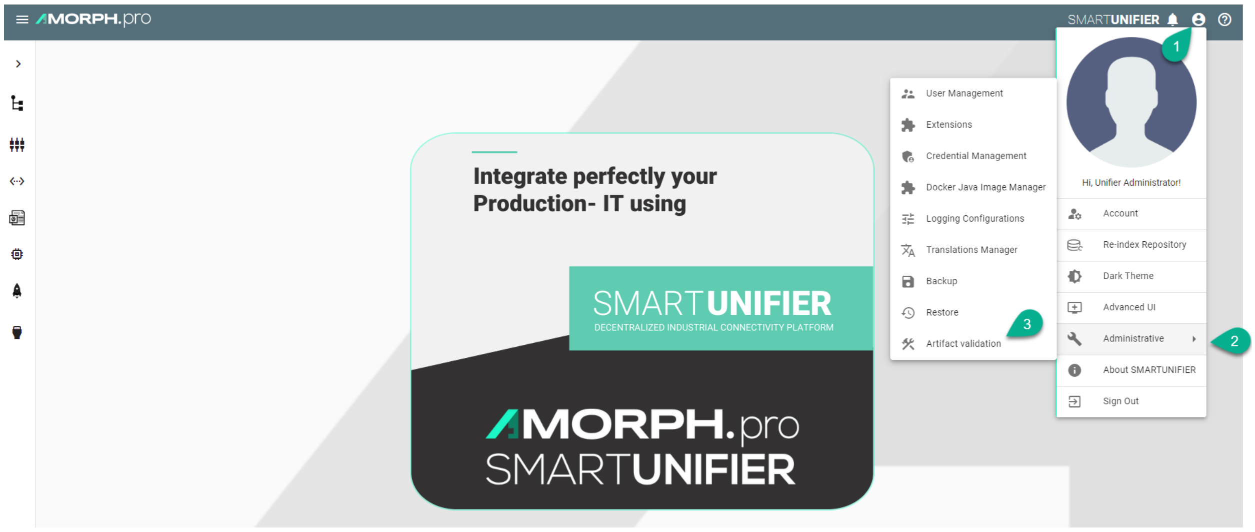
Model *
su.demo.dashboard.Analytics.latest

5.12 Configuration Components Validation

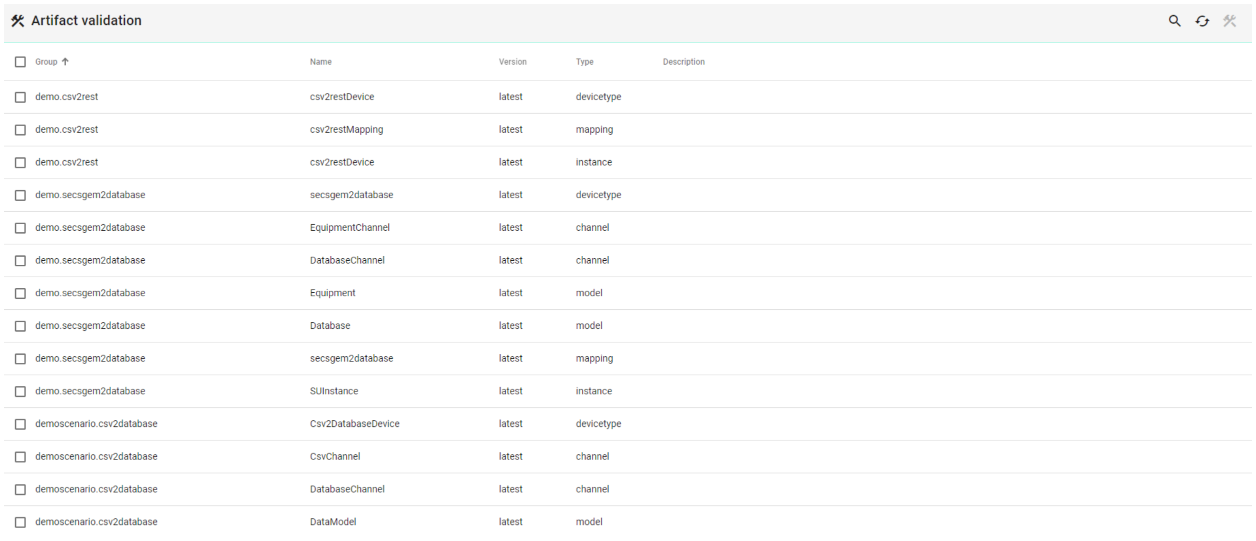
SMART**UNIFIER** provides the possibility to validate configuration components (artifacts) that are created. This is especially important when restoring a backup from an older version of a SMART**UNIFIER** Manager installation.

5.12.1 How to access

To access the Artifact Validation option, click on the **Account** icon **(1)**, go to the **Administrative** option **(2)** and select the **Artifact Validation** perspective **(3)**.

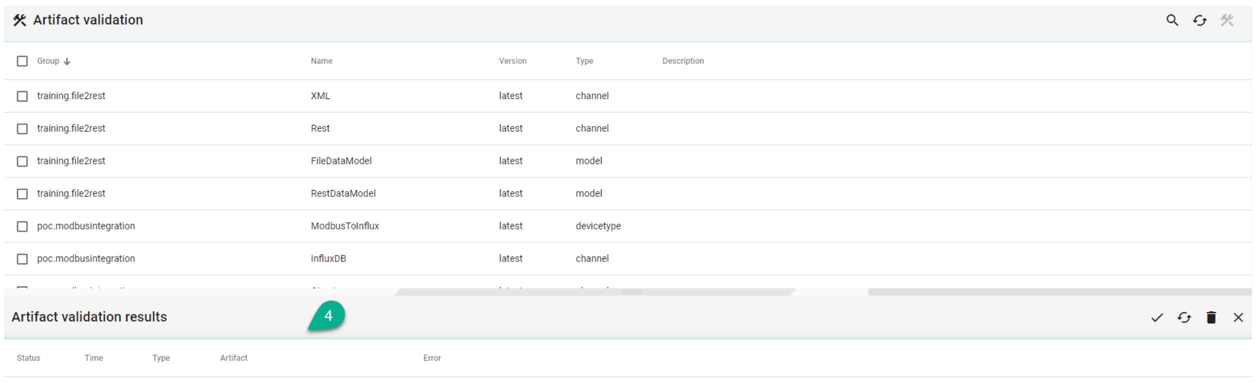


The Artifact Validation perspective is visible, displaying all the configuration components.


