MORPH.pro SMART**UNIFIER**

SMARTUNIFIER User Manual

Release 1.7.0

Amorph Systems GmbH

September 05, 2022

TABLE OF CONTENTS

1	Abou	t SMAR	TUNIFIER 2
	1.1	What is	SMARTUNIFIER 2
	1.2	What de	oes SMART UNIFIER do
	1.3		ant Use Cases with SMART UNIFIER
		1.3.1	Anything-To-Anywhere IT Interface
		1.3.2	Reusable Interfaces and Interface Models
		1.3.3	Integrate Legacy Equipment
		1.3.4	Implement Fab Communication Scenario
		1.3.5	Provide Base for Remote Maintenance and Health Monitoring
		1.3.6	Migrate to Industry 4.0
		1.3.7	Allow Unlimited Scalability 10
		1.3.8	Enable Internet of Things
	1.4	Connec	tivity Endpoints and Data Formats
		1.4.1	Connectivity Endpoints / Communication Protocols
		1.4.2	Data Formats
2	How	to integ	rate with SMARTUNIFIER 16
	2.1		ation Models
			What are Information Models 16
			How to create a new Information Model
			Node Types
			Data Types
	2.2		inication Channels
		2.2.1	What are Channels
		2.2.2	How to create a new Channel
		2.2.3	Channel Types and Configuration
		2.2.4	General Configurations
	2.3		gs
		2.3.1	What are Mappings
			How to create a new Mapping
			How to create Rules
	2.4		Types
			What are Device Types 88
			How to create a new Device Type
	2.5		inication Instances

		2.5.1	What are Instances
		2.5.2	How to create a new Instance
3	Conf	iguratio	on Component Management 93
	3.1		g Convention
	3.2		Filter
	3.3	_	nent Version Control
		3.3.1	How to release configuration components
	3.4	Operati	ions
		3.4.1	Add
		3.4.2	Edit
		3.4.3	Apply
		3.4.4	Exit Editing
		3.4.5	Save
		3.4.6	Save and Close
		3.4.7	Search
		3.4.8	Sort
		3.4.9	Reload
		3.4.10	Import
		3.4.11	Export
		3.4.12	Clone
		3.4.13	Delete
		3.4.14	Bulk Action
	_		
4	-	oyment	
	4.1		a Deployment
	4.2		Locally
	4.3		with Docker
	4.4		with SSH
	4.5		with AWS Fargate
		4.5.1	Prerequisites
		4.5.2	Architecture
		4.5.3	Planning the Deployment
			Deployment Steps
	4.6		Deploy, Run and Operate a Deployed Instance
		4.6.1	How to Deploy an Instance
		4.6.2	How to Run an Instance
		4.6.3	How to Stop an Instance
		4.6.4	How to Delete a Deployment of an Instance
		4.6.5	How to Un-deploy an Instance
		4.6.6	How to Edit a Deployment of an Instance
	4.7		ations
		4.7.1	How to access Notifications
		4.7.2	How to manage Notifications
	4.8		monitor a deployed Instance
		4.8.1	Log Viewer
		4.8.2	Dashboard
	4.9		nal Options

		4.9.1	Encryption of Communication Instances	. 131
		4.9.2	Protect Communication Instances	. 132
		4.9.3	VM Arguments	. 133
5	Admi	inistrati	ion	135
	5.1		Directory Integration (ADI)	
		5.1.1	AD Group Mapping	
	5.2		and Restore	
		5.2.1	How to access	
		5.2.2	Backup	
		5.2.3	Restore	
		5.2.4	Manager Backup	
	5.3		el Types Manager	
		5.3.1	How to access	
		5.3.2	About Layers	
		5.3.3	How to create a new Channel Type	
	5.4		Java Image Manager	
		5.4.1	How to access	
		5.4.2	Add a New Docker Java Image	
		5.4.3	Edit a Docker Java Image	
		5.4.4	Delete a Docker Java Image	
	5.5	Deplov	ment Endpoints	
		5.5.1	What are Deployment Endpoints	
		5.5.2	How to access	
		5.5.3	Deployment Endpoints Types	
		5.5.4	Deployment Endpoints States	
		5.5.5	Deployment Endpoints Operations	
	5.6		itial Management	
		5.6.1	How to access	
		5.6.2	Add Credentials	
		5.6.3	Edit Credentials	
		5.6.4	Delete Credentials	. 164
		5.6.5	Using Credential Manager when configuring the Communication Channels	
	5.7	User M	anagement	
		5.7.1	About User Management	
		5.7.2	How to access	
		5.7.3	Add a new user	
		5.7.4	Edit a user	
		5.7.5	Delete a user	
	5.8	Loggin	g Configurations	
		5.8.1	How to access	
		5.8.2	Add a new logging file	
		5.8.3	Edit a logging file	
		5.8.4	Delete a logging file	
	5.9		ions	
		5.9.1	OpcUa Model Import	
		5.9.2	JSON Model Import	
		5.9.3	AWS IoT SiteWise Model Export	
			-	

6	Getti	ing Help 188
	6.1	Troubleshooting
		6.1.1 Communication Instances
		6.1.2 SMARTUNIFIER Manager
	6.2	FAQ
	6.3	Glossary

Integrate perfectly your Production-IT using



MORPH.pro SMARTUNIFIER

CHAPTER

ONE

ABOUT SMARTUNIFIER

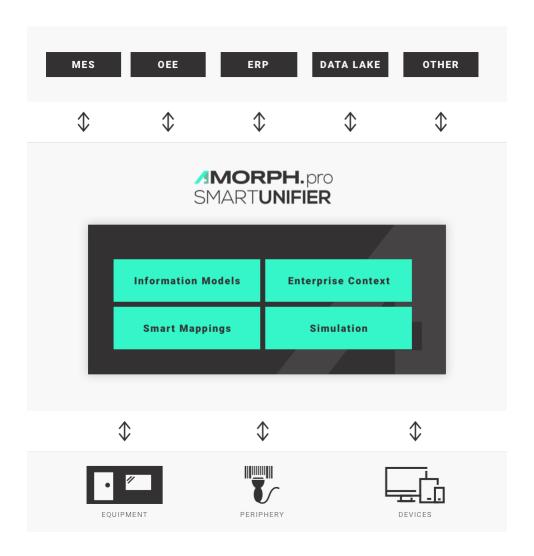
You are new to SMARTUNIFIER?

- Learn about the *SMARTUNIFIER* connectivity platform
- Learn about the connectivity use cases you can address with SMARTUNIFIER
- Check out the supported connectivity endpoints and data formats

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

- SMARTUNIFIER 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 SMARTUNIFIER several Communication Partners can be connected simultaneously.
- With SMARTUNIFIER 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.
- SMARTUNIFIER 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 SMARTUNIFIER 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

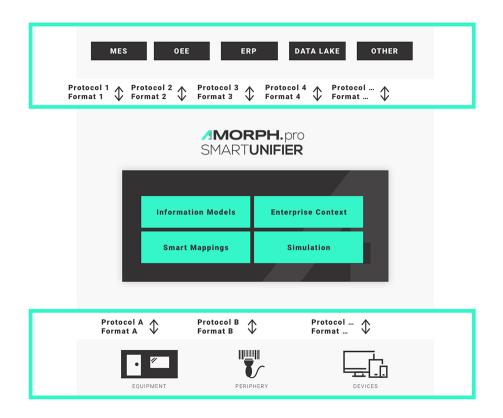
Easy, fast and flexible bi-directional interconnection of multiple IT systems and equipment within a production facility.

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.



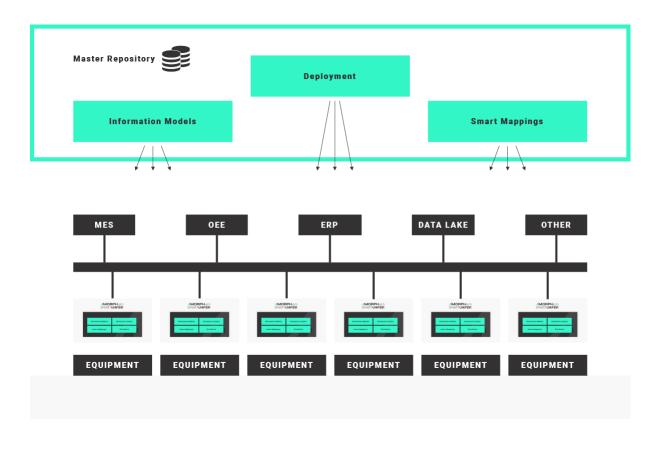
1.3.2 Reusable Interfaces and Interface Models

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.



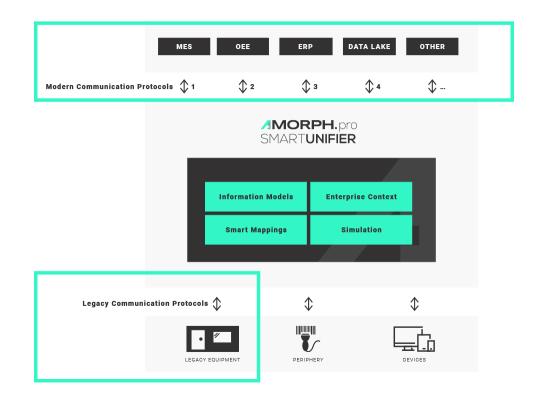
1.3.3 Integrate Legacy Equipment

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 SMARTUNIFIER 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 SMARTUNIFIER 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.

М	IES			DATA LAKI	E
	START 4. FINIS ORDER ORDE		ITY	RESULTING QUALITY DATA	6. DETAILED PROCESS DATA
	MOR SMART	PH. pr UNIFIE	0 2		
Informati	on Models	Enterp	rise Contex	t	
Smart N	Aappings	Sir	nulation		
4	1			•	^
1. ID	2. RELEASE ORDER	3. START ORDER	4. FINISH ORDER	5. RESULTING QUALITY DATA	6. DETAILED PROCESS DATA
ID-READER		EC	QUIPMENT		

1.3.5 Provide Base for Remote Maintenance and Health Monitoring

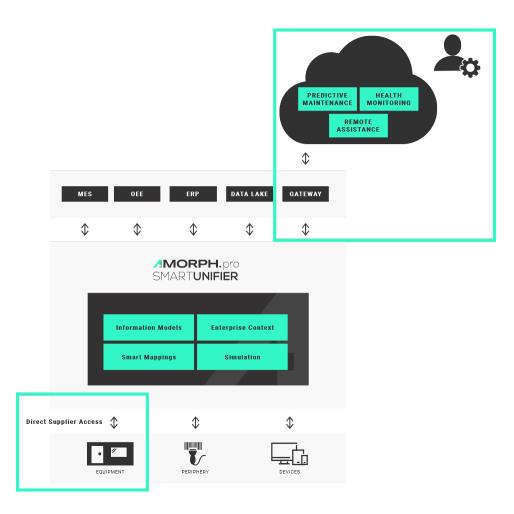
Establish new services and business models by giving secured multi-channel access to equipment and device data in real-time.

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.



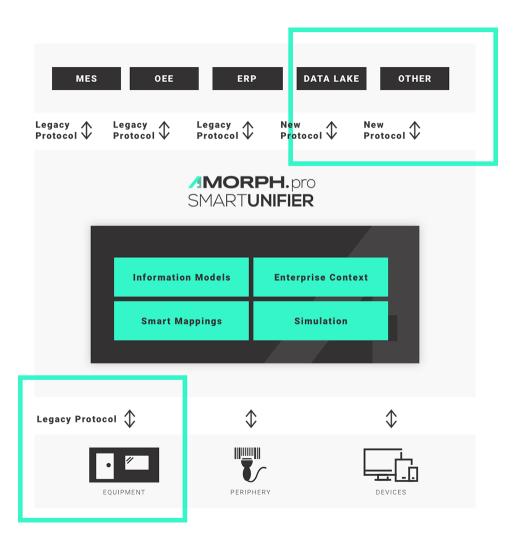
1.3.6 Migrate to Industry 4.0

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.

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

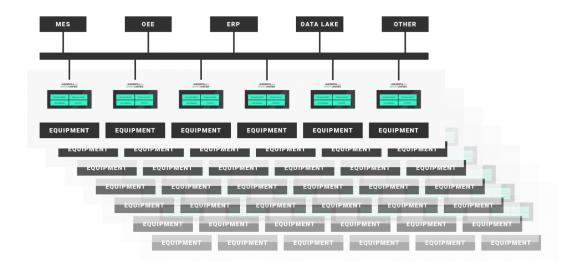
This is because SMART**UNIFIER** is not a traditional middleware having a central limiting message bus. Nor does SMART**UNIFIER** contain any central performance and latency limiting database for providing its communication features.

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 SMARTUNIFIER Instances can be managed centrally:

- central configuration of Information Models and Smart Mappings
- central Operations Monitoring of installed SMARTUNIFIER Instances.

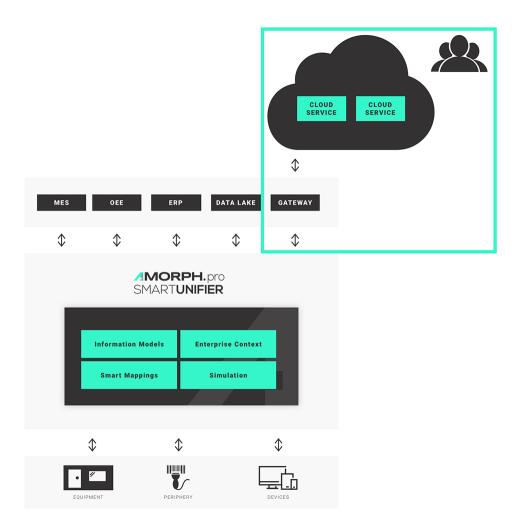
Thus, SMART**UNIFIER** is an essential piece of Industry 4.0 for any manufacturing enterprise – allowing fab-wide and enterprise-wide management of production communication and IT integration infrastructure.



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 SMARTUNIFIER.

Format	Description			
ADLink OpenSplice	Connectivity to ADLink OpenSplice middleware via Data Distribu-			
	tion 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 Ser-	Interface to applications running in AWS ECS			
vice (ECS)				
AWS Elastic Compute	Interface to applications running in AWS EC2			
Cloud (EC2)				
AWS IoT	Interface to AWS IoT			
AWS IoT Greengrass	Interface to AWS IoT Greengrass via MQTT			
AWS IoT Sitewise	Interface to AWS IoT SiteWise 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 identi-			
	fication system)			
Beckhoff	Interface to Beckhoff PLC via Beckhoff OPC-UA Server			
CNC	Connectivity to various CNC controllers (e.g., ABB, Fanuc, Heiden-			
	hain, 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			
НТТР	Send request to HTTP servers			
HTTPS	Send request to HTTPS servers			
InfluxDB	Interface to InfluxDB			
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 com-			
	munication partner			
JDBC	Access databases through SQL and JDBC (refer to SQL Databases)			
	continues on next page			

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
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 applica-
	tions
MQTT	Connectivity by implementing MQTT Client
NoSQL Databases	Cassandra, MongoDB, Hbase
OEE	Interface to various Overall Equipment Efficiency (OEE) Applica-
	tions 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 in-
	stances per SMARTUNIFIER Communication Instances
PLC	Connectivity to various PLCs (e.g., Allen-Bradley, B&R, FANUC,
	General Electric (GE), Hilscher, Honeywell, Krauss Maffei, Mit-
	subishi, Toshiba, Wago) via TCP/IP
PM	Interface to a various Predictive Maintenance Systems that support
1 101	standard communications via e.g., HTTP, REST, SQL
REST	Communication via REST using REST Server / REST Client (Web-
	services)
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 commu-
	nications via e.g., HTTP, REST, SQL
SECS/GEM	Communication with semiconductor or photovoltaic equipment us-
beeb/ dem	ing SECS/GEM interface protocol for equipment-to-host data com-
	munications (TCP/IP).
Siemens Industrial Edge	Deployment of SMARTUNIFIER Communication Instances via
Siemens muustriai Euge	Siemens Industrial Edge Platform
Siemens MindSphere	Interface to MindSphere via REST
(REST)	
Siemens MindSphere (MQTT)	Interface to MindSphere via MQTT
Siemens S7 PLC/TCP	Interface to Siemens S7 1500 / 1200 / 400 / 300 via TCP protocol
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
SOAP	Communication via SOAP (Webservices)

continues on next page

Table 1 – continued from previous page					
Format	Description				
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, Mari-				
	aDB, MSSQL, OracleDB, PostgreSQL, SQLServer and others				
ТСР	Communication from/to any (binary) TCP based protocol				
SFTP	Upload and Download files to/from SFTP servers				
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.

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		

Table 2:	Data	Formats
----------	------	---------

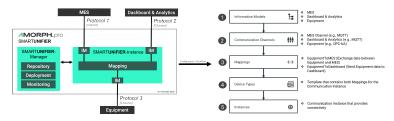
Note: In case a customer requires another data format not listed in the table, please contact Amorph Systems.

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

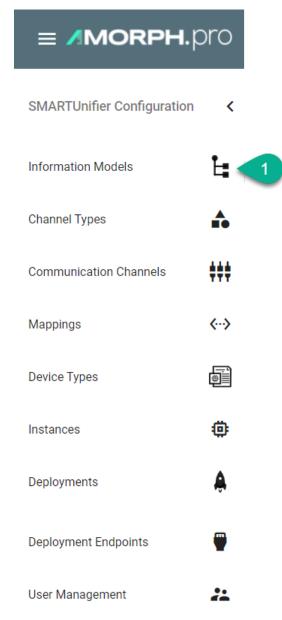
Within the SMART**UNIFIER** an Information Model describes the communication related data that is available for a device or IT system. One device or one IT system therefore is represented by one Information Model. An Information Model consists of so-called *Node Types*. Information Models are build up 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.

The kind of *Node Types* to be used depends on the protocol of the device or IT system. Before creating the Information Model take a look in the chapter *Communication Channels* to see which *Node Types* the Channel you want to use is supporting.

2.1.2 How to create a new Information Model

Follow the steps described below to create an Information Model:

• Select the SMARTUNIFIER 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).

MORPH.pro				SMART UNIFIER 🖯 🔿
La Information Models				Q (±) + 43
> Group 1	Name	Version	Description	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.

MORPH.pro	Smartunifier 🖯 🧿
+ Add Model	4 🖬 🖬 د 🗴
	Group * equipment Name * EquipmentMode(Description

- 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).

+ Add Model				🛡 🖬 🖒 X
M EquipmentM	M EquipmentModel	۹ \$	Group * equipment	
5	+ Add Node →	Variable	Name *	
	Сору	E Event	EquipmentModel	
		Command	Description	
		Array		
		List		
		Property		

2.1.3 Node Types

Model node types are elements within an Information Model. Model node types are 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 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)

+ Add Model		3 🛛 🖬 🖆 🗙
EquipmentModel	Q C MyFirstVariable	
	2 Description Data Type * String	x

Properties

What are Properties

Properties are working similar to *Variables*. Properties can be used for XML attributes when XMLfiles 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)

+ Add Model		3 🛛 🖬 🖆 X
I EquipmentModel ▲	Q C MyFirstProperty	
	2 Description Data Type * String	

Events

What are Events

SMARTUNIFIER is an event-driven software. In this context an event is an action or occurrence recognized by SMARTUNIFIER, often originating asynchronously from an external *data source* (e.g., equipment, device), that may be handled by the SMARTUNIFIER. Computer events can be generated or *triggered* by external IT systems (e.g., received via a *Communication Channel*), by the SMARTUNIFIER itself (e.g., timer event) or in other ways (e.g., time triggered event). Typically, events are handled asynchronously with the program flow. The SMARTUNIFIER 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)

+ Add Model		3 ♥ 🖬 ıb ×
EquipmentModel A	Q C 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)

+ Add Model				B	i占 ×
EquipmentModel	ontTunol MyFirstEvent	<u>a</u> :	ID * MyFirstEvent		_
	+ Add Node →	Variable 4	Description		
	Сору	Array	Data Type *		
	X Cut	List	MyFirstEventType		×
	Remove	Property			

- 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

+ Add Model		7 🛛 🖬 🖆 🗙
EquipmentModel MyFirstEvent[WyFirstEventType] NEWNODE	Q 5 Variable1	B
	Description Data Type * String	×

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 SMARTUNIFIER 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)

+ Add Model	2 🛡 🖬 16 🗙
EquipmentModel	ID * MyFirstCommand Description

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)

Add Model		5 🛛 🗈 🖒 >
EquipmentModel A MyFirstCommand A Parameters A Reply A	C C Description Data Type * MyFirstCommandRequest	x
• Select the Reply no	de from the tree (6)	
• Enter a Data Type	(7)	
Enter a Data TypeClick the "Apply" b		
		<mark>8 ඉ ස</mark> ාර
• Click the "Apply" b		<mark>8 🖉 ව</mark> ැර

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)

+ Add Model				V	占 ×
EquipmentModel G MyFirstCommand G Parameters (MyFirstCommandRec)) Reb) (MyFirstCommandRec))	 Parameters + Add Node ▶ 	9 Array List Property	Description Data Type * MyFirstCommandReply		x

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

+ Add Model		12 🛛 🖬 🖒 🗙
C EquipmentModel G MyFirstCommand O Parameters [MyFirstCommandRequest]	ID * Request1	13
 ■ [NEW NODE] ▲ ③ Reply [MyFirstCommandReply] 	Description	
	Data Type * String	X

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)

+ Add Model		4 🛛 🖬 🖒 🗙
M EquipmentModel ▲ ▲ [NEW NODE][1] ▲	Q MyFirstArray	
	Description Data Type *	x
	3 5 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)

+ Add Model		3 🛛 🗈 🗠 X
EquipmentModel	Q 0 MyFirstList	
	Description Data Type * String	x

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)

+ Add Model		S 🖬 🗗 🗙
EquipmentModel MySimpleVariable [String]	Q C MySimpleVariable	
	Description	
	1 String	×

Туре	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-	Immutable date-time object that represents a date-time, often viewed as year-
Time	month-day-hour-minute-second.
LocalTime	Immutable date-time object that represents a time, often viewed as hour-minute-
	second.
OffsetDate-	Immutable representation of a date-time with an offset.
Time	

Table 1: Predefined Data Types	5
--------------------------------	---

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., "MyFirst-ComplexVariableType" (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 Vi MyFirstComplexVariable[MyFirstComplexVariableType] MyStringVariable [String] ViyHytVariable [Int]	<u>م</u> ټ	ID * MyFirstComplexVariable Description	
		Data Type * MyFirstComplexVariableType	×

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

2.2 Communication Channels

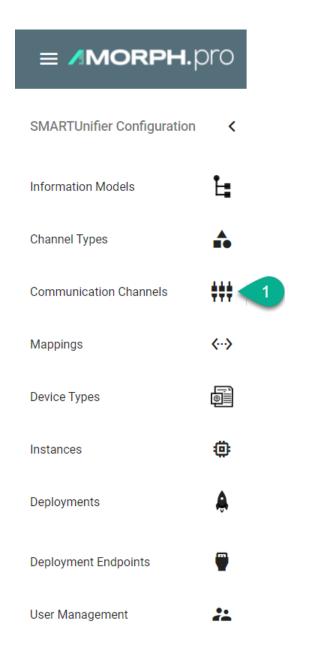
2.2.1 What are Channels

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 SMARTUNIFIER 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)

‡‡‡ Communie	cation C	Channels						Q (4) + 63
Group Filter	<	Group 🛧	Name	Version	Model	De	escription	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

		🖲 (i) 🖬 🖒 🗙
3	Group * demo Name * EquipmentChannel Description	6
4	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: Important to note is that the configuration of a Channel can be overwritten accordingly. For example: The configuration done in the Communication Channel view can be changed in the Device Type or 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

LA Equipment Medel

Supported File Formats:

- CSV
- JSON
- XML

Information Model Requirements

The first Node after the root node [™] must be of type Event [⊑].

CSV

- The node after the Event must be of type *List* 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.

	CSVDemoEvent [CSVDemoEvent]
Pressure, Temperature, Timestamp, PartNr	∧ L csvData [CSVData]
17.5,20,2020.06.11-06:56:31,0001	V Pressure [Double]
18.9,22,2020.06.11-07:56:31,0002	V Temperature [Double]
	V Timestamp [String]
	V PartNr [Int]

XML

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



How to use File Reader with CSV

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

⊕ Add Communication Channel		🍬 🛞 🖬 🖒 X
	Group * demo.scenario1 Name * FileReaderChannel Description Model * demo.EquipmentModel.latest	
1	Channel type *	2

- 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

quipmentModel 3	۹ ۵	
E csvDemoEvent (CsVDemoEventType) C csvData (CsVData) P Pressure (String) Termoerature (String) Termoerature (String) Partivr (String)	4	FileConsumer to String In files * C:\unifier\nput Process folder * C:\unifier\Process Our fold
		Error Folder * C.\Unifier\Error]
		Length * 250
		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	↔ ✓ X
EquipmentModel Q C CevDemoEvent[CSVDemoEventType] CevData [CSVData] Presure [String] Temperature [String] Timesterno [String] PertNr [String]	CSV String to Model Segarator *

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:lat	test	• < ×
EquipmentModel EquipmentModel SovData [CSVDemoEventType] Pressure [String] Tremestare [String] Trestamp [String] V Partive [String]	Q 0	9 FileConsumer to String Enable event EventFileNameFilter File same filter (BegEs) .*
	B	CSV String to Model Exable event CevModelConfiguration Message filter (RegEx)

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 for-	Format of the timestamp	YYYY-MM-DD HH:mm:ss
mat		
ignoreFirstLine	Delay between checks of the file for new con-	true,false
	tent in milliseconds	
TailFromEnd	Set to true to tail from the end of the file, false	true, false
	to tail from the beginning of the file	
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	C:\FileConsumer\Error
	in the implementation	
CharSet	Encoding of the file in use	UTF-8, UTF-8 BOM, etc
ProcessFolder	Regular Expression for the message filter used	C:\FileConsumer\Process
	in the implementation	

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

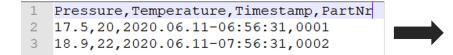
- JSON
- XML

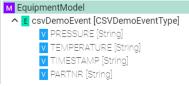
Information Model Requirements

The first Node after the root node [™] must be of type Event [€].

