MORPH.pro SMART**UNIFIER**

SMARTUNIFIER User Manual

Release 1.6.0

Amorph Systems GmbH

Apr 14, 2022

ABOUT SMARTUNIFIER

1	Abou	It SMAR	TUNIFIER 2
	1.1	What is	SMARTUNIFIER
	1.2	What d	oes SMART UNIFIER do
	1.3	Importa	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 11
		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	Informa	ation Models
		2.1.1	What are Information Models
		2.1.2	How to create a new Information Model 17
		2.1.3	Node Types
		2.1.4	Data Types
	2.2	Commu	unication 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	Mappin	gs
		2.3.1	What are Mappings
		2.3.2	How to create a new Mapping
		2.3.3	How to create Rules
	2.4	Device	Types
		2.4.1	What are Device Types 73
		2.4.2	How to create a new Device Type
	2.5	Commu	inication Instances

		2.5.1	What are Instances
		2.5.2	How to create a new Instance
3	Conf	iguratio	n Component Management 78
	3.1	Naming	g Convention
	3.2	Group	Filter
	3.3	Compo	nent Version Control
		3.3.1	How to release configuration components
	3.4	Operat	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 88
		3.4.12	Clone 89
		3 4 13	Delete 89
		3.4.14	Bulk Action 90
		0.1111	
4	Depl	oyment	93
	4.1	What is	s a Deployment \ldots \ldots \ldots 33
	4.2	Deploy	Locally
	4.3	Deploy	with Docker
	4.4	Deploy	with SSH
	4.5	Deploy	with AWS Fargate
		4.5.1	Prerequisites
		4.5.2	Architecture
		4.5.3	Planning the Deployment
		4.5.4	Deployment Steps
	4.6	How to	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	Notifica	ations
		4.7.1	How to access Notifications
		4.7.2	How to manage Notifications
	4.8	How to	monitor a deployed Instance
		4.8.1	Log Viewer
		4.8.2	Dashboard
	4.9	Additio	nal Options

		4.9.1	Encryption of Communication Instances	. 115
		4.9.2	Protect Communication Instances	. 116
		4.9.3	VM Arguments	. 117
5	Adm	inistrat	ion	119
Ŭ	5.1	Active	Directory Integration (ADI)	. 119
	0.12	5.1.1	AD Group Mapping	. 120
	5.2	Backur	and Restore	122
	0.1	5.2.1	How to access	123
		5 2 2	Backup	123
		523	Restore	126
		52.0	Manager Backup	120
	53	Channe	al Types Manager	120
	5.5	5 3 1	How to access	120
		532	About Lawars	120
		5.2.2	How to graate a new Channel Type	. 130
	E /	J.J.J Doctor		. 131
	5.4			. 133
		5.4.1		. 133
		5.4.2	Add a New Docker Java Image	. 135
		5.4.3	Edit a Docker Java Image	. 136
		5.4.4	Delete a Docker Java Image	. 136
	5.5	Deploy	ment Endpoints	. 136
		5.5.1	What are Deployment Endpoints	. 136
		5.5.2	How to access	. 136
		5.5.3	Deployment Endpoints Types	. 137
	5.6	Creden	itial Management	. 142
		5.6.1	How to access	. 142
		5.6.2	Add Credentials	. 144
		5.6.3	Edit Credentials	. 145
		5.6.4	Delete Credentials	. 145
		5.6.5	Using Credential Manager when configuring the Communication Channels	. 146
	5.7	User M	lanagement	. 149
		5.7.1	About User Management	. 149
		5.7.2	How to access	. 149
		5.7.3	Add a new user	. 150
		5.7.4	Edit a user	. 152
		5.7.5	Delete a user	. 153
	5.8	Loggin	g Configurations	. 154
		5.8.1	How to access	. 154
		5.8.2	Add a new logging file	. 156
		5.8.3	Edit a logging file	. 157
		5.8.4	Delete a logging file	. 158
	5.9	Extens	ions	. 158
		5.9.1	AWS SiteWise Model Export Plugin	. 158
6	Getti	no Holt		161
0	61	Trouble	eshooting	161
	0.1	611	Communication Instances	161
		0.1.1	Communication instances	. 101

	6.1.2	SN	ΛA	RT	UN	IFI	ER	Μ	an	ag	er	•				•	•		•				•		•	•	 •	 •	163
6.2	FAQ .		• •			•		•			•	 •	•	•	•	•	•		•	•	 •	•	•		•	•	 •	 •	165
6.3	Glossar	У	• •			•		•			•	 •	•	•	•	•	•		•	•	 •	•	•		•	•	 •	 •	174

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 SMARTUNIFIER.
- By applying so called *Information Models*, SMARTUNIFIER 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 LAK	E
1. ID 2. RELEASE 3. S ORDER 3. S	START 4. FINIS ORDER ORDI	SH 5. RESL QUAL DATA		5. RESULTING QUALITY DATA	6. DETAILED PROCESS DATA
	MOF SMART	RPH.pr UNIFIEI	ro २		
Informati	on Models	Enterp	rise Conte	xt	
Smart M	Aappings	Sir	mulation		
*	1 4	,	•	^	+
1. ID	2. RELEASE ORDER	3. START ORDER	4. FINISH ORDER	5. RESULTING QUALITY DATA	6. DETAILED PROCESS DATA
ID-READER		E	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 incre-

mentally migrate to a state-of-the-art Industry 4.0 IT architecture, but still keep the existing IT infrastructure fully operable.

MES	OEE	ERI	P	DATA LAKE	OTHER	2
Legacy Protocol 🗸	Legacy Protocol	Legacy Protocol	Nev Pro	otocol 🗘 🕺	New Protocol 🗘	
		AMOR SMARTU	ε ΡΗ. β JNIFI	oro ER		
	Informatio	n Models	Ente	rprise Contex	t	
	Smart Ma	appings		Simulation		
Legacy Protoco	oi 🗘	1			\$	
• EQ	UIPMENT	PERIP	HERY		Devices	

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).



	AMC SMAF	ORPH.¤ ?⊤UNIFII	oro ER	
	Information Models	Ente	rprise Context	
	Smart Mappings		Simulation	
	\Diamond	\Diamond	<	\triangleright
EC		PERIPHERY	DEV	

1.4 Connectivity Endpoints and Data Formats

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

1.4.1 Connectivity Endpoints / Communication Protocols

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

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						
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						
FTP	Upload and Download files to/from FTP servers						
HTTP	Send request to HTTP servers						

able 1. Connectivity Lindpoints

continues on next page

	Table 1 – continued from previous page
Format	Description
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)
JMS	Send and receive messages to/from a JMS Queue or Topic using plain JMS
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 SMARTUNIFIER 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
DECT	standard communications via e.g., H11P, RES1, SQL
KES I	Communication via REST using REST Server / REST Client (web-
SAD MII	Interface to SAD MII
SAD REC	Interface to SAP via remote function call (REC)
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-
	ing SECS/GEM interface protocol for equipment-to-host data com-
	munications (TCP/IP).
Siemens Industrial Edge	Deployment of SMARTUNIFIER Communication Instances via
	Siemens Industrial Edge Platform
Siemens MindSphere	Interface to MindSphere via REST
(REST)	

continues on next page

-	Table 1 – continued from previous page
Format	Description
Siemens MindSphere	Interface to MindSphere via MQTT
(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)
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.

Table 2. Data formats					
Format	Description				
Binary	Handling of any binary communication format (e.g., fixed/variable				
	lengths fields, headers/footers)				
CSV	Handle CSV (Comma separated values) payloads				
JSON	Encode and decode JSON formats				
TEXT	Handling of any text-based communication format				
XML	Encode and decode XML formats				

Table 2. Data Formate

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 [¹] + 42
> Group ↑	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 ♥ 🖬 ıb ×
EquipmentModel	Q C MyFirstVariable	
	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 🔽 🖬 🖒 🗙
M EquipmentModel A	Q 10 * 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	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	A 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] NetWnCDE	Q 0 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 SMART**UNIFIER** through a *Mapping*. A command contains one or multiple simple or structured *Variables*. Also a command has a return parameter that likewise can be a simple or complex *data type*.

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

How to create a Command

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

+ 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 🗸 🖬 16 X
EquipmentModel Q C MyFirstCommand Parameters Reply A 4	Description Data Type * MyFirstCommandReques(×
 Select the Reply node from the tr Enter a Data Type (7) Click the "Apply" button (8) 	ee (6)
+ Add Model	8 V B 15 ×
EquipmentModel A A A A A A A A A A A A A	Description Data Type * MyFirstCommandReply ×

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				S 🖬 16 ×
CupmentModel C MyFirstCommand C Parameters (MyFirstCommandRec)) Reby (MyFirstCommandRec))	 Parameters + Add Node > 	Q C Variable 9 Array List Property	Description Data Type * MyFiratCommandReply	×

- 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 🛡 🖻 🖒 🗙
EquipmentModel GMyFirstCommand OParameters [MyFirstCommandRequest]	Q 10 * Request1	13
 V [NEW NODE] ▲ Reply [MyFirstCommandReply] 	Description	
	11 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 😺 🖻 16 X
EquipmentModel	<u>्</u> २ 1	iD * MyFirstArray
	2 3	Description Data Type * String × Size * 5

Lists

What are Lists

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

How to create a List

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

+ Add Model		3 🛛 🖬 🖒 🗙
EquipmentModel	Q 0 MyFirstList	
	Description Data Type * String	×

2.1.4 Data Types

There are two kinds of Data Types:

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

How to create a Variable as a Simple Type

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

+ Add Model		🛡 🖬 16 ×
EquipmentModel	ID * MySimpleVariable Description	
	Data Type * String	X

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

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		S 🖬 🖒 🗙
Q Q V MyEirstComplexVariable [MyFirstComplexVariableType] 1 W MyEirstQuarable [MyFirstComplexVariableType] 1 W MyEirstQuarable [MyFirstComplexVariableType] 1	S 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)

👯 Communica	ation Ch	annels					ፍ 🖞 + ድ
Group Filter	<	Group 🛧	Name	Version	Model	Description	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

		🔖 🛞 🖬 165 🗙
3	Group * demo Name * EquipmentChannel Description	6
4	Model * emo.EquipmentModel.latest *	
5	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 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 * demo.EquipmentModel.latest ~	
0	Channel type * File tailer (CSV)	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
- 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

### Channel Configuration: demo.scenario1:FileTailerCh	annel:latest	\leftrightarrow \checkmark X
CaujomentModel CSVDemoEventType] Pressure [String] Timestamic [String] RardWr [String] PartWr [String]	C C File Tailer to String File * C:EquipmentFiles/Equipment.csv DurationConfigurationType Length * 250 Unit * Milliseconds	
		·

- 5. Enter the **separator** which is used in the CSV-file as well as the **string delimiter**, the **eol delimiter** and the **timestamp format** if one is used.
- 6. If the CSV file contains a header enable **ignoreFirstLine**.

el:latest	↔ ✓ ×
CSV String to Model Equator * 5 CSV String to Model Equator * 5 Constrained and the second sec	
	el:latest CSV String to Model

7. 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.

- 8. Check the **routes checkbox**.
- 9. Enter a **regular expression** for the message filter.
- 10. 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		↔ ✓ X
EquipmentModel SvDemoEventType Pressure[String] Timestamp[String] Timestamp[String] PertVr[String]	Q	
	CSV String to Model Enable event CsvModelConfiguration Message filter (RegEx) *	

Description of configuration properties:

Property	Description	Example
Separator	Separator type, used in the csv file	, , ;
Delimiter	Values that have an additional delimiter like "Date", "Time"	"
Eol Delimiter	Defining Carriage return and/or Line Feed	\r, \n
Timestamp 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	

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 ^M 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]
	✓ Timestamp [String]
	V PartNr [Int]
	Pressure, Temperature, Timestamp, PartNr 17.5,20,2020.06.11-06:56:31,0001 18.9,22,2020.06.11-07:56:31,0002

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 * FileRederChannel Description Model * demo.EquipmentModel.latest	
1	Channel type * File reader (CSV)	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 InFolder
 - Enter a path for the process folder ProcessFolder
 - Enter a path for the output folder **OutFolder**
 - Enter a path for the error folder ErrorFolder
 - Specify the **polling interval**
 - Select the CharSet according to the file in use

FileConsumer to String In folger * C:UnifierUnput	
Process folder * C:\Unifier\Process	
our folder * C∖Unifier\Out	
Error folder * C:\Unifier\Error	
DurationConfigurationType	
Length * 250	
Unit * Milliseconds	•
Charset * UTF-8	
	FileConsumer to String Infoider * C:WinferVinput Process folder * C:WinferVinoss Oct.WinferVinoss Oct.WinferVinos Oct.WinferVi

- 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 IgnoreFirstLine

### Channel Configuration: doc.filereader:FileReaderChannel:latest	O ✓ >
EquipmentModel Q C CevDemoEvent[CSVDemoEventType] CevData [CSVData] Presure [String] Temperature [String] Timesterno [String] PertNr [String]	CSV String to Model Seavator *

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 **FileNameFilter**
 - 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 **CsvModelConfiguration**
 - Start of processing
 - If the entire content of the file is processed on this event enter a wildcard in the RegEx field
 - If the processing starts at a specific line enter a regular expression in the **RegEx** field to identify the line
- 9. Click the **Apply** button, then the **Close** button and save the Channel by clicking the **Save** button

### Channel Configuration: doc.filereader:FileReaderChannel:latest		0 🗸 >
EquipmentModel EguipmentModel E	*	9 FileConsumer to String EventFileNameFilter File name filter (RegEx)
	8	CSV String to Model Enable event CsvModelConfiguration Massage filter (RegEs)