Artifact validation

Group ↑	Name	Version	Type	Description
demo.csv2rest	csv2restDevice	latest	devicetype	
demo.csv2rest	csv2restMapping	latest	mapping	
demo.csv2rest	csv2restDevice	latest	instance	
demo.secsgem2database	secsgem2database	latest	devicetype	
demo.secsgem2database	EquipmentChannel	latest	channel	
demo.secsgem2database	DatabaseChannel	latest	channel	
demo.secsgem2database	Equipment	latest	model	
demo.secsgem2database	Database	latest	model	
demo.secsgem2database	secsgem2database	latest	mapping	
demo.secsgem2database	SUInstance	latest	instance	
demoscenario.csv2database	Csv2DatabaseDevice	latest	devicetype	
demoscenario.csv2database	CsvChannel	latest	channel	
demoscenario.csv2database	DatabaseChannel	latest	channel	
demoscenario.csv2database	DataModel	latest	model	

Press **F8** from the keyboard to open the **Artifact Validation Results** view **(4)**.


Artifact validation

Group	Name	Version	Type	Description
training.file2rest	XML	latest	channel	
training.file2rest	Rest	latest	channel	
training.file2rest	FileDataModel	latest	model	
training.file2rest	RestDataModel	latest	model	
poc.modbusintegration	ModbusToInflux	latest	devicetype	
poc.modbusintegration	InfluxDB	latest	channel	

Artifact validation results

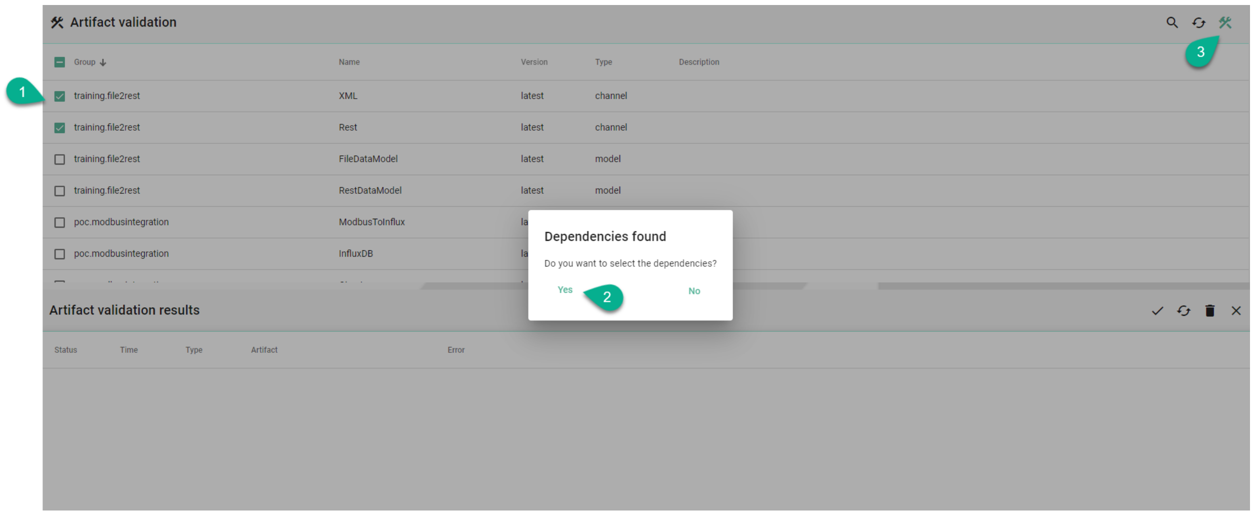
Status	Time	Type	Artifact	Error

Note: The Artifact Validation features can only be accessed by user accounts with an administrator role assigned.

5.12.2 How to Validate Artifacts

Follow the steps described below to validate artifacts:

- From the Artifact Validation perspective select the desired artifacts **(1)**
- If dependencies are found, a pop-up will appear. Click on the **Yes** button **(2)** to select the dependencies
- Click on the **Validate** button **(3)**


Artifact validation

	Group	Name	Version	Type	Description
✓	training.file2rest	XML	latest	channel	
✓	training.file2rest	Rest	latest	channel	
	training.file2rest	FileDataModel	latest	model	
	training.file2rest	RestDataModel	latest	model	
	poc.modbusintegration	ModbusToInflux	la		
	poc.modbusintegration	InfluxDB	la		