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.





XML

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



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 *	
1	demo.EquipmentModel.latest	2

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

tit Channel Configuration: demo.scenario1:FileTailerC	annel:latest	\leftrightarrow \checkmark X
EquipmentModel CSVDemoEventType] Pressure [String] Tempersture [String] Timestamo [String] PartVr [String]	File Tailer to String Tak* C:XEquipmentFiles:Equipment.csv DurationConfigurationType Length* 20 Unit* Milliseconds	
	Tail from end Reopen between chunks Charser * 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:la	est	↔ ✓ ×
EquipmentModel ScyDemoEvent[CsvDemoEventType] Pressure [String] Timpersture [String] Timestamp (String] PartNr [String]	CSV String to Model	
	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[ype] Pressure[String] Tmestamp[String] Parthr[String]	Q ≎ 7 8	File Tailer to String Enable event File name filter (RegEx)	9
	7 8	CSV String to Model Enable event Message filter (RegEx) 1	

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 for-	Format of the timestamp	YYYY-MM-DD
mat		HH:mm:ss
File	Path to the csv file	C:\test.csv
Delay Millis	Delay between checks of the file for new content in mil-	250
	liseconds	
TailFromEnd	Set to true to tail from the end of the file, false to tail from	true, false
	the beginning of the file	
ReopenBe-	If true, close and reopen the file between reading chunks	true, false
tweenChunks		
routes	Path of a node in the Information Model	true, false
messageFilter-	Regular Expression for the message filter used in the im-	.*
RegEx	plementation	

Databases

InfluxDB

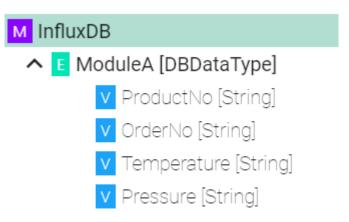
Characteristics - InfluxDB

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

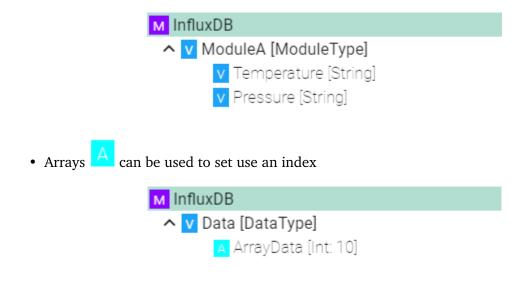
Inserts using Events

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



How to configure InfluxDB

- 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** if data should be written every 10 seconds enter a flush interval of 10000ms

## Channel Configuration: demo:InfluxDB:latest	G	>
InfluxDB Q C InfluxDB Q C <th>InfluxDB Uit * http://127.0.0.18086 Uit * InfluxDB1 Uit definition State * 100 Flush Interval (ms) Length * 100 Uit * 100 Uit</th> <th></th>	InfluxDB Uit * http://127.0.0.18086 Uit * InfluxDB1 Uit definition State * 100 Flush Interval (ms) Length * 100 Uit	

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

tt Channel Configuration: demo:InfluxDB:latest		\leftrightarrow \checkmark	×
InfluxDB I UpdateWeatherData [WeatherDataEvent] 3 I zipCode [String] Coation [String] 1 I coation [String] temperature [Float] 1 I numidity [Float] 1 numidity [Float] MachineData [MachineDataEvent] 1 0 I powerConsumption [Float] 1 1 I numidity [Sloat] 1 1 I numidity [Float] 1 1 I numimes [int: 500] 1 1	WeatherData	text if different from the event name) omma separated: location=New York,street=xxx)	
			_

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

M InfluxDB Q< ↓ ▲ UpdateWeatherData [WeatherDataEvent] InfluxDB ▼ zipCode [String] 5 ▼ location [String] € ▼ temperature [Float] Enable	
V zipCode [String] V location [String] V temperature [Float]	
V location [String] V temperature [Float]	
v temperature (Float)	
humidity (Float) InfluxDbEventContext InfluxDbEventContext InfluxDbEventContext	
v powerConsumption (Float)	
point contain paint () ready runtimes (int 500)	
InfluxDbVariableInEventContext	
 Name (optional if different from variable name) 	
· · · · · · · · · · · · · · · · · · ·	
🗹 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:lates	t	\leftrightarrow \checkmark X
 Channel Configuration: demo:influxDB:lates InfluxDB UpdateWeatherData [WeatherDataEvent] ip:Code [String] iccation [String] iccat	C C C InfluxDB Enable InfluxDbEventContext 8 C Extended configuration InfluxDbVariableInEventContext Name (optional if different from variable name) Temperature I 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 In-fluxDB.
 - (Optional) Enter Tags separated by commas e.g., (location=NewYork, street=xxx)

👯 Channel Configuration: demo:InfluxDB:latest			↔ ✓ ×
 InfluxDB UpdateWeatherData [WeatherDataEvent] zipCode [String] ceation [String] temperature [Float] numidity [Float] MachineData [MachineDataEvent] powerConsumption [Float] runtimes [int 500] 	Q 0	InfluxDB Enable InfluxDbComplexVariableContext InfluxDbArrayInComplexVariableContext Index name (optional, default is 'index') Field name (optional if different from variable name) Tags (optional, comma separated: location=New York,street=xxx)	

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	Interval to write data	10
Measurement	Name of the measurement stored in influxdb	WeatherData
Tag names	Optional tag to be added to the measurement	Type=Station

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* **c** which represent a database table.
- In case of relational databases: Tables which are dependent on each other require a *List* **_**.
- Columns of databases are represented by Variables **∨**.

				M DatabaseModel
* MATERIAL_NR	ORDER_NR	PRODUCT	CUSTOMER	∧ E DatabaseInsert [Database
1 HS787FSTC	121	1 AXY_200	DemoCompany1	V MaterialNr [String]
2 HS787FSTC	123	3 AXY_150	DemoCompany2	V OrderNr [String]
3 HS777FSTC	120	0 AXY_100	DemoCompany1	V Product [String]
4 HS767FSTC	123	3 AXY_200	DemoCompany1	V Customer [String]

Select

- The *Command* defines that after a request is made, a reply with a result is expected.
- Parameters ¹/₉ within a Command represent a collection of query parameter query parameters are defined as Variables [∨].
- Reply [®] within a Command represents the result of the Command results are defined as Variables [♥].

DetaleseMedal

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

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		↔ ✓ ×
SOLServer Solution Solu	SQL Database SQL Database DatabaseConnection Tipe* SQL Server Reconnect interval * 10 Jacc VRL* Jdbc sqBserver/localhost:1433.databaseName=unifier.trustServerCertificate=truel Criedemsta * Username and password	
	Username and password Username * admin Password * 	

Property	Description	Example
Туре	Type of the database	MariaDB, SQLServer, ORACLE, HSQLDB, DB2, PostgreSQL
ReconnectInterval	Time to reconnect if connec- tion 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		\circ \checkmark X
DatabaseModel 3 • ① DatabaseCommand [Command_DatabaseCommand] 3 • ① Parameters [QueryParameters] • ① materialWr [String] • ① orderk [Int] • ⑦ really (QueryReply) • ⑦ readuct [String] • ⑦ coduct [String]	٩ :	SQL Database Custom Query DatabaseCommandContext S0L Gerry* SELECT CUSTOMER_INAME FROM CUSTOMER WHERE ORDER_INUMBER * \$(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		\diamond \checkmark X
DatabaseModel OatabaseCommand [Command_DatabaseCommand] OatabaseCommand [Command_DatabaseCommand] OatabaseViewParameters] orentriv [m] Orentriv [m] orentriv [m] outproduct [String] outproduct [String]	6 🛛 Cust Datab Da Da Coli	Atabase an Ouery aseCommandContext ssign database column abaseCommandVariableColumn mn* ERIAL_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: demoscenario.json2database:SQ	Database:latest	↔ ✓ X
Database DatabaseInsertEvent [DatabaseInsertEventType] Eculomentio [String] OrderVr [String] Materially [String] Ouality [String]	Q C SQL Database 2 ■ Insett DatabaseEventContext Table + SU,DEMO,UCI_TABLE Schema dbo	

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* **∨** with a *Simple Data Type* represents the key-value pairs.
 - *Variables* vith a *Custom Data Type* represent objects that can contain key-value pairs.
 - With *Lists* vou 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.

Add Communication Channel		🖣 🛞 🖬 🖒 🗙
	Group * demo.mgtt	
	Name *	
	Host	
	Description	
	Model *	
	demo.mqtt.MQTT.latest 👻	
a	Channel type *	2
-		
1		2

- 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 *SMAR*-*TUNIFIER 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		\leftrightarrow \checkmark
MATT 3 MatEvent (MatteventType) v eoupmentio (String) v orderkiv (String) materialikvi (String) v oueity (String) v oueity (String)	Q * MQTT to String Hear * 127.0.0.1 Part * 1883 Reconnect interval Length * 5 Umit * Seconds	
	Connection timeout Lengs * 60 Uns * Seconds	
	Keep alive interval Length * 60 Unit * Seconds	v
Channel Configuration: demo.mqtt:Host:latest		↔ ✓
MATT 3 MattEvent [MattEventType] equipment of [String] or orevir (String) materially: [String] cuality [String]	Persistence folder * temp Client ID OGS * QOSO Retained 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		0 V X
MQTT N E MattEvent [MattEventType] 5 V eculamentia (String) V anaterialiki (String) v austerialiki (String) v auslity (String)	Q6	MQTT to String □ Consumers ☑ Producers TopicConfiguration Teple * TopicName

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

### Channel Configuration: demo.mqtt:Host:latest			↔ ✓ X
MOTT V temperature [Double] V temperature [Double] V adumentati [String] V orden/vr [String] V material/vr [String] V auality [String]	Q 0	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.

SON	0	
FileEvent [FileEventType]	MQT	'T to String
v orderNr [String]		onsumers
V materialNr [String] V quality [String]	🗆 P	roducers
	Jsoi	to Model
	2 🛛 🖬	vents configuration
	· · · · · · · · · · · · · · · · · · ·	rents configuration
	Jso Field	ToModelEventConfiguration
	Jso Field	ToModelEventConfiguration

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.

the channel Configuration: doc.mqtt:MQ11:latest		~~~ ×
Mattevent [MatteventType] Mattevent [MatteventType] Gaugementa (String) Gauge Aref (String) Gauge (String)	Q C Json to Model Image: Configuration JsonToModelEventConfiguration Image: Configuration JsonToModelVariableConfiguration JsonToModelVariableConfiguration Field came Image: Configuration Image: Configuration<	

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.

SON	Hostname Verification	
FileEvent [FileEventType] constraint [String] constraint [String] constraint [String]	2 ILS Configuration	
v quality [String]	C:\AWS\AWS_IoT_Certificates\Certificates\Test\AmazonRootCA1.pem	
	3 Client	
	Crt file *	
	C:\AWS\AWS_IoT_Certificates\Certificates\Test\6940074661-certificate.pem.cn	t
	Key file *	
	C:\AWS\AWS_IOT_Certificates\Certificates\Test\6940074661-private.pem.key	
	Client Password	
	None	• •
	Protocol *	
	TLSv1.2	v
	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			↔ ✓ ×
SISON FileEvent [FileEventType] If equipmentic [String] If equipmentic [String] If equipmentic [String] If equipmentic [String] If equipmentic [String]	م ث 2 3 4	Disconnected Buffer Suffer size * So Persist Buffer Delete Oldest Message	

Property	Description	Example
host	URL of the MQTT Broker.	test.
		mosquitto.org
port	Port of the MQTT Broker.	1883
reconnectInter- val	Time interval to reconnect to the MQTT Broker after loss of connection in seconds	5
connection- Timeout	Time interval the connection times out in seconds	60
keepAliveInter- val	Time the session persists in seconds	60
persistence- Folder	Path to a folder for the persistence store of the MQTT	temp
clientId	Identifies an MQTT client which connects to an MQTT Bro- ker	MyClientID
username	Client username	Username
password	Client password	Password
hostnameVeri- fication	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
disconnected- Buffer	Offline buffering of data	true, false
bufferSize	Amount of message allowed in the buffer	5000
persistBuffer	Buffer persistence	true, false
deleteOld- estMessage	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 ⊻ with a Simple Data Type.
 - Variables ⊻ 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	
	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			↔ ✓ ×
M Device 3 ▲ V DemoData [DemoDataType] votrage [int]	<u>م</u> ن 4	Modbus/TCP Client #* tocahost #************************************	
	6	umt - Umt - Milliseconds 	• •
		Length * 5000 Unit * Milliseconds	Ť
	7	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

	↔ ✓ ×
Q Modbus/TCP Client 9 ☑ TCP client connection configuration	
10 ≥ Autorefresh Length * 2 Uent * Sead function code * FCO4	
12 Max update interval Length * 500 Unit * Milliseconds	
	Q C Modbus/TCP Client 9 CP client connection configuration 10 ☑ Autorefresh Length * 2 Unit * Seconds 11 FC04 12 Max update interval Length * Sooil 00 Unit *

- 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, SMARTUNIFIER assumes that the Information Model structure is

in line with the register addresses.

### Channel Configuration: demo.modbus:Modbus:latest			↔ ✓ ×
M Device	۹ ۵	<u> </u>	
		Modbus/TCP Client	
Voltage [int]		✓ TCP client connection configuration	
		14 variables configuration	
		15 Type' DWORD	*
		16 Address	÷

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	Exam-
		ple
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 connec-	5
	tion	
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	FC04
	server	
Max update interval	Minimum time between requests to the Modbus server (if	60
	autorefresh is not used)	
Variable configuration	Format of variable	DWORD
Туре		
Variable configuration	Address of the variable on the modbus server	0
Address		

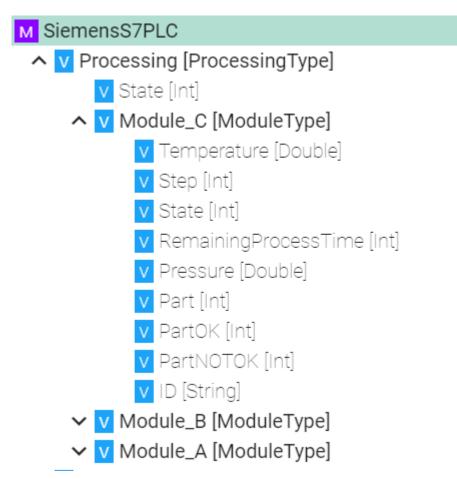
OPC-UA

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 ⊻ 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.

Edit Communication Channel: demo:OPCUA:latest ~	🍆 🕑 🖬 🖒 🗙
Group * demo Name * OPCUA Description Model * demo.PLC.latest Channel type * OPC UA Client 2 2	

- 3. Make sure the root model node is selected to configure the OPC-UA Client
- 4. Enter an Application name
- 5. Configure the serverTcpConfiguration
- Enter an Ip Adress
- Enter the **Port**
- Define an Endpoint
- Set a Request timeout
- 6. Configure the defaultSubscriptionAttribute
- Define a **Publishing interval** and select the **Unit**

3	Q \$	
ProcessingModule [ProcessingModuleType]	OPC-UA Client	
V Temperature (Double)	Application name *	
▼ State [int] ▼ Pressure [Double]	DemoApplication	
V Name [String]	Server Tcp Configuration	
	5 IP Address *	
	127.0.0.1	
	Port *	
	4840	
	Endpoint path *	
	demo	
	Request timeout *	
	5000	
	Subscription configuration	
	6 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		\leftrightarrow \checkmark X
M PLC	Q 0 Monitoring Parameters	
V Temperature [Double] V State [Int]	7 Sampling interval	
V Pressure (Double)	Length *	
V Name [String]	100	
	Unit *	
	Milliseconds	.
	Queue size *	
	1	
	M 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		↔ ✓ X
PLC YocessingModule [ProcessingModule Type] Temperature [Double] State [int] Pressure [Double] Name [String]	Q Image: Subscription groups 8 Subscription Group 9 Group name* Demd Image: Subscription configuration 10 Publishing interval Length * 1000 Umit * Milliseconds	+
	11 Monitoring Parameters Sampling interval Length * 100 Unit * Milliseconds Queue size * 1 13 ≥ Discard oldest	

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

### Channel Configuration: demo.opcua:OPCU	JA:latest		↔ ✓ ×
PLC ProcessingModule [ProcessingModuleType] Temperature [Double] State [int] Pressure [Double] Name [String]	<u>(</u>) (14)	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		\leftrightarrow \checkmark \times
PLC V Temperature [Double] State [nt] Vressure [Double] State [nt] Vressure [Double] Vressure [Double] Vressure [Double] Vressure [Double]	Q ≎ OPC-UA Client I To Enable node configuration I Nodes configuration Node is -	
	ns=3:s="DB_Processing_Module"."Temperature¶ Subscription group name	

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 inter-	Interval in which Notification Messages are sent	1
val		
Sampling inter-	Sampling interval of monitored items	10
val		
Queue size	Max number of messages stored in the publish	1
	queue	
Node id	Id of the item	s='DB_Processing_Module

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.

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

	↔ ✓ ×
REST Server Path prafix * demo Endpoint ID * localhost Path * 8000 Default content type * text/csv	5
Web App	
	Path prefix * demo Endpoint IP * localiost Port * 800 Default content type * text/csv Urb App

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
DefaultCon-	Is used to indicate the media type	application/json, application/xml,
tentType	of the resource	text/html, text/csv
webapp	Possibility to host an application	true, false

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: demo:RESTClient:latest		↔ ✓ ×
MEESTClient 1 ▲ @ GetSampleData [Command_GetSampleData] . ▲ @ Parameters [ParameterType] . ▲ @ urlVariables [urlVariableType] . ⑧ [Int] . ● ame [String] . ● description [String] .	C C Client Default content type * application/json 3 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. (Optional Headers) Enable the checkbox **Headers** for the configuration of the following fields:
 - Enter the name of the header
 - Enter the value
- 7. (Optional Headers) You can add multiple header entries by clicking the Add button
- 8. (Optional Headers) Headers can be deleted by clicking the **delete** button

## Channel Configuration: doc.filetailer:RestClient:latest	○ ✓ X
RestClient Q GetSampleData [Command_GetSampleData] A @ Parameters [GetSampleDataType] VurVariable gutVariableType] UrVariable gutVariableType] UrVariable gutVariableType] UrVariable gutVariableType] UrVariable gutVariableType] Vortaget [Int] Parameters [String] Vortaget [String] Vortaget [String] Vortaget [String] Society [REST Client Command routes CommandRouteConfiguration URL * http://localhost.8081/api/v1/dataPoint/S(id) Authentication

- 9. (Optional Authentication) Enable the checkbox **Authentication** for the configuration of the following fields:
 - Select the **Type** of the Authentication
 - Enter the Username and Password or select it from the Credentials Manager

### Channel Configuration: doc.filetailer:RestClient:latest		\diamond \checkmark \times
RestClient GetSampleData [Command_GetSampleData]	Q C REST Client	
 Parameters (GetSampleDataType) Q (m)	 Command routes CommandRouteConfiguration <pre>URL * http://localhost.8081/spl/v1/dataPoint/\$(id) tittp:method * eET</pre>	
	Username and password Username * admin Password * 	

- 1. (Optional URL Parameters) Select a custom variable and enable the **command routes** checkbox
- 2. (Optional URL Parameters) 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		• ✓ X
RestClient GetSampleData [Command_GetSampleData] OParameters [GetSampleDataType] v untVariables [urtVariableType] ① : [n] ①	11 🛛 Com	and routes dRouteConfiguration mand routes andComplexVariableContext see ters

Property	Description	Example
URL	URL of the REST API.	http://localhost:8081/api/
		v1/dataPoint/\${id}
HttpMethod	HTTP method for the action performed	GET, POST, PUT
	by the Client.	
HeaderName and	To provide server and client with addi-	Retry-After: 12
Header Value	tional information	
Default Content	Is used to indicate the media type of the	application/json
Туре	resource.	
RouteHeaderConfig-	Headers represent the meta-data asso-	Name, Value
uration	ciated with the API request	
Authentication Type	Type of the Authentication	Basic, Digest, Kerberos, NTLM,
		SPNEGO
Content Type of Pa-	Type of the Parameter	Parameters, Body, Header,
rameter Nodes		None
Field Name	For non-matching Information Model	String
	nodes and API spelling	
WaitTimeoutDura-	Timeout in seconds until request is fail-	10
tion	ing	

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 , Command c or Variable V
- The following Node Types can be used under the Event Node:

- *Variables* ∨ with a *Simple Data Type* represents the key-value pairs.
- *Variables* vith a *Custom Data Type* represent objects that can contain key-value pairs.



Configuration - SECS/GEM Client

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

Add Communication Channel			كا 🖬 🕙 🔷
	Group *		
	demo.secsgem		
	Name *		
	SECS/GEM		
	Description		
	Model *		
	demo.secsgem.Equipment.latest	•	
	Channel type *	* 2	
-	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 CateEvent [DataEventType] CateEvent [String] CateContended to the second	۹ \$	SECS GEM Client		
	5	DataFormats C80 * U4 SPT0 * U4 U4 U4 U4 U4 U4 U4		
	6	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		↔ ✓ ×
Equipment CataEvent [DataEventType] Oressure [String] temperature [String]	T5 Length * 10000 Unit * Milliseconds	•
6	T6 Length * 5000 Unit * Milliseconds	
	T7 Length * 10000	
	Unit * Milliseconds	•
	Leigth * 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		↔ ✓ ×
Equipment CataEvent [DataEventType] Consume [String] Consume [String] Consume [String] Consume [String] Consume [String] Consume [String] Consume Statement Consumers (String) Consumer	Q ○ 7 FrameworkConfiguration Logging	8
	Stateful variable Enable Log Data Stateless variable	
	Enable Log Data Event Enable Enable	
	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			\leftrightarrow \checkmark \times
9	Equipment EdutaEvent[DataEventType] Cressure[String] Veneesture[String]	<u>व</u> ः 10	SECS GEM Client	12
		1	SecsEventContext	

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

	\leftrightarrow \checkmark \times
Q C SECS GEM Client ✓ Events SecsEventContext 14 ✓ variables SecsEvariableContext Type * U1 td (required for events and Svs) So □ Is SV Name (optional for commands)	
	SECS GEM Client C Events SecsPariableContext Type * U Id (required for events and Svs). 56 Is SV

- 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

Property	Description	Example
Ір	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
Т6	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
Туре	Type of variable	U1
Id	Variable Id	V56
Туре	Commands - Type of the message	S2F41
Id	Commands Id	C33
RCMD	Name of command if it is different from the command Id	C1

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 \mathbf{M} can be of type *Event* \mathbf{E} or *Variable* \mathbf{V} .
- The following Node Types can be used under the Event Node or Variable Node:
 - *Variables* **∨** with a *Simple Data Type* represent measurements.
 - *Variables* vith 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

M SiemensS7PLC
Processing [ProcessingType]
V State [Int]
🔨 🗸 Module_C [ModuleType]
V Temperature [Double]
V Step [Int]
V State [Int]
RemainingProcessTime [Int]
V Pressure [Double]
V Part [Int]
V PartOK [Int]
V PartNOTOK [Int]
V ID [String]
🗸 🔽 Module_B [ModuleType]
🗸 🗸 Module_A [ModuleType]

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.de	mo.dashboard:SiteWise:latest ~	🔖 🕀 🖬 🖒 X
	Group * su demo dashboard Name * SiteWise	
•	Description Model * su.demo.dashboard.Analytics.latest Channel type * AWS Sitewise	2

- 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		\leftrightarrow \checkmark \times
M Analytics	Q C AWSSiteWise	4
V OrderNo [String] V ProductNo [String] V StartOrder [Boolean]	Group name * su demo dashboard	
Vuantity [int] Vuantity [int] VuastPartStarted [int] VuastPartOK [int]	Model name + Analytics	
✓ LastPartNOTOK [int] ✓ PercentFailure [int]	Credentials Profile * default	
	AWS Region * eu central-1	

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.