Description of configuration properties:

Property	Description	Example
Separator	Separator type, used in the csv file	, , ;
Delimiter	Values that have an additional delimiter like	11
	"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	

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

## Channel Configuration: demo:InfluxDB:latest		↔ ✓ ×
 InfluxDB UpdateWeatherData [WeatherDataEvent] zipCode [String] location [String] temperature [Float] humidity [Float] MachineData [MachineDataEvent] owerConsumption [Float] runtimes [int: 500] 	Q ♀ InfluxDB Url * http://127.0.0.1:8086 Database name * InfluxDB1 Credentials None Batch size * 1000 Flush interval (ms) * 100	

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

InfluxDB	۹ \$		
E UpdateWeatherData [WeatherDataEvent]		InfluxDB	
V location [String] V temperature [Float]	4	Z Enable	
v humidity [Float]		InfluxDbEventContext	
MachineData [MachineDataEvent]		Measurement (optional if different from the event name)	
v powerConsumption (Float)		WeatherData	
		Tag names (optional, comma separated: location=New York,street=xxx)	
	10 10 10	Type=Station	

Configuration of tags

- 5. Select the variable which should be a **tag**
- 6. Enable Extended Configuration
 - Enter a **name** if it differs from the variable name
 - Enable the checkbox **isTag**

+++ Channel Configuration: demo:InfluxDB:latest		()	~ ×
 InfluxDB UpdateWeatherData [WeatherDataEvent] ipiCode [String] temperature [Float] numidity [Float] MachineData [MachineDataEvent] powerConsumption [Float] runtimes [Int: 500] 	Q ↓ Influx Influx Influx Influx Influx Influx Influx	DB ble DbEventContext xtended configuration uxDbVariableInEventContext t (optional if different from variable name) ode sTag	

Configuration of fields

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

### Channel Configuration: demo:InfluxDB:latest	\leftrightarrow \checkmark \times
 M InfluxDB Q C Q UpdateWeatherData [WeatherDataEvent] Q ipCode [String] Q location [String] V temperature [Float] N humidity [Float] A MachineDataEvent] Q powerConsumption [Float] runtimes [Int 500] 	InfluxDB Carbon Enable InfluxDbEventContext Extended configuration InfluxDbVariableInEventContext Name (optional if different from variable name) Temperature IsTag

Array Configuration

- 9. Select the Array
- 10. To configure the Array select Extended Configuration
 - (Optional) Enter an **index** name
 - (Optional) Enter a **field** name if the event node name differs from the actual name in In-fluxDB.
 - (Optional) Enter tags separated by commas e.g., (location=NewYork, street=xxx)

ttt Channel Configuration: demo:InfluxDB:latest		↔ ✓ X
 InfluxD8 UpdateWeatherData [WeatherDataEvent] Ipocode [String] location [String] temperature (Float] humidity [Float] MachineData [MachineDataEvent] powerConsumption [Float] Turtimes [int: 500] MachineData2 [MachineData2Type] 	InfluxDB InfluxDbComplexVariableContext InfluxDbComplexVariableContext InfluxDbArrayInComplexVariableContext InfluxDbArrayInComplexVariableContext Index name (optional, default is 'index') Field name (optional if different from variable name) Tags (optional, comma separated: location=New Year)	ne) ork,street=xxx)

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 **∨**.

MATERIAL_NR	ORDER_NR	PRODUCT	CUSTOMER
HS787FSTC	121	AXY_200	DemoCompany1
HS787FSTC	123	AXY_150	DemoCompany2
HS777FSTC	120	AXY_100	DemoCompany1
HS767FSTC	123	AXY_200	DemoCompany1
	MATERIAL_NR HS787FSTC HS787FSTC HS777FSTC HS767FSTC	MATERIAL_NR ORDER_NR HS787FSTC 121 HS787FSTC 123 HS777FSTC 120 HS767FSTC 123	MATERIAL_NR ORDER_NR PRODUCT HS787FSTC 121 AXY_200 HS787FSTC 123 AXY_150 HS777FSTC 120 AXY_100 HS767FSTC 123 AXY_200

Select

- The *Command* defines that after a request is made, a reply with a result is expected.
- Parameters ¹/₂ within a Command represent a collection of query parameter query parameters are defined as Variables [∨].

• Reply ^ℚ within a Command represents the result of the Command - results are defined as Variables [∨].

					M DatabaseModel
*	MATERIAL_NR	ORDER_NR	PRODUCT	CUSTOMER	 C DatabaseCommand [Command_DatabaseComman Parameters [QueryParameters]
1	HS787FSTC	121	AXY_200	DemoCompany1	v orderivr [int]
2	HS787FSTC	123	AXY_150	DemoCompany2	v materialNr [String]
3	HS777FSTC	120	AXY_100	DemoCompany1	∧ Reply [QueryResult]
4	HS767FSTC	123	AXY_200	DemoCompany1	✓ customer [String] ✓ product [String]

How to configure the SQL-Database

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

Channel Configuration: doc.database:SQLServer:latest			<> ✓
SOLServer	٩ \$		
DatabaseCommand [Command_DatabaseCommand] O Parameters [DatabaseCommandType] v materialWr [String]		SQL Database	
V orderNr [int]		DatabaseConnection	
Reply [DatabaseCommandReplyType]		Type *	
v customer [String]	2	SQLServer	Ŧ
		Reconnect interval *	
		10	
		Jdbc URL *	
		jdbc:sqlserver://localhost:1433;databaseName=unifier;trustServerCertificate=true	
		Credentials *	
		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-	10 (in milliseconds)
	tion fails	
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

- 1. Select the **command node** in the tree on the left.
- 2. Check the **custom query** checkbox and enter the **sql query**.

+++ Channel Configuration: demo.scenario1:SQLDatabase:latest		\circ \checkmark X
DatabaseModel Command_DatabaseCommand] Parameters [QueyParameters] Corent/[int] Corent/[int] Corent/[int] Corent/[int] Customer [String] Customer [String] Customer [String]	<u></u>	SQL Database Custom Query DatabaseCommandContext S0L Guery * 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 \times
DatabaseCommand [Command_DatabaseCommand] DatabaseCommand [Command_DatabaseCommand] DatabaseCommand [Command_DatabaseCommand] materialiki(String) Corrective[int] Correcti	Q \$ SQL Database B Custom Query DatabaseCommandContext Image: Assign database column DatabaseCommandVariableColumn Column * MATERIAL_NRI	

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:SQLData	abase:latest	\diamond \checkmark X
Database Solution DatabaseInsertEvent [DatabaseInsertEventType] Ecoupment [String] Vorent/[String] MaterialN/[String] Quality [String]	2	SQL Database inset DatabaseEventContext Table SupEMO_UCI_TABLE Schema do

Protocols

MQTT

Characteristics - MQTT

Information Model Requirements

- The first Node after the root node **M** must be of type *Event* **E**.
- The following Node Types can be used under the Event Node:
 - *Variables* v with a *Simple Data Type* represents the key-value pairs.
 - *Variables* v with a *Custom Data Type* represent objects that can contain key-value pairs.
 - With *Lists* 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.

How to configure the MQTT Channel

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

Add Communication Channel		× ئا 🖬 🕀 🐳
	Group * demoscenario.json2database Name * Host	
	Description Model * demoscenario.json2database.JSON.latest *	
0	Channel type * MQTT (JSON)	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
 - 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: doc.mqtt:MQTT:latest		↔ ✓ X
MQTT 3 Muttevent [MutteventType] conserver (String] conserver (String] cuality [String] cuality [String]	Q C MQTT to String Host * 127.0.0.1 Pars * 1883 Reconnect interval * 5 Connection timeout * 60 Resplate interval * 60 Persistence folder * temp Cilent ID 000 * 1 Petatula None	

- 5. Select the **event node** in the tree on the left.
- 6. Enable either **producer** or **consumer** depending on the use case and enter a **topic name**.
- 7. Click the **Apply** button.

### Channel Configuration: doc.mqtt:MQTT:latest		↔ ✓ ×
MQTT	MQTT to String	
	TopicConfiguration 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.

### Channel Configuration: demoscenario.json2dat	abase:Host:latest	↔ ✓ ×
JSON If FileEvent [FileEventType] V ecupemento [string] O order/kr [string] materialikr [string] o cuality [string]	Q	4
	Json to Model 2 Sents configuration	
	JsonToModelEventConfiguration Field name PrivateFileEvent∥	

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.

### Channel Configuration: doc.mqtt:MQTT:latest		\leftrightarrow \checkmark \times
MOTT S MattEvent [MattEventType] C coder/v [String] C materialiv/ [String] C auality [String]	Q \$\$ Json to Model Image: Sevents configuration JsonToModelEventConfiguration 2 VariableConfiguration JsonToModelEventConfiguration Field name 3 Equipment ID	4

Certificates

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

- 1. Enable Hostname Verification (optional)
- 2. Enable the Tls Configuration checkbox
- Enter the **path** to the **CA (certificate authority) certificate** of the CA that has signed the server certificate

Note: Make sure the CA certificate is valid.

- 3. Enable the **Client** checkbox
- Enter the **path** to the **client certificate**. The client certificate identifies the client just like the server certificate identifies the server.
- Enter the **path** to the **private client key**.

- If applicable select to enter a **password** or to add from the **Credentials Manager**.
- Select the **protocol** from the Drop-Down.

Channel Configuration: demoscenario.json2database:	lost:latest	\leftrightarrow \checkmark
JSON	A Instname Verification	
FileEvent [FileEventType] veuprmento [String] orderNr [String] maserialNr [String] valuestralNr [String]	TLS Configuration CA on file C1AWSLIOT_Certificates\Certificates\Test\AmazonRootCA1.pem	
	Clent Cr.VWSVJWSJ.oT_Certificates\Certificates\Test\6940074661-certificate pem.cr key file + C.\WSVJWSJ.oT_Certificates\Certificates\Test\6940074661-private pem.key Clent Pasword None	n
	Protocol * TLSv1.2	

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 **DisconnectedBuffer** checkbox.
- 2. Set the Buffer Size defines the number of messages being hold e.g., 5000.
- 3. (Optional) Enable PersistBuffer.
- 4. (Optional) Enable DeleteOldestMessage.

M JSON Q 1 © Disconnected Buffer I FileEventType] Buffer stare Sof I order/r [String] 2 Sof I materialive [String] 3 Persist Buffer I ouality [String] 3 Delete Oldest Message	

Description of configuration properties:

Property	Description	Example
host	URL of the MQTT Broker.	test.
		mosquitto.org
port	Port of the MQTT Broker.	1883
reconnectInter-	Time interval to reconnect to the MQTT Broker after loss of	5
val	connection in seconds	
connection-	Time interval the connection times out in seconds	60
Timeout		
keepAliveInter-	Time the session persists in seconds	60
val		
persistence-	Path to a folder for the persistence store of the MQTT	temp
Folder		
clientId	Identifies an MQTT client which connects to an MQTT Bro-	MyClientID
	ker	
username	Client username	Username
password	Client password	Password
hostnameVeri-	Hostname Verification	true, false
fication		
tls	Encryption	true, false
producers	Data producer	true, false
consumer	Data consumer	true, false
protocol	TLS protocol version	TLSv1.1,
		TLSv1.2
disconnected-	Offline buffering of data	true, false
Buffer		
bufferSize	Amount of message allowed in the buffer	5000
persistBuffer	Buffer persistence	true, false
deleteOld-	Delete oldest message in buffer	true, false
estMessage		

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 vith a Custom Data Type.

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]

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	
1	Description Model * demo.PLC.latest Channel type * OPC UA Client	

- 1. Make sure the root model node is selected to configure the OPC-UA Client
- 2. Enter an **applicationName**
- 3. Configure the serverTcpConfiguration
- Enter an **ipAdress**

- Enter the **port**
- Define an **endpoint**
- Set a **requestTimeOut**
- 6. Configure the defaultSubscriptionAttribute
- Define a **publishingInterval**
- 7. Configure monitoringParameters
- Set a **samplingInterval**
- Enter a **queueSize**
- Enable discardOldest depending on the use case

### Channel Configuration: demo:OPCUA:latest		\circ \checkmark X
PLC ProcessingModule [ProcessingModuleType] Temperature [Double] State [Int] Pressure [Double] Varme [String]	Q C OPC-UA Client Application name * DemoApplication	Î
	5 Server TCP configuration IP address * 127.0.1 TCP port * 4840 Endpoint path * demo RequestTimeOut * 5000	
	6 defaultSubscriptionAttributes requestedPaulishingisterval * 1000 monitoringParameters	
	7 Samping interval * 1000 Ourse size * 1 Subscription groups	

- 8. Select the complex variable node.
- 9. Enable the complexVariables

### Channel Configuration: demo:OPCUA:latest		○ ✓ X
V PLC V ProcessingModule [ProcessingModuleType] V Temperature [Double] V State [int] V Pressure [Double] V Name [String]	<u>Q</u> : 9	OPC-UA Client ComplexVariables

- 10. Assign OPC-UA data block variables to corresponding variables in the Information Model by selecting the variable in the tree
- 11. Assign data block
 - Enable the **variables** checkbox
 - Enter the **nodeId**

### Channel Configuration: demo:OPCUA:latest		↔ ✓ ×
PLC If TrocessingModule [ProcessingModuleType] If Temperature [Double] Gase [Int] Pressure [Double] Name [String] Name [String]	Q C OPC-UA Client Image: complexVariables OpcUaComplexVariableContext Image: complexVariableNodeContext Node id * Image: complexVariableNodeContext Image: complexVariableNodeContext Node id * Image: complexVariableNodeContext Image: complexVariableNodeContext	

REST

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.