Dependencies found

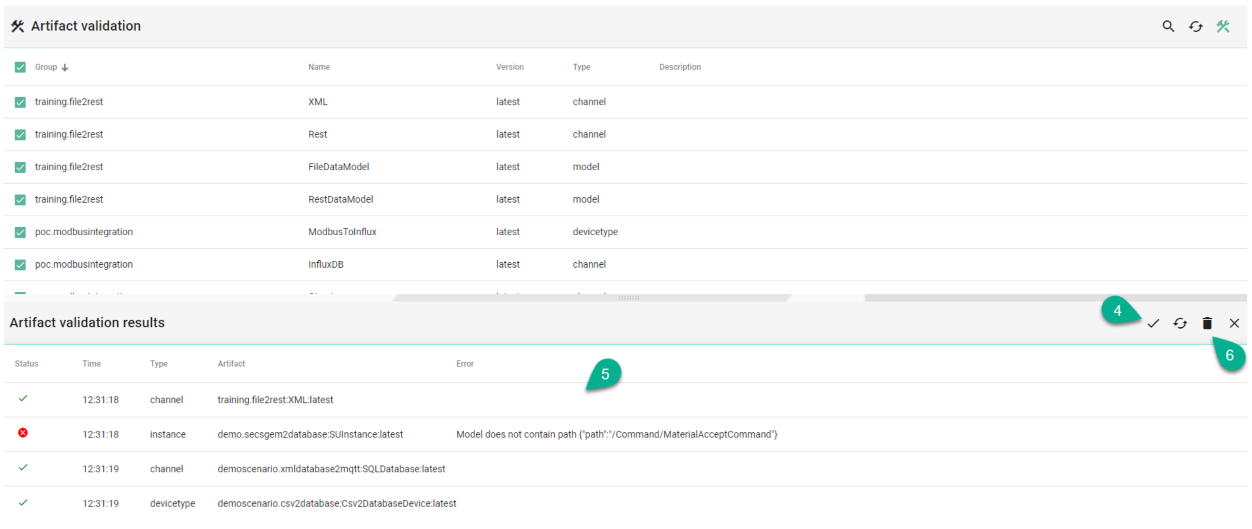
Do you want to select the dependencies?

Yes No

Artifact validation results

Status	Time	Type	Artifact	Error

- Make sure the **Artifact Validation Results** view is opened and click on the **Show valid** button **(4)** to see the results **(5)**



Group	Name	Version	Type	Description
training.file2rest	XML	latest	channel	
training.file2rest	Rest	latest	channel	
training.file2rest	FileDataModel	latest	model	
training.file2rest	RestDataModel	latest	model	
poc.modbusintegration	ModbusToInflux	latest	devicetype	
poc.modbusintegration	InfluxDB	latest	channel	

Status	Time	Type	Artifact	Error
✓	12:31:18	channel	training.file2rest:XML:latest	
✗	12:31:18	instance	demo.secsgem2database:SUInstance:latest	Model does not contain path {"path":"/Command/MaterialAcceptCommand"}
✓	12:31:19	channel	demoscenario.xmldatabase2mqtt:SQLDatabase:latest	
✓	12:31:19	devicetype	demoscenario.csv2database:Csv2DatabaseDevice:latest	

- If an artifact is valid, the **Status** column will show a green valid icon
- If an artifact is NOT valid, the **Status** column will show a red **X** icon and the **Error** column will provide details
- To delete the validation results, click on the **Clear** button **(6)**

CHAPTER SIX

GETTING HELP

Having trouble? We would like to help!

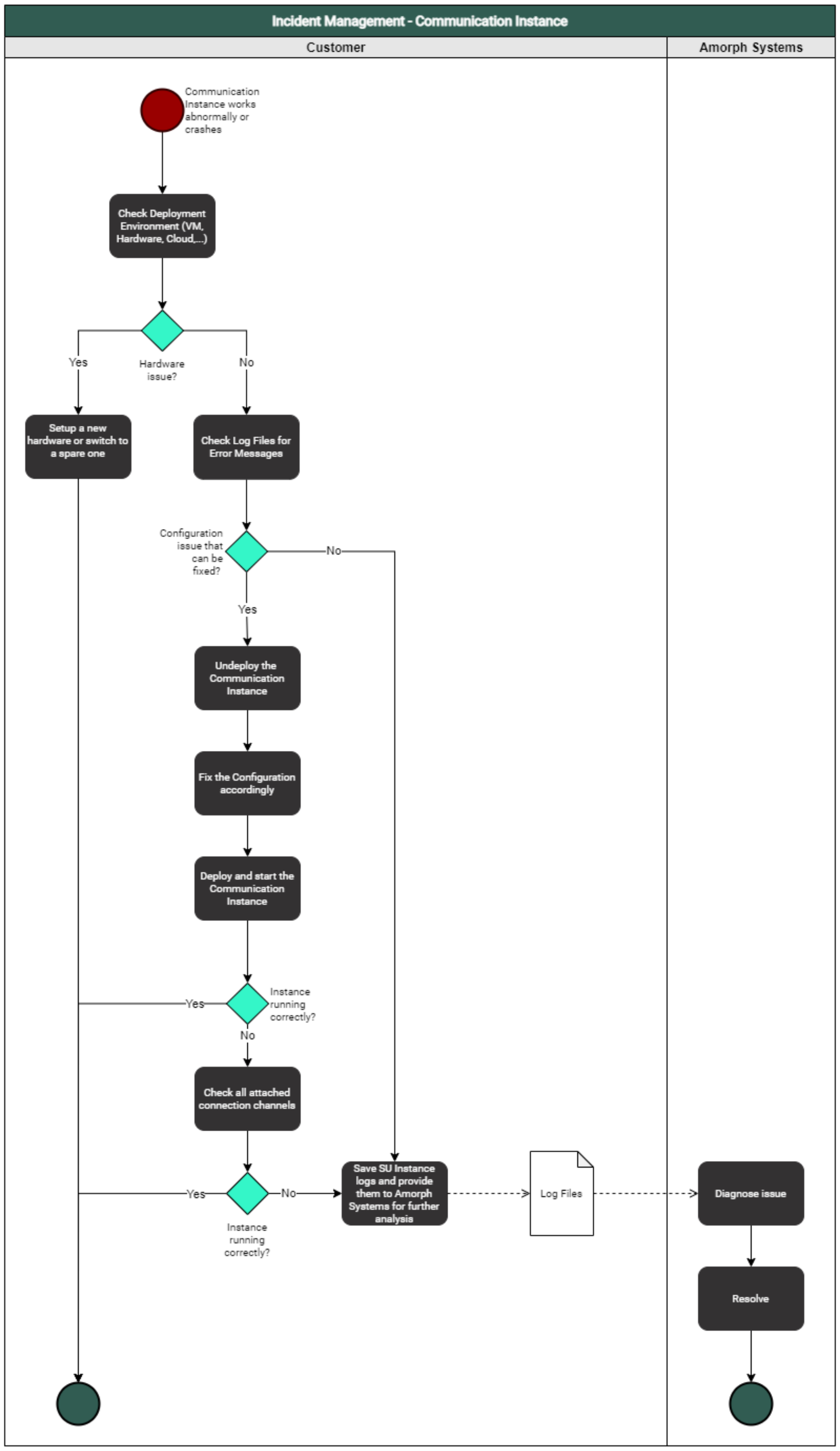
- In case of malfunctioning SU Instances check out the *Troubleshooting* section
- Try the *FAQ* - it's got answers to regularly asked questions
- Check out the *Glossary* if some terminology is not clear