FrameworkConfigura	tion
--------------------	------

Logging	
Stateful variable	
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	
\checkmark	EnableLogging
\checkmark	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

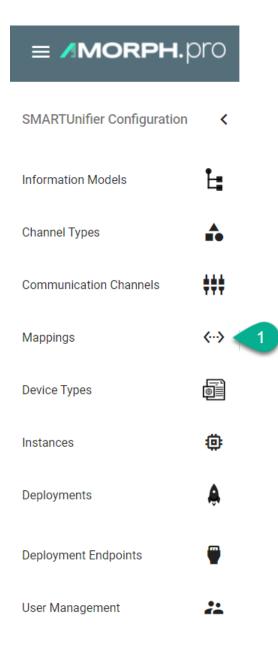
2.3.1 What are Mappings

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)

<··>	Mappings					
H K	Group 🛧	Name	Version	Models	Description	2
UP FILTE						

- 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	3		
CSVToREST			
Description			
	Information model identifier *	4 +	
Models Short name CSV 5	Information model identifier * demoscenario.csv2rest:CsvDataModel:latest	4 + + <u>→</u> ī	
Short name			

2.3.3 How to create Rules

Follow the steps described below to create Rules:

• Select the "Edit" button (1).

>	↔ Mappings				ዲ 🖞 + ይ
Ŀ	Group 🛧	Name	Models	Description	1
***	doc.mapping	DataMapping	DataModel, DataModel		Ċ / ī

- 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.ı	mapping:DataMapping:latest ∽			2	0 B ić	5
Configuration			Rules		Single Rule Multi Rule	(
Group * doc.mapping Name * DataMapping						1
Description						
Models Short name * CsvDataModel	Information model identifier * doc.mapping:CsvFile:latest	+				
Short name * RestServerModel	Information model identifier * doc.mapping:RestServer:latest	· ô				

Graphical

How to create a Rule

Follow the steps described below to create *Rules*:

• Select the "Edit" button (1).

> *	↔ Mappings				Q 🖞 + 😏
Ŀ:	Group 🛧	Name	Models	Description	1
***	doc.mapping	DataMapping	DataModel, DataModel		Ċ / 🔳

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

Edit Mapping: doc.mapping:DataMappir	l g :latest ∽			2
onfiguration			Rules	•
Broup *				
oc.mapping				
ame * DataMapping				
Description				
Models		+		
Short name * Information moto CsvDataModel doc.mappin	del identifier * g:CsvFile:latest	· 0		
Short name * Information more RestServerModel doc.mapping	del identifier * g:RestServer:latest	* 0		

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 includ	es only one Trigger.
-------	------------------------	----------------------

• Enter "Rule name" (3).

	pping:late	st ~				● (⊕ ×
odel svDataModel	÷ 0	Model RestServerModel	. n	Rule Configuration		↔ 💩 🖉 🗙
CsvFile	۹ \$	RestServer	Q 0	Rule name * CsvToRest	Rule description	
V Temperature [Double] V Pressure [Double]		V Temperature [Double] V Pressure [Double]	3			4 Trigger Type 🗶
				Trigger: Tree Member		Tree Member
				Cannot be empty		Fixed Rate Scheduler
				actions [Target <=> Source]		Fixed Delay Schedule

- 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).

^{odel} svDataModel	÷ 🗆	Model RestServerModel	÷ 13	Rule Configuration	0 🙆 💆
CsvFile DemoEvent [DemoDataType] Temperature [Double]	۵ \$	RestServer V DemoData [DemoDataType] V Temperature [Double]	۵ \$	Rule name * CsvToRest R -	ule description
V Pressure [Double]		V Pressure (Double)			Trigger Ty
			2	Trigger: Tree Member CsvDataModel/DemoEvent	
				actions [Target <=> Source]	

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).

aMapping:late	st ~			s (3)
÷ 0	Model RestServerModel	÷ 0	Rule Configuration	↔ 💩 💆 >
۹ \$	RestServer ZernoData [DemoDataType]	٩ \$	Rule name * CsvToRest	Rule description
	V Temperature [Double] V Pressure [Double]			Trigger Typ
		1	Trigger: Cron Expression 0 */5 * ? * *	
			actions Tarnet <=> Source	
	 		Comparison of the set of the	Model Clinic Rule Configuration Q MestServer Q Rule name * Q DemoData [DemoDataType] CorRest Temoerature [Double] Pressure [Double] Trigger: Oran Expression

Fixed Delay Scheduler

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

							· · · · ·	(⊕ ×
63	Model RestServerModel	Ŧ	:3	Rule Configuration			0 🙆	v ×
Q 🗘	M RestServer	٩	\$	Rule name * CsvToRest		Rule description		
	V Temperature [Double] V Temperature [Double] V Pressure [Double]			1	2	3	Trig	ger Type 🖠
				Initial Delay * 2	Period * 5	Unit * SECONDS		
				actions [Target <=> So	urce]			
•		RestServerModel RestServer N @ DemoDataType] Temperature [Double]	RestServerModel RestServer O O C RestServer O C C	Cline RestServerModel Cline Q Image: Cline Q Image: Cline MilestServer Q Image: Cline Q Image: Cline MilestServer Q Image: Cline Image: Cline	RestServerModel C Rule Configuration Q M. RestServer Q No DemoData [DemoData Type] CsvToRest Temperature [Double] Pressure [Double] Pressure [Double] Initial Delay *	RestServerModel C Rele Configuration Q Image: Configuration Rele name * M DemoData [DemoData Type] RestServer Image: Configuration CevToRest Image: Configuration Image: CevToRest Image: CevToRest Image: CevToRest	RestServerModel Ci Rule Configuration Q Temperature [Double] Rule name * CsvToRest Rule description Imitial Delay * Period * Unit * SECONDS	RestServerModel Rule configuration Q Material RestServer No DemoData (DemoData Type) Rule name * Image: Temperature (Double) Pressure (Double) Image: Pressure (Double) Image: Temperature (Double) Image: Temperature (Double) Image: Temperature (Temperature) Image: Temperature (Temperature) Image: Temperature) Image: Temperature (Temperature) Image: Temperature) Image: Temperature (Temperature) Image: Temperature) Image: Temperature) Image: Temperature) Image: Tempera

Actions

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

Edit Mapping: doc.mapping:Da	taMapping:late	st v		♦ (⊙ ×	
Model CsvDataModel	÷ 0	Model RestServerModel	, Π	Rule Configuration	↔ 💩 💆 ×
CsvFile CsvFil	م ¢ 5	V Temperature [Double]	۵ ټ	Rule name * CsvToRest	Rule description //
V Pressure [Double]		V Pressure [Double]			Trigger Type 🏚
				Trigger: Tree Member CsvDataModel/DemoEvent	0
				actions [Target <=> Source]	
			6	∧ V DemoData [DemoDataType]	
				L	

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 no	de's children
Simple	Complex

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).

Model CsvDataModel	÷ 🗆	RestServerModel	÷ 🗆	Rule Configuration	↔ 🙆	
CsvFile	۵ ټ	RestServer	۵ ټ	Rule name * CsvToRest	Rule description	
V Temperature (Double) V Pressure (Double)		V Temperature [Double] V Pressure [Double]			т	rigger Typ
				Trigger: Tree Member CsvDataModel/DemoEvent		
				actions [Target <=> Source]	8	
				RestServerModel/DemoData [DemoDataType	e) Definition Node A E DemoEvent [DemoDataType]	

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).

^{Model} CsvDataModel	÷ 🖸	Model RestServerModel	÷ 0	Rule Configuration	<> 🙆 💆 ×
CsvFile CsvFile Is DemoEvent [DemoDataType] V Temperature [Double] V Pressure [Double]	م ≎	RestServer V DemoData [DemoDataType] V Temperature [Double] V Pressure [Double]	٩ \$	Rule name * CsvToRest	tule description
Lessa e [Donne]		Liezzai e (poppie)		Trigger: Tree Member CsvDataModel/DemoEvent	Trigger Type
				actions [Target <=> Source]	8
					Temperature [Double] Temperature [Double]

- 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.

svDataModel	÷ 🖸	RestServerModel	÷ 🗆	Rule Configuration				<> 🛆	
CsvFile	۹ \$	RestServer V DemoData [DemoData Type] V Temperature [Double]	٩ \$	Rule name * CsvToRest			Rule description		9
V Pressure [Double]		V Pressure [Double]						Trigg	ger Typ
				Trigger: Tree Member CsvDataModel/DemoEvent					
				actions [Target <=> Source]					
				RestServerModel/DemoData/Temperature	[Double] 🥫	;=	CsvDataModel/DemoEvent/Temperature	[Double] <	> 🗖
					[Double]	;=	CsvDataModel/DemoEvent/Pressure	[Double] <	> 🗖

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).

Edit Mapping: doc.mapping:DataMapping:latest ~

Trigger: Tree Member CsvDataModel/DemoEvent actions [Target <-> Source] Condition Node Condition Node			~			
Image: Construction Image: Construction Image: Constatet construction Image: Const		÷ 🛙		÷ 0	Rule Configuration	(> 🙆 💆 🗙
Trigger Tree Member CevDataModel/DemoEvent actions [Target> Source] 1 Condition Node 1 Condition Node 2 CoNDATAMOdel/DemoEvent/Temperature [Double] 0 != RestServerModel/DemoData/Temperature [Double] -> 0	DemoEvent [DemoDataType] Temperature [Double]	٩ \$	DemoData [DemoDataType] V Temperature [Double]	۵ \$		Rule description
CevDataModel/DemoEvent	V Pressure (Double)		V Pressure [Double]			Trigger Type 🏚
Condition Node CONDITIONS + CONDITIONS + Block Operator: AND ~ 6 4 3 Co-data/del/DemoData/Temperature [Double] • = • RestServerModel/DemoData/Temperature [Double] • • •						0
Condition Node CONDITIONS + CONDITIONS + Block Operator: AND ~ 6 4 3 Co-data/del/DemoData/Temperature [Double] • = • RestServerModel/DemoData/Temperature [Double] • • •						
CONDITIONS + 6 4 3 Block Operator: AND = 6 4 3 2 ConDutaModel/DemoEvent/Temperature [Double] = = RestServerModel/DemoDutar/Temperature [Double] <> 0					actions [Target <=> Source]	1 4
Block Operator: AND - 6 4 3 Con/DataModel/DemoEvent/Temperature [Double] 1 = • RestServerModel/DemoData/Temperature [Double] <> 1					Condition Node	ā ^
Block Operator: AND ~ 2 CsoBataModel/DemoEvent/Temperature [Double] = != ~ RestServerModel/DemoData/Temperature [Double] <> =						0
					Block Operator: AND - 6	3
				2	CsvDataModel/DemoEvent/Temperature [Double]	!= 🔻 RestServerModel/DemoData/Temperature [Double] <> 🛅
					CONDITION: DROP TREE MEMBER	
5						
THEN (>					THEN	**
CONDITION: THEN					CONDITION: THEN	
ADD ELSE EXPRESSION					ADD ELSE EXPRESSION	

- 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).

(⊕ ×

^{Model} CsvDataModel	. D	RestServerModel	÷ 0	Rule Configuration	<> 🙆 🎽
M CsvFile	۵ \$	■ RestServer V DemoData [DemoDataType]	٩ \$	Rule name * CsvToRest Rule description	
V Tennovar (Dennovar) (Pennovar) V Tennovar) V Tensure (Double) V Pressure (Double)		V Temperature (Double)		Trigger: Tree Member CsvDataModel/DemoEvent	Trigger Ty
				actions [Target <=> Source]	
				Condition Node	Ō
				CONDITIONS +	
				Block Operator: AND CovDataModel/DemoEvent/Temperature [Double]	re [Double] <> 📋
				CsvDataModel/DemoEvent/Temperature [Double] CsvDataModel/DemoEvent/Temperature [Double]	<
				CONDITION: DROP TREE MEMBER 9 8	
				THEN	
				CONDITION: THEN	
				l ¹	

- 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.

Model CsvDataModel	÷ 🖸	Model RestServerModel	÷ 🛙	Rule Configuration	O 🙆 🦁 >
M CsvFile	۹ :	RestServer	٩ \$	Rule name * CsvToRest	Rule description
V Temperature [Double] V Pressure [Double]		 Temperature [Double] Pressure [Double] 			Trigger Type
				Trigger: Tree Member CsvDataModel/DemoEvent	
				actions [Target <=> Source]	
				Condition Node	
				CONDITIONS + 10	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 👻	11,
				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:Da	ataMapping:late	rst ∽			♥ (③ ×
Model CsvDataModel	÷ 0	Model RestServerModel	÷ 🗆	Rule Configuration	0 💩 💆 🗙
CsvFile Image: Constant Composition Image: Constant Composition <th>۵ \$</th> <th>RestServer Z DemoData [DemoData Type] Temperature [Double] Pressure [Double]</th> <th>۵ \$</th> <th>Rule name * CsvToRest Trigger: Tree Member CsvDataMode//DemoEvent</th> <th>Rule description</th>	۵ \$	RestServer Z DemoData [DemoData Type] Temperature [Double] Pressure [Double]	۵ \$	Rule name * CsvToRest Trigger: Tree Member CsvDataMode//DemoEvent	Rule description
				csvDataModer/DemoEvent actions [Target <=> Source] Condition Node	ء ج 1 ^ أ
				CONDITIONS + Block Operator: AND × ConOstaModel/DemoEvent/Temperature [Double] [] 1 ConOstaModel/DemoEvent/Temperature [Double] [] ConOstaModel/DemoEvent/Temperature [] [Double] [] ConOstaModel/DemoEvent/Temperature [] [Double] [] ConOstaModel/DemoEvent/Temperature [] [Double] [] ConOstaModel/DemoEvent/Temperature [] [Double] [] THEN	RestServerModel/DemoData/Temperature [Double] <> 20 Control =
			13	RestServerModel/DemoDesa [DemoDesaType] -	OsubasModel/DemoEvent (DemoDasType) (> 0

Actions with Custom Conditions

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

^{Model} CsvDataModel	÷ 0	Model RestServerModel	÷ 0	Rule Configuration	<> 🗠 🧹
CsvFile	٩ \$	RestServer	۵ \$	Rule name * CsvToRest	Rule description
V Denotvent (peniodat rype) V Orderhr (nt) V Temperature (Double) V Pressure (Double)		V ceniova (periodara peri V OrderN/[Int] V Temperature (Double) V Pressure (Double)		Trigger: Tree Member CsvDataModel/DemoEvent	Trigger 1
				actions [Target <=> Source] Condition Node	ñ
				CONDITIONS +	
			2	1 CsvOataModel.Doublevariable RestServerMode	l.coudewriaite (csudetwodel.intvariatie > 50 && Cudetwodel.intvariatie <100
				THEN	3
				CONDITION: THEN	
				ADD ELSE EXPRESSION	

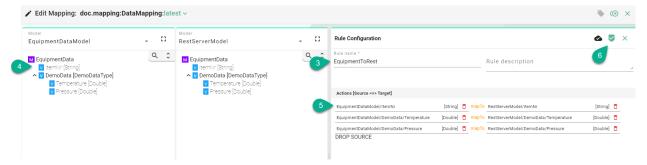
• For the "THEN" section use drag and drop or click the "Source Code" button (3) to input code.

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.



- 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.

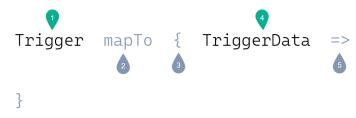
🖍 Edit Mapping: doc.ma	apping:DataMapping:latest ~			× ئا 🕃 🟵 📎
Configuration			Rules	(⁷ q
Group * doc.mapping			EquipmentToRest Trigger: Model/EquipmentDataModel/Variable/ItemNr	/ 0
Name * DataMapping				
Description				
Models		+		
Short name EquipmentDataModel	Information model identifier * doc.mapping:EquipmentData:latest	<u> </u>		
Short name RestServerModel	Information model identifier * doc.mapping:EquipmentData:latest	•		

Code-based Rules

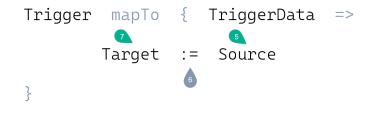
More complex scenarios, which are currently not supported by the graphical view can be implemented via the code editor in the Scala programming language. Similar to Mappings via drag and drop, there is no knowledge of the underlying communication protocol (e.g., MQTT, OPCUA, etc.) needed. Protocols are hidden behind the corresponding Information Models.

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:



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 Types
```

Rule 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 1: Scheduled Rule

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

(continues on next page)

(continued from previous page)

4	})
5	}

Same Type Assignments

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

Listing 2: Type Assignment with Events

event1 := event2

Logging

1

1

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

Listing 3: Rule with Logging

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

Model db	÷ 🖸	Model file	÷ 🖸	Rule Configuration
Database Command_D OtabaseSelect [Command_D OtabaseSelect [Command_D OtabaseSelect [String] OtabaseSelect [String] OtabaseSelect [String] OtabaseSelect [String]		Equipment EditeSvent [FileEventType] OrcerNumber [string] Orber [string] Date [string] Quality [string] Quality [string] Quantity [string]	Q \$	Rule name* DataToMQT Rule description 1 If (file.fileEvent mapTo) (peerst =>

Examples

Variable to Event Mapping

In this case the mapping of the Complex Variable *CurrentOrder* in the *EquipmentModel* and of a Simple Variable in the *EnterpriseModel* to the *EquipmentNewOrderStart* Event in the *MesModel* is described.

- Trigger: EquipmentModel.StartNewOrderFlag (line 1)
- TriggerInstance of EquipmentModel.Alarm: variable (line 1)
- Since values are assigned to an Event, call the function *send*, on the EquipmentNewOrder-StartEvent (line 2) and define the TriggerInstance **event** (line 2).
- The Targets are defined by entering the path of the variables in the event **event.EquipmentId** (line 4).

Listing 4: Rule - StartOrder - Variable/Event

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

Event to Variable Mapping

In this case the mapping of values inside the *TransferNewOrder* Event from the *MesModel* into variables from the *EquipmentModel* is described.

- 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 *NewME*-*SOrderFlag* with data from the MesModels *TransferNewOrder* Event.
- Targets are defined by entering the path of the variables like **Equipment-Model.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 *Transfer*. *NewOrder* 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 5: Rule - TransferNewOrder - Event/Variable

```
MesModel.TransferNewOrder mapTo { event =>
1
      Try {
2
        EquipmentModel.NewOrder.OrderNr := event.OrderNr
3
        EquipmentModel.NewOrder.MaterialNr := event.MaterialNr
4
        EquipmentModel.NewOrder.Quantity := event.Quantity
5
        EquipmentModel.NewMESOrderFlag := true
6
      }
7
    }
8
```

Commands Mapping

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.

- 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 assign 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 6: Rule - File2MqttWithDB - Event/Commands

```
file.FileEvent mapTo {event =>
    database.DatabaseCommand.execute(command => {
        Try {
            command.orderNr := event.orderNr
            command.materialNr := event.materialNr
            CommunicationLogger.log(event, command)
        }
    }, reply => {
```

(continues on next page)

1

2

3

4

5

6

7

8

(continued from previous page)