### Channel Configuration: demoscenario.csv2res	st:RESTServer:latest		↔ ✓ ×
RestDataServerModel RestComplexVar [RestComplexVarType] Quality [String] Orderkir [Int] Materiality [String] Oustomer [String] Product [String]	1 <u>2</u> 3	EST Server ath prefix * lemo Endpoint (p * localhost Port * 8000 Default content type * text/csv	
	4	Web App SSL	

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

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		<> < ×
RESTClient GetSampleData [Command_GetSampleData] O Parameters [ParameterType] V urtVariables [urtVariableType] I of [int] Reply [ReplyType] I of [int] I ame [String] I description [String]	Q C REST 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	O √ X
RestClient Q > GetSampleData [Command_GetSampleData] Parameters [GetSampleDataType] G[m] Reply [GetSampleDataReplyType] G[m] Rener [String] Gescription [String] 	REST Client Command routes CommandRouteConfiguration Ust* Mtp://ostbost.0801/api/v1/dataPoint/\$(xi) Mtp://ostbost.

- 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		\leftrightarrow \checkmark \times
M RestClient A G GetSampleData [Command_GetSampleData]	Q C REST Client	
 Parameters (GetSampleDataType] Q (m)	Command routes CommandRouteConfiguration URL.* http://ocalhost.8081/api/v1/dataPoint/\$(id) Herg method * GET Headers G Authentication Type * BASIC Oredemains * Username and password	
	Username and password Username * admin Passeord * ****	

- 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	Q Comman Comman 11 € Com Parame FieldN	ient and routes dRouteConfiguration mand routes indComplexVariableContext refs ame

Description of configuration properties:

Property	Description	Example	
URL	URL of the REST API.	http://localhost:8081/api/	
		v1/dataPoint/\${id}	
HttpMethod	HTTP method for the action performed	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 V with a Simple Data Type represents the key-value pairs.
- *Variables* vith a *Custom Data Type* represent objects that can contain key-value pairs.



SECS/GEM Client

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

Add Communication Channe	4	
	Crews *	
	droup -	
	Nome #	
	SECO /OEM	
	SECS/GEIM	
	Description	
	Description	
	Model -	
	demo.Equipment.latest	
		2
	Channel type *	
	SecsGem Client	· · · ·

- 3. Make sure the root model node is selected to configure the SECS/GEM Client
- 4. Enter the device configuration:
- input the equipment-to-host **Ip** address
- type in the TCP **Port** for the communication
- input the **Device Id**
- 5. Enter the Data Formats
- Input **CEID** format for event Ids
- Enter **RPTID** format for report Ids
- Input ALID format for alarm Ids
- 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.

M Equipment	Q 0	
CataEvent [DataEventType]	SECS GEM Client	
v pressure [String]	lp *	
 remberatore (oring) 	localhost	
	Port *	
	80	
	Device Id *	
	3012	
	Data Formate	
	5 Data Pormats	
	U4	
	POTIO *	
	U4	
	ALID *	
	U4.	
	Timeouts	
	6	
	45000	
	T5 '	
	10000	
	т6 *	
	5000	
	77 *	
	10000	

- 7. Select the logging type for the required Node Types:
- Check the Enable box
- Check the **Log Data** box

Channel Configuration: demo:SECS/GEM:lates		()
Cequipment CataEvent [DataEventType] Constant [DataEventType] Constant [String]	Q ↓ 7 FrameworkConfiguration	
V temperature (String)	Logging	
	Stateful variable	
	Enable Log Data	
	Stateless variable	
	Enable Log Data	
	Event	
	Z Enable ✓ Log Data	
	Command	
	Enable Log Data	

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

	### Channel Configuration: demo:SECS/GEM:latest				\sim	×
9	Equipment GataEvent [DataEventType] Oressure [String]	٩٥		SECS GEM Client	12	
	v temperature (String)	10	10	Events SecsEventContext		
			1	10 ° 32		

- 10. Click to check the **Events** box
- 11. Enter the event context Id which will trigger the event in the Information Model
- 12. Click on the **Apply** button
- 13. Select the variable in the tree

### Channel Configuration: demo:SECS/GEM:latest		↔ 🗸
Equipment CateEvent[DataEventType] To verseure [String] Verseure [String]	<u>Q</u> ()	SECS GEM Client

- 14. Click to check the **variables** box and configure the Secs variable context
 - select the variable **Type**
 - enter the variable Id

- click the Is SV box to check if the variable is a SV
- input the variable **Name**

Description of configuration properties:

Property	Description	Example
Ip	IP address of the Equipment	http://localhost
Port	TCP port for the communication	5000
Device Id	Id of the equipment	NJ-300
CEID	Format for event Ids	U4
RPTID	Format for report Ids	U4
ALID	Format for alarm Ids	U4
Timeouts	Time interval the connection times out in milliseconds	45000
T3	Reply timeout in the HSMS protocol	10000
T5	Connect Separation Timeout in the HSMS protocol	5000
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{c} 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



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.

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.dem	✓ Edit Communication Channel: su.demo.dashboard:SiteWise:latest ∨					
	Group * su demo.dashboard Name * SiteWise					
0	Model * Sudemo.dashboard.Analytics.latest	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 lot SiteWise service you are using

# Channel Configuration: su.demo.dashboard:SiteWise:latest		
Analytics	Q. 0	4
ModuleA [DBDataType]	AWSSiteWise	
V OrderNo [String]	Group name *	
V ProductNo (String) V StartOrder (Boolean)	3 su.demo.dashboard	
V Quantity [Int]	Model name *	
V LastPartStarted [Int]	Analytics	
V LastPartOK [Int]	Credentials Profile *	
V PercentFailure [Int]	default	
V Temperature [Double]	AWS Region *	
V Pressure [Double]	eu-central-1	
E ModuleB [DBDataType]		
ModuleC [DBDataType]		

2.2.4 General Configurations

These configurations apply for all Communication Channel Types.

Framework Configuration

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

The following Framework Logging Configurations are available:

- Stateful Variable
- Stateless Variable
- Event
- Command

For each configuration there are two ways to use logging:

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

FrameworkConfiguration

Logging		
Stateful variable		
Stateless variable		
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**.



Event Logging Output

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

→Event/ReleaseOrder
```

Event Data Logging Output

```
[INF0 ] - 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)

<··> N	Mappings					
) Ж	Group 个	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				⊕ 🖬 ı£ ×
Configuration			Rules	7 α
Group * demoscenario.csv2rest Namo * CSVTOREST				
Description				
Models		4 +		
Short name	Information model identifier *			
csv 5	demoscenario.csv2rest:CsvDataModel:latest	Ť Ö		
Short name	Information model identifier *			
rest	demoscenario.csv2rest:RestDataModel:latest	* 🖬 <	6	

2.3.3 How to create Rules

Graphical

How to create a Rule

Follow the steps described below to create *Rules*:

• Select the "Edit" button (1).

>	↔ Mappings				Q 💾 + 49
Ŀ	Group 🛧	Name	Models	Description	1
	doc.mapping	DataMapping	DataModel, DataModel		ڭ 🗸 🖡

• Select the "Add Rule" button at the top right corner (2).

Edit Mapping: doc.mappin	g:DataMapping:latest 🗸			🔖 🕢 🖯 🖬 🖒 🗙
Configuration			Rules	2 ्
Group * doc.mapping Name * DataMapping Description				
Models Short name * CsvDataModel Short name * RestServerModel	Information model identifier * doc.mapping.model:CsvFile:latest Information model identifier * doc.mapping.model:RestServer:latest	+ 6		

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

• Enter Rule name (3).

Edit Mapping: doc.mapping:Data	aMapping:late	st ~					♥ (④ ×
Model CsvDataModel	÷ []	Model RestServerModel	. E		Rule Configuration		↔ 💩 💐 ×
CsvFile CsvFile TemoEvent [DemoEventType] Temoerature [Double] Pressure [Double]	<u>२</u> २	RestServer ZemoData [DemoData Type] V DemoData [DemoData Type] V Temperature [Double] Pressure [Double]	Q	≎ 3	Rule name * CsvToRest	Rule description	
					Trigger: Tree Member		Tree Member
					Cannot be empty		O Fixed Rate Scheduler
					actions [Target <=> Source]		Fixed Delay Scheduler

- 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).

Edit Mapping: doc.mapping:Dat	taMapping:late	st ~				● (⊕ ×
Model CsvDataModel	÷ 🗆	Model RestServerModel	÷ 🗆	Rule Configuration		0 🙆 🖉 X
M CsvFile DemoEvent [DemoDataType]	<u>م</u> \$	RestServer V DemoData [DemoData Type] V Temperature [Double]	۵ \$	Rule name * CsvToRest	Rule description	ĥ
V Pressure [Double]		V Pressure (Double)		Trigger: Tree Member		Trigger Type 🟚
			2	CsvDataModel/DemoEvent		0
				actions [Target <=> Source]		

Fixed Rate Scheduler

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

Edit Mapping: doc.mapping:Data	taMapping:late	st ∽				♥ (⊕ ×	
Model CsvDataModel	÷ 0	Model RestServerModel	÷ 13	Rule Configuration		0 🙆 🖉 🗙	
M CsvFile CsvF	۹ \$	RestServer V DemoData [DemoDataType] V Temperature [Double]	۹ \$	Rule name * CsvToRest	Rule description		
V Pressure [Double]		V Pressure (Double)		Trigger: Cron Expression		Trigger Type 🏚	
				* 0*/5*?**		ō	
				actions [Target <=> Source]			

Fixed Delay Scheduler

• Input the trigger initial start delay (1), the period delay (2) and the unit (3).

Edit Mapping: doc.mapping:Da	ataMapping	:late	rst ∽							● (⊕ ×
Model CsvDataModel	Ŧ	0	Model RestServerModel	Ŧ	::	Rule Configuration	1			0 🙆 🦁 X
CsvFile	٩	¢	M RestServer	C	२ ३	Rule name * CsvToRest			Rule description	
Temperature [Double] Pressure [Double]			Temperature [Double] Pressure [Double]			1	2	3		Trigger Type 🏚
						Initial Delay * 2	Period * 5	Unit * SECOND:	6	_ Ō
						actions [Target <=> So	ource]			
						actions [Target <=> So	ource]			

Actions [Target <=> Source]

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

Edit Mapping: doc.mapping:Dat	taMapping:lat	est ~	\$ (⊕ >		
Model CsvDataModel	÷ 11	Model RestServerModel	÷ 11	Rule Configuration	0 🕹 💆 X
CsvFile CsvFile DemoEvent [DemoDataType] V Temperature [Double] V Pressure [Double]	<u>م</u> ث 1	RestServer Composition RestServer Composition RestServer Composition RestServer RestRestRestRestRestRestRestRestRestRest	٩ \$	Rule name * CsvToRest	Rule description
				Trigger: Tree Member CsvDataModel/DemoEvent	
			2	actions [Target <=> Source]	

A popup appear 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	ode'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).

Edit Mapping: doc.mapping:Dat	taMapping:late	st ~	♥ (⊙ ×			
Model CsvDataModel	÷ 0	Model RestServerModel	÷ 0	Rule Configuration		0 💩 🖉 X
CsvFile	¢ ≎	RestServer V DemoData [DemoDataType] Temperature [Double]	<u>م</u> ټ	Rule name * CsvToRest		Rule description
V Pressure [Double]		V Pressure [Double]				Trigger Type 🏚
				CsvDataModel/DemoEvent		Ö.
				actions [Target <=> Source]		2
				RestServerModel/DemoData	[DemoDataType] 📋 :=	Definition Node

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).

verModel 🖵 🖸	Pule Configuration	
	Kule comgaration	<> 🙆 🖉 X
Server emoData [DemoDataType] I Temperature [Double] Pressure [Double]	Rule name * CsvToRest	Rule description
	Trigger: Tree Member CsvDataModel/DemoEvent	
	actions [Target <=> Source]	2
	RestServerModel/DemoData/Temperature [Double] 🧰 :=	Temperature [Double]
	RestServerModel/DemoData/Pressure [Double]	Definition Node Cannot be empty
6	imoData (DemoData Type) @ Temperature (Double) @ Pressure (Double)	ImpOata [DemoData Type] Imperature [Double] Trigger: Tree Member CsvDataModel/DemoEvent actions [Target <=> Source] RestServerModel/DemoEvent RestServerModel/DemoEvent Imperature Double] Imperature Double] Imperature Imp

- After all mandatory fields have been filled out, select the "Apply" button (9) to save the newly created Rule.
- The 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.

Edit Mapping: doc.mapping:Dat	aMapping:lat	est ~					♦ ⊕ ×
Model CsvDataModel	÷ 🛙	Model RestServerModel	÷ 0	Rule Configuration			↔ 🗠 🖉 ×
CsvFile CsvFile DemoEvent [DemoDataType] Temperature [Double]	۵ \$	RestServer V DemoData [DemoData Type] Temperature [Double]	۵ ټ	Rule name * CsvToRest		Rule description	9
V Pressure [Double]		V Pressure [Double]					Trigger Type 🏚
				Trigger: Tree Member CsvDataModel/DemoEvent			0
				actions [Target <=> Source]			
				RestServerModel/DemoData/Temperature	[Double] 📋 :=	CsvDataModel/DemoEvent/Temperature	[Double] <> 📋
				RestServerModel/DemoData/Pressure	[Double] 📋 :=	CsvDataModel/DemoEvent/Pressure	[Double] <> 📋

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).