6.1 Troubleshooting

6.1.1 Communication Instances

Determine if there is an issue with the deployment environment (VM, Cloud, other Hardware) where the Communication *Instance* is operated on.

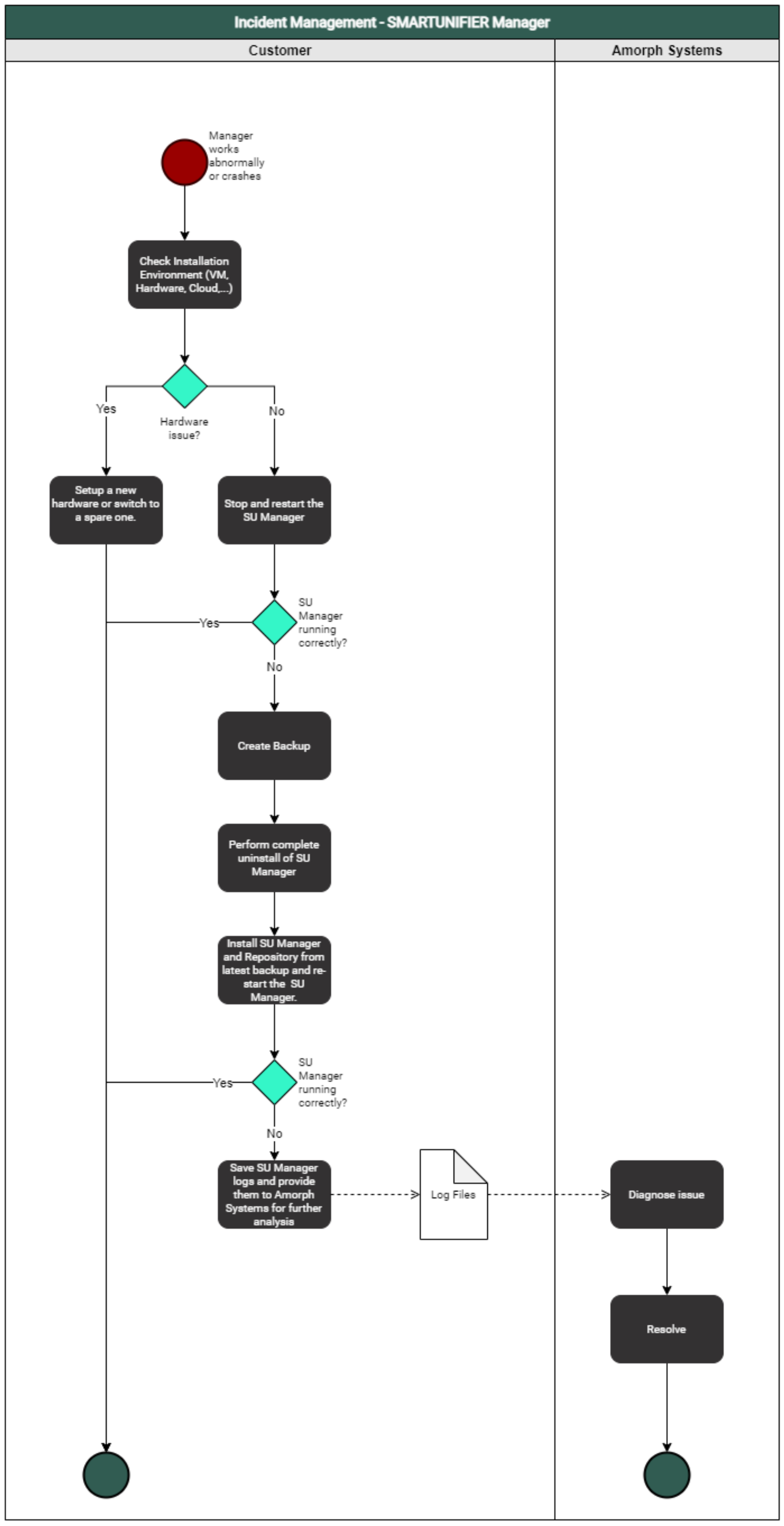
- In case of a HW problem setup a new HW (or switch to a spare HW). Ensure to place the correct security certificates on the new HW. Perform a new *deployment* of a new SU Instance with *SU Manager* on the new HW.
- In case, the HW is operating correctly navigate to the log file of the deployed instance **./SmartUnifierManager/deploy/<deployment-id>** and check for error messages.
  - If there is a configuration issue which can be fixed:
    - Undeploy the Communication Instance
    - Fix the configuration issue accordingly
    - *Deploy* and start the Communication Instance
  - If there is a configuration issue which can not be fixed save the log files and contact Amorph Systems through the Support Portal for further assistance


Incident Management - Communication Instance
Customer
Amorph Systems
Communication Instance works abnormally or crashes
Check Deployment Environment (VM, Hardware, Cloud,...)
Hardware issue?
Yes
No
Setup a new hardware or switch to a spare one
Check Log Files for Error Messages
Configuration issue that can be fixed?
No
Yes
Undeploy the Communication Instance
Fix the Configuration accordingly
Deploy and start the Communication Instance
Instance running correctly?
Yes
No
Check all attached connection channels
Instance running correctly?
Yes
No
Save SU Instance logs and provide them to Amorph Systems for further analysis
Log Files
Diagnose issue
Resolve

6.1.2 SMARTUNIFIER Manager

Determine if it is a HW problem on the HW where SMART**UNIFIER** Manager is operated.