```
mqtt.MqttEvent.send(event1 => {
9
           Try {
10
             event1.Quality := event.quality
11
             event1.OrderNr := event.orderNr
12
             event1.MaterialNr := event.materialNr
13
             event1.Customer := reply.customer
14
             event1.Product := reply.product
15
             CommunicationLogger.log(reply, event1)
16
             }
17
        })
18
      })
19
    }
20
```

Mapping with Lists

The following scenario describes a Rule that is mapping incoming data from a file to MQTT. The MQTT Model contains a List called *DataList*. **Note** that lists can only be mapped in the code view.

- 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 7: Rule - FileToMQTT - Lists

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

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:

ndel n1	*	0	m2	Ŧ	Rule Configuration			<> 🙆 💆
E2ETestModel	C	२ ३	M E2ETestModel	٩	\$ Rule name * Simple		Rule description	
 ✓ C E2ECommand [Command_E2EComm ▲ E2EEvent [E2EEventType] ▲ VariableNode [String] 	nand]		C E2ECommand [Command_E2EComma E E2EEvent [E2EEventType] VariableNode [String]	nd]	1 Trigger: Tree Member			Trigger Ty
ArrayNode [String: 1] ListNode [String] PropertyNode [LocalDateTime]			ArreyNode (String: 1) ListNode (String) PropertyNode (LocalDateTime)		m1/E2EEvent			
V VariableNode [String] A ArrayNode [String: 1] ListNode [String]			VariableNode [String] ArrayNode [String: 1] ListNode [String]		actions [Target <=> Source]			
PropertyNode [LocalDateTime]			 PropertyNode [LocalDateTime] 		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 <i>node</i>	Any Target node	Any Source <i>node</i>
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:**

todel n1	*	53	m2	÷ 🖸	Rule Configuration				<> 🐟 💆
Z E2ETestModel	C	ર 🙎	E2ETestModel ComplexVariable [ComplexType]	٩ \$	Rule name * Simple			Rule description	
C E2ECommand [Command E2EComm E2EECommand [Command E2EComm Setternet [E2EEventType] VariableNode [String] ArrayNode [String] PropertyNode [LocalDateTime] VariableNode [String] ArrayNode [String]	nand]		VariableNode (Smng) ArrayNode (String, 1) Listvice (Smng) ProperyNode (LocalDateTime) E2ECommand (Command, E2ECommand Command, E2ECommand, E2ECommand	ind]	Trigor: Tree Member m1/E2EEvent actions [Target <> Source]				Trigger Typ
ListNode (String) PropertyNode (LocalDateTime)			ListNode (String) PropertyNode (LocalDateTime)		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
	•	continues on next nage

continues on next page

Trigger	Target	Source
		Array
	List	Variable
		Variable of a custom type
		List
		Array
	Array	Variable
		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	Variable of a custom type
		List
	Event	Array Variable
	Event	
		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
Droporty of a sustant tra-	Variable	Variable
Property of a custom type	variable	
		Variable of a custom type
		List
		Array
	Custom type Variable	Variable
		Variable of a custom type
		List
		Array
		continues on next page

Table 2 – continued from previous page

Trigger	 Z – continued from previous Target 	Source
	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	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
	I	
		Array
	Array	Array Variable continues on next page

Table 2 – continued from previous page

Trigger	Target	Source
		Variable of a custom type
		List
		Array
Command	Variable	Variable
		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
		continues on next page

Table 2 – continued from previous page

Trigger	Target	Source
		Variable
		List
		Array
	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

Table 2 – continued from previous page

2.4 Device Types

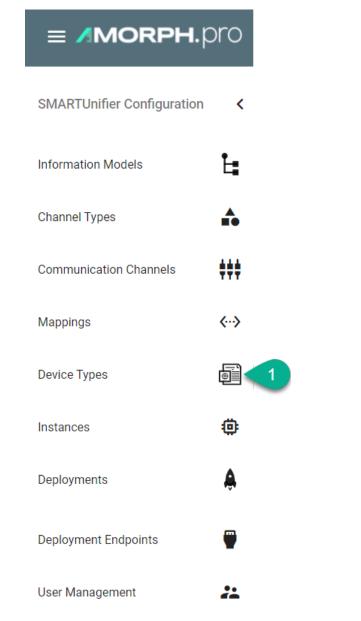
2.4.1 What are Device Types

With SMARTUNIFIER Device Types it is possible to have multiple Communication *Instances*, which share common configuration parameters. A Device Type contains one or multiple Mappings. Each Mapping contains one or multiple Information Models and its associated Communication Channel. Within a SMARTUNIFIER Device Type it is possible to over-write existing Communication Channel configurations. Device Types are especially important, when integrating several similar pieces of equipment or devices. In this case, the Device Type can be reused for all Instances (i.e., one instance represents one equipment).

2.4.2 How to create a new Device Type

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

• Select the SMARTUNIFIER Device Type Perspective (1).



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

	Device Types				< 🖞 + ታ
ER <	Group 🕈	Name	Version	Description	2
GROUP FILTE					

- The creation of a Device Type is split up into two parts. First provide the basic information about the Device Type like the Group, the Name, and the Version. Optionally, provide a short description (3).
- In the next step provide one or multiple *Mappings* previously created. To do so click the "Add Mapping" button (4). After selecting a Mapping (5) the associated Information Models show up. In case the wrong Mapping was selected click the "Delete Mapping" button to remove the Mapping from the Device Type (6). Now select a *Communication Channel* for each *Information Model* from the Drop-Down (7).
- Similar to the Communication Channel view it is possible to change the configuration of the Channel within the Device Type view. In case of changes in the configuration click the "Configure" button (8). This action over-writes previous configurations.
- The new Device Type can be saved by clicking the "Save" button at the top right corner (9).

Edit Device Type: demoscenario.cs	v2rest:SUDeviceType:latest	~		🎙 🛞 🖬 🖒 🗡
3	Group * demoscenario.csv2rest			9
	Name * SUDeviceType			
	Description			
	Mappings		4 +	
	5 demoscenario.csv2rest	:CSVtoREST:latest	· 16	
	Models	Channels		
	CSVDataModel	demoscenario.csv2rest:CSV:latest		
	RESTDataModel	demoscenario.csv2rest:RESTServer:latest	<u> </u>	

2.5 Communication Instances

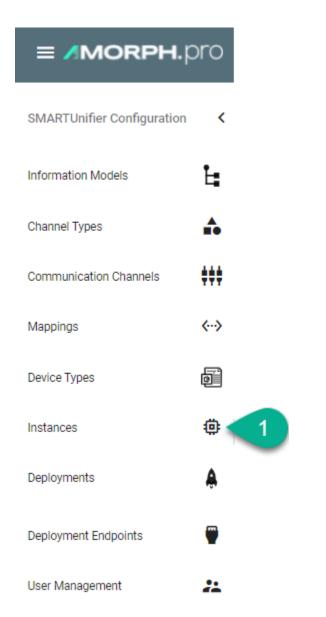
2.5.1 What are Instances

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*.

2.5.2 How to create a new Instance

Follow the steps described below to create a SMARTUNIFIER Instance.

• Select the SMARTUNIFIER Instances Perspective (1).



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

۲	Instances				ር 🖞 + ይ
ER <	Group 🛧	Name	Version	Description	2
GROUP FILTE					

- Select a Device Type from the Drop-Down (3)
- The details for the Instance are automatically taken from the Device Type (4). However, Group, Name, Version and the Description can still be changed.
- The Mapping defined in the Device Type show up in the Mapping area (5).
- To change the existing configuration or if no configuration has been made yet, click the "Con-

figure" button (6)

Edit Instance:demoscenario.csv2res	st:SUInstance:latest ~			× طا 🖬 🕀 🖗
3	Device Type demoscenario.csv2rest:SUD	eviceType:latest		
4	Group * demoscenario.csv2rest			
	Name * SUInstance			
	Description			
	Mappings			
	Mapping demoscenario.csv2rest:C	SVtoREST:latest		
	Models	Channels		
5	CSVDataModel	demoscenario.csv2rest:CSV:latest	- * 🕌	
•	RESTDataModel	demoscenario.csv2rest:RESTClient:latest	- *	
	Advanced Settings		~ 7	

• 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 SMARTUNIFIER.

Edit Instance:demoscenario.csv2rest	t:SUInstance:latest ∽			🍬 🕀 🖬 🖒 🗙
	SUInstance			9
	Description			•
	Mappings			
	Mapping demoscenario.csv2rest:	:CSVtoREST:latest	-	
	Models	Channels		
	CSVDataModel	demoscenario.csv2rest:CSV:latest	÷ 🗘	
	RESTDataModel	demoscenario.csv2rest:RESTCllent:latest	× 🌣	
	Advanced Settings		^	
	Framework Version com.amorphsys.i40.adapter	r:SmartUnifierFrameworkLibraryDefinition:1.5.0-SNAPSHOT	•	
8	CSV String to Model com.amorphsys.i40.adapter	channel:CsvStringToModelLayer:1.5.0-SNAPSHOT	•	
	File Tailer to String com.amorphsys.i40.adapter	r.channel:FileTailerToStringLayer:1.5.0-SNAPSHOT	<u> </u>	
	REST Client com.amorphsys.i40.adapter	r.channel:Restimplementation:1.5.0-SNAPSHOT		

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

CHAPTER

CONFIGURATION COMPONENT MANAGEMENT

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

- Group Filter
- Component Version Control
- Operations

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

3.1 Naming Convention

Each *Configuration Component* created with SMART**UNIFIER** has defined a Group, a Name, and a Version.

We recommend the following naming convention for better comprehensibility.

Group Identifies the integration scenario across all integration scenarios within the *SMARTUNIFIER Manager*.

Name Is the name of each component, which is part of the integration scenario, such as: *Models*, *Channels*, *Mappings*, *Device Types*, *Instances* as well as *Deployment Endpoints*.

Version Defines the version of the component - Suggested format: 1.0.0 / 1.0.1 / 2.0.0.

3.2 Group Filter

With the Group Filter it is possible 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).

La Information Mode	els			ላ 🖞 + 🕁
Group Filter <	Group 🗸	Name	Description	
v Show All	demoscenario.xmldatabase2mqtt	Database		Ċ / Ō
scenario1 v demoscenario csv2rest	demoscenario.xmldatabase2mqtt	Equipment		Ċ 🖍 🙃
json2database xmldatabase2mqtt	demoscenario.xmldatabase2mqtt	Host		Ċ 🖍 🙃
	demoscenario.json2database	Database		Ċ 🖍 🗇
	demoscenario.json2database	JSON		Ċ 🧪 🗇
	demoscenario.csv2rest	CsvDataModel		Ċ 🧪 🙃
	demoscenario.csv2rest	RestDataModel		Ċ 🧪 🖻
	demo.scenario1	DatabaseModel		Ċ 🧪 🙃
	demo	DBModel		Ċ 🧪 🙃
	demo	EquipmentModel		ථ 🧪 📋

In order 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).

🔓 Information Mode	els Filtered by: csv2rest × 3			Q (L) + G
Group Filter <	Group 🗸	Name	Description	
 ✓ Show All ✓ demo 	demoscenario.csv2rest	CsvDataModel		Ċ 🖌 🗇
scenario1 demoscenario csv2rest 2	demoscenario.csv2rest	RestDataModel		Ċ 🗡 🙃
json2database xmldatabase2mqtt				

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, SMARTUNIFIER 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 SMARTUNIFIER 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.

>	✔ Edit Model: su.demo.dashboard:MESSimulator:latest ∨	1 🗣 🔞 🗟 🗗 🗴
Ŀ	Equipment	Group *
	[FileEvent [FileEventType] OrderNumber [String]	demo.scm
***	V ProductNumber [String] V Date [String] V Quality [String]	Name * Equipment
<·· >	Volumity (string)	Description
6		

- 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	
2	1.0.0	
3	Ok	Cancel

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

>	Edit Model: demo.scm:Equipment	:1.0.0 ~			×
Ŀ	M Equipment	latest	٩ \$	Group	
•	 FileEvent [FileEventType] OrderNumber [String] ProductNumber [String] 	1.0.0		demo.scm	
***	V Date [String]			Name Equipment	
<>	V Quality [String] V Quantity [String]			Description	

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).

Ŀ,	formation Models Filtered by: demoscenario ×			ସ୍ ∐ + <i>€</i>
н К	Group 🛧	Name	Description	1
FILT	demoscenario.csv2rest	CsvDataModel		Ċ 🗡 🗇
GROUP	demoscenario.csv2rest	RestDataModel		Ů 🖍 🙃
	demoscenario json2database	Database		Ů 🖍 🗊
	demoscenario json2database	JSON		Ů 🧪 🗊
	demoscenario.xmldatabase2mqtt	Database		Ċ 🖍 🗇

The component is opened in the edit mode.

Edit Model: demoscenario.csv2rest:CsvDataModel:latest ~	 (c) 🦁 🖬 ۱۵ >
CevDataModel Q \$ CevDemoEvent [CSVDemoEventType] PARTNR [String] V TIMESTAMP [String] VTIMESTAMP [String] V PRESSURE [String] VPRESSURE [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.

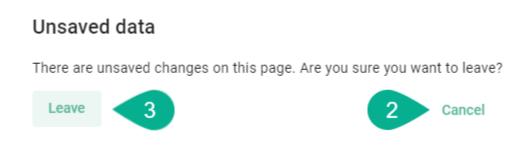
Edit Model: demoscenario.csv2rest:CsvDataModel:latest ~		🔖 🕑 🦉 🖬 🖒 🗙
CsvDataModel CsvDataModel Image: CsvDemoEvent(CSVDemoEventType) Image: CsvDatamolecont(SVDemoEventType) Image: CsvDatamolecont(SVDe	Croup * demoscenari Name * CsvDataMode	1
	Description	

3.4.4 Exit Editing

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

Edit Model: demoscenario.csv2rest:CsvDataModel:latest ~			🍋 🕑 💆 🖬 🖆 🗙
CsvDataModel CsvDamoEvent [CSVDemoEventType] ATTRR [Sring] TIMESTAINP [Sring] TIMESTAINP [Sring] TPRESSURE [Sring]	Q C demoscenario. Name * CsvDataModel Description	:sv2rest	

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.



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).

CsvDataModel A CsvDemoEvent[CSVDemoEventType] V PATTUR [String] TIM/ESTAM/P [String] V FRESSURE [String]	-	sup * moscenario.csv2rest ma * vDataModel	U
		escription	

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

Edit Model: demoscenario.csv2rest:CsvDataModel	:latest ~	🔖 🕲 🖬 🖒 🗙
CsvDataModel CsvDemoEvent [CSVDemoEventType] CsvDatnik [Sring] ThMESTANA [Sring] TEMPERATUR [Sring] PRESSURE [Sring]	Group * demoscenario.csv2rest Name * CsvDataModel Description	
	2 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)**.

CsvDataModel CsvDataModel CsvDataModel DATTINR [String] TiMESTAMP [String] TemPERATUR [String] TemPERATUR [String]	Group * demoscenario.csv2rest Name * CsvDataModel	1
	Description	

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

E I	formation Models Filtered by: demoscenario ×			Q 💾 + 49
› د	Group 🛧	Name	Description	
FILTER	demoscenario.csv2rest	CsvDataModel		Ů 🖍 🗇
GROUP	demoscenario.csv2rest	RestDataModel		Ů 🖍 🗊
	demoscenario.json2database	Database		Ċ 🖍 🗇
	demoscenario.json2database	JSON		Ů 🖍 🗊
	demoscenario.xmldatabase2mqtt	Database		ů 🖍 🙃
	demoscenario.xmldatabase2mqtt	Equipment		Ċ 🖍 🗇
	demoscenario.xmldatabase2mqtt	Host		Ů 🖍 🗊
	2 The ent	ry 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.

🗄 Information Mode	als			1 Q 🖞 + 5
Group Filter <	Group 🕹	Name	Description	<u>^</u>
 ✓ Show All ✓ demo 	demoscenario.xmldatabase2mqtt	Database		Ċ 🖍 🗇
scenario1 demoscenario csv2rest	demoscenario.xmldatabase2mqtt	Equipment		Ċ 🖍 🙃
json2database xmldatabase2mqtt	demoscenario.xmldatabase2mqtt	Host		Ċ 🖍 Ō

Enter a search term (2).

				Х Ц́ + Ф
Group Filter <	Group 🕁	Name	Description	
 ✓ Show All ✓ demo 	demoscenario.csv2rest	CsvDataModel		Ů 🖍 ō
scenario1 ✓ demoscenario csv2rest	demoscenario.csv2rest	RestDataModel		Ů 🖍 ō
json2database xmldatabase2mqtt				

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

🖁 csvl				× Ů + ↔
Group Filter <	Group 🦊	Name	Description	3
 ✓ Show All ✓ demo 	demoscenario.csv2rest	CsvDataModel		ථ 🖍 🗇
scenario1 v demoscenario csv2rest	demoscenario.csv2rest	RestDataModel		Ċ 🖍 🙃
json2database xmldatabase2mqtt				

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 Mode	IS Filtered by: demoscenario ×			Q 🖞 + 🖸
Group Filter <	Group	Name 🕈 🗾	Description	
 ✓ Show All ✓ demo 	demoscenario.csv2rest	CsvDataModel		ů 🖍 🗊
scenario1 demoscenario csv2rest	demoscenario.json2database	Database		Ċ 🖍 🗇
json2database xmldatabase2mqtt	demoscenario.xmldatabase2mqtt	Database		ů 🧪 🙃
	demoscenario.xmldatabase2mqtt	Equipment		ů 🖍 🗊
	demoscenario.xmldatabase2mqtt	Host		ů 🧪 🙃
	demoscenario.json2database	JSON		ů 🖍 🗊
	demoscenario.csv2rest	RestDataModel		Ů 🖍 🗊

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 Mode	Filtered by: demoscenario X			५ 🖞 + <i>२</i>
Group Filter <	Group 🥹	Name	Description	1
 Show All demo 	demoscenario.xmldatabase2mqtt	Database		Ů 🖍 Ō
scenario1 v demoscenario csv2rest	demoscenario.xmldatabase2mqtt	Equipment		Ů 🖍 Ō
json2database xmldatabase2mqtt	demoscenario.xmldatabase2mqtt	Host		Ů 🖊 🗇

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.

```
💳 model demoscenario.csv2rest CsvDataModel latest.json 🔀
     ₽ (
  1
           "info": {
  2
  3
               "identifier": {
                    "id": "8b6bc3f2-192d-4870-8bad-b7c2f5e191b9",
  4
                    "version": "latest"
  5
  6
               },
                                              2
  7
               "externalIdentifier": {
                    "name": "CsvDataModel",
 8
                    "group": "demoscenario.csv2rest"
 9
               },
 11
               "externalDescriptor": {
                    "description": " "
 13
               }
 14
           },
15
           "members": [{
                    "id": "csvDemoEvent",
16
17
                    "description": "",
                    "definitionType": "Event",
 18
 19
                    "typeName": "CSVDemoEventType"
               }
21
           ],
           "types": {
22
               "CSVDemoEventType": {
23
                    "id": "CSVDemoEventType",
24
25
                    "members": [{
                             "id": "PARTNR",
26
                             "description": "",
 27
                             "definitionType": "Variable",
28
                             "typeName": "String"
29
30
                        }, {
                             "id": "TIMESTAMP",
31
                             "description": "",
"definitionType": "Variable",
 33
 34
                             "typeName": "String"
                        }, {
                             "id": "TEMPERATUR",
 36
                             "description": "",
"definitionType": "Variable",
                             "typeName": "String"
 39
 40
                        }, {
                             "id": "PRESSURE",
 41
                             "description": "",
 42
                             "definitionType": "Variable",
43
                             "typeName": "String"
44
45
                        }
46
                    1,
                    "category": "model",
47
48
                    "type": "StructuredVariable"
 49
               }
           },
51
           "rawModelString": "",
           "ignoreCompileErrors": false
 52
53
```

```
🔚 model_demoscenario.csv2rest_CsvDataModel_latest.json 🔀
     ₽(
  2
            "info": {
  3
                "identifier":
                      "id": "",
  4
  5
                      "version": "latest"
  6
                 },
                 "externalIdentifier": {
  7
                      "name": "CsvDataModel2",
  8
  9
                      "group": "demoscenario.csv2rest"
                 }.
                 "externalDescriptor": {
                      "description": " "
 13
                 },
    "version": "latest"
                                                 3
 14
            },
 16
            "members": [{
                      "id": "csvDemoEvent",
 17
                      "description": "",
"definitionType": "Event",
 18
 19
                      "typeName": "CSVDemoEventType"
                 }
            ],
 23
            "types": {
                 "CSVDemoEventType": {
 24
                      "id": "CSVDemoEventType",
 26
                      "members": [{
                                "id": "PARTNR",
"description": "",
"definitionType": "Variable",
 27
 28
 29
                                "typeName": "String"
 31
                           ), {
                                "id": "TIMESTAMP",
                                "description": "",
"definitionType": "Variable",
 34
                                "typeName": "String"
 36
                           }, {
                                "id": "TEMPERATUR",
 37
                                "description": "",
"definitionType": "Variable",
"typeName": "String"
 38
 39
 40
 41
                           }, {
                                "id": "PRESSURE",
 42
                                "description": "",
"definitionType": "Variable",
 43
 44
                                "typeName": "String"
 45
 46
                           }
 47
                      ],
 48
                      "category": "model",
                      "type": "StructuredVariable"
 49
 50
                 }
 51
            },
            "rawModelString": "",
            "ignoreCompileErrors": false
 53
54
      1.
```

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

🗄 Information Mode	IS Filtered by: demoscenario X			Q 💾 + 49
Group Filter <	Group	Name 🛧	Description	4
 Show All ✓ demo 	demoscenario.csv2rest	CsvDataModel		Ċ 🖊 Ō
scenario1 v demoscenario csv2rest	demoscenario.json2database	Database		Ů 🖍 🗇
json2database xmldatabase2mqtt	demoscenario.xmldatabase2mqtt	Database		Ċ 🖍 🙃

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

					×
↓ > This PC > Downloads			ې ن) Search Downloads	
New folder					•
Name	Date modified	Туре	Size		^
∨ Today (1)					
/// model_demoscenario.csv2rest_CsvData	15.09.2021 16:26	JSON File	2 KB <	5	
> Yesterday (1)					
File name: model_demoscenario.csv2rest_Cs	vDataModel_latest.json		val 🗸	ON-Datei (*.json)	~
			6	Open C	ancel .::

The imported component is now listed (7).

占 Information Mode	S Filtered by: csv2rest ×			Q [¹] + 43
Group Filter <	Group 🛧	Name	Description	
 Show All demo 	demoscenario.csv2rest	7 CsvDataModel2		Ů 🖍 🙃
scenario1 demoscenario csv2rest	demoscenario.csv2rest	CsvDataModel		Ů 🖍 🙃
json2database xmldatabase2mqtt	demoscenario.csv2rest	RestDataModel		Ů 🖊 🗇

3.4.11 Export

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

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

🗄 Information Mode	els Filtered by: csv2rest ×			Q [¹] + G
Group Filter <	Group 🛧	Name	Description	
 ✓ Show All ✓ demo 	demoscenario.csv2rest	CsvDataModel		1 🖒 🗡 🙃
scenario1 ✓ demoscenario csv2rest	demoscenario.csv2rest	RestDataModel		Ů 🖍 Ō
json2database xmldatabase2mqtt				

3.4.12 Clone

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

CsvDataModel CsvDatemoEvent [CsVDermoEventType] PARTNR [String] TitleSTAMP [String] TEMPERATUR [String] PRESSURE [String]	Q > Group * demoscenario.csv2rest 	4

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



Cancel

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

✔ Edit Model: demoscenario.csv2rest:CsvDataModel [CLONE]:latest ∨	× di 🖬 🖉 🕥 🧳
CovDataModel [CLONE] Q	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 Mod	els Filtered by: csv2rest ×			Q [¹] + <i>C</i> ₂
Group Filter <	Group 🛧	Name	Description	1
 Show All demo 	demoscenario.csv2rest	CsvDataModel [CLONE]		ů 🖍 ī
scenario1 demoscenario csv2rest	demoscenario.csv2rest	CsvDataModel		Ů 🖍 Ō
json2database xmldatabase2mqtt	demoscenario.csv2rest	RestDataModel		ů 🖍 🖻

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

Delete Model

Are you sure you want to delete this Information Model?



The component is deleted.

La Information Mode	Is Filtered by: csv2rest ×			Q (¹) + 5
Group Filter <	Group 🕈	Name	Description	
Show All → demo	demoscenario.csv2rest	CsvDataModel		Ú 🗡 🗇
v demoscenario	demoscenario.csv2rest	RestDataModel		Ů 🗡 ō
json2database xmidatabase2mqtt		Entry successfully deleted	04 [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

>	Å (Deploy	. 🛛 🕆 😤						۹	Ľ	+ 4	э ф
Ŀ	× X		Group 🛧	Name	Version	Deployment Type	State					
***	GROUP FILTER	- 1	ex1	CSVtoRESTDeviceType	latest	Local	NotDeployed	►	Ŧ	5 C	2 /	1
-	GROUF		ex2	JSONDatabaseChannel	latest	Local	NotDeployed	►	Ŧ	5 C	2 /	1
			ex3	OpcUa Device Type	latest	Local	NotDeployed	►	Ŧ	80 Q	2 /	1
۲												
\$												
۲												

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

	Deploy	■ 주 ¶ _×	4						۹	Ľ	+ -	с ф
>		Group 🕇	Name	Version	Deployment Type	State						
ILTER		ex1	CSVtoRESTDeviceType	latest	Local	NotDeployed	►		Ŷ.			
2 GRd		ex2	JSONDatabaseChannel	latest	Local	NotDeployed		•	Ŷ.			
		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)**.

A 1	Deploy	■ 🌱 🗣							Q	Ľ	+	Ð	۵
÷	= :	Group 🕈 5	Name	Version	Deployment Type	State							
P FILTE	V	ex1	CSVtoRESTDeviceType	latest	Local	NotDeployed	►	з.	Ŧ	8	Ξ×	1	
GROUF	~	ex2	JSONDatabaseChannel	latest	Local	NotDeployed	►		Ŷ	8	Ξ×	/	i i
		ex3	OpcUa Device Type	latest	Local	NotDeployed	►		Ŧ	8	ΞM	1	Î

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)



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).

		12					_							
10	eploy	F	■ 🕹 😤				11			۹	Ľ	+	c,	¢
н С		:	Group 🕇	Name	Version	Deployment Type	State							
O FILTE	\checkmark		ex1	CSVtoRESTDeviceType	latest	Local	Stopped		•	$\widehat{\tau}_{*}$	8	Ξ۲	/	Î.
GROUF	\checkmark		ex2	JSONDatabaseChannel	latest	Local	Stopped		•	$\widehat{\tau}_{\star}$	8	Ξ۲	/	Î.
	\checkmark		ex3	OpcUa Device Type	latest	Local	NotDeployed	►	•	Ŧ	8	Ξ×	/	Î
GROU	 ✓ 													

Deploym	ents		
ex1:CS\	/toRESTDeviceType:late	🗸	
ex2:JS0	NDatabaseChannel:late	···· 🗸	
ex3:0pd	cUa Device Type:latest	×	13

Note: Protected Instances will not work using bulk actions.

CHAPTER

DEPLOYMENT

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

- Local on the same environment the SMARTUNIFIER Manager is running on
- Docker on containerized environments
- SSH remote on Linux machine
- Fargate on the AWS Cloud using fully managed service AWS Fargate

Learn how to operate and monitor your SMARTUNIFIER Instances.

Learn about notifications.

Learn about additional *deployment options*.

4.1 What is a Deployment

With the SMARTUNIFIER Deployment capability you can deploy your SMARTUNIFIER *Communication Instances* to any IT resource (e.g., Equipment PC, Server, Cloud) suitable to execute SMARTUNIFIER 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 SMARTUNIFIER Communication Instance to your local computer where the SMARTUNIFIER Manager is running on.
- *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
 - 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:

≡ /ΜΟRPH .β	oro
SMARTUnifier Configuration	<
Information Models	Ŀ.
Channel Types	
Communication Channels	***
Mappings	⟨ ··⟩
Device Types	
Instances	٥
Deployments	A 🚺
Deployment Endpoints	-
User Management	**

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

A [eployment						م ٿ +	£,
› «	Group 🛧	Name	Version	Deployment Type	State	3	Local	
ILTE							Docker	
GROUP FILTER							AWS	
GR							SSH	

- In the Add Deployment view a set of configuration parameters is required (4)
 - Select the SMARTUNIFIER 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).

⊕ Add Local Deployment	в њ ×
Instance * demo.filetailer:SUInstance.latest	5
Endpoint 10 * demo:LocalEndpoint1	v
Log File Configuration * Info	•
Z Enable Encryption	
Same folder as deployment Custom Path:	
4 ✓ Protected	
VM Arguments false	
JMX Properties JMX Host Name *	
localhost JMX Port * 33	
☐ JMX Authenticate ☐ Authenticate	
Cocal Only Use SSI	

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

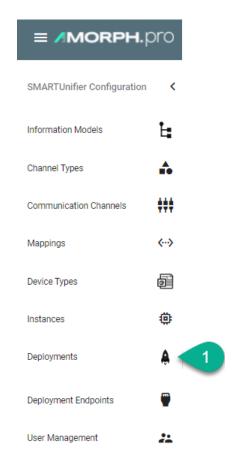
Note: The Instance configuration folder can be copied to another location and started, but the Instance will not be monitored by the SMART**UNIFIER** Manager.

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:



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

>	â I	Deployment				ද එ	+ -9 🕸
Ŀ	› «	Group 🛧	Name	Version	Deployment Type	State	Local
♣	P FILTER					3	Docker
###	GROU						AWS

- Select the SMARTUNIFIER 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)

⊕ Add Docker Deployment		14 B 16 ×
	Instance + demoscenario.csv2rest:SUInstance:latest •	
	Endpoint ID * demoscenario.csv2rest:LocalDockerEndpoint +	
	image * adoptopenjdk:11-jre-hotspot 🗸	
	Log File Configuration *	
8	Enable Encryption	
9	Protected	
	Volumes + 10	
1	Local Container /home/developer/temp/files : /files	
13	VM Arguments	
-	JMX Properties	
	Default Arguments	
	xms * 32m	
	256m	

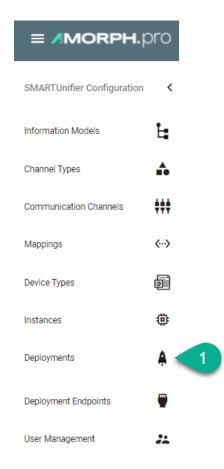
- (Optional) Add Volumes to store persisting data that can be used by the Docker container (10).
 - Enter the local path of the directory or the name of an existing volume (11)
 - Enter the mount path inside the container (12)
- (Optional) Add VM Arguments (13)
- When all mandatory fields are filled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

4.4 Deploy with 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:



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

٩	Deployment					२ 🖞 + <i>छ</i>
۰ «	Group 🛧	Name	Version	Deployment Type	State	Local
FILTER						Docker
GROUP						AWS
					3	SSH

- 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			⊡ 15 ×
3	Instance ** demo:CSVIRESTDeviceType:latest feedon:CSVIRESTDeviceType:latest feedon:CSVIRESTDeviceType:latest Lagenet:D * demo:VM1 LageTile Configuration * Info Frotected VML Arguments false Default Arguments XMS * 325m XMS * 325m Additional	· •	6

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

4.5 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 SMARTUNIFIER 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.

4.5.1 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

SMARTUNIFIER complies with the security best practices in IAM and does not need root privileges. We recommend to create one user dedicated for SMARTUNIFIER. The IAM user follows the general rule of least privileges and allows only policies needed for the deployment of SMARTUNIFIER 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 buck-
	ets via the AWS Management
	Console.
arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess	Provides full access to AWS
	CodeBuild via the AWS Man-
	agement Console. Also at-
	tach AmazonS3ReadOnlyAccess
	to provide access to down-
	load build artifacts, and at-
	tach IAMFullAccess to create
	and manage the service role for
	CodeBuild.
arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryFullAcce	
	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 re-
	sources, including VPCs, Auto
	Scaling groups, and CloudFor-
	mation stacks.
arn:aws:iam::aws:policy/CloudWatchFullAccess	Provides full access to Cloud-
	Watch.

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 SMARTUNIFIER 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)

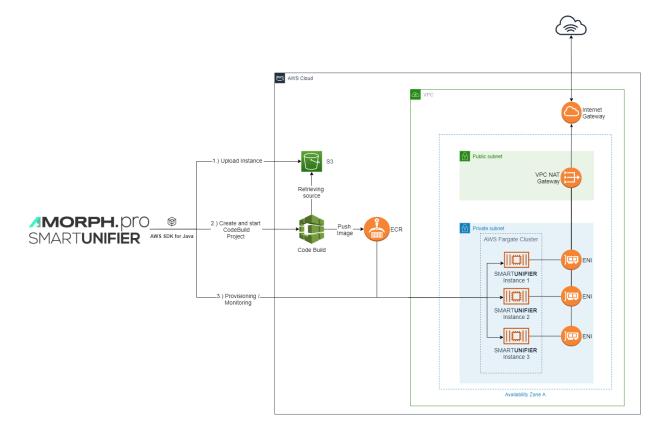
```
]
     },
     {
         "Sid": "ECRPullPolicy",
         "Effect": "Allow",
         "Action": [
             "ecr:BatchCheckLayerAvailability",
             "ecr:GetDownloadUrlForLayer",
             "ecr:BatchGetImage"
         ],
         "Resource": [
             "*"
         ]
     },
  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

4.5.2 Architecture

The deployment of SMARTUNIFIER-Instances on AWS Cloud is handled by the SMARTUNIFIER 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 SMARTUNIFIER Manager on AWS Cloud please see the SMARTUNIFIER Installation Manual.