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	1٠	Rule -	StartOrder	_ '	Variable	/Fvent
LISUING	т.	Rule -	StartOruer	-	variable/	LIVEIII

```
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 2: Rule - TransferNewOrder - Event/Variable

```
MesModel.TransferNewOrder mapTo { event =>
1
2
      Try {
        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 3: Rule - File2MqttWithDB - Event/Commands

```
file.FileEvent mapTo {event =>
1
      database.DatabaseCommand.execute(command => {
2
        Try {
3
           command.orderNr := event.orderNr
4
           command.materialNr := event.materialNr
5
           CommunicationLogger.log(event, command)
6
        }
      }, reply => {
8
        mqtt.MqttEvent.send(event1 => {
9
10
           Try {
             event1.Quality := event.quality
             event1.OrderNr := event.orderNr
             event1.MaterialNr := event.materialNr
```

(continues on next page)

7

11

12

13

(continued from previous page)

```
14 event1.Customer := reply.customer
15 event1.Product := reply.product
16 CommunicationLogger.log(reply, event1)
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 4: 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
    }
```

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 5: Scheduled Rule

```
1 def rule_ScheduleNode(): Unit = {
2     _trigger.fixedRateScheduler("0/1 * * * * ? *") mapTo(() => {
3     model1.StringVariable := model2.StringVariable
4     })
5  }
```

Type Assignment

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

Listing 6: T	vpe Assignmer	t with Events
--------------	---------------	---------------

```
event1 := event2
```

Logging

1

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

Listing 7: Rule with Logging

```
EquipmentModel.Alarm mapTo {variable =>
1
      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.

Edit Mapping: demoscenario.xm	ıldatabase2r	nqtt:OrderInformationToMQTT:latest	t¥		s (🕁 🗙
Model db	÷ 0	Model file	÷ 0	Rule Configuration	△ ♥ ×
 ✓ Database ▲ OatabaseSelect [Command_Databa ▲ Parameters [RequestType] ▲ Order-Humber [String] ▲ Reply [ReplyType] ▲ Customer [String] 	Q 3	Capital Control (String) Correct-unter (String) Correct-unter (String) Coate (St	<u>م</u> :	Rule name * DataToMQTT ** file.fileVent moTo (revert ->	Rule description

Target Source Combinations

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

Trigger	Target	Source
Any node	Any node	Has to be child of trigger or simple type
Fixed Rate Scheduler		
Fixed Delay Scheduler		

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:



Complex

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

Sample Assignments:

todel n1	÷ 🖸	m2	÷ 🖸	Rule Configuration				<> 🙆 💆
I E2ETestModel ✓ V ComplexVariable [ComplexType]	۹ 2	E2ETestModel	٩ ٥	Rule name * Simple			Rule description	
C E2ECommand [Command_E2EComm E E2EVent [E2EEVentType] VariableNode [String] ArrayNode [String:1] ListNode [String]	and]	VariableNoce [String] ArrayNode [String: 1] ListNoce [String: 1] FropertyNode [String] CE2ECommand [Command_E2EComma	and]	Trigger: Tree Member m1/E2EEvent				Trigger
PropertyNode [LocalDateTime] VariableNode [String] ArrayNode [String: 1] ListNode [String]		ZEEvent [EZEEventType] VariableNode [String] ArrayNode [String: 1] ListNode [String]		actions [Target <=> Source]				
PropertyNode [LocalDateTime]		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] <> 🕻
				2				Each node assigne individually from "F2FEvent" (1)

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 SMARTUNIFIER Device Type.

• Select the SMARTUNIFIER Device Type Perspective (1).



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

	Device Types				ላ 🖞 + ድ
ک د	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		> d (⊕ 🖬 ıb ×		
3	Group * demoscenario.csv2rest			9
	SUDeviceType			
	Mappings		4 +	
	5 demoscenario.csv2rest:	CSVtoREST:latest	* •	
	Models	Channels		
	CSVDataModel	7 demoscenario.csv2rest:CSV:latest	<u>* \$ 8</u>	
	RESTDataModel	demoscenario.csv2rest:RESTServer:latest	¢	

2.5 Communication Instances

2.5.1 What are Instances

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

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).

٢	Instances				< 🖞 + ታ
> ~	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 ∽			🔖 💮 🖬 🖆 🗙
3	Device Type demoscenario.csv2rest:SUE	DeviceType:latest	*	
4	Group * demoscenario.csv2rest			
	Name * SUInstance			
	Description			
	Mappings			
	Mapping demoscenario.csv2rest:0	CSVtoREST: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 Taller to titring com.amorphsys.140.adapter.channel:FileTailerToStringLayer:1.5.0-SNAPSH0T				
	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).

🗄 Information Mode	ls			Q 💾 + 69
Group Filter <	Group 🕁	Name	Description	
Show All	demoscenario.xmldatabase2mqtt	Database		Ċ 🖍 🗇
scenario1 demoscenario	demoscenario.xmldatabase2mqtt	Equipment		Ċ 🖍 🗊
json2database xmldatabase2mqtt	demoscenario.xmldatabase2mqtt	Host		Ċ / Ō
	demoscenario.json2database	Database		Ċ 🖍 🗇
	demoscenario.json2database	JSON		Ů 🖍 🗊
	demoscenario.csv2rest	CsvDataModel		Ċ 🖍 🗇
	demoscenario.csv2rest	RestDataModel		Ċ 🖍 🗊
	demo.scenario1	DatabaseModel		Ċ / Ō
	demo	DBModel		Ċ 🖍 🗊
	demo	EquipmentModel		Ċ 🖍 🗇

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).

La Information Mode	Filtered by: csv2rest × 3			ር 🖞 + <i>ና</i> ታ
Group Filter <	Group 🕹	Name	Description	
 Show All demo scenario1 demoscenario csv2rest json2database xmldatabase2mqtt 	demoscenario.csv2rest	CsvDataModel		Ľ 🖍 🗊
	demoscenario.csv2rest	RestDataModel		Ċ 🖍 🙃

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 🔍 🕄 🖬 🖆 🗙
L M Equipment ▲ FileEvent [FileEventType] ● © Creat-Number [String] ● © Dest [String] ● © Quartity [String] ● ●	Q ≎ droup * demo.scm Name * Equipment Description	
 Enter a version number. Click Ok to confirm. 		
	component-mar	amagement//_images/Operations/scm/ReleasePc

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

>	Edit Model: demo.scm:Equipment	t:1.0.0 ~	×
L	Equipment FileEvent [FileEventType] Contect Lumber [String]	latest Q 0	- Group demo.scm
***	V ProductNumber [String] V Date [String] V Quality [String]		Name Equipment
↔	V Quantity [String]		Description
Ð			

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,	nformation Models Filtered by: demoscenario ×			Q (¹) + 43
> ≅	Group 🕈	Name	Description	1
P FILTE	demoscenario.csv2rest	CsvDataModel		Ċ 🗡 🗇
GROUF	demoscenario.csv2rest	RestDataModel		ů 🧪 🖻
	demoscenario.json2database	Database		Ċ 🖍 🗇
	demoscenario.json2database	JSON		ů 🧪 🖻
	demoscenario.xmldatabase2mqtt	Database		Ů 🖍 🗊

The component is opened in the edit mode.

Edit Model: demoscenario.csv2rest:CsvDataModel:latest	/	s (0) 🖉 🖻 16 ×
CsvDataModel CsvDataModel CsvDataModel PARTUR [String] TIMESTANP [String] TIMEFEATUR [String] PRESSURE [String]	Q \$	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 ~	🗞 🕑 🖬 и́с 🗙
CsvDataModel Q C CsvDemoEvent [CSVDemoEventType] PARTIR [String] THINESTAMP [String] TEIMEERATUR [String] TEIMEERATUR [String] PRESSURE [String]	Group * demoscenario.csv2rest Name * CsvDataModel
	Lescription

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 ~		🔖 🕲 🖬 🖆 🗙
CovDataModel	Group * demoscenario.csv2rest Name * CsvDataModel Description	1

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



3.4.5 Save

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

Edit Model: demoscenario.csv2rest:CsvDataModel:latest ~	» (© 🖉 🖬 اله ×
CsvDataModel CsvDemoEvent (CSVDemoEventType) PARTNR [String] VTIMESTAMP [String] VTIMESTAMP [String] V PRESSURE [String]	Group * 1 demoscenario.csv2rest Name * CsvDataModel
	Description

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

Edit Model: demoscenario.csv2rest:CsvDataModel:latest ~	🍋 🕲 🖉 🖬 ഥ ×
CsvDataModel CsvDataModel PARTNR [Gring] TimberStaff [Sring] PRESSURE [Sring] PRESSURE [Sring] Comparison of the state of the sta	Group * demoscenario.csv2rest Name * CsvDataModel Description

3.4.6 Save and Close

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

Edit Model: demoscenario.csv2rest:CsvDataMode	:latest ~	ان 🕲 🕲 🕲 🖒 🗙
CsvDataModel CsvDataModel CsvDataModel PARTINE [String] TIMESTAVIP [String] TIMESTAVIP [String] PRESSURE [String]	Group * demoscenario.csv2rest Name * CsvDataModel	0
	Description	

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

Information Models Filtered by: demoscenario ×				Q (L) + 43	
× د	Group 🛧	Name	Description		
0 FILTE	demoscenario.csv2rest	CsvDataModel			Ů 🖍 🗊
GROUF	demoscenario.csv2rest	RestDataModel			Ů 🖍 🗊
	demoscenario.json2database	Database			Ů 🖍 🗊
	demoscenario json2database	JSON			Ů 🖍 🗊
	demoscenario.xmldatabase2mqtt	Database			Ů 🖍 🙃
	demoscenario.xmldatabase2mqtt	Equipment			ů 🧪 🙃
	demoscenario.xmldatabase2mqtt	Host			Ů 🖍 🗊
	2 The entr	y 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	ls			1 9 4 +	C ₂
Group Filter <	Group 🕁	Name	Description		Î
 ✓ Show All ✓ demo 	demoscenario.xmldatabase2mqtt	Database		Ů 🖍 🗊	Г. – I
scenario1 demoscenario	demoscenario.xmldatabase2mqtt	Equipment		Ů 🖍 🗊	ī -
json2database xmldatabase2mqtt	demoscenario.xmldatabase2mqtt	Host		凸 🖍 🗊	

Enter a search term (2).

				× 凸 + 分
Group Filter <	Group 🗸	Name	Description	
 Show All demo scenario1 demoscenario cev2rect 	demoscenario.csv2rest	CsvDataModel		ථ 🖍 🗇
	demoscenario.csv2rest	RestDataModel		Ċ 🖍 🖬
json2database xmldatabase2mott				

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

t csv				× Ľ + &
Group Filter <	Group 🗸	Name	Description	3
 Show All demo 	demoscenario.csv2rest	CsvDataModel		Ů 🖍 🗊
scenario1 demoscenario	demoscenario.csv2rest	RestDataModel		ථ 🖍 🗊
json2database xmldatabase2mgtt				

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.

La Information Mode	Filtered by: demoscenario X			ላ 🖞 + 🕁
Group Filter <	Group	Name 🛧 🚺	Description	
 ✓ Show All ✓ demo 	demoscenario.csv2rest	CsvDataModel		ů 🧪 🖻
scenario1 ✓ demoscenario	demoscenario.json2database	Database		ů 🧪 🖻
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.

La Information Models Filtered by: demoscenario X					
Group Filter <	Group 🕁	Name	Description	1	
 ✓ Show All ✓ demo 	demoscenario.xmldatabase2mqtt	Database		Ċ 🖍 🗊	
scenario1 demoscenario	demoscenario.xmldatabase2mqtt	Equipment		Ů 🖍 🗊	
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 🔀
     ₽(
           "info": {
  3
               "identifier": {
  4
                    "id": "8b6bc3f2-192d-4870-8bad-b7c2f5e191b9", •
                    "version": "latest"
  5
  6
               },
                                              2
               "externalIdentifier": {
  7
                    "name": "CsvDataModel",
 8
  9
                    "group": "demoscenario.csv2rest"
               },
 11
                "externalDescriptor": {
                    "description": " "
 13
               }
 14
           },
           "members": [{
 15
                    "id": "csvDemoEvent",
 16
 17
                    "description": "",
                    "definitionType": "Event",
 18
                    "typeName": "CSVDemoEventType"
19
               }
           ],
           "types": {
22
23
                "CSVDemoEventType": {
24
                    "id": "CSVDemoEventType",
25
                    "members": [{
                             "id": "PARTNR",
"description": "",
26
27
                             "definitionType": "Variable",
28
                             "typeName": "String"
29
                        ), {
                             "id": "TIMESTAMP",
                             "description": "",
"definitionType": "Variable",
 33
                             "typeName": "String"
34
                        }, {
                             "id": "TEMPERATUR",
36
                             "description": "",
"definitionType": "Variable",
37
                             "typeName": "String"
 39
 40
                        }, {
                             "id": "PRESSURE",
41
42
                             "description": "",
                             "definitionType": "Variable",
43
                             "typeName": "String"
44
                        }
45
46
                    1,
47
                    "category": "model",
                    "type": "StructuredVariable"
48
49
               }
50
           },
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.

La Information Mode	Is Filtered by: demoscenario ×			ዲ 💾 + ታ
Group Filter <	Group	Name 🛧	Description	4
✓ Show All ✓ demo	demoscenario.csv2rest	CsvDataModel		Ċ 🖍 🗊
scenario1 demoscenario	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			۶ 🗸 ک	O 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	<u>کا</u>	ON-Datei (*.json)	~		
			6	Open	Cancel .::

The imported component is now listed (7).

🗄 Information Mode	Filtered by: csv2rest ×			Q [¹] + 43
Group Filter <	Group 个	Name	Description	
 ✓ Show All ✓ demo 	demoscenario.csv2rest	7 CsvDataModel2		ů 🖍 ô
scenario1 demoscenario	demoscenario.csv2rest	CsvDataModel		Ċ 🖍 🗊
json2database xmldatabase2mqtt	demoscenario.csv2rest	RestDataModel		Ċ 🖍 🗇

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	S Filtered by: csv2rest ×			Q 💾 + 43
Group Filter <	Group 🛧	Name	Description	
 ✓ Show All ✓ demo 	demoscenario.csv2rest	CsvDataModel		1 🖒 🗡 🙃
scenario1 demoscenario	demoscenario.csv2rest	RestDataModel		Ċ 🖍 🙃
json2database xmldatabase2mqtt				

3.4.12 Clone

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

✔ Edit Model: demoscenario.csv2rest:CsvDataModel:latest ~	🔍 (@ 💆 🖻 нб 🗙
CsvDataModel CsvDamoEvent [CSVDemoEventType] A C csvDamoEvent [CSVDemoEventType] A Till Sstraw[Cstring] A TELMEERATURE [String] A PRESSURE [String] A C constraints of the string	Greep *

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 CovDernoEvent[CSVDemoEventType] PARTNR [String] TIMESTAMP [String] TEMPERATUR [String] PRESSURE [String]	Group * demoscenario.csv2rest Name * CsvDataModel [CLONE] Description

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

3.4.13 Delete

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

La Information Mode	Els Filtered by: csv2rest ×			Q (¹) + 43	
Group Filter <	Group 🛧	Name	Description	1	
 ✓ Show All ✓ demo 	demoscenario.csv2rest	CsvDataModel [CLONE]		Ċ 🗡 🖥	
scenario1 demoscenario	demoscenario.csv2rest	CsvDataModel		Ů 🖍 🗊	
json2database xmldatabase2mqtt	demoscenario.csv2rest	RestDataModel	RestDataModel		

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

Delete Model

Are you sure you want to delete this Information Model?



The component is deleted.

🗄 Information Mod	els Filtered by: csv2rest ×			Q (¹) + 5	
Group Filter <	Group 🕈	Name	Description		
 ✓ Show All ✓ demo 	demoscenario.csv2rest	CsvDataModel		Ů 🗡 Ō	
scenario1 demoscenario csv2rest	demoscenario.csv2rest	RestDataModel	RestDataModel		
jeonäätabase xmitäatabase2mytt		Entry successfully deleted	ok [2]		

3.4.14 Bulk Action

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

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

- Start
- Stop
- Deploy
- Undeploy

>	Å C	eploy	. 🛛 🕆 😤							۹ ۱	4) +	G	۵
Ŀ	، ۳		Group 🕇	Name	Version	Deployment Type	State						
***	FILTE		ex1	CSVtoRESTDeviceType	latest	Local	NotDeployed	•	-	5	E۲	1	Î
~	GROUF		ex2	JSONDatabaseChannel	latest	Local	NotDeployed	•	-	5	E۲	1	i
			ex3	OpcUa Device Type	latest	Local	NotDeployed	•		5	Ξ۲	1	i
ì													
۲													
A													
•													

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

A 1	Deploy	•	= ♀ ¥	4						۹	Ľ	+	£,	¢
ж Ч		:	Group 个	Name	Version	Deployment Type	State							
	~		ex1	CSVtoRESTDeviceType	latest	Local	NotDeployed	►		Ŧ		≣≁	/	Î
GR			ex2	JSONDatabaseChannel	latest	Local	NotDeployed	►		Ŧ		Ξ×	/	
			ex3	OpcUa Device Type	latest	Local	NotDeployed	►	-	Ŧ	5	Ξ×	/	Î

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).

â.	Deploy	•	• *	₽,							Q	Ċ	+	÷	\$
› «		:	Group ↑ 5		Name	Version	Deployment Type	State							
PILTE	~		ex1		CSVtoRESTDeviceType	latest	Local	NotDeployed	►		Ŧ	5	Ξv	1	Î
GROUF	~		ex2		JSONDatabaseChannel	latest	Local	NotDeployed	►	-	Ŧ	5	Ξ×	1	Î
0			ex3		OpcUa Device Type	latest	Local	NotDeployed	►		Ŧ	8	Ξ×	1	

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	Deploy	F	■ 🕹 😤				11		۹	Ľ	+	G	٥
, ж	Z	:	Group 个	Name	Version	Deployment Type	State						
o FILTE	\checkmark		ex1	CSVtoRESTDeviceType	latest	Local	Stopped	►	$\widehat{\tau}_{*}$	8	Ξ×	1	Î.
GROUF	\checkmark		ex2	JSONDatabaseChannel	latest	Local	Stopped	►	₹,	5	Ξ۲	1	Î.
0	\checkmark		ex3	OpcUa Device Type	latest	Local	NotDeployed	►	Ŧ	5	Ξ×	1	Î

Deploym	ents		
ex1:CSV	/toRESTDeviceType:late	e 🗸	
ex2:JS0	NDatabaseChannel:lat	e 🗸	
ex3:0pc	Ua 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).

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

• Select the SMARTUNIFIER Deployment perspective (1).



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

								2
>	â I	Deployment					< 🖞	+ -9 \$
Ŀ	> ب	Group 🛧	Name	Version	Deployment Type	State	3	Local
.	JP FILTE							Docker
** *	GROL							AWS

- Select the SMARTUNIFIER Communication Instance to be used in the Deployment (4).
- Select the *log file level* (5). We recommend the log level of type *Info* in case of a normal deployment scenario.
- (Optional) Enable *Encryption* (6).
- (Optional) Enable *Protection* (7).
- (Optional) Add VM Arguments (8).
- When all mandatory fields are filled click the "Save" button (9).

Add Local Deployment		9 🖬 🖆 🗙
4	Instance * demoscenario.csv2rest:SUInstance:latest 🗸	
5	Log File Configuration + Info -	
6	Z Enable Encryption	
	Same folder as deployment Custom Path:	
7	Z Protected	
8	VM Arguments	
-	JMX Properties	
	JAX Host Name * localhost	
	JMX Port *	
	JMX Authenticate	
	Authenticate	
	🗹 Local Only	
	Use SSL	

4.3 Deploy with Docker

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

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

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

• Select the SMARTUNIFIER Deployment perspective (1).



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

>	Å	Deployment				م (')	+ -9 🕸
Ŀ	> ~	Group 个	Name	Version	Deployment Type	State	Local
♣	P FILTE					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 🖬 ம ×
4	Instance - demoscenario.csv2rest:SUInstance:latest	
5	Endpoint ID * demoscenario.csv2rest:LocalDockerEndpoint 🗸	
6	image * adoptopenjdk:11-jre-hotspot *	
0	tog File Configuration *	
8	Enable Encryption	
9	Protected	
	Volumes + 10	
11	Local Container /home/developer/temp/files : /files	
13	VM Arguments	
•	JMX Properties	
	Default Arguments	
	sms " 32m	
	xux - 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 a *SSH Deployment Endpoint*. The Deployment Endpoint specifies the location where you want the Instance to run.

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

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

• Select the SMARTUNIFIER Deployment perspective (1).



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

A	Deployment					२ 🖞 + <i>छ</i>
› «	Group 🕇	Name	Version	Deployment Type	State	Local
FILTEI						Docker
ROUP						AWS
0					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			∂ 1 <u>6</u> ×
4	Instance * demo: CSVERESTDevice Type: latest Engenit D * demo: VM 1 Log File Configuration * Info Protected VM Arguments false JMX Properties Default Arguments XMS * 32m XMS * 256m Image: Heap Dump On Out Of Memory Error Additional	· · ·	5

• 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/AmazonEC2ContainerRegistryFullAcces	s Provides administrative access
	to Amazon ECR resources.
arn:aws:iam::aws:policy/AmazonECS_FullAccess	Provides administrative access
	to Amazon ECS resources and
	enables ECS features through
	access to other AWS service 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": [
            "*"
        ]
    },
    {
        "Sid": "ECRPullPolicy",
        "Effect": "Allow",
        "Action": [
```