- In case of a HW problem, setup a new HW or switch to a spare HW. Perform installation of SU Manager and Repository from latest backup and re-start the Manager on the new HW.
- In case HW is operating correctly stop and restart the Manager
- If the Manager is still not running correctly:
  - Create a Backup
  - Perform a complete uninstall of the Manager
  - Install the Manager with the Repository from the latest backup and start the Manager
- If the Manager is still not working navigate to **./SmartUnifierManager/log** and save the log files (debug.log and info.log) and contact Amorph Systems through the Support Portal for further assistance


Incident Management - SMARTUNIFIER Manager
Customer
Amorph Systems
Manager works abnormally or crashes
Check Installation Environment (VM, Hardware, Cloud,...)
Hardware issue?
Yes
No
Setup a new hardware or switch to a spare one.
Stop and restart the SU Manager
SU Manager running correctly?
Yes
No
Create Backup
Perform complete uninstall of SU Manager
Install SU Manager and Repository from latest backup and re-start the SU Manager.
SU Manager running correctly?
Yes
No
Save SU Manager logs and provide them to Amorph Systems for further analysis
Log Files
Diagnose issue
Resolve

6.2 FAQ

Does SMARTUNIFIER provide caching/buffering of data?

Yes, SMART**UNIFIER** is capable of supporting caching of messages using file buffer (Spool) for message transfer to external middleware like MQTT. This functionality can be provided as part of a SMART**UNIFIER** Communication Channel and dependent on the used communication protocol of the respective channel.

Is it possible to set different buffering options for different channels?

Yes, each communication channel of SMART**UNIFIER** can provide a different buffer size and further options.

Does SMARTUNIFIER enable data pre-processing, cleansing, filtering and optimization of data?

Yes, this is a core feature of SMART**UNIFIER**. SMART**UNIFIER** provides powerful capabilities for any kind data preprocessing, cleansing, filtering and optimization. The capabilities of SMART**UNIFIER** in this respect range from simple calculations, unit conversions, type conversions and reformatting up to arbitrary processing algorithms of any complexity.

Does SMARTUNIFIER enable data aggregation?

Yes, SMART**UNIFIER** enables data aggregation and reformatting with any level of complexity.

Does SMARTUNIFIER provide short term data historian features?

Yes, historic telemetric data (of variable time horizons; size limited by used HW) can be monitored by usage of SMART**UNIFIER**'s logs which can record all communication activities of a SMART**UNIFIER** Instance incl. telemetric data. SMART**UNIFIER**'s Log data can afterwards be forwarded by usage of a dedicated Communication Channel to any (and also multiple) upper-level monitoring or analytics system. Alternatively SMART**UNIFIER**'s Logs can be accessed directly by any external IT application (remote access to HW device is required).

Yes, SMART**UNIFIER** can create any number of OPC-UA Servers and/or Clients within just one Communication Instance.

Does SMARTUNIFIER support standard number of connections to OPC-UA Clients?

Yes, SMART**UNIFIER** supports a virtually unlimited number of client connections per OPC-UA Server. Physically the number of connections is limited by number of subscriptions per session, number of data objects and size per subscription as well as by HW and network constraints. SMART**UNIFIER** allows to operate multiple OPC-UA Servers and/or OPC-UA Clients within each single SMART**UNIFIER** instance for northbound and/or southbound communication.

Does SMARTUNIFIER support brokering to MQTT Server?

Yes, SMART**UNIFIER** supports any number of MQTT connections. One single SMART**UNIFIER** Instance can connect to one or multiple MQTT brokers (e.g., for different target systems) and is able to communicate bi-directional.

Which southbound protocols are offered with SMARTUNIFIER?

SMART**UNIFIER** supports many protocols like e.g.,

- Siemens S7, S7-2
- OPC-UA
- Beckhoff
- MQTT
- Modbus-TCP
- file-based (different formats like CSV, XML, JSON, any binary format)
- SQL

. . . and many more to come continuously. Specific protocols can be provided based on customer request. Therefore please contact Amorph Systems (www.amorphsys.com).

Does SMARTUNIFIER enable pre-aggregation of additional sensor data and/or more devices (rule based), for e.g., temperature monitoring?

Yes, SMART**UNIFIER** allows to connect any number of telemetric data sources to a SMART**UNIFIER** Instance. Rule-based pre-aggregation and pre-processing of additional sensor data is supported with any level of complexity. This ranges from simple pre aggregation/pre-processing up to complex utilization of advanced AI or ML algorithms.

Does SMARTUNIFIER support processing of active cloud commands? (e.g., System Manager AWS / AWS Agent)

Yes, SMART**UNIFIER** provides a RESTful API to execute Shell Commands (e.g., Start/Stop Instance, etc.). Thus, active cloud commands are supported. In addition, also commands from other external IT-Systems (e.g., MES, ERP, AWS Systems Manager etc.) are possible. Furthermore if required SMART**UNIFIER** can be fully executed and operated within Cloud Environments (e.g., within AWS Cloud).

Which northbound protocols are supported by SMARTUNIFIER?

SMART**UNIFIER** supports many northbound protocols, like e.g.,

- OPC-UA
- MQTT
- WebSphere
- HTTP / REST
- any file based protocol
- SQL/any database
- Splunk
- Vantiq

. . . and many more to come continuously. Specific protocols can be provided based on customer request. Therefore, please contact Amorph Systems (www.amorphsys.com).