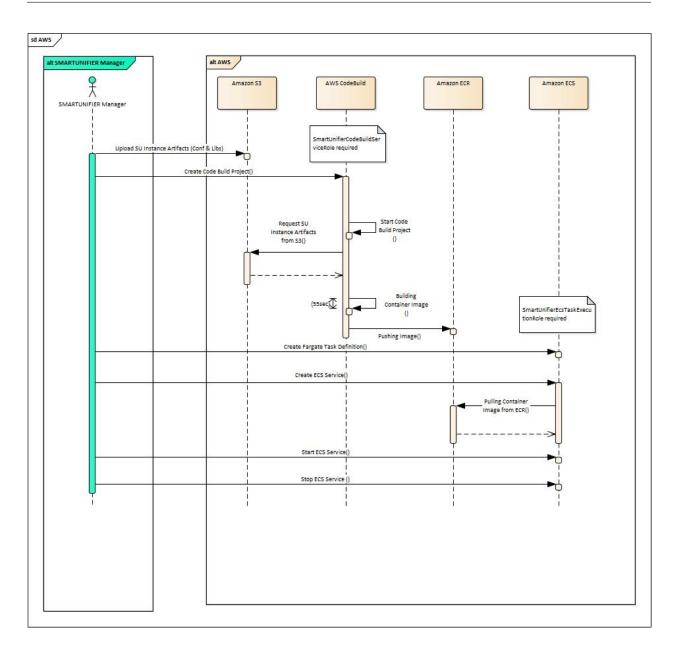
SMART**UNIFIER** is using the AWS SDK for Java to make deployments of Instances to AWS Fargate. Following AWS Services are used during the deployment process:

- 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 SMARTUNIFIER 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.



4.5.3 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			
0.25 vCPU	0.5GB, 1GB, and 2GB	pings) <= 5		
0.5 vCPU	Min. 1GB and Max. 4GB, in 1GB increments	> 6		

4.5.4 Deployment Steps

Expected Time

- Deployment of an SMARTUNIFIER 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 SMARTUNIFIER Instance on AWS Fargate:

			≡ / MO	RPH.pro		
			SMARTUnifier Co	nfiguration <		
			Information Models	Ŀ		
			Channel Types	.		
			Communication Ch	annels		
			Mappings	<··>		
			Device Types	٥		
			Instances	٥		
		1	Deployments	٨		
			Deployment Endpo	nts 🔮		
			User Management	22		
•	Click the "Add" bu	tton (2) .				
•	Select AWS (3).					2
>	🌲 Deployment					م ٿ +
	> Group ↑	Name	Version	Deployment Type	State	Lo
•	GROUP FILTER					Doc 3 AW

- Select the SMARTUNIFIER 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).

	MORPH.pro		SMARTUNIFIER 😫 🔞
>	⊕ Add AWS Deployment		9 B 16 ×
Ŀ	4	Instance + demoscenario.csv2rest:SUInstance:latest *	•
	5	Endpoint ID * demo:AWSAccount1 ~	
***		vpcs * vpc-0b7318756fa7fcfab *	
<··>		Subnet * subnet-09ec948d12803ed24 *	
ē		Security Group * SMARTUnifierIntegrationTestECSContainer_SG *	
ب		Bole * arn:aws:lam:: role/SMARTUnifierIntegrationTestCodeBuildServiceRole	
•		ssbucket * smartunifier-integration-test •	
-		Cluster * arn:aws:ecs:eu-central-1: :cluster/SMARTUnifierIntegrationTestCluster *	
		Republiky * Su-instance *	
		25 vCPU ▼	
		Memory * 512 *	
	6	Log File Configuration *	
		Enable Encryption	
	8	Protected	

• Go back to the list view by clicking the "Close" button and deploy your SMARTUNIFIER Instance by clicking the "Deploy" button (8).

🌲 Deployment					Q 💾 + 67 🏟
Group 🕈	Name	Version	Deployment Type	State	8
su.demo.dashboard	SUInstance	1.0.0	AWS	NotDeployed	

• You can start and stop the Instance using SMARTUNIFIER 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.

4.6 How to Deploy, Run and Operate a Deployed Instance

4.6.1 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.

A	Deployment					Q 💾 🕂 🕁 🏚
k v	Group 🛧	Name	Version	Deployment Type	State	1
FILT	su.demo.dashboard	SUInstance	latest	Local	NotDeployed	> = 🕹 🏭 🖾 🗡 🗑
GROUP						

4.6.2 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.

\$	Deployme	ent					९ 💾 + 🕁 🏚
ہ د	□ :	Group 🛧	Name	Version	Deployment Type	State	2
o FILTE		su.demo.dashboard	SUInstance	latest	Local	Stopped	📃 > = 🕹 🏭 🖾 🥕 📋
GROUF							

4.6.3 How to Stop an Instance

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

A D	eployn	ment					ላ 💾 + 😏 🌣
н К	<u>~</u>	Group 🛧	Name	Version	Deployment Type	State	
P FILTE	~	su.demo.dashboard	SUInstance	latest	Local	Started	> 🔹 🖑 🖽 🖾 🗡 📋
GROUI							3

4.6.4 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*.

Å (Deploy	mei	nt					
ER <	\checkmark	:	Group 🛧	Name	Version	Deployment Type	State	
IP FILTE	\checkmark		su.demo.dashboard	SUInstance	latest	Local	Stopped	> = y II 🖬 🔪 🧃
GROUF								4

4.6.5 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).

A I	eployme	nt					ላ 💾 + 😏 🏟
er 、		Group 🛧	Name	Version	Deployment Type	State	
° FILTE		su.demo.dashboard	SUInstance	latest	Local	Stopped	> = 🛃 🖬 🖬 🖍 🗎
GROUF							6

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



• 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.

A 1	Deployme	ent					ላ 💾 + 😏 🏚
ER <		Group 🛧	Name	Version	Deployment Type	State	
P FILTE		su.demo.dashboard	SUInstance	latest	Local	NotDeployed	> = 🗄 🖬 🖬 🗡 🗊
GROUF						8	

4.6.6 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.

â I	Deployme	ent					ዓ 💾 + 😔 🌣
н К		Group 🛧	Name	Version	Deployment Type	State	
P FILTE		su.demo.dashboard	SUInstance	latest	Local	NotDeployed	> = 🕹 🖽 🖬 🖉 🧨 🗎
GROUF							9

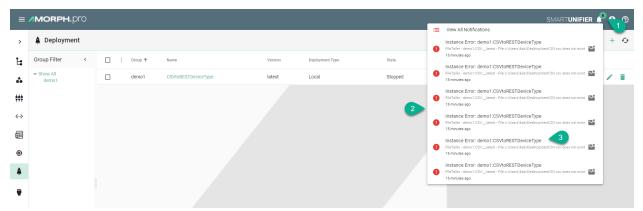
4.7 Notifications

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.

4.7.1 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.



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

				4		
>	Status Dashboard: demo1:CSVtoRESTDeviceTy	vpe:latest				×
ъI	ttt Channels			^	Status	Stopped
	Info	Туре	Status	Model	© Time start 2022	
î.	demo1:RESTServer.latest	Rest Server	Stopped	demo1:RestDataModel:latest		
ŧŧ	demo1:CSV:latest	File tailer (CSV)	Stopped	demo1:CsvData:latest		
>						
1						
0					CPU Usage	(0% / 0%)
A						
•	Log Viewer: 90596ff9-bae1-4039-9688-e184al	o70f698:latest		A		
	2022-02-15 11:23:34,448 - [INFO] - [INFO] - Main - About to start c:\Users\raa\Desktop\Unifier\SmartUnifierManager-windows-x64.15.02.2			t.Instance596ff9bae140399688e184ab70f698 from folder		
	2022-02-15 11:23:34,465 - [DEBUG] - [DEBUG] - Main - About to create 2022-02-15 11:23:34,585 - [INFO] - [INFO] - Main - Instance demo1: 2022-02-15 11:23:34,587 - [DEBUG] - [DEBUG] - Main - PID 67800 writt	<pre>new instance of instance.p90.m596ff9b CSVtoRESTDeviceType:latest process s en to file c:\Users\raa\Desktop\Unifie</pre>	ael40399688el84ab70f69 tarting on PID:67800 r\SmartUnifierManager-	windows-x64_15_02_22\deploy\90596ff9-bae1-4039-9688-e184ab70f698\pid.txt	Memory Usage	(0%)
	2022-02-15 11:23:34,762 - [DEBUG] - [DEBUG] - ConfigurationResolver\$ 2022-02-15 11:23:34,762 - [DEBUG] - [DEBUG] - ConfigurationResolver\$		ource. Resolving path=			
	2022-02-15 11:23:34,767 - [DEBUG] - [DEBUG] - RestServerImplementati	on - demol:RESTServer:latest - unMar		figuration Config(SimpleConfigObject(("endpoint":("defaultContentType":		
	<pre>{ enableOatalogging :false, enableLogging :false}, statefulvariableL {"enableOatalogging":false, "enableLogging":false}}}, "pathPrefix":"de</pre>		ablecogging (talse), s	raretessyaliaoieroÄätuä :		
	2022-02-15 11:23:36,030 - [INFO] - [INFO] - RestServerImplementati					
				oggingAwareConfiguration(ArtifactLoggingConfiguration(false,false),ArtifactLoggingConfig		
	2022-02-15 11:23:36,031 - [DEBUG] - [DEBUG] - RestServerImplementati					
				ServerEndpoint(127.0.0.1,8091,ApplicationJson,None,None) exist. Creating a new one		
				•		
	4			*		

4.7.2 How to manage Notifications

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

≡	MORPH.	oro								SMART UNIFIER 🚑 8. ()
	•								=	View All Notifications
>	🌲 Deployme	nt								Instance Error: demo1:CSVtoRESTDeviceType + 4
Ŀ	Group Filter	<	Gr	oup 🕈	Name	Version	Deployment Type	State	1	25 minutes ago
•	 Show All demo1 		🗌 de	mo1	CSVtoRESTDeviceType	latest	Local	Stopped	9	Instance Error: demo1:CSVIORESTDeviceType FileTailer - demo1:CSV_latest - File c.\Users\kab\DesktopItestCSV.csv does not exist.
***									0	Instance Error: demo1:CSVt0RESTDeviceType FileTailer - demo1:CSV_latest - File c:\Users\kab\DesktopitestCSV.csv does not exist 24 4 minutes ago
↔									0	Instance Error: demo1:CSVtoRESTDeviceType FileTailer- demo1:CSVlatest -File c:\Users\kab)DesktopitestCSV.csv does not exist 24 Aminutes ago
.									0	Instance Error: demo1:CSV10RESTDeviceType FileTaller- demo1:CSV_latest-File c:\Users\kab\DesktopitestCSV.csv does not exist 24 minutes ago
A									0	Instance Error: demo1:CSVtoRESTDeviceType FileTailer - demo1:CSV_latest - File c\Users\kabiDesktopitestCSV.csv does not exist 24 minutes ago
۲									_	

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

Notification	ons				Q	Ð
	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	×	Î
	23	InstanceError	FileTailer - demo1:CSV:latest - File c:\Users\kab\Lesktop\testCSV.csv does not exist	15 Feb, 2022 11:23:49	×	Î
	22	InstanceError	FileTailer - demo1:CSV:_Jatest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:46	×	Î
	21	InstanceError	FileTailer - demo1:CSV:_Jatest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:43	×	Î
	20	InstanceError	FileTailer - demo1:CSV:_Jatest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:40	×	
	19	InstanceError	FileTailer - demo1:CSV:_Jatest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:37	×	Î
	18	InstanceError	FileTailer - demo1:CSV:_Jatest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 10:42:14	5	

After selection a pop-up appears providing two options.

5		6				
Nor					۹	. •
~	: ID	Event Type	Message	Event Time 🗸		
~	24	InstanceError	FileTailer - demo1:CSVlatest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:52	×	Î
~	23	InstanceError	FileTailer - demo1:CSV_latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:49	×	
\checkmark	22	InstanceError	FileTailer - demo1:CSV:jatest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:46	×	Î
~	21	InstanceError	FileTailer - demo1:CSV:_latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:43	R	Î
~	20	InstanceError	FileTailer - demo1:CSV:latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:40	×	Î
~	19	InstanceError	FileTailer - demo1:CSV:_latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:37	R	Î
¥	18	InstanceError	FileTailer - demo1:CSV:latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 10:42:14	*	Î

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.

= /	MO	RPH.	pro				SMARTUNIFIER	8	0
>	🌲 No	tificati	ons					۹	c,
Ŀ		:	ID	Event Type	Message	Event Time 🕹			
.			24	InstanceError	FileTailer - demo1:CSVlatest - File c:\Jsers\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:52	2	5	i.
***			23	InstanceError	FileTailer - demo1:CSVlatest - File c:\Users\kab\Desktop\testCSV csv does not exist	15 Feb, 2022 11:23:49	2	5	Î
«·· »			22	InstanceError	FileTailer - demo1:CSVlatest - File c:\Users\kab\Desktop\testCSVcsv does not exist	15 Feb, 2022 11:23:46	2	5	Î
Ð			21	InstanceError	FileTailer - demo1:CSVlatest - File c:\Users\kab\Desktop\testCSVCsv does not exist	15 Feb, 2022 11:23:43	2	5	Î.
0			20	InstanceError	FileTailer - demo1:CSVlatest - File c:\Users\kab\Desktop\testCSV csv does not exist	15 Feb, 2022 11:23:40	2	5	Î
			19	InstanceError	FileTailer - demo1:CSVJatest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:37	2	5	Ĩ.
Â			18	InstanceError	FileTailer - demo1:CSVJatest - File c:\Jsers\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 10:42:14	2	5	i.
•									

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

≡	MORPH.pro						SMART UNIFIER A 🕘 🧿
>	A Notifications						Q G
Ŀ		:	ID	Event Type	Message	Event T	ime 🗸

⟨ ··⟩							
þ							
۲							
\$							
•							

4.8 How to monitor a deployed Instance

4.8.1 Log Viewer

SMART**UNIFIER** comes with an integrated log viewer, which helps to gain insights in deployed and running Communication Instance.

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

Log Levels

- **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 an production environment. The DEBUG log level should be used for troubleshooting an 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 operation

Logs can be accessed by clicking the "Log" button (1).

>	A I	Deployment					९ 💾 + 🕁 🏚
Ŀ	ہ د	Group 🛧	Name	Version	Deployment Type	State	1
•	o FILTE	demoscenario.csv2rest	SUInstance	1.0.0	Local	Started	> = 🖑 🔢 🖍 🗊
***	GROUF						

Log Viewer comes with the following features:

- Font size adjustability (2)
- Searching, based on a regular expression (Regex) (3)
- Start/Stop to "freeze" the current logging in order to investigate already printed log lines (4)
- Follow Tail, to skip through to the latest log line (5)

Log Viewer for deploy	rment ld 06484d90-fe4e-49b0-873a-42234162506c:1.0.0	11 1	2
2021-08-26 15:58:51,668 - [DEBUG] - 2021-08-26 15:58:51,762 - [DF0] -	[aa1] - Biol 1: Start IS Early Estates Estates (asses).4689.4680.668734520322382584; asses).5.1.5.1.5.1.5.1.5.1.5.1.5.1.5.1.5.1.5.	av. 🔮	
2021-08-26 15:58:51,668 - [DEBUG] - 2021-08-26 15:58:51,762 - [INFO] -	[and] - min - PD 3512 within to file Correction/BETRITION/BETRITION/Part (and and visco) fileming-vision-abi (add) passes/a-file-add) add (add) add) add (add) add) add (add) add)	18.p4d90.nfete4	2968873a
2021-08-26 15:58:51,765 - [DEBUG] - 2021-08-26 15:58:51,911 - [DEBUG] -	[main] - Nain - PD 35432 written to file C1Project120MURUITER1/amini/Fastures1cq0iaaer/SastUnifferUnger-sindem-ads-3/deplay/0608609-fels-0008-773-a223462308C/pid.tt main] - ConfigurationResoLverS - UmresoLver 04: Config Loaded From Fesser 04: 090-10, 0 - 640-4008-773-4224462308C588.ion; Benzlung path-, Complete config: Config/SimpleConfigDirect(1"commands";11,"events";11"res	rane[i]terRev[i	
2821-88-26 15:58:51,912 - [DEBUG] -	and) = configurationsolves = config lander from resource eds.dog land.efset-non-offset	nt/csvDemoEvent	enn, si
2021-08-26 15:58:51,915 - [DEBUG] - 2021-08-26 15:58:52,856 - [INFO] -	[main] = ConfigurationResolver5 = Werged config: Config(SimpleConfigDipert(["commands"]]),"werts:[["respectation"]; "respectation"]),""," "respectation"; "," "respectation; ","," "respectation; ",","," respectation; ",",",",",",",",",",",",",",",",",",",	':{"type":"Seco	ands"}}
	<pre>mail > Usering(recently): usering(recently): (second): (secon</pre>	Jui(/Evenu/csvo	Jenoeven
1 2021-08-26 15:58:52,868 - [DEBUG] -	main) - ConfigurationResolver\$ - 'Unresolved' Config loaded from resource 0648.4d90 1.0.0 fe4e-4950-873a-42234162506c config-channel-c3798ec0-4064-4576-956d-082d38bc56a8-layer-filetailer2string-0.ison: Resolving path=. Complete config: Config(SimpleConfigObject(("de	lay":{"length":	:250, "un
2021-08-26 15:58:52,869 - [DEBUG] -	main] - ConfigurationResolvers - Merged config: Config(SimpleConfigObject(["delay":["length":250, "unit":["type":"Milliscconds"]},"file":"C:\Projects\\SMARTURIFEE\\Source_for_Demo\LData\\CSV-files\\Equipment1.csv", "reopenSetweenChunks":true,"tailFronEnd":true]))		
2021-08-26 15:58:52,892 - [INFO] - 2021-08-26 15:58:52,893 - [INFO] -	[mais] - FileTallerfotringuyer - 0708ees44064-0370-05086-0822888C5483:1999-FileTallerTotring-0 artificity channel 0709066-08240806C5643:1999-FileTallerTotring-0 with config FileTallerTotring-0 with config FileTallerTotring-0 artificity of the after of the artificity of the artif	_Demo\Data\CSV-	-files()
2821-08-26 15:58:52,949 - [DEBUG] -	main] - Vuriabilatype - Vuriabila defined MATINR, String		
2021-08-26 15:58:52,950 - [DEBUG] -	(main) - TVariableType - Variable defined TIMESTAMP, String		
2021-08-26 15:58:52,951 - [DEBUG] - 2021-08-26 15:58:52,952 - [DEBUG] -	jazioj - Varizbielype - Varizbie defined TEMPERATUR, String maini - Varizbielype - Varizbie defined TESERT. String		
2021-08-26 15:58:52,952 - [DEBUG] - 2021-08-26 15:58:52,961 - [DEBUG] -	main) - twisionelpe - warizone ensurine Messions, string main - stysticationelpe - versions described and the string - stystication of the string - stystication of the string - style style style style string - style		
2021-08-26 15:58:52,975 - [DEBUG] -	[main] = ConfigurationResolver\$ - 'Unresolved' Config loaded from resource 0648.4690_1.0.0_fe4e-4900-873a-42234162506c_config-channel-6876fa4a-2c9b-47c9-ac96-f8de6ec65f3b.json: Resolving path=. Complete config: ConfigUspect(["endpoint":["defaultContent]]	pe":{"type":"Ar	pplicat
2021-08-26 15:58:52,976 - [DEBUG] -	[main] - ConfigurationResolver\$ - Merged config: Config(SimpleConfigObject({"endpoint":{"defaultContentType":"ApplicationJson"},"ip":"localhoss","port":8001),"pathPrefix":"demo"}))		
	[min] - RestServer[mplementation - demoscenario.cov/rest:RESTServer:_1.0.0 - unMarshallConfiguration configuration Config(SimpleConfigO		
	(main) - Retriever[inplementation - democranic.csv/retrikSTServer_1.0.0 - Configuring channel 8076/abs/200-a7C0-a2C0-8486acc5750:1.0.0 with config Restienver[inplementationConfiguration(demo,Resterverindpoint(localhost,8001,Application)ton,None,None)) (main) - Retriever[inplementation - democranic]. Configuration: Restienver[inplementationConfiguration(demo,Resterverindpoint(localhost,8001,Application)ton,None,None))		
2021-08-26 15:58:53,000 - [DEBUG] -	main) - RestServerImplementation - demoscenario.csv2rest:RESTServer: 1.0.0 - No server for endpoint RestServerEndpoint(localhost.8001.Application]son.None.None) exist. Creating a new one		
2021-08-26 15:58:53,049 - [DEBUG] -	[main] - RestServerImplementation - demoscenario.csv2rest:RESTServer:1.0.0 - SSL not configured		
2021-08-26 15:58:53,055 - [DEBUG] -	[main] - Retiferver[inplementation - demoscenario.csv/next:NSTSterver:].0.0 - Configuration[loaded RestierverImplementationConfiguration(demo,RestServerEndpoint(localhost,B091,ApplicationIson,None,None)) main] - Yurajbellove - Yurajbellove - Variable defined Temoratur. String		
	NBIN : TWATEDIALY & VALUED CONTINUE C		
2021-08-26 15:58:53.070 - [INFO 1 -	main] - RestServerImplementation - demoscenario.csv2rest:RESTServer: 1.0.0 - localbost:8001/demo/Variable/RestDemoData added to GET bandler		
3 2021-08-26 15:58:53,071 - [INFO] -	[main] - RestServerImplementation - demoscemanio.csv2rest:RESTServer:1.0.0 - localhost:8091/demo/Variable/RestDemoData added to PUT handler		
1 2021-08-26 15:58:53,071 - [INFO] -	[and] - Interveringiamentation - descense do convertation environmentation environmentation and det of NU handle (and) - Interveringiamentation - descense convertation environmentation environmenta adde to 90 handle (and) - Interveringiamentation - descense do convertation environmentation e		
5 2021-08-26 15:58:53,072 - [INFO] - 6 2021-08-26 15:58:53,073 - [INFO] -	[Bi1] - KetServerippieRentation - demoscharto.cs/2/PetrikeSiserveri_1.e.u.e. iocs/index/VetipdeKots/Vetipale/VetipdeKots/VetipdeKots/Vetipale/Vetipale/VetipdeKots/Vetipale/VetipdeKots/Vetipale/VetipdeKots/Vetipale/Vetipale/VetipdeKots/Vetipale/Vetipa Vetipale/V		
2021-08-26 15:58:53.074 - [INFO] -	nain - RestServerImplementation - demoscenario.csv2rest:RESTServer: 1.0.0 - localhost:8091/demo/Variable/Pressure added to GET handler		
2021-08-26 15:58:53,074 - [INFO] -	main] - RestServerImplementation - demoscenario.csv2rest:RESTServer:_1.0.0 - localhost:8091/demo/variable/RestDemoData/Variable/Pressure added to PUT handler		
2021-08-26 15:58:53,074 - [INFO] - 2021-08-26 15:58:53,077 - [INFO] -	[main] - Restistereringlesentation - demoscenerio.csu/pertilSiSterer_1.0.0 - localhost:BB01/demol/wriable/Pressure added to PGST handle?B31-86-36 15:85:3,076 - [DBB05] - [main] - TAdapter - Init mappings [main] - TAdapter - Handling rules of mapping mapping.psz1.p5:86 + macdedAP995Haas32804777 ValueSAB.g200477 - Handling FalueSAB.g200478 - January Santas January January Santas January Santas January January Santas January Santas January January Santas January January January Santas January Janu		
2021-08-26 15:58:53,078 - [INFO] -	main - Maketer - Found rule sxylext on method rule sxylext		
2021-08-26 15:58:53,079 - [DEBUG] -	main] - TAdapter - Setting up mapping csv2rest		
	main) - EventDefinition - Listemer added for /Model/c3708ec8-4864-4576-9864-8824388-56a8/Event/csvDemoEvent		
2021-08-26 15:58:53,080 - [D1806] - 2021-08-26 15:58:53,081 - [DEBUG] -	main) - EventUmFinition - Listemer added for /Node/LC798ec8-4064-4576-906d-082d3Hbc56s8/Event/csvDemoEvent, mapping.pe222.p)549.mafc44de790934aaa328b9477.v1_0_0_Nsppings/c44de790934aaa328b94775\$Lambda\$633/Boo0000008080520400000000520400000000520400000000520400000000		
	umilip - / Mulpiter - John Lung, applementations : umiliper-io-transfel - CityString[df0606]Layer - demoscenario.csv2rest:CSV:_1.0.0 - Starting channel c3708ec0-4064-6576-9b6d-002d38bc56a8:1.0.0		
2021-08-26 15:58:53,273 - [DEBUG] -	unifier-io-thread-0] - DefaultModelImplementation - demoscenario.csv2rest:CSV: 1.0.0 - onStateChange('Stopped', Starting) invoked		
2021-08-26 15:58:53,273 - [DEBUG] -	[unifier-io-thread-8] - ModaliplementationFim - State changed From Stopped to Starting2021-08-26 15:58:53,274 - [DEBUG] - [unifier-io-thread-0] - DefaultModelImplementation - demoscenario.csv2rest:CSV:_1.0.0 - prestart() invoked		
2021-08-26 15:58:53,274 - [INFO] - 2021-08-26 15:58:53,277 - [DEBUG] -	(unffer-io-thread-0) - filefaller[obtringsyer - c1798ec-0864-0576-0964-08201065081:]per-filetaller2tring file taller layer (unffer-io-thread-0) - filefaller - c1798ec-0864-0576-0964-08805088:]per-filetaller2tring-file taller layer		
2021-08-26 15:58:53,278 - [DEBUG] -	[unifier-io-thread-8] - FileTailer - c3798ec8-4064-4576-9b6d-482d38bc56a8:layer-filetailer2string-8 - About to create tailer instance on file FileTailerToStringLayerConfiguration(C:\Projects\SMARTUNIFIER\Source_for_Demo\Data\CSV-files\Equipment1.csv,DurationConfigurationConfiguration(C)	tionType(258,M5	illised
2021-08-26 15:58:53,279 - [INFO] -	[unifier-io-thread-0] - FileListenerAdapter - c1798ec0-4064-4576-9b64-082d18bc56a8:layer-filetailer2string-0 - Tailer initialized for file C:\Projects\SWARUMIFIER\Source_for_Demo\Data\CSV-files\Equipment1.csv		
	unifier in thread eg > FileTailer - (3788c8-4064-4376-366d-882d38bc55a8:layer-filetaileristring.0 - Tailer Started for (:Projects)SMRTUNIFIEN/Source_for_Demo(Data/CSV-files/Equipment).csv. Read delay 250 ms, ReopenBetweenChunks-true, TailFromEnd-true		
2021-08-26 15:58:53,280 - [INFO] - 2021-08-26 15:58:53,283 - [DEBUG] -	umfife-i-o-thread-0) - CxViringTeModellayer - democemaria.cxVzext(SV_1.0.0 - Channel cJDNecd-0060/402/300554811.0.0 successfully started umfife-i-o-thread-0) - Defaultioning - democemaria.cxVzext(SV_1.0.0 - Online(-America).comected invoked		
2021-08-26 15:58:53,283 - [DEBUG] -	[unifier-io-thread-0] - ModelImplementationFsm - State changed from Starting to Connected		
2021-08-26 15:58:53,284 - [DEBUG] -	[unifier-io-thread-0] - DefaultModelImplementation - demoscenario.csv2rest:CSV: 1.0.0 - postStart() invoked2021-08-26 15:58:53,284 - [IN#0] - [unifier-io-thread-1] - RestServerImplementation - demoscenario.csv2rest:RISTServer: 1.0.0 - Starting channel 6876fa4a-2ci	5-47c9-ac96-f8d	de6ec65
2021-08-26 15:58:53,286 - [DEBUG] -	[unifier-io-thread-1] - DefaultModelImplementation - demoscenario.csv2rest:RESTServer:10.0 - onStateChange('Stopped', Starting) invoked		
2021-08-26 15:58:53,286 - [DEBUG] - 2021-08-26 15:58:53,627 - [INFO] -	unffar-io-thread-1 - Modlipplasentation'ss - State changed from Stopped to Starting Uniffar-io-thread-1 - Mestimerine/Indexemplosentation - demoscare/io-curver: 18,0# - Onamel 6076/434-2020-4769-6064-6065/10:1.0.8 successfully started		
2021-08-26 15:58:53.628 - [DFRUG] -	uminer-to-tm?mod-1) = ketterveringutematation = demostemario.csv/retrixolsterver:_1.e.e.= (uminet_ae/name_cc)=-acto-scone=hostecconial.ie.e succession1.e.e succes		
2021-08-26 15:58:53,628 - [DEBUG] -	unifier-io-thread-1 - ModelImplementation(sm - State changed from Starting to Connected		
2021-08-26 15:58:53,628 - [DEBUG] -	junfjer-jo-treest-ji - Defailviskilapiseettion - demosimario.csv2restijkilSterver_1.0.0- ostitart() invoked mainj - Main - Instance demosimario.csv2restijkilstance_168 uscessivily started		