{

(continues on next page)

(continued from previous page)

```
"ecr:BatchCheckLayerAvailability",
             "ecr:GetDownloadUrlForLayer",
             "ecr:BatchGetImage"
         ],
         "Resource": [
              "*"
         1
     },
     {
         "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	Instance	Wo	orkload
		(Number	of	Map-
		pings)		
0.25 vCPU	0.5GB, 1GB, and 2GB	<= 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 SMARTUNIFIER 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 set up an AWS Deployment Endpoint please refer to chapter: *AWS Endpoint*. Follow the steps described below to deploy a SMART**UNIFIER** Instance on AWS Fargate:

• Select the SMARTUNIFIER Deployment perspective (1).

			≡ / 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).					6
>	🌲 Deployment					م ٿ +
ŧ	> Group ↑	Name	Version	Deployment Type	State	Lo
•	GROUP FILT					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 SMARTUNIFIER 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 *	
đ		Security Group * SMARTUnifierIntegrationTestECSContainer_SG *	
ۍ ۵		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					
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.

Å I	Deployme	nt					ዓ 💾 + 🕁 🏟
> #		Group 🛧	Name	Version	Deployment Type	State	1
FILTE		su.demo.dashboard	SUInstance	latest	Local	NotDeployed	> = 🕹 🖽 🖾 🗡 📋
GROUF							

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.

\$ C)eployme	nt					९ 💾 + 🕁 🏚
» ж	□ :	Group 🛧	Name	Version	Deployment Type	State	2
PILTE		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).

Å □	eploy	/mer	nt						ላ 🖞 +	- - - - - -
х х	\checkmark	:	Group 🕈	Name	Version	Deployment Type	State			
P FILTE	\checkmark		su.demo.dashboard	SUInstance	latest	Local	Started	► ■	£× ∎ ∎	/ 1
GROUF								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	ymer	nt					< 🖞 + ኇ 🔅
× ×	~	:	Group 🛧	Name	Version	Deployment Type	State	
° FILTE	\checkmark		su.demo.dashboard	SUInstance	latest	Local	Stopped	> = 🖑 🔢 🖉 🗡 🗊
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).

ا ھ	Deployme	ent					ዓ 💾 + 🕁 🏟
× ×	□ :	Group 🛧	Name	Version	Deployment Type	State	
FILTE		su.demo.dashboard	SUInstance	latest	Local	Stopped	> = 🕹 🏭 🖬 🖉 🖍 🛢
GROUF							5

• 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.

Å C	eploym	ent						< 🖞	+ 4	5 \$
х х		Group 🛧	Name	Version	Deployment Type	State				
° FILTE		su.demo.dashboard	SUInstance	latest	Local	NotDeployed	►	- 🕹 👪	2	
GROUP						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.

I	Deployme	nt					
× H		Group 🛧	Name	Version	Deployment Type	State	
° FILTE		su.demo.dashboard	SUInstance	latest	Local	NotDeployed	> = 🕹 II 🖬 🗡 🗎
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			>
## Channels				A Statue Stoppe
Info	Туре	Status	Model	© Time start 2022-02-15 11:45:2
demo1:RESTServer.latest	Rest Server	Stopped	demo1:RestDataModel:latest	
demo1:CSV:latest	File tailer (CSV)	Stopped	demo1:CsvData.latest	
				@ CPU Usage (0% / 0%
🖬 Log Viewer: 90596ff9-bae1-4039-9688-e184at	o70f698:latest		A 🗕 — A 🚍 Q	
2022-02-15 11:23:34,448 - [INFO] - [INFO] - Main - About to start c:\Users\raa\Desktop\Unifier\SmartUnifierManager-windows-x64_15_02_2	instance instance.p90.m596ff9bae14 2\deploy\90596ff9-bae1-4039-9688-e	0399688e184ab70f698.vlate: 184ab70f698\.	t.Instance596ff9bae140399688e184ab70f698 from folder	
2022-02-15 11:23:34,465 - [OEBUG] - [DEBUG] - Nain - About to create 2022-02-15 11:23:34,465 - [DFW] - [NFO] - Nain - Instance denoil 2022-02-15 11:23:34,587 - [OEBUG] - [DEBUG] - Nain - PID 67000 writh 2022-02-15 11:23:34,762 - [OEBUG] - [DEBUG] - ConfigurationResolver5 2022-02-15 11:23:34,762 - [OEBUG] - [DEBUG] - ConfigurationResolver5	new instance of instance.p80.m596 CSVtoRESTDeviceType:latest proce en to file c:\Users\raa\Desktoy\Ur - 'Unresolved' Config loaded from - Config loaded from resource	<pre>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>	8.ulatest.instance500ff9baal403956686.Blaab704698 wlndows-x64_15_02_22/deploy/98596ff9-bael-4039-9688-e184ab704698\pld.txt	Memory Usage (09)
<pre>2022-02-15 11:23:34,767 = [DEBUG] = [DEBUG] = RestServerImplementati ("type":"ApplicationJson"),"ip":"127.0.0.1","port":8091),"frameworkCG ("enabledatlogging":false,"enablelogging":false),"statefulVariableL {"enableDatlogging":false,"enablelogging":false)}},"pathPrefix":"der</pre>	on - demol:RESTServer:latest - u onfiguration":{"loggingConfigurati ogging":{"enableDataLogging":false mo"}))	<pre>inMarshallConfiguration con lon":("commandLogging":("en e,"enableLogging":false),"!</pre>	fguration Config(SimpleConfig0bject(("endpoint":("efmultContentType": ableDataLogging":false,"enableLogging":false),"eventLogging": tatelessVariableLogging":	
2022-02-15 11:23:36,030 - [INFO] - [INFO] - RestServerImplementation RestServerImplementationConfiguration(demo,RestServerEndpoint(127.0. 2022-02-15 11:23:36,031 - [DEBUG] - [DEBUG] - RestServerImplementati	on - demol:RESTServer:latest - 0 0.1,8091,ApplicationJson,None,None on - demol:RESTServer:latest - 0	Configuring channel 377e520 e),FrameworkConfiguration(I Configuration:	7-661-4614-bce7-6f9a2e65ab08:latest with config oggingAwareConfiguration(ArtifactLoggingConfiguration(false,false),ArtifactLoggingCor	fig
<pre>kestberverimplementation.oniguration(demo,RestServerEndpoint(127.0.) 2022-02-15 11:23:36,033 - [DEBUG] - [DEBUG] - RestServerImplementati</pre>	0.1,8091,ApplicationJson,None,None on - demo1:RESTServer:latest - N),FrameworkConfiguration() to server for endpoint Rest	oggingAwarecon+iguration(Arti+actioggingconfiguration(false,false),ArtifactloggingCor ServerEndpoint(127.0.0.1,8091,Application]son,None,None) exist. Creating a new one	118

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.pro											
>	🌲 Deployment						View All Notifications	+	c,			
Ŀ	Group Filter		Group 🛧	Name	Version	Deployment Type	State	Ť	25 minutes ago			
A	✓ Show All demo1		demo1	CSVtoRESTDeviceType	latest	Local	Stopped	0	TelFaller - demont.csv:_latest - File c:\Users\kab\Desktop\testCSV.csv does not exist Sf minutes ago	1	Î.	
***								9	Instance Error: demo1:CSVtoRESTDeviceType FileTailer- demo1:CSV:latest - File c:\Usersikab\DesktopttestCSV.csv does not exist 24 24 minutes ago			
<··>								9	Instance Error: demo1:CSVtoRESTDeviceType FileTailier - demo1:CSV:latest - File c:\Users\kab\Desktop\testCSV.csv does not exist 24 minutes ago			
0								0	Instance Error: demo1:CSVtoRESTDeviceType FileTailer- demo1:CSV_lstest-File c:\Users\kab\Desktop1testCSV.csv does not exist 24 minutes ago			
٨								9	Instance Error: demo1:CSVtoRESTDeviceType FileTailer - demo1:CSVlatest - File c:\Users\kab\DesktopttestCSV.csv does not exist S 24 minutes ago			
۲								_				

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

	🌲 Notifi	ications				Q 43
3		: 10	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	
4		23	InstanceError	FileTailer - demo 1:CSV_latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:49	R
		22	InstanceError	FileTailer - demo1:CSV_latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:46	8
		21	InstanceError	FileTailer - demo1:CSV:latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:43	R
		20	InstanceError	FileTailer - demo1:CSV:latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:40	8
		19	InstanceError	FileTailer - demo1:CSV_latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:37	R
		18	InstanceError	FileTailer - demo1.CSVlatest - File c:\Users\kabiDesktopitestCSV.csv does not exist	15 Feb, 2022 10:42:14	

After selection a pop-up appears providing two options.

≜ N	of 📰					Q	c,			
1	:	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	8	Î.			
~		23	InstanceError	FileTailer - demo1:CSV:_Jatest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:49	8	Ĩ.			
\checkmark		22	InstanceError	FileTailer - demo1:CSV:Jatest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:46	8	Î			
~		21	InstanceError	FileTailer - demo1:CSV:_latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:43	8	i i			
\checkmark		20	InstanceError	FileTailer - demo1:CSV:Jatest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:40	8	Î			
~		19	InstanceError	FileTailer - demo1:CSV:_latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 11:23:37	8	i i			
~		18	InstanceError	FileTailer - demo1:CSV:latest - File c:\Users\kab\Desktop\testCSV.csv does not exist	15 Feb, 2022 10:42:14	5	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.	oro				8	0
>	🌲 No	otificatio	ons				٩	£
Ŀ		:	ID	Event Type	Message Event	Time ↓		
.			24	InstanceError	FileTailer - demo1:CSV_latest - File c:\Users\kab\Desktop\testCSV.csv does not exist 15 11	Feb, 2022 :23:52	5	ii.
***			23	InstanceError	FileTaller - demo 1:CSV:latest - File c:\Users\kab\Desktop\testCSV.csv does not exist 11	Feb, 2022 :23:49	5	Î
⟨·· ⟩			22	InstanceError	FileTailer - demo1:CSVlatest - File c:\Users\kab\Desktop\testCSV.csv does not exist 11	Feb, 2022 :23:46	~	Î
R			21	InstanceError	FileTailer - demo1:CSVlatest - File c:\Users\kab\Desktop\testCSV.csv does not exist 11	Feb, 2022 :23:43	5	i.
23			20	InstanceError	FileTaller - demo 1:CSV:latest - File c:\Users\kab\Desktop\testCSV.csv does not exist 11	Feb, 2022 :23:40	5	Î
÷			19	InstanceError	FileTailer - demo1:CSVlatest - File c:\Users\kab\Desktop\testCSV.csv does not exist 15	Feb, 2022 :23:37	5	I.
Ą.			18	InstanceError	FileTailer - demo1:CSV_latest - File c:\Users\kab\Desktop\testCSV.csv does not exist 15 10	Feb, 2022 :42:14	5	i.
•								

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

≡	MORPH.pro							8 0
>	A Notifications						······································	Q 69
Ŀ,		:	ID	Event Type	Message	Event Time 🔸		
•								

⟨·· ⟩								
Ð								
۲								
Å								
•								

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).

>	≜ D	eployment					Q 💾 + 67 🏟
Ŀ,	> ∝	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 deployment Id 06484d90-fe4e-49b0-873a-42234162506c:1.0.0	2 AO A	٩	II ↓	×
1 2014-03 3 3 5535.464 - 1190 1 - 0101 - 010 - 1 Ann - Annor to start startes appende adversame 2014 (2014)	<pre>windows-x64-3\deploy\06484d90-fe4e-49b0-873a-4] - [main] - Main - About to start instance in</pre>	223 3 c\. stance.p8648.p	5 1d)8.nfete49b	8873442234:
[32] 431 (443) [5153,101] [006] [and) configurational over (introductor) (introduc	Config(SimpleConfig(Deject(('commands':[],"even ts':[('messagefilterMegk':'.'', ''modelPath':['p ","stringDelimiter':'\",","usitTimeout':("length urationType(10,Seconds),List(CsvEventConfigurat	ts":[{"message ath":"/Event/c ":10,"unit":{" ion(EventPath(ilterRegEx": svDemoEvent"} type":"Second (Event/csvDem	<pre>, "mode], "ignorel "}})) oEvent), Sor</pre>
[1] 2024-0-3 DiSCLAD [100] [201] - Discretizing of the second state of the second s	<pre>ig_patn=. Complete conig: Lonig()sipleConig() .csv","reopenBetweenChurkS"true,"failFromEnd" ingLayerConfiguration(C:\Projects\SMMRTUNIFIER)</pre>	oject({ delay true})) Source_for_Dem	:{ length :25	les\Equipm
[2] 2014-05 ESESSION - DESC. [DESC] - Intributive - vertain effective Unrealing for a first constant for / Intributive - vertain effective Unrealing for a first constant for / Intributive - Intributive - Vertain effective Unrealing for a first constant for / Intributive - Intributive - Vertain effective Unrealing for / Intributive - Vertain effective Unrealing	Config(SimpleConfigObject({"endpoint":{"defaul	tContentType":	["type":"Appl	icationJso
31 201-04:3 15:55:37.0% [DB0] [cm1] estip="testime-relationer statuses of encourse and encourses	<pre>localhost","port":8991,"pathPrefix":"demo"})) point(localhost,8091,ApplicationJson,None,None)</pre>)		
[2] All-4.5 [1551,46] - [006] - [015] - Bit] - B	estDemoDataType			
[4] 2014-02.1 51:53.(07). [100] - [aii] - Instrumeringhementation - denoscenario curverstallSizerer_1.4.4. iscolbast:BRI(denoVriable/Instrumentation added to SGT bandler B 2014-04.5 ISS:S3.(07). [100] - [aii] - Instrumeringhementation - denoscenario curverstallSizerer_1.4.4. iscolbast:BRI(denoVriable/Instrumentation/Instrumentation) B 2014-04.5 ISS:S3.(07). [100] - [aii] - Instrumeringhementation - denoscenario.curverstallSizerer_1.4.4. iscolbast:BRI(denoVriable/Instrumentation) B 2014-04.5 ISS:S3.(07). [100] - [aii] - Instrumeringhementation - denoscenario.curverstallSizerer_1.4.4. iscolbast:BRI(denoVriable/Instrumentation/Instrumentation) D 2014-04.5 ISS:S3.(07). [100] - [aii] - Instrumeringhementation - denoscenario.curverstallSizerer_1.4.4.4. iscolbast:BRI(denoVriable/Instrumentation) D 2014-04.5 ISS:S3.(07). [100] - [aii] - Instrumeringhementation - denoscenario.curverstallSizerer_1.4.4.4. iscolbast:BRI(denoVriable/Instrumentation) D 2014-04.5 ISS:S3.(07). [100] - [aii] - Instrumeringhementation - denoscenario.curverstallSizerer_1.4.4.4. iscolbast:BRI(denoVriable/Instrumentation) D 2014-04.5 ISS:S3.(07). [100] - [aii] - Instrumeringhementation - denoscenario.curverstallSizerer_1.4.4.4. iscolbast:BRI(denoVriable/Instrumentation) D 2014-04.5 ISS:S3.(07). [100] - [aii] - Instrumeringhementation - denoscenario.curverstallSizerer_1.4.4.4. iscolbast:BRI(denoVriable/Instrumentation) D 2014-04.5 ISS:S3.(07). [100] - [100] - Instrumeringhementation - denoscenario.curverstallSizerer_1.4.4.4. iscolbast:BRI(denoVriable/Instrumentation) D 2014-04.5 ISS:S3.(07). [100] - [100] - Instrumeringhementation - denoscenario.curverstallSizerer_1.4.4.4. iscolbast:BRI(denoVriable/Instrumentation) D 2014-04.5 ISS:S3.(07). [100] - [100] - Instrumeringhementation - denoscenario.curverstallSizerer_1.4.4.4. iscolbast:BRI(denoVriable/Instrumentation) D 2014-04.5 ISS:S3.(07). [100] - [100] - Instrumeringhementation - denoscenario.curverstallast:BRI(denoVriable/Instrumentation) D 2014-04.5 ISS:S3.(07). [100] - [100] - [100] - [100] - [