Does SMARTUNIFIER support international naming standards (example: EUROMAP 77, PackML)?

Yes, SMART**UNIFIER** is specifically designed to strongly support the incorporation of international standards (e.g., EUROMAP 77, 82, 83, 84, AutomationML, PackML, DFQ, SEMI SECS/GEM etc.) as well as company standards, by offering the capability to be able to build up specific SMART**UNIFIER** Information Models complying with these standards and incorporating full data semantics. There will be a one-time effort to implement such a standard in SMART**UNIFIER** as a respective Information Model and afterwards this Standard can be used for any communication across the whole customer IT Infrastructure. Also this includes flexible mapping from legacy protocols to new standard protocol and vice versa.

Does SMARTUNIFIER offer the ability to integrate with other systems and applications through REST Server APIs and Web Services for Operational purpose?

Yes, SMART**UNIFIER** features a REST API for operational purpose (e.g., instance start/stop service, configuration etc.)

Does SMARTUNIFIER offer a way to realize a flexible, configurable dataflow?

Yes, SMART**UNIFIER** features a configurable and highly performant rule-based engine (SmartMappings) based on different northbound and/or southbound input sources for realizing any dataflow (workflow) that is required in industrial environments. This covers communication sequences for identification, processing start, processing execution, processing end, results data pro-vision as well as detailed process data provision. Also commands from any upper-level IT-System can be processed and further transmitted to the production equipment (e.g., recipe management, NC program transfer etc.) External data flow engines / visualization apps (e.g., Node-Red, Grafana) can be connected.

Does SMARTUNIFIER enable Central Software Management?

Yes, all Information Models, Mappings and Deployment Features can be managed centrally. Furthermore, SMART**UNIFIER** features an easy to use REST API for operational purpose (e.g., instance start/stop service, configuration etc.).

Does SMARTUNIFIER enable Container Deployment?

Yes, SMART**UNIFIER** operation and deployment is fully based on Container-Technology (Docker). SMART**UNIFIER** Manager and Instances can be operated and deployed inside Docker Containers to any End Point within the network running Docker environment.

Which Operating System SMARTUNIFIER is supporting?

SMART**UNIFIER** runs on Windows, Linux, Mac and other OS supporting Java RT and Docker.

Does SMARTUNIFIER support onPrem Edge-Analytics?

Yes, SMART**UNIFIER** can be connected to any Edge-Analytics System SMART**UNIFIER** Logs can provide detailed information about all communication activities. These log data can either be provided by a dedicated Communication Channel to any upper level Analytics System (in any required format) or can be made locally accessible to any agent running locally on the HW.

Does SMARTUNIFIER support DevOps CI/CD Pipeline for installations and update?

Yes, SMART**UNIFIER** supports remote installation/update of Software from SMART**UNIFIER** Manager via Docker Registry SMART**UNIFIER** Instances (running in Docker Containers) can be updated, monitored and controlled remotely. Docker registry is also accessible from external systems if required.

Does SMARTUNIFIER enable Software Scalability?

Yes, SMART**UNIFIER** can scale from connection of one single equipment/device to virtually any number of equipment/devices by means of its decentralized architecture.

Does SMARTUNIFIER support the architecture of distributed systems?

Yes, SMART**UNIFIER** itself is a fully distributed and scalable IT system. With this architecture SMART**UNIFIER** is able to collaborate in any small or large IT environment. SMART**UNIFIER** is open to reliably collaborate in large sites.

Does SMARTUNIFIER provide the ability to directly communicate with other Devices or IT-Systems through standard protocols and also supports Load-Balancing?

Yes, SMART**UNIFIER** can communicate with any other Devices or IT-Systems and also address load balancers for optimized feeding of data to any message brokers or data forwarder.

Does SMARTUNIFIER provide the ability for data to be ingested as a consolidated batch (File Transfer)?

Yes, SMART**UNIFIER** can use any file in any format as input source and also as output destination.

Does SMARTUNIFIER provide the ability to create custom connectors to ingest data from arbitrary sources?

Yes, the capability to be able to realize custom connectors for any data source is one of the core elements of SMART**UNIFIER**‘s architecture.

Is SMARTUNIFIER able to push operational data to an Edge-Gateway?

Yes, SMART**UNIFIER** can receive operational data from any device or IT-System and push it to an Edge-GW. E.g., OPC-UA, MQTT and HTTP/REST are supported. Also, many other protocols can be used therefore.

Does SMARTUNIFIER provide Software Monitoring?

Yes, each SMART**UNIFIER** Instance creates detailed logs that document every communication activity. These logs can be made accessible to any external system e.g., by a dedicated monitoring communication channel. Moreover, SMART**UNIFIER** Manager comes with a built-in Monitoring Dashboard that allows monitoring of the distributed SMART**UNIFIER** Instances.

Does SMARTUNIFIER support Monitoring integration?

Yes, this is possible; Each SMART**UNIFIER** Instance creates detailed logs that document every communication activity. These logs can be made accessible to any external system e.g., by a dedicated monitoring communication channel. In addition, SMART**UNIFIER** is able to send any kind of monitoring message (e.g., based on status changes or other events (e.g., time triggered) to any (or multiple) upper level monitoring system in any required format.

Does SMARTUNIFIER provide certificate handling?

Yes, SMART**UNIFIER** can handle certificates and establish state-of-the-art secured connections (e.g., TLS, secured MQTT, secured OPC-UA, etc.).

Is it possible with SMARTUNIFIER to limit access to data?

Yes, SMART**UNIFIER** Instances work on independent Windows/Linux computer units. Data may be stored temporarily on these HW devices as logs or for buffer (cache) purposes. This temporary data can be protected by assigning the HW with appropriate access rights and user roles.

Does SMARTUNIFIER support services for security supervision and security monitoring?