4.8.2 Dashboard

- In order to monitor an Instance, access the Dashboard view by clicking the "Dashboard" button (6).
- If the Instance is in the state NotDeployed the Dashboard cannot be accessed.

Å 1	Deployme	ent					Q (¹) + 5-
R 、		Group 🛧	Name	Version	Deployment Type	State	
IP FILTE		test	Instance1	latest	Local	Started	> = 🖑 🖽 🖾 🗡 🗊
GROUI							6

- 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

Status Dashboard: test:Instance1:latest

nfo	Туре	Status	Model	Messages	 ⊘ Status ③ Time start 202: 	Started 2-03-15 11:50:08
est:Out:latest	In Memory	ConnectedState	test:Model1:latest	4 / 9800	⑦ Time Up	3h 13min 47s
est:in:latest	In Memory	ConnectedState	test:Model1:latest	4 / 9800	_	
Log Viewer: 5d7d062a-	8d5c-4b43-97b9-72f7141	5ba61:latest		A	⊕ CPU Usage	(0% / 19%)
2022-03-15 15:03:49,963 - [INFO] - 107701t?)	VariableDefinition - InMemoryModelIm	plementation-Get Variable: /Model/fbea8	4f4-b045-46e1-909f-a46dae86233a/Event/e1/Variable 4f4-b045-46e1-909f-a46dae86233a/Event/e1/Variable 85e-0c09-473b-a24b-7a3eae5fbd79/Event/e1/Variable	e/v1=Some(?		
2022-03-15 15:03:49,064 [INFO] 2022-03-15 15:03:49,064 [INFO] 2022-03-15 15:03:50,0695 [INFO] 2022-03-15 15:03:50,0695 [INFO] 2022-03-15 15:03:50,0695 [INFO] 2022-03-15 15:00:50,0665 [INFO] 2022-03-15 15:00:50,0665 [INFO] 2022-03-15 15:00:50,0666 [INFO]	Variablebrinition - InKenorykoozlin Variablebrinition - InKenorykoozlin	Diementation-Set Variable: //bode1/96806 Diementation-Set Variable: //bode1/free8 Diementation-Set Variable: //bode1/free8 Diementation-Set Variable: //bode1/fbe8 Diementation-Set Variable: //bode1/9808 Diementation-Set Variable: //bode1/9808 Diementation-Set Variable: //bode1/9808 Diementation-Set Variable: //bode1/9808	$\label{eq:2} \begin{array}{c} 444-3654-646-36976-4666-8672323076-00-0742/Write3b0 \\ 65-600-372-36-26675-7072-00-0742/Write3b0 \\ 454-3054-642-3097-466668523376-10-072-10-074-074-074-074-074-074-074-074-074-07$	<pre>//-2-Some(-1-48350113) ///1-Some(254767X) //2-Some(1532943131) //1-Some(1512942131) ///2-Some(1512943131) ///2-Some(1512943131) ///2-Some(1512943131) //2-Some(1512943131) //2-Some(1512943131)</pre>	Memory Usage	(8%)
1022-03-15 15:03:51,004 - [INFO] 1022-03-15 15:03:51,005 - [INFO] 1022-03-15 15:03:51,005 - [INFO] 1022-03-15 15:03:51,616 - [INFO] 1022-03-15 15:03:51,616 - [INFO] 1022-03-15 15:03:51,616 - [INFO] 1022-03-15 15:03:51,616 - [INFO] 1022-03-15 15:03:51,617 - [INFO]	VariableDefinition - InMemoryModelIm WariableDefinition - InMemoryModelIm VariableDefinition - InMemoryModelIm VariableDefinition - InMemoryModelIm VariableDefinition - InMemoryModelIm VariableDefinition - InMemoryModelIm	plementation-Set Variable: /Model/98dd8 plementation-Get Variable: /Model/7be80 plementation-Set Variable: /Model/7be80 plementation-Set Variable: /Model/7be88 plementation-Set Variable: /Model/7be88 plementation-Set Variable: /Model/7be88	444-005-4621-0097-4640680338/tvent/cl/variabl 68-009-4730-264-78406547037/tvent/cl/variabl 68-009-4730-264-784064603334/tvent/cl/variabl 644-0054-6621-0097-46406803331/tvent/cl/variabl 644-0054-6621-0097-46406803331/tvent/cl/variabl 644-0054-6621-0097-46406803331/tvent/cl/variabl 644-0054-6621-0097-46406803331/tvent/cl/variabl 644-0054-6621-0097-46406803331/tvent/cl/variabl 644-0054-6621-0097-46406803331/tvent/cl/variabl 644-0054-6621-0097-46406803331/tvent/cl/variabl	<pre>//1-Some(+ko?? B)?) //1-Some(140576809) //2-Some(2140578699) //1-Some(172727379024) //1-Some(17272739024) //1-Some(172752_7272) //1-Some(172752_7272)</pre>	Messages / sec	(8)

4.9 Additional Options

4.9.1 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	Ŧ
	Log File Configuration * Info	Ŧ
1	Zenable Encryption	
2	Same folder as deployment	
3	Same folder as deployment Custom Path:	
-		

4.9.2 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 *	•
Enable Encryption	
V 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).

A Deployment				Q 🖞 + & 🏚		
Group Filter <	Group 🛧	Name	Version	Deployment Type	State	
 Show All ex1 	ex1	CSVtoRESTDeviceType		Local cted Instance e instance name to continue.	NotDeployed	2
			3 Instanc	e Namek Cancel		

Note: Protected Instances will not work with Bulk actions.

4.9.3 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	Ŧ
Log Pile Configuration * Info	•
Enable Encryption	
Protected	
VM Arguments	
1 JMX Properties	
JMX Host Name * localhost	
2 JMX Pert * 1280	
JMX Authenticate	
Use SSL	
Default Arguments	
XMS*	
4 32m	
XMX *	
5 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.

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	6
argument	0
-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
- Use Extensions

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.

	Active Directory Users and Comp	outers		- 🗆 X
	File Action View Help			
<	⊨ → 2 💼 🤾 🗈 🗙 🗉] 🖸 📑 🛛 🗊 🖏 🐄 🍞 🚨 🍇		
	Active Directory Users and Com	Name	Туре	Description
	> 🚞 Saved Queries	ADadmin SmartUnifier	User	
· ·	✓ jii test.local	Administrator	User	Built-in account for administering the computer/domain
	> 🧰 Builtin	Allowed RODC Password Replication Group		Members in this group can have their passwords replicated to all read-only domain controllers in the
	> Computers	arpad.kiss	User	······································
	> 📓 Domain Controllers	Axel F	User	
	> ForeignSecurityPrincipal:	🕹 bogdan.irimie	User	
	> Managed Service Accour	d cd_21	User	
	📔 Users	Cert Publishers		Members of this group are permitted to publish certificates to the directory
		Cloneable Domain Controllers		Members of this group that are domain controllers may be cloned.
		sorneliu.bogdan	User	
		DefaultAccount	User	A user account managed by the system.
		Replication Group	Security Group	
		A DnsAdmins	Security Group	DNS Administrators Group
		A DnsUpdateProxy		DNS clients who are permitted to perform dynamic updates on behalf of some other clients (such as I
_		R Domain Admins		Designated administrators of the domain
		Bomain Computers		All workstations and servers joined to the domain
		Domain Controllers		All domain controllers in the domain
		Bomain Guests		All domain quests
		Domain Users		All domain users
		Enterprise Admins		Designated administrators of the enterprise
		Enterprise Key Admins		Members of this group can perform administrative actions on key objects within the forest.
		Enterprise Read-only Domain Controllers		Members of this group are Read-Only Domain Controllers in the enterprise
		Siuliano De Sabata	User	members of this gloup are read-only bornain controllers in the enterprise
		Giulica	User	Contul meu pentru experimente
		Roup 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
		Lenkins Server	User	balle in account for gatest access to the compared adminin
		A DO	User	
		Key Admins	Security Group	Members of this group can perform administrative actions on key objects within the domain.
		Rev Admins		Members of this group are afforded additional protections against authentication security threats. See
		RAS and IAS Servers		Servers in this group can access remote access properties of users
		Read-only Domain Controllers		Members of this group are Read-Only Domain Controllers in the domain
		Sascha Fischer	User	memory of this group are need only bornam controllers in the domain
		Schema Admins		Designated administrators of the schema
		SmartUnfierAll	Security Group	oraginarea daministrators of the schema
		SmartUnifier Admin	User	
		SUAdmin	Security Group	
		SUWriter	Security Group	
		Not the second s	User	
	< >	<	030	
	× >	`		>

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 SMARTUNIFIER roles is defined in the **applica-tion.conf** file from the **conf** folder.

```
🔚 application.conf 🔣
     # <u>https://www.playframework.com/documentation/latest/Configuration</u>
 1
     include "default.conf"
  2
  3
     apiPrefix = adapter
  4
  5
     play = {
  6
      server = {
  7
         http.port = 9000
  8
         http.address = "0.0.0.0"
  9
 10
         #http.port=disabled
 11
         #https.port=9443
 12
         #https.keyStore.path="path_to_keystore"
 13
         #https.keyStore.password="keystore_password"
 14
       3
 15
       http.secret.key = "ChangeMySecret"
 16 }
 17
     authentication {
 18
             activedirectory {
 19
                 host = "192.168.0.132",
 20
                 port = 389,
                 baseDN = "DC=test,DC=local",
 21
                 useSSL = false,
 23
                 user = "jenkins@test.local",
                 password = "Aiurea05",
 24
 25
                 groupmapping = {
 26
                     Administrator = "SUAdmin",
 27
                     Writer = "SUWriter",
                     Reader = "SmartUnifierAll"
 28
 29
                 }
 30
             }
 31
         }
 32
     unifiermanager {
       tempFolder = "temp"
 33
 34
       compiler {
 35
         scala {
          javaHome = "jre"
 36
 37
         3
 38
         management {
 39
         dontDeleteWorkspace = true
 40
         }
 41
       }
 42
       model {
 43
         loadCodeFromScalaFile = false
 44
       }
 45
       deployment {
        javaHome = "jre"
 46
 47
         local = {
 48
          deploymentFolder = "deploy"
 49
           softRefreshInterval = 2000
           hardRefreshInterval = 5
 50
 51
           logStatusInterval = 30
 52
         monitorLogs = true
 53
         }
 54
         docker {
 55
           baseimage {
 56
             autocreate = false
 57
             jreImage = "adoptopenjdk/11-jre-hotspot"
 58
           }
 59
         }
 60
       }
 61
     3
```

As seen above (1) in the left side are the SMARTUNIFIER 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.

	MORPH.pro Smartunifier	
User ID		
JOD@test.local		
Password		

5.2 Backup and Restore

SMART**UNIFIER** provides the possibility to manually backup and restore the repository. The repository represents a central location in which all the configuration components are stored:

- Information Models
- Communication Channels
- Mappings
- Node Types

• Instances

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 the **Backup** perspective (3) or the **Restore** perspective (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					ረ ታ 🛆
Repository Database					
5 Group	Name	Version	Туре	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.

Backup				ር ታ 🙆
Repository Database				
Group	Name	Version	Туре	Description
demo1	RESTServer	latest	channel	
demoscenario.csv2rest	DataModel	latest	model	
demoscenario.csv2rest	RestDataModel	latest	model	
demoscenario.csv2rest	CSVtoREST Dependencies foun	d	mapping	
_ ex	Modell Do you want to select the d		model	
🗋 ex	CSv Yes 6	No	channel	
🗆 ex	Rest	latest	channel	

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

Backup 7	Q & Ø
Repository Database	9
8 🔽 Name	
INSTANCE INFORMATION	
INSTANCE_DEPLOYMENT_SETTINGS	
DEPLOYMENT_ENDPOINT_SETTINGS	
CHANNEL_TYPES	
BASELIMAGES	
USER_ACCOUNT	
VSER.ROLE	
CREDENTIALS	
INSTANCE_STATES	
VITIFICATIONS	

• 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.

Save As			
÷ → ▼ ↑ 📕 > NBRAA > Desktop > Exported Files 🛛 🚺			
Organize 🔻 New folder			I ∷ •
Name ^ versioning-local-NBRAA-01-10-16_3Atar.gz versioning-local-NBRAA-01-10-16_3Btar.gz	Date modified Type 1/10/2022 4/34 PM GZ File 1/10/2022 4/38 PM GZ File	Size 14 K8 14 K8	
2 File name: backup-NBRAA-03-07-15_48.tar.gz			
Save as type: GZ File (*.gz)			
Hide Folders			13 Save Cancel

Cancel

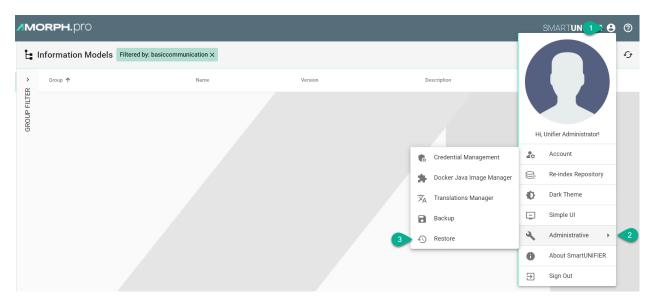
5.2.3 Restore

The Restore feature allows to copy the SMART**UNIFIER** configuration components from a backup to the original location.

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

Follow the steps described below to restore the SMARTUNIFIER 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!

4 Choose file No file chosen

Cancel

- 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.

Restore					९ म 🗟 🕙
Repository Database					
6 Delete all artifacts before restore					
8 Group	Name	Version	Туре	Description	
demoscenario.csv2rest	CSVtoREST This artifact will be overwritten.	latest	mapping		
7 demoscenario.csv2rest	DataModel A 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.

Restore						Q & 🔂 🚯
Repository Database						11
Delete all artifacts before restore						
Group	Name		Version	Туре	Description	
demoscenario.csv2rest	CSVtoREST A This artifact will be overwritten.		latest	mapping		
demoscenario.csv2rest	DataModel This artifact will be overwritten.		latest	model		
demoscenario.csv2rest	RestDataModel A This artifact will be overwritten.	Dependencies found	latest	model		
		Do you want to select the dependencies? Yes 10 No				

- 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.



• Click on the **Close** button to finish (12).

5.2.4 Manager Backup

In order to backup SMARTUNIFIER Manager make a copy of the SMARTUNIFIER 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	Туре	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						
题 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).

≡	MORPH.pro		
>			
Ŀ			

⟨··⟩	Integrate perfectly your	Hi,	Unifier Administrator!
f	Production-IT using	20	Account
۲		Cî,	Re-index Repository
A		Ф	Dark Theme
	SMART UNIFIER 2	÷	Advanced UI
•	ADVANCED IT-INTEGRATION PLATFORM	٩	Administrative
		0	About SMARTUNIFIER
		€	Sign Out
	AMORPH.pro		
	AMORPH.pro SMARTUNIFIER		

• Click on the **Channel Types** button (3) to open the Channel Types perspective.

≡	MORPH.pro SMARTUNIFIER &	0
>		
Ŀ		
A	3	
## #	Integrate perfectly your	
⟨··⟩	Production-IT using	
í		
٥	SMARTUNIFIER	
â	ADVANCED IT-INTEGRATION PLATFORM	
•		
	SMARTUNIFIER	

• The main view of the Channel Types is visible.

>	🏠 Channel Types						م ٿ) + 6
Ŀ	Group Filter <	Group 🛧		Name	Version	Description		
A	 ✓ Show All ✓ com 	com.amorphsys.unifier.e	hannel	Sql Database	1.0.0		Ċ	Z 1
***	 ✓ amorphsys ✓ unifier channel 	com.amorphsys.unifier.o	hannel	InfluxDB	1.0.0		Ċ	/ 1
<>		com.amorphsys.unifier.o	hannel	In Memory	1.0.0		Ċ	/ 1
		com.amorphsys.unifier.	hannel	Iso-On-Tcp Client	1.0.0		Ċ	/ 1
Í		com.amorphsys.unifier.e	hannel	Modbus Tcp Client	1.0.0		Ċ	/ 1
٥		com.amorphsys.unifier.o	hannel	OPC UA Server	1.0.0		Ċ	/ 1
â		com.amorphsys.unifier.e	hannel	OPC UA Client	1.0.0		Ċ	/ 1
		com.amorphsys.unifier.o	hannel	Rest Server	1.0.0		Ċ	/ 1
		com.amorphsys.unifier.o	hannel	Rest Client	1.0.0		Ċ	/ 1
		com.amorphsys.unifier.o	hannel	SecsGem Client	1.0.0		Ċ	/ 1
		com.amorphsys.unifier.o	hannel	Websocket client (JSON)	1.0.0		Ċ	/ 1
		com.amorphsys.unifier.o	hannel	Websocket client (XML)	1.0.0		Ċ	/ 1
		com.amorphsys.unifier.	hannel	Websocket client (CSV)	1.0.0		Ċ	/ 1
		com.amorphsys.unifier.e	hannel	SFTP file writer (JSON)	1.0.0		Ċ	/ 1
		com.amorphsys.unifier.	hannel	SFTP file writer (XML)	1.0.0		ن ث	1

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.

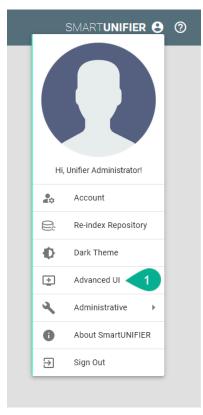
As an example for such a layer stack you can see below the layer stack for the *MQTT Communication Channel*:

- 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 SMARTUNIFIER menu and select Advanced UI.



2. Go to the Channel Types perspective by clicking the **Channel Types** button.

≡ ⊿MORPH .β	oro
SMARTUnifier Configuration	<
Information Models	Ŀ
Channel Types	A 2
Communication Channels	###
Mappings	< ·· >
Device Types	
Instances	٥
Deployments	٨
Deployment Endpoints	•
User Management	**

3. Click on the Add button in the upper right corner.

>	🏠 Channel Types					۹ 🖞 +	÷
Ŀ	Group Filter <	Group 🛧	Name	Version	Description	3	ĵ
	 Show All ✓ com 	com.amorphsys.unifier.channel	SqlDatabase	1.0.0		Ů 🖊	
***	 amorphsys unifier channel 	com.amorphsys.unifier.channel	InfluxDB	1.0.0		Ċ 🖍	1

- 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.

>	> ③ Add Communication Channel Type			6 🖬 🖒 😣
Ŀ,	te Group *			
•				
***	Name " MQTT_JSON			
< ·· >	↔ Version * 1.0.0			
đ				
۲				
A	A		DEVICE	
•	S Layer type * MQTT to String	*	MQTT to String	
**	Layer type * String to Json	-	String to Json	
	Layer type * Json to Model	*	Json to Model	
			SMARTUNIFIER	

5.4 Docker Java Image Manager

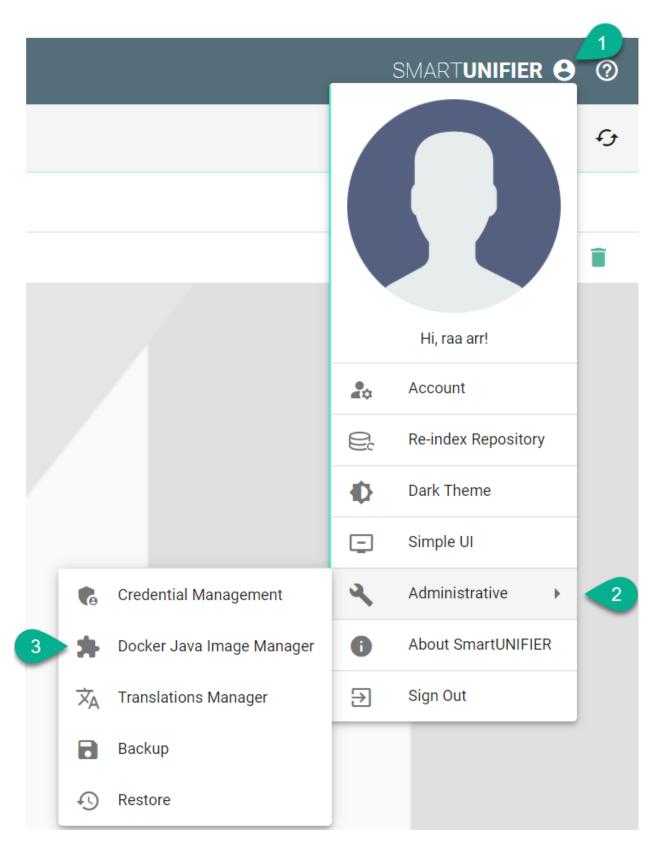
SMART**UNIFIER** supports the Deployment of Instances using Docker Containers using different Java base images. With the Docker Java Images Manager, the user can create and maintain different versions of Docker Java images.