[1] 2014-33 31533104 - 1090 - 1-001 - Instru-Findprenettal - descourse convertailSiseer_1.4.8 - Loadback:USIGenver1.4.0.TextBifdenver1ablemetation-interaction and and and and and and and and and an	ter - Init mappings ingafc444e790934aaa328b9477			
4) 241 (494-5) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1985); [181] - Kattyler - String & Bayley Collect: (2014-65) ESS3,487 - 1	0000008005e2840@49e4c2d5			
[4] 2014-33 D:SS3(20) - [D005] - [arlife=i-bread=] - behalvestinglementatio - encoursin convertionyL&# - obtanchong("synthesis in the intervention") - intervention -</td><td>- preStart() invoked</td><td></td><td></td><td></td></tr><tr><td>1 2014-3 15553.27 • 190) • enfor-in-trends - interaction - control default and the second state - interaction - i</td><td>_ror_pendo(bara(cv-risk(cquipment).cv),buratic , ReopenBetweenChunks=true, TailFromEnd=true</td><td>nconfiguration</td><td>ype(250,0111</td><td>seconds),</td></tr><tr><td>[1] 3214-3 15553.34. [U003] - [u015.4] -</td><td><pre>v2rest:RESTServer:1.0.0 - Starting channel 68</pre></td><td>76fa4a-2c9b-47</td><td>c9-ac96-f8de6</td><td>xc65f3b:1.4</td></tr><tr><td>Exception 2.15:05.16.4 (DBC) - Unitaria-tornessi - Unitaria-tor</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></tbody></table>				

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 [eployn	nent					ې (ئ) ج
×		Group 🕇	Name	Version	Deployment Type	State	
° 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

tt Channels						
Info	Туре	Status	Model	Messages	⊘ Status⊙ Time start	Started 2022-03-15 11:50:08
test:Out:latest	In Memory	ConnectedState	test:Model1:latest	4 / 9800	Time Up	3h 13min 47s
test:in:latest	In Memory	ConnectedState	test:Model1:latest	4 / 9800		
🛃 Log Viewer: 5d	7d062a-8d5c-4b43-97b9-72f7141	5ba61:latest		A .	– A Q ↓	(0% / 19%)
2022-03-15 15:03:49,963 2022-03-15 15:03:49,963 n87781t7) 2022-03-15 15:03:49,964 n87781t7)	- [INFO] - VariableDefinition - InMemoryModelIn - [INFO] - VariableDefinition - InMemoryModelIn - [INFO] - VariableDefinition - InMemoryModelIn	splementation-Set Variable: /Model/f splementation-Get Variable: /Model/f splementation-Set Variable: /Model/S	bea&474-b045-46e1-989f-a46dae86233a/Event/e1/Variab bea&4f4-b045-46e1-989f-a46dae86233a/Event/e1/Variab 8dd805e-0c09-473b-a24b-7a3ee85fbd79/Event/e1/Variab	le/v2=Some(-148350113) le/v1=Some(? le/v1=Some(?		
2022-03-15 15:03:49,964 2022-03-15 15:03:49,964 2022-03-15 15:03:59,065 2022-03-15 15:03:59,065 2022-03-15 15:03:59,065 2022-03-15 15:03:59,066 2022-03-15 15:03:59,066 2022-03-15 15:03:59,066 2022-03-15 15:03:59,966 2022-03-15 15:03:51,09,966 2022-03-15 15:03:51,003 2022-03-15 15:03:51,003	INFO Verlabbefaltion -InterryDeall INFO Verlabbefaltion -InterryDeall	plementation-Get Veriale. //doi/1 00.demotion-Set Veriale. //doi/ 10.demotion-Set Veriale. //doi/00.demotion-Set Veriale.	be844-985-642.987-642.987-64264863330/tert/1/ariab be844-985-642.987-846.987-946.886.983.97/tert/1/ariab be844-985-642.997-446.986.933.7/tert/1/ariab be844-985-642.997-446.986.933.7/tert/1/ariab be844-985-642.997-486.985.97/tert/1/ariab be844-985-642.997-486.985.7/tert/1/ariab be844-985-642.997-485.987-885.7/tert/1/ariab be844-985-642.997-485.987-885.7/tert/1/ariab be844-985-642.997-485.486.983.7/tert/1/ariab be844-985-642.997-485.486.885.337/tert/1/ariab be844-985-642.997-485.486.885.337/tert/1/ariab be844-985-642.997-485.485.857.7/tert/1/ariab be844-985-642.997-657.497.7/tert/1/ariab	In (v2)see(-14359113) In (v2)see(-14359113) In (v2)see(-15359113) In (v2)see(-15359413) In (v2)see(-15359413)	🖪 Memory Us	age (8%)
2022-03-15 15:03:51,005 2022-03-15 15:03:51,005 2022-03-15 15:03:51,005 2022-03-15 15:03:51,616 2022-03-15 15:03:51,617 2022-03-15 15:03:51,617 2022-03-15 15:03:51,617 2022-03-15 15:03:51,617 2022-03-15 15:03:51,977 2022-03-15 15:03:51,977	LING 0. - Verlauberisticus - Linkenyöksill LING 1. Verlauberisticus - Tinkenyöksill	pjementation-Get Veriable //Mode // pjementation-Set Veriable //Mode //	bed64-b98-e61.967-e62.967-e646e6033V/tert/21/07/e18 bed64-b98-e60-360-786e567607/tert/21/07/e18 bed64-b98-e61.967-e646e66333V/tert/21/07/e18 bed64-b98-e61.967-e666e66333V/tert/21/07/e18 bed64-b98-e61.967-e666e66333V/tert/21/07/e18 bed64-b98-e61.967-e666e6633V/tert/21/07/e18 bed64-b98-e61.967-e666e6633V/tert/21/07/e18 bed64-b98-e61.967-e666e6633V/tert/21/07/e18 bed64-b98-e61.967-e666e6633V/tert/21/07/e18 bed64-b98-e61.967-e666e6633V/tert/21/07/e18 bed64-b98-e61.967-e666e6633V/tert/21/07/e18	ler(v2-Sene(214058869) ler(v2-Sene(21405869) ler(v2-Sene(21405869) ler(v2-Sene(723739034) ler(v2-Sene(723739034) ler(v3-Sene(727379034) ler(v2-Sene(723739034) ler(v2-Sene(723739034) ler(v2-Sene(723739034) ler(v2-Sene(723739034) ler(v2-Sene(723739034) ler(v2-Sene(723739034) ler(v2-Sene(723739034))	P 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	-
	Log File Configuration * Info	Ŧ
1	Enable Encryption	
2	Same folder as deployment	
3	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 💾 + 67 🌣		
Group Filter <	Group 🛧 Name	Version Deployment Type	State			
✓ Show All ex1	ex1 CSVtoRESTDeviceType	latest Local	NotDeployed			
				2		
		Enter the instance name to continue.				
		3 Instance Name				
		Ok 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	
🚺 🔽 JMX Properties	
JMX Host Name * localhost	
2 JMX Pert * 1280	
JMX Authenticate	
Cocal Only	
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 *	
Into	
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	-

• 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.

File Action View Help			
♦ ♦ 2] 🖸 📑 🛛 🖬 🔧 🗽 🗑 🍞 🚨 🎘		
Active Directory Users and Com	Name	Туре	Description
> 🦰 Saved Queries	ADadmin SmartUnifier	User	
✓ mi test.local	Administrator	User	Built-in account for administering the computer/domain
> Builtin	Allowed RODC Password Replication Group	Security Group	Members in this group can have their passwords replicated to all read-only domain controllers i
> Computers	arpad.kiss	User	
Domain Controllers	🛃 Axel F	User	
Managed Service Account	🐁 bogdan.irimie	User	
	🖁 cd_21	User	
0303	💐 Cert Publishers	Security Group	Members of this group are permitted to publish certificates to the directory
	A Cloneable Domain Controllers	Security Group	Members of this group that are domain controllers may be cloned.
	🛃 corneliu.bogdan	User	•
	DefaultAccount	User	A user account managed by the system.
	Robert Replication Group	Security Group	Members in this group cannot have their passwords replicated to any read-only domain contro
	A DnsAdmins	Security Group	DNS Administrators Group
	A DnsUpdateProxy	Security Group	DNS clients who are permitted to perform dynamic updates on behalf of some other clients (su
	A Domain Admins	Security Group	Designated administrators of the domain
	A Domain Computers	Security Group	All workstations and servers joined to the domain
	A Domain Controllers	Security Group	All domain controllers in the domain
	A Domain Guests	Security Group	All domain guests
	A Domain Users	Security Group	All domain users
	A Enterprise Admins	Security Group	Designated administrators of the enterprise
	Enterprise Key Admins	Security Group	Members of this group can perform administrative actions on key objects within the forest.
	Kenterprise Read-only Domain Controllers	Security Group	Members of this group are Read-Only Domain Controllers in the enterprise
	🔏 Giuliano De Sabata	User	
	🖁 Giulica	User	Contul meu pentru experimente
	K Group Policy Creator Owners	Security Group	Members in this group can modify group policy for the domain
	🛃 Guest	User	Built-in account for guest access to the computer/domain
	Jenkins Server	User	
	aol 😵	User	
	Key Admins	Security Group	Members of this group can perform administrative actions on key objects within the domain.
	A Protected Users	Security Group	Members of this group are afforded additional protections against authentication security threa
	RAS and IAS Servers	Security Group	Servers in this group can access remote access properties of users
	Read-only Domain Controllers	Security Group	Members of this group are Read-Only Domain Controllers in the domain
	Sascha Fischer	User	- · ·
	💐 Schema Admins	Security Group	Designated administrators of the schema
	A SmartUnfierAll	Security Group	-
	SmartUnifier Admin	User	
	SUAdmin 🕄	Security Group	
	SUWriter	Security Group	
	🔍 vah	User	

5.1.1 AD Group Mapping

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

The mapping between the AD groups and the 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
 50
           hardRefreshInterval = 5
 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.

	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					ፍ ት 🙆
	Repository Database					
5	Group	Name	Version	Туре	Description	
	demo1	RESTServer	latest	channel		
4	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				ፍ 🕁 🙆
Repository Database				
Group	Name	Version	Туре	Description
demo1	RESTServer	latest	channel	
demoscenario.csv2rest	DataModel	latest	model	
demoscenario.csv2rest	RestDataModel	latest	model	
demoscenario.csv2rest	CSVtoREST Depende	encies found	mapping	
🗋 ex	Modell Do you wan	nt to select the dependencies?	model	
🗋 ex	CSv	No	channel	
🗋 ex	Rest	latest	channel	

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

	B Backup 7	Q & Ø
	Repository Database	9
8	Vame Name	
	✓ INSTANCE_INFORMATION	
	INSTANCE_DEPLOYMENT_SETTINGS	
	Z DEPLOYMENT_ENDPOINT_SETTINGS	
	V LOGGING_SETTINGS	
	CHANNELTYPES	
	Z BASE_IMAGES	
	V USER_ACCOUNT	
	V USER-ROLE	
	CREDENTIALS	
	☑ INSTANCE_STATES	
	V NOTIFICATIONS	

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

Backup Repository?

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



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

Save As			
\leftrightarrow \rightarrow \checkmark \land NBRAA \rightarrow Desktop \rightarrow Exported Files 11			
Organize 🔻 New folder			III 👻 😮
Name versioning-local-NBRAA-01-10-16_34 tar.gz versioning-local-NBRAA-01-10-16_38 tar.gz	Date modified Type Size 1/10/2022 4:34 PM GZ File 1/10/2022 4:38 PM GZ File	: 14 KB 14 KB	
12 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 Yes 5

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 🕁 🗟 🕙
	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 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 43	a 0
Repository Database							11
Delete all artifacts before restore							
Group	Name		Version	Туре	Description		
demoscenario.csv2rest	CSVtoREST This artifact will be overwritten.		latest	mapping			
demoscenario.csv2rest	DataModel This artifact will be overwritten.		latest	model			
demoscenario.csv2rest	RestDataModel This artifact will be overwritten.	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				
nanual	31.08.2021 07:10	File folder				
repository	31.08.2021 07:10	File folder				
🚽 scala	31.08.2021 07:10	File folder				
versioning	07.09.2021 17:16	File folder				
💿 UnifierManager.bat	31.08.2021 07:10	Windows Batch File	1 KB			

5.3 Channel Types Manager

By default, the Channel Types Manager displays all *Channels* included in your current version of SMART**UNIFIER**.

Communication Channels that should be used within the configuration of a SMART**UNIFIER** Communication Instance have to exist in the Channel Types Manager. How to add new Channel Types is shown in the *section below*.

5.3.1 How to access

Follow the steps bellow to access the Channel Types Manager:

• Click on the Account icon (1) and select the Advanced UI (2).

≡	MORPH.pro		SMARTUNIFIER 😫 🕐
>			
Ŀ			

⟨·· ⟩	Integrate perfectly your	Hi,	Unifier Administrator!
f	Production-IT using	20	Account
۲		Ę,	Re-index Repository
4		Ф	Dark Theme
-0-	SMART UNIFIER 2	÷	Advanced UI
•	ADVANCED IT-INTEGRATION PLATFORM	4	Administrative +
		0	About SMARTUNIFIER
		€	Sign Out
	AMORPH.pro		
	SMARTUNIFIER		

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

	≡ Morph .pro	SMARTUNIFIER 🖯 🧿
>		
Ŀ		
A		
***	Integrate perfectly your	
‹·›	> Production-IT using	
j		
٥	SMARTUNIFIER	
â	ADVANCED IT-INTEGRATION PLATFORM	
•		
	MORPH.pro	
	SMARTUNIFIER	

• The main view of the Channel Types is visible.

>	🏠 Channel Types					٩ (٢	์) +	- G
Ŀ	Group Filter <	Group 个	Name	Version	Description			
A.	 ✓ Show All ✓ com 	com.amorphsys.unifier.channel	Sql Database	1.0.0		Ċ	/	Î
***	 amorphsys unifier channel 	com.amorphsys.unifier.channel	InfluxDB	1.0.0		Ċ	1	Î.
~~~	CTEATING P	com.amorphsys.unifier.channel	In Memory	1.0.0		Ċ	1	Î
_		com.amorphsys.unifier.channel	Iso-On-Tcp Client	1.0.0		Ċ	1	Î.
٥I		com.amorphsys.unifier.channel	Modbus Tcp Client	1.0.0		Ċ	1	Î
۲		com.amorphsys.unifier.channel	OPC UA Server	1.0.0		Ċ	1	Î
<b>A</b>		com.amorphsys.unifier.channel	OPC UA Client	1.0.0		Ċ	1	Î
		com.amorphsys.unifier.channel	Rest Server	1.0.0		Ċ	1	Î.
-		com.amorphsys.unifier.channel	Rest Client	1.0.0		Ċ	1	Î
		com.amorphsys.unifier.channel	SecsGem Client	1.0.0		Ċ	1	Î
		com.amorphsys.unifier.channel	Websocket client (JSON)	1.0.0		Ċ	1	Î.
		com.amorphsys.unifier.channel	Websocket client (XML)	1.0.0		Ċ	1	Î
		com.amorphsys.unifier.channel	Websocket client (CSV)	1.0.0		Ċ	1	Î
		com.amorphsys.unifier.channel	SFTP file writer (JSON)	1.0.0		Ċ	/	i.
		com.amorphsys.unifier.channel	SFTP file writer (XML)	1.0.0		Ċ	F	Î