Yes, SMART**UNIFIER** creates detailed logs regarding all communication activities (and other activities) it performs. With SMART**UNIFIER** it is possible to integrate with any external security supervision/monitoring system (e.g., Splunk) and provide on-line log files (in any required format) to these systems by usage of a dedicated monitoring communication channel.

Does SMARTUNIFIER support End-to-End transport encryption (to Northbound and Southbound)?

Yes, SMART**UNIFIER** can support End-to-End transport encryption for southbound and northbound communication channels.

Does SMARTUNIFIER enforce secure individual authentication for all users?

Yes, SMART**UNIFIER** supports individual user authentication.

Does SMARTUNIFIER support Windows Active Directory (AD)?

Yes, SMART**UNIFIER** supports *Windows Active Directory*.

Does SMARTUNIFIER support a (configurable) secure remote access?

Yes, Secure remote access to SMART**UNIFIER** Manager and SMART**UNIFIER** Instances is possible by standard Windows or Linux tools (e.g., SSH).

Can SMARTUNIFIER protect unsecured Shop Floor devices from office network through isolation?

Yes, a SMART**UNIFIER** Instance can be deployed locally near an equipment/device and map any unsecured equipment/device interface into a secured protocol (e.g., OPC-UA, MQTT). This way "unsecured data streams" coming from an equipment/device can be transferred to any northbound system in a secured way (isolation of the equipment/device). The same principle can be also applied when sending control parameters (e.g., screwer params, NC programs, recipes, ...) or commands from a northbound system to the equipment/device.

Does SMARTUNIFIER support malware protection concepts (e.g., support of standard Anti-Virus Software)?

Yes, SMART**UNIFIER** works with any standard malware protection software incl. McAffee, NOD and many others.

Is SMARTUNIFIER secure by design (e.g., secure coding guidelines, use of open source code, pen-testing)?

SMART**UNIFIER** was developed according to state-of-the-art coding principles and on request we are willing to let perform any checks, verifications, pen testing as required to validate the software. Especially for realizing communication channels and implementing protocols, state-of-the-art Open Source Libraries are used and constantly updated to the newest versions available.

Does SMARTUNIFIER support a range of transmission/infrastructure protocols (e.g., IPV4/IPv6)?

Yes, with SMART**UNIFIER** (depending on used HW) IP4/IP6 are supported.

- LAN: Up to 4x Gbit Ethernet Intel i211
- Wireless LAN: 802.11ac dual antenna + BT 4.2
- Cellular communication: LTE/WCDMA/GSM/GNSS

USB: Up to 8 ports, 2x USB 3.0, Up to 6x USB 2.0

- RS232 serial port

Also other transmission/infrastructure protocols can be supported on request but may require additional HW.

Does SMARTUNIFIER provide the ability to handle intermittent connectivity of sources (data/event redelivery and failure modes)?

Yes, intermittent connectivity of sources can be handled by SMART**UNIFIER** Communication Channels. Based on rules, data/event redelivery can take place, failure modes can be activated, and escalation procedures to northbound systems can be triggered.

Does SMARTUNIFIER reduce unnecessary traffic on shop floor network to protect device interfaces from traffic overload?

Yes, a SMART**UNIFIER** instance can be deployed locally nearby the equipment on any suitable HW device. The SMART**UNIFIER** instance can then be configured to communicate to the connected southbound equipment/devices by using a separate physical network port and this way isolate the device from unnecessary traffic coming from the northbound network.

Does SMARTUNIFIER support low Latency between Southbound and Northbound Interfaces?

Yes, SMART**UNIFIER** provides high performance / low latency by its distributed architecture consisting out of small SMART**UNIFIER** Instances (i.e., no central bottlenecks like e.g., a middleware broker/database). Furthermore, SMART**UNIFIER** features an integrated compiler that creates native Bytecode for the interfaces to be executed within the SMART**UNIFIER** Instances. This makes the SMART**UNIFIER** highly performant, since no slow scripting language nor any slow interpreter is used to provide the connectivity functionality.

Is it possible with SMARTUNIFIER to ensure a consistent setting of time stamps for events (NTP)?

Yes, this is possible.

Is it possible to use UNICODE for operational data?

Yes, it is possible to use UNICODE with SMART**UNIFIER** (e.g., for OPC-UA).

Is stability of SMARTUNIFIER s API given? Is the API stable across releases?

Yes, SMART**UNIFIER** is a standard product from Amorph Systems. Interface stability is given and stable across new product releases. Furthermore, interfaces are versioned and under controlled release management (i.e., different versions of interface, Information, Models and Mappings can be maintained and deployed in a controlled mode).

Which tools for development, deployment and error analysis can be used with SMARTUNIFIER ?

For extension, deployment and error analysis of SMART**UNIFIER** (e.g., development of new Information Models, pre-processing, aggregation etc.) widely-used and accepted state-of-the-art development environments and powerful standard tools may be used, e.g., Eclipse, Maven/sbt, Jenkins, Docker. For Error Analyses detailed logs created by SMART**UNIFIER** can be used and analysed with any analytics tools.

What is the cost model of SMARTUNIFIER ?

Please refer to Amorph Systems (www.amorphsys.com) for prices for SMART**UNIFIER** . In general, the following policies apply:

- SMART**UNIFIER** Manager is free of charge
- For SMART**UNIFIER** Instances a yearly license fee is charged

Does Amorph Systems offer reliable support for SMARTUNIFIER ?

For many years, Amorph Systems is providing first class support and intensive care to all of its customers. This covers all products and solutions that were delivered and operated in Industrial Areas as well as in Air Traffic Industry. For customer references please refer to Amorph Systems (www.amorphsys.com).

What support levels (SLAs) are supported?

Different levels of services (8x5, 8x7 up to 24x7) are available upon request from Amorph Systems (www.amorphsys.com).

Does SMARTUNIFIER support multiple languages?