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.

*	ocker Java Image Manager		Q + 5
> ~	group 个	name	
> FILTE	demo.basiccommunication	DJI1	/ 1
GROUF			

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).

=	SMARTUNIFIER 🔒 🕜		
>	🇯 Docker Java Image Manager		Q + 5
Ŀ	> group ↑ ∰	name	1
;;;	OUP FILT		
⟨ ⟩	GRO		

- 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).

≡	MORPH.pro		SMARTUNIFIER	8 0
>	🕀 Add Docker Java Image		3 🖬	ıс ×
Ŀ:		Group *		
## #	2	> demo		
‹·· >		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).

=	MORPH.pro	Smartunifier \varTheta 🕜	
>	🗯 Docker Java Image Manager		Q + 5
Ŀ:	> group ↑	name	
***	비 H demo d	Image1	() 🗡 🔋

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).

=	MORPH.pro		Smart unifier 😔
>	🛸 Docker Java Image Manager		Q + 5
Ŀ:	> group ↑	name	9
***	H H demo	Image1	/ 1
	Inc		

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.



• The main view of the Deployment Endpoints is visible.

>	P C	eployment Endpoints				ዓ 💾 + 😏
Ŀ	х х	Group 个	Name	Туре	State	
***	P FILTER	default	Default	Local	Running	Þ 🗉 🖒 🗡 🥫
<>	GROUP					
6						
٥						
â						
•						

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 SMARTUNIFIER Deployment Endpoints perspective (1).
- Click on the Add Endpoint button (2).
- Select the Deployment Type Local from the pop-up (3).

							2	
>	e 1	eployment Endpoints					ላ 🖞 🕂	-C3
Ŀ	› «	Group 🛧	Name	Туре	State	3	Local	
***	P FILTER	default	Default	Local	Running	►	Docker AWS	
(··)	GROUP						SSH	
6								
٥								
A								
•								

- 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 * c2-Program Filesiava 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 SMARTUNIFIER Deployment Endpoints perspective (1).

	≡ ∕MORPH .β	oro
	SMARTUnifier Configuration	<
	Information Models	Ŀ
	Channel Types	.
	Communication Channels	## #
	Mappings	< ·· >
	Device Types	ð
	Instances	٥
	Deployments	٨
1	Deployment Endpoints	•
	User Management	**

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

					2	
P (Deployment Endpoints				ላ 💾 🕂	+ •
>	Group 🛧	Name	Туре	State	Local	
TER					3 Docker	-
P FIL	default	Default	Local	Running	AWS	
GROU	demo	LocalEndpoint1	Local	Stopped	•	
-					SSH	

- 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)**.

		5
① Add Docker Endpoint		() 🗈 🖒 ×
4	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 SMARTUNIFIER Deployment Endpoints perspective (1).

	≡ / MORPH.β	oro
	SMARTUnifier Configuration	<
	Information Models	Ŀ
	Channel Types	.
	Communication Channels	** *
	Mappings	< ·· >
	Device Types	þ
	Instances	٢
	Deployments	â
1	Deployment Endpoints	•
	User Management	**

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

e C	eployment Endpoints				2 Q 🖞 +	£
× «	Group ↑	Name	Туре	State	Local	
FILTE	default	Default	Local	Running	Docker	Î
GROUF	demo	LocalEndpoint1	Local	Stopped	AWS	Î
-					SSH	

- 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 SMARTUNIFIER to connect to your AWS account.
- Save the new Endpoint by clicking the **Save** button (5):

Add AWS Endpoint		() 🖯 🔁 () 🛠
	Group * demo Name * AWSAccount1	5
4	Account ID * Region * eu-central-1 * Access Key ID *	
	Secret Access Key *	

SSH

SMARTUNIFIER supports the Deployment of Instances using SSH protocol.

• Navigate to the SMARTUNIFIER Deployment Endpoints perspective (1).

	≡ ⊿MORPH .β	oro
	SMARTUnifier Configuration	<
	Information Models	Ŀ
	Channel Types	•
	Communication Channels	;;;
	Mappings	⟨…⟩
	Device Types	ē
	Instances	٥
	Deployments	٨
1	Deployment Endpoints	-
	User Management	**

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

P 0	eployment Endpoints				오 법 +	£,
, ,	Group 🕈	Name	Туре	State	Local	
PILTE	default	Default	Local	Running	Docker	Î
GROUF	demo	LocalEndpoint1	Local	Stopped	AWS	
					SSH	

- 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).

⊕ Add SSH Endpoint		() 🖬 🔂 🗙
	Group * demo	5
	Name * VM1	
	Hostname * 192.168.56.101	
	Pert * 22	
	Username * ubuntu	
4	Pasword	
	Private Key	
	Connect Timeout * 5000	
	Java Home * usr/iib/ym/java-11-openjdk-amd64/	
	Deployment Folder * /opt/amorph/smartunifier/deploy/	
	Soft Refresh Interval * 5000	
	Hard Refresh Interval * 5	

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

Deploymen	nt Endp	oints				Q [¹] + G
Group Filter	<	Group 🛧	Name	Туре	State	
✓ Show All default		default	Default	Local	Running	ト 🛛 🖉 🧵
demo		demo	DockerE1	Docker	Stopped	> = Ċ 🗡 🔋
		demo	SSHE2	SSH	Failure	 ٺ

For the **Failure** state, hover over it (1) and a pop-up will display the error (2).

	eployment Endpoints					Q	Ľ	+ 9
×	Group 🛧	Name	Туре	State				
P FILTER	default	Default	Local	Running	►	• (t /	
GROUP	demo	DockerE1	Docker	Stopped	•	= (ٹ ٹ	
	demo	SSHE2	SSH	Fallure 1	•	= (ٹ ٹ	
			2	Failed to create and clear for 192.148.38.101:32				

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)**.

e (Deployment Endpoints				Q (L) + 67
> ~	Group 🕈	Name	Туре	State	
FILTE	default	Default	Local	Stopped	> = 🖒 🖍 🖡
GROUF	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)**.

•	Deployment Endpoints						Q	Ľ	+ -	9
> er	Group 🛧	Name		Туре	State					
GROUP FILTER	default	Default		Local	Stopped	•		5 /	r i	i
GROUF	demo	LocalEndpoint1		Local	Failure	•	- C	ט ע	r i	i
		4	Error Java Home directory does	not exist °c∶\Users\ra	a\Desktop\Unifier\SmartUnifierManager-windows-x64_states_00_07_22\r' OK 5					

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)**.

	Deployment Endpoints				Q [¹] + G
> ~	Group 🕇	Name	Туре	State	
FILTE	default	Default	Local	Stopped) = 🖒 🗡 📋
GROUF	demo	LocalEndpoint1	Local 2	Stopped	🔰 Č 🖉 🖡
					1

Delete Endpoint

To remove a Deployment Endpoint, click on the **Delete** button (1) and confirm the action (2).

e 1	eployment Endpoints				Q 💾 + G
> ~	Group 🛧	Name	Туре	State	
GROUP FILTER	default	Default	Local	Stopped	ト = 白 / 盲
GROUF	demo	LocalEndpoint1	Local	Stopped) = Ċ / ī
			Delete	Endpoint	
			Are you su	re you want to delete this endpoint?	
				Cancel Delete	

Edit Endpoint

To edit a Deployment Endpoint, click on the **Edit** button (1).

	Deployment Endpoints				Q 💾 + G
er 、	Group 🕇	Name	Туре	State	
0 FILTE	default	Default	Local	Stopped	> = 🖒 🗡 🥫
GROUF	demo	LocalEndpoint1	Local	Stopped	 المراجع
-					1

In the Deployment Endpoint edit view update the configuration **(2)** and click on the **Save** button **(3)**.

Edit Local Endpointdemo:LocalEndpoint1	(⊕ 🖬 🖒 ×
group * demo Name * LocalEndpoint1 Jaxa Home * C'Program Files/java Deployment Folder * Local Sott Refresh Interval * 5000 Hard Refresh Interval * 6 Log Status Interval * 30 Image: Monitor Logs	3

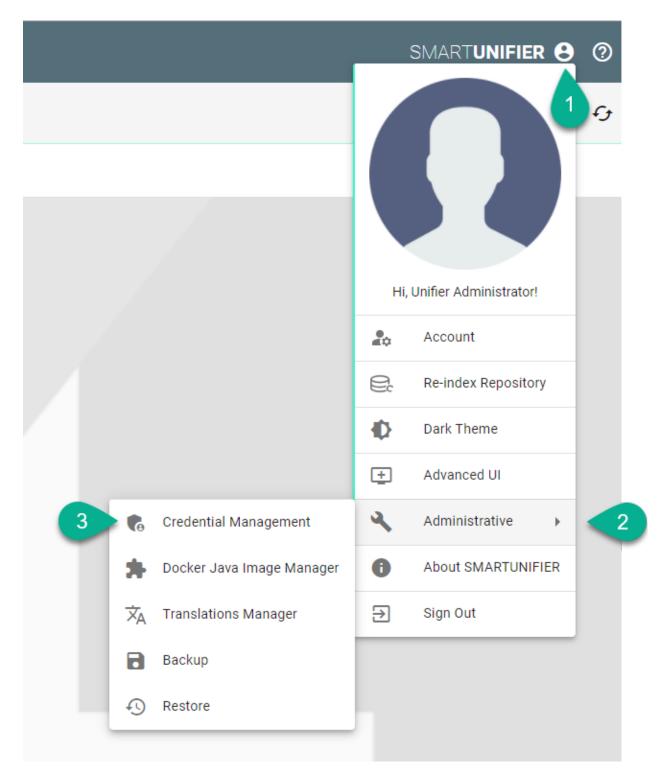
5.6 Credential Management

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.

Credential Management			< 💾 + ይ
Name	Description	Туре	

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.

6 Credential Management			ላ 🖞 + ኇ
Name	Description	Туре	Password 1
			Username and password

- Type a name for Credentials (3).
- Add description (4) (optional).
- Input the Username and Password (5).

Add Credential	🖬 16 ×
3 Name * TestSQLCredentials 4 Description 4 Credentials to access the Test SQL Server Username * testsqluser 9 Password * typethepassword	6

• Click on the Save and Close button (6).

Credential Management			Q [¹] + 63
Name	Description	Туре	
TestSQLCredentials	Credentials to access the Test SQL Server	UserNameAndPassword	/ 1

5.6.3 Edit Credentials

To edit the credentials, select the **Edit** button (1).

🔹 Credential Management			ዲ 🖞 + 🕁
Name	Description	Туре	
TestSQLCredentials	Credentials to access the Test SQL Server	UserNameAndPassword	1
			4

The Edit Mode is visible, the configuration can be edited (2) and then save the session by selecting the **Save and Close** button (3).

Edit Credential: TestSQLCredentials		5;	×
2 Name * TestSQLCredentials Description Credentials to access the Test SQL Server Username * testsqluser Password * typethepassword	3	3	

5.6.4 Delete Credentials

To delete credentials, select the **Delete** button (1).

Credential Management			Q [¹] + 45
Name	Description	Туре	
TestSQLCredentials	Credentials to access the Test SQL Server	UserNameAndPassword	/ 1
			٥

A pop-up confirmation appears, select the **Delete** button (2).



5.6.5 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 Database Communication Channel configuration:

• Click on the Database credentials field (1).

### Channel Configuration: demo.scenariodb:DataBase:latest		↔ ✓ X
M db Q ♀ ▲ ② DatabaseSelect [Command_DatabaseSelect] ▲ ③ Parameters [RequestType] ▲ ③ OrderH [Simp] ▲ ③ OrderH [Simp] ▲ ③ Customer [String]	SQL Database Database connection Type * SOLServer Reconnect interval * 10 Jose Uni * Jobe Uni * Job	· · ·

• Select the Username and password credentials reference option (2).

Database connection	
Гуре *	
SQLServer	
Reconnect interval *	
10	
JDBC Url *	
dbc:sqlserver://192.168.0.111:1433;databaseName=unifier	
Stabaee eredentiale #	
Jsername and password	

• Click on the Credentials Manager Selector option (3).

ype *		
QLServer		•
econnect interval *		
0		
DBC Url *		
lbc:sqlserver://192.168.0.111:1433;databaseName=unifier		
atabase credentials *		
Isername and password credentials reference		•
Username and password credentials reference		

• Select one of the credentials from the list (4).

atabase connection			
pe *			
QLServer			
econnect interval *			
)			
DBC Url *			
bc:sqlserver://192.168.0.111:1433;data	-handbland		
uc.sqiserver.//192.108.0.111.1433,086	abaseName=unifier		
	abasename=unifier		
atabase credentials *			
atabase credentials * sername and password credenti	ials reference		
atabase credentials *	ials reference		
sername and password credent Username and password credent	ials reference		
atabase credentials * sername and password credenti Username and password crede	ials reference		

• If the credentials are not saved in the Credentials Manager, click on the Add credentials option (5).

atabase connection			
/pe *			
QLServer			
econnect interval *			
0			
DBC Url *			
lbc:sqlserver://192.168.0.111:1433;data	abaseName=unifier		
atabase credentials *			
sername and password credentials *	als reference		
serifance and passivora creacing			
Username and password crede	ntials reference		
Oradontiale Managar Calenter	entials reference		
	ntials reference		
Oradontiale Managar Calenter	ntials reference		
Oradontiale Managar Calenter	intials reference		

• Input the credentials details (6) and click on the Save and Close button (7).

	7
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).

Database connection		
ype *		
QLServer		
econnect interval *		
0		
DBC Url *		
dbc:sqlserver://192.168.0.	11:1433;databaseName=unifier	
atabase credentials *		
Jsername and passwo	rd credentials reference	

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.

≡	MORPH.p	oro							SMART UNIFIER 🕐
>	😕 User Mana	gement							Q + 5
Ŀ	User ID 🛧	Email	First Name	Last Name	Language	Role	Status	Created	
•	admin		Unifier	Administrator	en	Administrator	Active	2021-10-26 00:00:00.000	/ 1

⟨··⟩									

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 SMARTUNIFIER User Management perspective (1).

≡	MORPH.pro						SMARTUNIFIER 🔒 🤅
>							
Ŀ							
A			naufaatlu waxu				
## #		Production	perfectly your			Hi,	Unifier Administrator!
⟨··⟩		Troductio	in th using			20	Account
j						e;	Re-index Repository
۲			SMART UNIFIER			Ð	Dark Theme
â						Ģ	Simple UI
_				**	User Management 1	٩	Administrative
•				6	Credential Management	0	About SMARTUNIFIER
			DRPH.pro RT UNIFIER	*	Docker Java Image Manager	€	Sign Out
				蒜	Logging Configurations		
		SIMAF		Â	Translations Manager		
				8	Backup		
				Ð	Restore		

• Click the "Add User" button (2).

😕 User Man	agement							2 + 5
User ID	Email	First Name	Last Name	Language	Role	Status	Created	
admin		Unifier	Admin	en	Administrator	Active	2020-07-13 00:00:00.000	/ 1

- 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 SMARTUNIFIER.
 - Administrator: Full read and write access for the SMARTUNIFIER Configuration and Administration.
 - Reader: Only read access for the SMARTUNIFIER Configuration
 - Writer: Read and write access for the SMARTUNIFIER 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).

⊕ Add User			4 B ×
3	User ID * JohnDoe2		
	Email First Name * John		
	Lost Nome * Doe Longuage *		
	English Role: Wilter Administrator		
	Reader Writer Status:Active		
	Active Inactive Credentials	 □ × 	
	Paseword *	© 9 / 30 ower character	
	contains at least one u contains at least one u contains at least one t contains at least one s contains at least 4 cha	ipper character ligit character pecial character	
	Confirm password *	<u>ର</u>	

5.7.4 Edit a user

This procedure describes how to edit an existing user account.

• Select the SMARTUNIFIER User Management perspective (1).

≡	MORPH.pro						SMARTUNIFIER 🔒 🤅
>							
Ŀ							
A			naufaatlu waxu				
## #		Production	perfectly your			Hi,	Unifier Administrator!
⟨··⟩		Troductio	in th using			20	Account
j						e;	Re-index Repository
۲			SMART UNIFIER			Ð	Dark Theme
â						Ģ	Simple UI
_				**	User Management 1	٩	Administrative
•				6	Credential Management	0	About SMARTUNIFIER
			DRPH.pro RT UNIFIER	*	Docker Java Image Manager	€	Sign Out
				蒜	Logging Configurations		
		SIMAF		Â	Translations Manager		
				8	Backup		
				Ð	Restore		

• Click the "Edit" button (2).

🕹 User Managem	ent							Q + 43
User ID 🕇	Email	First Name	Last Name	Language	Role	Status	Created	
JohnDoe2		John	Doe	en	Reader	Active	2021-03-26 00:00:00.000	2 / 1
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

C Edit User: John Doe		4 ■ ×
3	ters ID * sbinboo2 Email First hare * John Lest hare * Doe Lesters * Boe Reder Administrator Reder	
	Writer Image: Status-Active Active Image: Status-Active Inactive Image: Status-Active	
	Change Password	

• After editing, click the "Save" button (4).

5.7.5 Delete a user

This procedure describes how to delete a user account.

• Select the SMARTUNIFIER User Management perspective (1).



• Click the "Delete" button (2).

😕 User Managem	nent							Q + 5
User ID 🕇	Email	First Name	Last Name	Language	Role	Status	Created	2
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	Z 1

Confirm by selecting the "Delete" button (3).

🕰 User Management						Q + <i>f</i> y		
User ID 🕇	Email	First Name	Last Name	Language	Role	Status	Created	
JohnDoe2		John	Doe	en	Reader	Active	2021-03-26 00:00:00.000	Z 1
admin		Unifier	Administrator	en	Administrator	Active	2021-03-26 00:00:00.000	Z 1
				Delete use Are you sure y	r ou want to delete this user JohnDoe Cancel Delete			

The user account is deleted and no more visible in the SMART**UNIFIER** User Management perspective.

💤 User Management						Q + 43		
User ID 🕇	Email	First Name	Last Name	Language	Role	Status	Created	
admin		Unifier	Administrator	en	Administrator	Active	2021-03-26 00:00:00.000	Z 1

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.

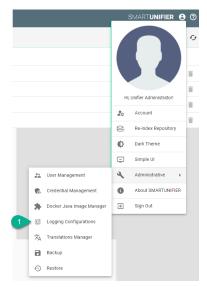
MORPH.pro SMARTUNIFIER 🖯 🧿						
🔥 Logging Co	onfigur	ations				Q (L) +
Group Filter	<	Group 🕇	Name			
✓ Show All default		default	Trace			ن o ن
		default	Debug			<u>ن</u> و
		default	Warning			<u>ن</u> 💿
		default	Info			<u>ن</u> 💿

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:

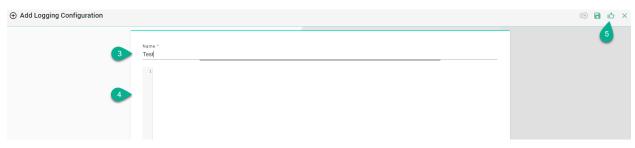
• Select the Logging Configurations perspective (1).



• Click on the "Add" button (2).

Logging Config	urations		ር 🖞 🕇 ፍ
Group Filter <	Group 🛧	Name	2
✓ Show All default	default	Trace	Ċ 💿 🗎
	default	Debug	ڭ • ٿ
	default	Warning	ن o آ
	default	Info	ڻ 💿 📋

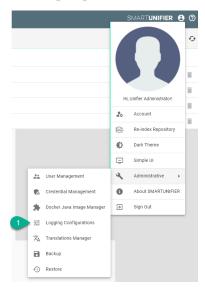
- Input the file Name (3) and the configuration (4).
- Click on the Save and Close button to exit (5).



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).

Logging Config	Logging Configurations		
Group Filter <	Group 🕇	Name	2
 Show All custom 	custom	Test	ථ 🗡 🗂
default	default	Trace	<u>أ</u> الا الم
	default	Debug	<u>ن</u> الا
	default	Warning	ڭ © آ
	default	Info	Ů ◎ Ĩ

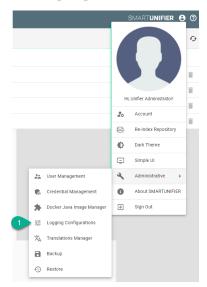
• Edit and click on the **Save and Close** button to exit (3).

Edit Logging Configuration: custom	Test	() 🔂 🕄
	Name * Test	3
	1	

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).

Logging Co	Logging Configurations			
Group Filter	<	Group 🛧	Name	2
✓ Show All custom		custom	Test	ů 🗡 🖬
default		default	Trace	ڭ 💿 🗎
		default	Debug	Ů • Î
		default	Warning	Ů • 🗊
		default	Info	් ම 🗎

• To confirm, click on the **Delete** button (3).

Delete Logging Configuration

Are you sure you want to delete this Logging Configuration?



5.9 Extensions

Note: Please contact Amorph Systems for guidance on how to enable and use extensions.

5.9.1 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 SMARTUNIFIER Information Model Perspective (1).
- Click on the **Extensions** button (2).
- Select the **OpcUa model generator: ADD** option (3).

=	M	DRPH.pro			[DEBUG] SMARTUNIFIER 🌲 😫 🞯
>	E I	nformation Models			<mark>2 क</mark> ५ 🖞 + छ
Ŀ:	_1	Group 🕇	Name	Description	Json model generator: ADD
***	FILTER	test	Test 0e38296/-687/-4438-b9b1-aa1d9c0bcda1		OpcUa model generator: ADD
<··>	GROUP				
6					
۲					
â					
•					

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).