**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.

≡ <b>⊿MORPH</b> .β	oro
SMARTUnifier Configuration	<
Information Models	Ŀ.
Channel Types	<b>A</b>
Communication Channels	<b>##</b> #
Mappings	<b>⟨</b> ·· <b>⟩</b>
Device Types	
Instances	٥
Deployments	۵
Deployment Endpoints	•
User Management	**

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

>	🏠 Channel Types					۹ (	· <b>1</b>	+	c,
Ŀ.	Group Filter <	Group 🛧	Name	Version	Description			3	Â
<b>A</b>	<ul> <li>Show All</li> <li>✓ com</li> </ul>	com.amorphsys.unifier.channel	SqlDatabase	1.0.0		Ċ	1	Î	
	<ul> <li>✓ amorphsys</li> <li>✓ unifier</li> <li>channel</li> </ul>	com.amorphsys.unifier.channel	InfluxDB	1.0.0		Ċ	1	Î	

- 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.

>	Add Communication Channel Type	6 E 16 8
Ŀ	Broun *	
•	4 com.amorphsys.unifier.channel	
***	Name " MQTT_JSON	
<b>&lt;</b> ·· <b>&gt;</b>	Version * 1.0.0	
Ð		
۲	Description	
Å	DEVICE	
•	5 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.

🎓 Docker Java Image Manager				
> ~	group 🛧	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).