Yes, SMART**UNIFIER** can support multiple languages. Currently the GUI is available in English and German language. In case more languages are required, please contact Amorph Systems (www.amorphsys.com)

Does Amorph Systems provide relevant training capabilities for operating SMARTUNIFIER and for engineering of Information Models and Mappings?

Yes, SMART**UNIFIER** is a simple to use standard product and was specifically designed as a powerful tool to enable the end customers themselves to provide seamless equipment/device as well as IT-Systems interconnectivity within their industrial environments.

Therefore, Amorph Systems trains customers to configure, deploy and operate SMART**UNIFIER** in their environments. Moreover, we can give advanced trainings, so that the customers can also implement new interfaces, new channels, new, Information Models and new Mappings on their own.

6.3 Glossary

Arrays
An Array (as an Information Model Node Type) is a container object that holds a fixed number of values of a single type.

Configuration Components
Configuration Components are Information Models, Communication Channels, Mappings, Device Type and Communication Instances, used to realize an integration scenario.

Commands
Commands (as an Information Model Node Type) are functions like the methods of a class in object-oriented programming. The scope of a Command is bounded to the Information Model it belongs.

Communication Channels
Communication Channels or simply Channels, refer to a transmission medium. A Channel is used to convey information from one or several senders (or transmitters). Communicating data from one location to another requires some form of pathway or medium. These pathways are called Communication Channels, and the information is transmitted with the help of communication protocols. Each Information Model has one or many Channels, and each Information Model can choose which Channel it subscribes to. The information is transmitted through the Communication Channels in both directions: from the external system to the SMART**UNIFIER** application and vice versa.

Custom Types
Custom Data types are defined by the user for a Node Type.

Data Types
Each Node Type has a Data Type. Data Types can be either a Simple Type or a Custom Type - depending on the Node Type.

Deployments
With the SMART**UNIFIER** Deployment capability Instances can be deployed to any IT resource (e.g., Equipment PC, Server, Cloud) suitable to execute a SMART**UNIFIER** Instance.

Deployment Endpoints
Deployment Endpoints are used to identify the location of a Deployment (e.g., AWS Fargate, Docker)

Device Types
Device Type contains one or multiple Mappings. Each Mapping contains one or multiple Information Models and its associated Communication Channel. Within a SMART**UNIFIER** Device Type it is possible to overwrite existing Communication Channel configurations. Device Types are especially important, when having to connect several similar equipment or devices that share the same communication parameters. In these cases it is sufficient to define only one Device Type and the settings of this Device Type can be reused across all Instances.

Events
Events (as an Information Model Node Type) represent an action or occurrence recognized by SMART**UNIFIER**, often originating asynchronously from an external data source (e.g., equipment, device). Computer events can be generated or triggered by external IT systems (e.g., received via a Communication Channel), by the SMART**UNIFIER** itself (e.g., timer event) or in other ways (e.g., time triggered event).

File Consumer
This Communication Channels offers the capability to read-in files (e.g., CSV, XML, and JSON). The File Consumer monitors an input folder that is specified in the configuration.

File Tailer
This Communication Channels offers the capability to read-in files (e.g., CSV, XML, and JSON). The File Tailer monitors a specific file, which is specified in the configuration.

InfluxDB
This Channel offers connectivity to an InfluxDB. InfluxDB is an open-source time series database.

Information Models
Information Model describes the communication related data, which is available for a device or IT system. Each device or IT system is represented by an Information Model.

Instances
An Instance represents an application that handles the connectivity. Instances can be deployed to any suitable IT resource (e.g., Equipment PC, Server, Cloud), and provide the connectivity functionality configured. Therefore, a SMART**UNIFIER** Instance uses one or multiple Mappings and selected Communication Channels from a previously defined Device Type.

Lists

A List (as an Information Model Node Type) representing collections of Node Types (e.g., Variables, Properties, Arrays, and other Lists).

Mappings
Mapping represents the SMART**UNIFIER** component that is defining when and how to exchange/transform data between two or multiple Information Models. In other words it is acting as a translator between the different Information Models. One Mapping consists of one or multiple Rules.

MQTT
This Communication Channel offers the capability to send data using the messaging protocol MQTT. MQTT is a lightweight publish/subscribe messaging transport for connecting remote devices with minimal network bandwidth.

Node Types
Node Types are elements within an Information Model. Node Types are Variables, Properties, Events, Commands and also collections such as Arrays and Lists. Each Node Type has a Data Type that defines if the Node Type is a predefined Data Type or a custom Data Type.

OPC-UA
Is a machine to machine communication protocol for industrial automation.

Predefined Types
Predefined Data Types (e.g., String, Integer, Double, etc.) are available for the definition types - Variables, Properties, Arrays, Lists (e.g., String, Integer, Boolean).

Properties
Properties (as an Information Model Node Type) are used to represent XML attributes.

REST Client
This Communication Channels offers the capability to consume and operate with resources exposed by REST Servers.

REST Server
This Communication Channels offers the capability to provide resources employing the HTTP communication protocol.

Rules
A Rule contains a Trigger that defines when the exchange/transformation takes place and a list of actions that are defining how the exchange/transformation is done.

Manager
The Web application SMART**UNIFIER** Manager is used to create and monitor SMART**UNIFIER** Instances.

Source
In the Mapping sources are Node Types that are mapped to targets.

SQL DB
This Communication Channel offers the capability to connect to a SQL Databases (e.g., MariaDB, SQLServer, PostgreSQL, ORACLE, HSQLDB, and DB2).

Target
In the Mapping targets are Node Types that receive data assigned from a source.

Trigger
The Trigger defines when the exchange/transformation data between two or multiple Information Models takes place.

User Management
User Management allows the administrator to create users accounts, to assign permissions as well as to activate or to deactivate the user accounts.

Variables
Variables (as an Information Model Node Type) are used to represent values.