UA Nodeset XML Import OpcUa Import Group * demo Name * OpcUa Model1 Description Type * Model Name * XML UA Nodeset XML Input: 1 ;2xml version=1.0; encoding='utf-8'?> 2 ;UANodeSet XML Input: 1 ;2xml version=1.0; encoding='utf-8'?> 3 ;UANodeSet XML Input: 3 ;UANodeSet	* 0	pcUa Model Generator Plugin		
demo Name * OpcUa Model1 Description Type * Model Name * XML UA Nodeset XML Input: 1 :2xml version='1.0' encoding='utf-8' >> 2:UANodeSet xmlns:xsi="http://www.wb.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/ 3 :ChamespaceUris> 2:UANodeSet Xmllis="Stormace" instance" xmlns:uax="http://opcfoundation.org/UA/ 3 :ChamespaceUris> 2:UANodeSet Xmllis="Stormace" instance" xmlns:uax="http://opcfoundation.org/UA/ 3 :ChamespaceUris> 2:UANodeSet Xmllis="Stormace" instance" xmlns:uax="http://opcfoundation.org/UA/ 3:ChamespaceUris> 3:CAlis Alias="Stormace" instance" instance" xmlns:uax="http://opcfoundation.org/UA/ 3:CAliases> 3:CAlis Alias="Chamespace": instance" instance" instance" instance" instance" instance" instance instance" instance instance instance instance insta	U	A Nodeset XML Import OpcUa Import		
Name * OpcUa Model1 Description Type * Model Name * XML UA Nodeset XML Input: 1 :2xml version="1.0" encoding="utf-8"?> 2 :UANodeset XML Input: 2 :UANodeset xML input: 2 :UANodeset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/ 3 (VanespaceUris> (VanespaceUris> (VanespaceUris> (Valias Alias="toing">>:unifier:opc-ua-server 3 (VanespaceUris> (Valias Alias="toing">>:unifier:opc-ua-server 4 (Valiases> 7 (Alias Alias="toing">>:unifier:opc-ua-server 5 (Valias Alias="toing">>:unifier:opc-ua-server 5 	Group			
OpcUa Model Description Type * Model Name * XML UA Nodeset XML Input: 1 ;2xml version=1.0* encoding="utf-8">> 2 ;UANodeSet XML Input: 1 ;2xml version=1.0* encoding="utf-8">> 2 ;UANodeSet XML input: 1 ;2xml version=1.0* encoding="utf-8">> 2 ;UANodeSet XML input: 1 ;2xml version=1.0* encoding="utf-8">> 2 ;UANodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/ 3	dem	0		
<pre></pre>	Name			
Type * Model Name * XML UA Nodeset XML Input: 1 :2xml version="1.0" encoding='utf-8'?> 2 :UANodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/ 3 :Characteriss 4 :Characteriss 5 :CANdeSet Xmlss:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/ 3 :Characteriss 4 :Characteriss 5 :CANdeSet Nodel:sister: 5 :CANdeSet: 6 :CANdeSet: 6 :CANdeSet: 7 :CANdeSet: 8 :CANdeSet: 9 :CANdeSet: 10 :CANdeSet: 11 :CANdeSet: 12 :CANdeSet: 13 :CANdeSet: 14 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 14 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 16 :CANdeSet: 17 :CANdeSet: 18 :CANdeSet: 19 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 11 :CANdeSet: 12 :CANdeSet: 13 :CANdeSet: 14 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 16 :CANdeSet: 17 :CANdeSet: 17 :CANdeSet: 18 :CANdeSet: 19 :CANdeSet: 19 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 11 :CANdeSet: 11 :CANdeSet: 12 :CANdeSet: 13 :CANdeSet: 14 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 16 :CANdeSet: 17 :CANdeSet: 17 :CANdeSet: 18 :CANdeSet: 18 :CANdeSet: 19 :CANdeSet: 19 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 11 :CANdeSet: 11 :CANdeSet: 12 :CANdeSet: 13 :CANdeSet: 13 :CANdeSet: 14 :CANdeSet: 15 :CANdeSet:	Opcl	Ja Model1		
Type * Model Name * XML UA Nodeset XML Input: 1 :2xml version="1.0" encoding='utf-8'?> 2 :UANodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/ 3 :Characteriss 4 :Characteriss 5 :CANdeSet Xmlss:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/ 3 :Characteriss 4 :Characteriss 5 :CANdeSet Nodel:sister: 5 :CANdeSet: 6 :CANdeSet: 6 :CANdeSet: 7 :CANdeSet: 8 :CANdeSet: 9 :CANdeSet: 10 :CANdeSet: 11 :CANdeSet: 12 :CANdeSet: 13 :CANdeSet: 14 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 14 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 16 :CANdeSet: 17 :CANdeSet: 18 :CANdeSet: 19 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 11 :CANdeSet: 12 :CANdeSet: 13 :CANdeSet: 14 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 16 :CANdeSet: 17 :CANdeSet: 17 :CANdeSet: 18 :CANdeSet: 19 :CANdeSet: 19 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 11 :CANdeSet: 11 :CANdeSet: 12 :CANdeSet: 13 :CANdeSet: 14 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 15 :CANdeSet: 16 :CANdeSet: 17 :CANdeSet: 17 :CANdeSet: 18 :CANdeSet: 18 :CANdeSet: 19 :CANdeSet: 19 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 10 :CANdeSet: 11 :CANdeSet: 11 :CANdeSet: 12 :CANdeSet: 13 :CANdeSet: 13 :CANdeSet: 14 :CANdeSet: 15 :CANdeSet:				
Model Name * XML UA Nodeset XML Input: 1 :?xml version="1.0" encoding="utf-8"?> 2 :UMNodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/ 3 3 4 5 6 1 states="String"> 7 < <td><<td> 6</td><td>Des</td><td>cription</td></td>	< <td> 6</td> <td>Des</td> <td>cription</td>	6	Des	cription
<pre>Name * XML Vame * XML UA Nodeset XML Input: 1 :?xml version="1.0" encoding='utf=8'?> 2:UANodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/ 3 cUnivurn:amorphsys:unifier:opc-ua-serverc/Uri> 4 cUnivurn:amorphsys:unifier:opc-ua-serverc/Uri> 5 cAliases> 7 cAlias Alias="Tomble">i=11c/Alias> 8 cAlias Alias="Tomble">i=11c/Alias> 9 cAlias Alias="Tomg?i=12c/Alias> 9 cAlias Alias="Tomg?i=12c/Alias> 9 cAlias Alias="String">i=12c/Alias> 9 cAlias Alias="Tomg?i=12c/Alias> 9 cAlias Alias="tring">i=12c/Alias> 9 cAlias Alias="Tomg?i=12c/Alias> 9 cAlias Alias="tring">i=12c/Alias> 9 cAlias Alias="t</pre>	Туре			
XML UA Nodeset XML Input: 1 ?>ml version="1.0" encoding="utf-8"?> 2 !UMNodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/ 3 (AlmaespaceUris> 4 (Urisum:amorphsys:unifier:opc-ua-server5 (Allases) 6 (Allases) 7 (Allas Allas="String">i=12(/Allas> 8 (Allases) 9 (Allas Allas="String">i=12(/Allas> 9 (Allases) 4 (Allases) 4 (Allases) 5 (Alefrence Reference") = "Asount" Browsellame="1:acunit" ParentNodeId="1=85"> 9 (StriplayMame>acunit 4 (Reference ReferenceType="Organizes" Isforward="false">i=55//Reference> 16 (Reference ReferenceType="Organizes" Isforward="false">i=55//Reference> 17 (Reference ReferenceType="Organizes" Isforward="false">i=55//Reference> 18 (Reference ReferenceType="Organizes" Isforward="false")>i=55//Reference> 19 (Reference ReferenceType="Organizes")>i=5/i=7/isfored(/Reference> 19 (Reference ReferenceType="Organizes")>i=5/i=7/isfored(/Reference>)	Mod	el		
XML UA Nodeset XML Input: 1 ?>ml version="1.0" encoding="utf-8"?> 2 !UMNodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/ 3 (AlmaespaceUris> 4 (Urisum:amorphsys:unifier:opc-ua-server5 (Allases) 6 (Allases) 7 (Allas Allas="String">i=12(/Allas> 8 (Allases) 9 (Allas Allas="String">i=12(/Allas> 9 (Allases) 4 (Allases) 4 (Allases) 5 (Alefrence Reference") = "Asount" Browsellame="1:acunit" ParentNodeId="1=85"> 9 (StriplayMame>acunit 4 (Reference ReferenceType="Organizes" Isforward="false">i=55//Reference> 16 (Reference ReferenceType="Organizes" Isforward="false">i=55//Reference> 17 (Reference ReferenceType="Organizes" Isforward="false">i=55//Reference> 18 (Reference ReferenceType="Organizes" Isforward="false")>i=55//Reference> 19 (Reference ReferenceType="Organizes")>i=5/i=7/isfored(/Reference> 19 (Reference ReferenceType="Organizes")>i=5/i=7/isfored(/Reference>)	Name			
<pre>1 :2xml version='1.0' encoding='utf-8'?> 2 :UANodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/</pre>				
<pre>2 UdWodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uax="http://opcfoundation.org/UA/</pre>				
<pre>3 <{NamespaceUris></pre>				
<pre>5 6 <aliases> 7 <alias aliase"=""></alias></aliases></pre>	З	<namespaceuris></namespaceuris>		
<pre>6</pre>				
<pre>7</pre>				
<pre>9 <alias alias="Organizes">i=35</alias></pre>				
<pre>10</pre>	8			
<pre>11 12 13 14 15 <td></td><td></td></pre>				
<pre>12 <uaodject browsename="1:acunit" nodeld="ns=1;s=acunit" parentnodeld="1=85"> 33 <uisplayname>acunit</uisplayname> 4 <references 54="" 55="" <references="" <referencespare='THasTypeDefinition"'>i=61</references> 56 <reference referencetype="Organizes">isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85isForward="false">i=85i=85i=85i=85i=85i=85i=85i=85i=85i=85<td></td><td></td></reference></uaodject></pre>				
<pre>33 <displaymame>acunit</displaymame> 44 <references 54="" <reference="" referencetype="HasTypeDefinition">i=61 55 <reference isforward="false" referencetype="Organizes">i=65</reference> 56 <reference referencetype="Organizes">i=5+/i=65</reference> 57 <reference referencetype="Organizes">i=5+/i=62</reference> 58 <reference referencetype="Organizes">i=5+/i=5>/i=5/i=62</reference> 59 <reference referencetype="Organizes">i=5+/i=5>/i=5/i=62</reference> 59 <reference referencetype="Organizes">i=5+/i=5/i=62</reference> 50 <reference referencetype="Organizes">i=5+/i=5+/i=5/i=62</reference> 50 <reference referencetype="Organizes">i=5+/i=5+/i=5/i=62</reference> 50 <reference referencetype="Organizes">i=5+/i=5+/i=5+/i=5+/i=5+/i=5+/i=5+/i=5+/</reference></references></pre>				
<pre>14 <references> 15 <reference referencetype="HasTypeDefinition">i=61</reference> 15 <reference lsforward="false" referencetype="Organizes">i=85</reference> 16 <reference referencetype="Organizes">i=1;s=/temperature</reference> 17 <reference referencetype="Organizes">i=1;s=/temperature</reference> 18 <reference referencetype="Organizes">i=1;s=/temperature</reference> 19 <reference referencetype="Organizes">i=1;s=/temperature</reference> 19 <reference referencetype="Organizes">i=1;s=/temperature</reference> 19 <reference referencetype="Organizes">i=1;s=/temperature</reference> 19 <reference referencetype="Organizes">i=1;s=/temperature</reference> 10 <reference referencetype="Organizes">i=1;s=/temperature</reference> 11 <reference referencetype="Organizes">i=1;s=/temperature</reference> 12 <reference referencetype="Organizes">i=1;s=/temperature</reference> 13 <reference referencetype="Organizes">i=1;s=/temperature</reference> 14 <reference referencetype="Organizes">i=1;s=/temperature</reference> 15 <reference referencetype="Organizes">i=1;s=/temperature</reference> 16 <reference referencetype="Organizes">i=1;s=/temperature</reference> 17 <reference referencetype="Organizes">i=1;s=/temperature</reference> 17 <reference referencetype="Organizes">i=1;s=/temperature</reference> 18 <reference referencetype="Organizes">i=1;s=/temperature</reference> 19 <reference referencetype="Organizes">i=1;s=/temperature</reference> 16 <reference referencetype="Organizes">i=1;s=/temperature</reference> 17 <reference <="" referencetype="Organizes" td=""><td></td><td></td></reference></references></pre>				
15 <reference referencetype="lasTypeDefinition">1+61/Reference> <reference isforward="false" referencetype="Organizes">i=85</reference> <reference referencetype="Organizes">isForward="false">i=85</reference> <reference referencetype="Organizes">isForward="false">i=85</reference> <reference referencetype="Organizes">isForward="false">i=85</reference> <reference referencetype="Organizes">isForward="false">isForward="false">i=85</reference> <reference referencetype="Organizes">isForward="false">isForward="false">isForward="false">isForward="false">isForward="false">isForward="false">isForward="false">isForward="false">isForward="false"</reference></reference>				
16 <reference isforward="false" referencetype="Organizes">i=85</reference> 17 <referencetype="organizes">i=sis>/temperature 18 <referencetype="organizes">i=sis>/temperature 19 <referencetype="organizes">i=sis>i=sis>i=sis>i=sis>i</referencetype="organizes"></referencetype="organizes"></referencetype="organizes">				
<pre>18 <reference referencetype="Organizes">ns=1;s=/airSpeed</reference> 19 <reference referencetype="Organizes">ns=1;s=/location</reference></pre>		<reference isforward="false" referencetype="Organizes">i=85</reference>		
<pre>19 <reference referencetype="Organizes">ns=1;s=/location</reference></pre>				
	19 20	<reference referencetype="Organizes">ns=1;s=/location</reference>		

• The Information Model is generated.

Edit Model: demo:OpcUa Model1:latest ~	🛸 🗣 🐨 🖉 🖬 🖆 🗙
CopcUa Model1 Q C GunitType] Itemperature [Double] @ arispeed [Double] @ arispeed [Double] @ location [String]	Greup * 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).

OpcUa Model Generator Plugin	9
UA Nodeset XML Import OpcUa Import	•
Group *	
demo	
Name *	
OpcUa Model2	
Description	
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.

✗ Edit Model: demo:OpcUa Model2:latest ∨	🖈 🕏 😨 🖉 🔁 🖈
OpcUa Model2 Q Q O Module_A [Module_AType] Pressure [Double] Io [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).

=	MORPH.pro	[DEBUG] SMART UNIFIER 🌲 😝 (
>	✗ Edit Model: demo:OpcUa Model1:latest ∨	🚺 🛊 🗞 🔞 🖉 🔒 🗠 ×				
Ŀ ## ↔	OpeCla Model1 Q C Securit (Acurit Type) Temperature (Double) G arSpee (Double) Coston (String)	Group * demo 2 OpcUa model generator: UPDATE Name * OpcUa Model1				
۵Ì		Description				
© A						
•						

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)**.

JS0	N Model Genera	tor Plugin
UA No	deset XML Import	OpcUa Import
Selector	Madali	
3 cHamespaceUris> 4 c/Uri>urn:amor 5 c/NamespaceUris 6 cAliasAiases> 7 cAliasAiases 9 cAliasAiases 10 cAliasAiases 11 c/Aliases> 12 cUAObject NodeI 13 cOisplayNames 14 cReferences 15 cReference 16 cReference		
Type *		
Variabl	e	
Name *		
XIVIL		
UA Node	eset XML Input:	
		encoding='utf-8'2>
		<pre>i="http://www.w3.org/2001/XML5chema-instance" xmlns:uax="http://opcfoundation.org/U</pre>
4	<uri>urn:amorphs</uri>	sys:unifier:opc-ua-server
5		
6	<aliases></aliases>	
		buble">i=11
		nteger">i=27
		rganizes">i=35
		asTypeDefinition">i=40
		<pre>"ns=1;s=bag" BrowseName="1:bag" ParentNodeId="1=85"> </pre>
		\$
		ferenceType="HasTypeDefinition">i=61
		ferenceType="Organizes" IsForward="false">i=85
17		<pre>ferenceType="Organizes">ns=1;s=/countedItems</pre>
18		ferenceType="Organizes">ns=1;s=/noBags
19		ferenceType="Organizes">ns=1;s=/motorSpeed
20		<pre>ferenceType="Organizes">ns=1;s=/motorTemperature</pre>
21		<pre>ferenceType="Organizes">ns=1;s=/passengerId</pre>
22		<pre>ferenceType="Organizes">ns=1;s=/scannerTemperature</pre>
23		
		<pre>d="ns=1;s=/countedItems" BrowseName="1:countedItems" ParentNodeId="ns=1;s=bag" Data</pre>
26		untedItems
27	<references></references>	Constant and Mills Theorem De Clarific Instantia CD - (De Constants)
28 29		<pre>ferenceType="HasTypeDefinition">i=63 ferenceType="BasTypeDefinition">i=63 ferenceType="BasTypeDefinition">i=63</pre>
30	<th><pre>ferenceType="Organizes" IsForward="false">ns=1;s=bag</pre></th>	<pre>ferenceType="Organizes" IsForward="false">ns=1;s=bag</pre>
31		

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).

UA Nodeset XML Import OpcUa Import	-
Selected Model: demo:0pcUa Model1:latest	
Namespace Index *	
2	
	5
Identifier Name	-
Identifier Name *	
Module_B	
Security Policy *	
None	
IP address *	
192.168.4.12	
TCP port *	
9999	
Endpoint path *	
server	

5.9.2 JSON Model Import

SMARTUNIFIER 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 SMARTUNIFIER Information Model Perspective (1).
- Click on the **Extensions** button (2).
- Select the Json model generator: ADD option (3).

MORPH.pro Information Models			SMARTUNIFIER	ł 🌲	8 0		
>	Ŀ,	nformation Models			² 🗼 વ	Ľ	+ G
Ŀ:	, ж	Group 🛧	Name	Description	3 Json model generator: ADD		
***	P FILTER	demo.test	commandTest			Ċ	î î
<>	GROUP	demo.test	testJson			Ů	Ô
j							
۲							
۵							
۲							

- Provide the following mandatory information: **Group** and **Name (4)**.
- Click on the Add item button (5).

Group *	
demoscenarios.extensions	
Name *	
FlightModel	
Fightwodel	
Description	
	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).

-	N Model Generator Plugin	9 > >
Group *		
demoso	cenarios.extensions	
Name *		
FlightM	odel	
Descri	ption	
items		
Type *		
Event		
Name *		
FlightEv	rent	
JSON Ir	iput:	
1 {	"version": 1,	
2	"intervalStart": 1599685200000,	
4	"intervalEnd": 1599771600000,	
5	"validTo": 1599739052657,	
6	"flights": [
7	{	
8	"id": 0,	
9	"fuid": "AY1511/20200910/D",	
10	"formatVersion": 0,	
11	"departureDate": "20200910",	
alle alle	"operationsDate": "20200910",	
12		

• The Information Model is generated.

Image: Project State Q_< ↓	
<pre>Description Description D</pre>	

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 ∨			🗯 🔖 🐨 🛡 🖬 🖆 🗙
	Group * demoscenarios.extensions Name * FlightModel Description	2	Json model generator: UPDATE

- 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).

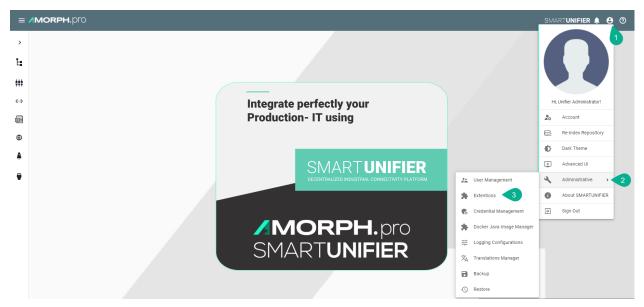
	Model:		
demosce	enarios.extensions:FlightModel:latest		3
Items			
Type *			
Event			
Name *			
FlightEve	entV2		
JSON In	but:		
15			
1 {			
2	"version": 2,		
2 3	"intervalStart": 1599685200000,	6	
2 3 4	"intervalStart": 1599685200000, "intervalEnd": 1599771600000,	6	
2 3 4 5	"intervalStart": 1599685200000,	6	
2 3 4 5	"intervalStart": 1599685200000, "intervalEnd": 1599771600000,	6	
2 3 4	"intervalStart": 1599685200000, "intervalEnd": 1599771600000, "validTo": 1599739052657,	6	
2 3 4 5	"intervalStart": 1599685200000, "intervalEnd": 1599771600000, "validTo": 1599739052657,	6	
2 3 4 5 6 7	"intervalStart": 1599685200000, "intervalEnd": 1599771600000, "validTo": 1599739052657, "flights": [{	6	
2 3 4 5 6 7 8	<pre>"intervalStart": 1599685200000, "intervalEnd": 1599771600000, "validTo": 1599739052657, "flights": [{</pre>	6	
2 3 4 5 6 7 8 9	<pre>"intervalStart": 1599685200000, "intervalEnd": 1599771600000, "validTo": 1599739052657, "flights": [{</pre>	6	

5.9.3 AWS IoT SiteWise Model Export

This extension allows you to export an SMARTUNIFIER 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)

* Extentions					Q + 9
Name 🛧	Extention Type	Description	State	Version	
AWS SiteWise Model export plugin	Export	Exports SMARTUNIFIER Information Models to AWS IoT SiteWise	Installed	1.0.0	4 🌣

How to export Information to AWS IoT SiteWise

We recommend to have one user dedicated for SMARTUNIFIER.

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 SMARTUNIFIER 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)

🗯 AWS SiteWise Model Export Plugin		2 ► 8
0	Region * eu-central-1 Access key id * AKIARU7XHLCLZDN5ZZVT Secret access key id *	

CHAPTER

GETTING HELP

Having trouble? We would like to help!

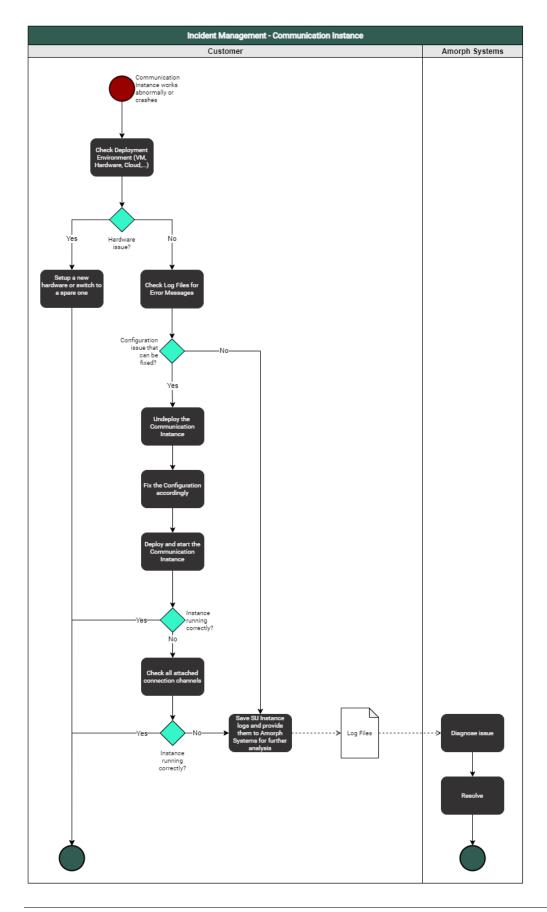
- In case of malfunctioning SU Instances check out the Troubleshooting section
- Try the FAQ it's got answers to regularly asked questions
- Check out the Glossary if some terminology is not clear

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.

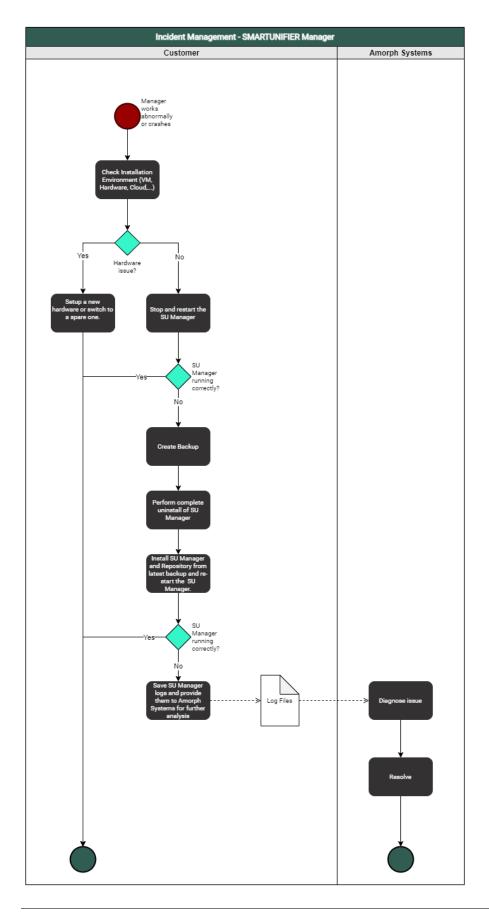
- In case of a HW problem setup a new HW (or switch to a spare HW). Ensure to place the correct security certificates on the new HW. Perform a new *deployment* of a new SU Instance with *SU Manager* on the new HW.
- In case, the HW is operating correctly navigate to the log file of the deployed instance ./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



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



6.2 FAQ

Does SMARTUNIFIER provide caching/buffering of data?

Yes, SMARTUNIFIER 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 SMARTUNIFIER 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 SMARTUNIFIER. SMARTUNIFIER provides powerful capabilities for any kind data preprocessing, cleansing, filtering and optimization. The capabilities of SMARTUNIFIER 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, SMARTUNIFIER 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 SMARTUNIFIER's logs which can record all communication activities of a SMARTUNIFIER Instance incl. telemetric data. SMARTUNIFIER'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 SMARTUNIFIER'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, SMARTUNIFIER 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. SMARTUNIFIER allows to operate multiple OPC-UA Servers and/or OPC-UA Clients within each single SMARTUNIFIER instance for northbound and/or southbound communication.

Does SMARTUNIFIER support brokering to MQTT Server?

Yes, SMARTUNIFIER supports any number of MQTT connections. One single SMARTUNIFIER 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?

SMARTUNIFIER 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?

SMARTUNIFIER 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, SMARTUNIFIER 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 SMARTUNIFIER Information Models complying with these standards and incorporating full data semantics. There will be a one-time effort to implement such a standard in SMARTUNIFIER 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?

SMARTUNIFIER runs on Windows, Linux, Mac and other OS supporting Java RT and Docker.

Does SMARTUNIFIER support onPrem Edge-Analytics?

Yes, SMARTUNIFIER can be connected to any Edge-Analytics System SMARTUNIFIER 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, SMARTUNIFIER supports remote installation/update of Software from SMARTUNIFIER Manager via Docker Registry SMARTUNIFIER 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, SMARTUNIFIER itself is a fully distributed and scalable IT system. With this architecture SMARTUNIFIER is able to collaborate in any small or large IT environment. SMARTUNIFIER 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, SMARTUNIFIER 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 SMARTUNIFIER 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, SMARTUNIFIER 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, SMARTUNIFIER creates detailed logs regarding all communication activities (and other activities) it performs. With SMARTUNIFIER 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, SMARTUNIFIER supports individual user authentication.

Does SMARTUNIFIER support Windows Active Directory (AD)?

Yes, SMARTUNIFIER 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, pentesting)?

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 SMARTUNIFIER (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, SMARTUNIFIER provides high performance / low latency by its distributed architecture consisting out of small SMARTUNIFIER Instances (i.e., no central bottlenecks like e.g., a middleware broker/database). Furthermore, SMARTUNIFIER features an integrated compiler that creates native Bytecode for the interfaces to be executed within the SMARTUNIFIER Instances. This makes the SMARTUNIFIER 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 SMARTUNIFIER Manager is used to create and monitor SMARTUNIFIER 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.