=	≡ <b>/MORPH.</b> pro					
>	🗯 Docker Java Image Manager		Q + 5			
Ŀ	> group ↑ ⊄	name				
***	OUP FILTE					
<b>⟨</b> ⟩	9 2 2					

- 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 🖬	ıc ×
Ŀ:		Group *		
<b>##</b> #	2	> demo		
<b>‹··</b> >		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).

=	≡ <b>/MORPH.</b> pro				8 0
>	🗯 D	ocker Java Image Manager		Q -	+ <i>•</i> ,
Ŀ	х Ж	group 个	name		
***	P FILTE	demo	Image1		1
	N				

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).

=	≡ <b>/MORPH</b> .pro sm				
>	<b>*</b> D	ocker Java Image Manager		Q + 5	
Ŀ	х Х	group 🛧	name	0	
***	° FILTE	demo	Image1	2 II.	
	n n				

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.

>	Deployment Endpoints			Q ( ¹ ) + 43
Ŀ	> Group ↑	Name	Туре	
•				
***	GROL			
<b>«··</b> >				
đ				
۲				
Å				
•				

**Note:** The Deployment Endpoints can only be accessed by user accounts with an administrator role assigned.

## 5.5.3 Deployment Endpoints Types

#### Docker

SMART**UNIFIER** supports the Deployment of Instances using Docker Containers. Before creating a new Deployment for an Instance using Docker. install Docker on your device and open up the Docker Remote API Interface. If you want to learn more about Docker and how to install it, visit the Docker Website. When your Docker Daemon is up and running you must provide a Docker endpoint.

• Navigate to the SMART**UNIFIER** Deployment Endpoints perspective (1).

	≡ <b>∕MORPH.</b> β	oro
	SMARTUnifier Configuration	<
	Information Models	ţ.
	Channel Types	<b>.</b>
	Communication Channels	<b>*</b> **
	Mappings	<b>⟨··⟩</b>
	Device Types	þ
	Instances	۵
	Deployments	â
1	Deployment Endpoints	
	User Management	**

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

۳	Deployment Endpoints			Q ( ¹ ) + 5
GROUP FILTER ~	Group 🛧	Name	Туре	Docker
				AWS

- 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
🕀 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).

	≡ <b>/MORPH</b> .β	oro
	SMARTUnifier Configuration	<
	Information Models	ţ.
	Channel Types	<b>*</b>
	Communication Channels	<b>*</b> **
	Mappings	<b>⟨</b> ⟩
	Device Types	þ
	Instances	۲
	Deployments	٨
1	Deployment Endpoints	•
	User Management	**

- Click the "Add" button (2).
- Select AWS (3).
|          |                            |      |      | 2      |
|----------|----------------------------|------|------|--------|
| •        | Deployment Endpoints     Q |      |      |        |
| ><br>~   | Group 🛧                    | Name | Туре | Docker |
| IP FILTE |                            |      | 3    | AWS    |
| GROU     |                            |      |      |        |

- 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 an credentials profile that allows SMARTUNIFIER to connect to your AWS account.
- Save the new Endpoint by clicking the "Safe" button (5):

		5
Add AWS Endpoint		() 🖬 🖒 ×
4	Broup * demo Name * AWSAccount1 Account ID * eu-central-1 profile * profile *	

### SSH

SMARTUNIFIER supports the Deployment of Instances using SSH protocol.

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

	≡ <b>∕MORPH.</b> β	oro
	SMARTUnifier Configuration	<
	Information Models	ţ.
	Channel Types	<b>.</b>
	Communication Channels	<b>*</b> **
	Mappings	<b>⟨··⟩</b>
	Device Types	þ
	Instances	۵
	Deployments	â
1	Deployment Endpoints	-
	User Management	**

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

•	Deployment Endpoints				2 Q 💾 + 6	,
× œ	Group 🕈	Name	Туре		Docker	
FILTE					AWS	
GROUP				3	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	
	Port * 22	
	Username * ubuntu	
4	Password	
	Private Key	
	Connect Timeout * 5000	
	Java Home * usr/lib/ym/java-11-openjdk-amd64/	
	Deployment Folder * /opt/amorph/smartunifier/deploy/	
	Soft Refresh Interval * 5000	
	Hard Refresh Interval * 5	

## 5.6 Credential Management

Within the Credential Manager the user can store and manage the credentials needed for the Communication Channel configuration (e.g., password for certificates, username and password for SQL Server).

### 5.6.1 How to access

Follow the steps bellow to access the Credential Management:

• Click on the Account icon (1), go to Administrative section (2) and select the Credential Management option (3).



• The Credential Management is visible.



**Note:** The Credential Management can only be accessed by user accounts with an administrator role assigned.

### 5.6.2 Add Credentials

Follow the steps described below to add credentials:

- Click on the Add button (1).
- Select an option (2) Password or Username and Password.

🍖 Credential Management			ላ 🖞 + 😏
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 4 5	Name * TestSQLCredentials Description Credentials to access the Test SQL Server Username * testsqluser Password * typethepassword	6

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

🗞 Credential Management				
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 🖞 + ብታ
Name	Description	Туре	
TestSQLCredentials	Credentials to access the Test SQL Server	UserNameAndPassword	/ 1
			0

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	<b>5</b> :	×
Name *         TestSQLCredentials         Description         Credentials to access the Test SQL Server         Username *         testsqluser         Password *         typethepassword	3	

## 5.6.4 Delete Credentials

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

Credential Management			Q ( ¹ ) + 4
Name	Description	Туре	
TestSQLCredentials	Credentials to access the Test SQL Server	UserNameAndPassword	/ 1
			4

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		↔ ✓ ×
M db     Q. ↓       C DatabaseSelect [Command_DatabaseSelect]       ↑ Parameters [RequestType]       ① OrderN [String]       ↑ Reply [ReplyType]       ✔ Customer [String]	SQL Database Database connection Type * SOLServer Reconnect interval * 10 JBBC UI * jdbc:sqlserver.//192.168.0.111.1433.databaseNamerunfier Database credentials * 11 Database credentials *	• •

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

Database connection	
Type *	
SQLServer	*
Reconnect interval *	
10	
JDBC Url *	
dbc:sqlserver://192.168.0.111:1433;databaseName=unifier	
Datahaoo orodontialo t	
atabaee eredentiale t	

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

atabase connection			
pe *			
QLServer			*
econnect interval *			
0			
DBC Url *			
bc:sqlserver://192.168.0.111:1433;databaseName=unifie	r		
atabase credentials *			
sername and password credentials reference		 	•
Username and password credentials reference	e		

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

atabase connection			
pe *			
QLServer			
connect interval *			
)			
DBC Url *			
ho:ogloon/or://102.169.0.111:1422:dot	-h		
uc.sqiserver.//192.108.0.111.1433,086	abaseName=unifier		
uchana aradantiala t	abasename=unifier		
nocsquserver.//192.108.0.111.1433,dat nabase credentials * sername and password credenti	abaseName=unifier		
sername and password credenti	abasename=unmer		
sername and password credent	adaseName=uniner		
stabase credentials * sername and password credent	adasewame=uniner ials reference entials reference		
atabase credentials * sername and password credent Username and password crede	abaseName=Uniner		
sername and password credent Username and password credent + Add credentials	abaseName=Uniner		

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

rpe *			
QLServer			
econnect interval *			
0			
DBC Url *			
lbc:sqlserver://192.168.0.111:14	433:databaseName=unifier		
atabase credentials *			
sername and password cr	edentials reference		
Username and password	credentials reference		
Username and password	credentials reference		
Username and password + Add credentials 5	credentials 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).

rver	
rver	
ict interval *	
*	
lserver://192.168.0.111:1433;databaseName=unifier	
e credentials *	
ame and password 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.pro								SMARTUNIFIER 🖯 🕐
>	🏞 User Managen	nent							Q + G
Ŀ.	User ID 个	Email	First Name	Last Name	Language	Role	Status	Created	
•	admin		Unifier	Administrator	en	Administrator	Active	2021-10-26 00:00:00.000	/ 1
<b>₩</b>									

**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 😫	0
>								
F								
•								
***		Productio	perfectly your			Hi,	Unifier Administrator!	
<b>&lt;</b> ··>		i iouuotio				20	Account	
i						e:	Re-index Repository	
۲			SMART <b>UNIFIER</b>			Ð	Dark Theme	
A						Ξ	Simple UI	
-				22	User Management 🚺	4	Administrative +	
•				6	Credential Management	0	About SMARTUNIFIER	
			<b>DRPH</b> .pro	*	Docker Java Image Manager	€	Sign Out	
				荘	Logging Configurations			
		SIMA		×A	Translations Manager			
				8	Backup			
				Ð	Restore			

• Click the "Add User" button (2).

↔ User Management								2 +	c,
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	ī

- 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			<b>4</b> ■ ×
3	User ID - JohnDoe2		
	Email First Name * John		
	Last Name * Doe Language * English		
	Role: Writer Administrator Reader		
	viruei Status Active Active Inactive	₩ 	
	Credentials Plasmad *	× ę. 0/10	
	contains at least one low     contains at least one low     contains at least one low     contains at least one spe     contains at least of the least 4 charace     contains at least 4 charace	er character er character character cial character ters	
	Confirm password *	<b>Q</b>	

### 5.7.4 Edit a user

This procedure describes how to edit an existing user account.

• Select the SMARTUNIFIER User Management perspective (1).

≡	AMORPH.pro					SMARTUNIFIER 8
>						
F						
•						
***	Produ	ction-IT using			Hi,	Unifier Administrator!
<b>&lt;</b> ·· <b>&gt;</b>	1000				20	Account
j					e:	Re-index Repository
۲		SMART <b>UNIFIER</b>			Ð	Dark Theme
â		ADVANCED IT-INTEGRATION PLATFORM	1			Simple UI
_			22	User Management 1	2	Administrative
			6	Credential Management	0	About SMARTUNIFIER
		<b>JORPH</b> .pro	*	Docker Java Image Manager	€	Sign Out
	CM		∃≓	Logging Configurations		
	51*1	ARIUNIFIER	ネ	Translations Manager		
			8	Backup		
			Ð	Restore		

• Click the "Edit" button (2).

24 User Management								Q + 5
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	Z 1

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

Unit () *         Jubboez         Email         For tame *         John         Last tame *         Doe         English	C Edit User: John Doe		<b>4</b> ■ ×
Roler Rader Administrator Reader Reader Witer StatussActive Active Active Inactive Change Password Change Password	8	tore ro - Johnboe2 Email Finst hare + John Last Name + Doe English Role: Reader Administrator Reader Writer Status Active Status Active Inactive Mactive 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).

2 User Management Q +								
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	× =

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

2 User Management	t							Q +	- cj
User ID 🕇	Email	First Name	Last Name	Language	Role	Status	Created		
JohnDoe2		John	Doe	en	Reader	Active	2021-03-26 00:00:00.000	1	i i
admin		Unifier	Administrator	en	Administrator	Active	2021-03-26 00:00:00.000	1	i i
				Delete user Are you sure you war	nt to delete this user JohnDoe2? Cancel Delete	3			

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

半 User Manage	ment							Q + G
User ID 🕇	Email	First Name	Last Name	Language	Role	Status	Created	
admin		Unifier	Administrator	en	Administrator	Active	2021-03-26 00:00:00.000	× =

## 5.8 Logging Configurations

Log files in SMART**UNIFIER** are generated using the log4j framework. The Logging Configuration features enables to create new *log Levels* configurations that can be selected when deploying a Communication Instance.

### 5.8.1 How to access

Follow the steps below to access the feature:

• Click on the Account icon (1), go to the Administrative option (2) and select the Logging Configurations perspective (3).



- Logging Configurations main view is visible, as seen below.
- There are four predefined log4j configurations that can be used as template when creating a new log level.

Note: The predefined log4j configurations can not be edited or deleted.

≡	MORPH.pro			SMART <b>UNIFIER (9</b> (2)
>	🛔 Logging Configu	rations		ረ 🖞 + 🕁
Ŀ	Group Filter <	Group 🛧	Name	
A	✓ Show All default	default	Trace	白 💿 🔋
***		default	Debug	Ů <b>◎</b> ≣
6.5		default	Warning	Ů <b>◎</b> 🔋
		default	Info	<u>ن</u> الا
۲				
<b>Å</b>				
•				

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 Configu	ations		Q 🖞 + 5
Group Filter <	Group 🛧	Name	2
✓ Show All default	default	Trace	ڭ 💿 🗎
	default	Debug	白 💿 🧻
	default	Warning	<u>ن</u> 💿 آ
	default	Info	白 💿 🧻

- 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 Configur	ations		Q ( ¹ ) + 47
Group Filter <	Group 🛧	Name	2
<ul> <li>Show All custom</li> </ul>	custom	Test	ڭ 🗡 🖡
default	default	Trace	Ċ 💿 🗊
	default	Debug	Ċ 💿 📋
	default	Warning	ථ 💿 📋
	default	Info	<u>ث</u> •

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

Edit Logging Configuration: custom	:Test			() 🗟 🖒 ×
	Name * Test]	 	 -	3

### 5.8.4 Delete a logging file

Follow the steps below to delete a log4j configuration file:

• Select the Logging Configurations perspective (1).



• Click on the "Delete" button (2).

🔒 Logging Configu	rations		Q [ ¹ ] + <i>G</i>
Group Filter <	Group 🛧	Name	2
✓ Show All custom	custom	Test	Ċ / ī
default	default	Trace	Ċ 💿 📋
	default	Debug	Ů <b>⊙</b> Ĩ
	default	Warning	Ċ 💿 📋
	default	info	Ľ 📀 🗎

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

#### Delete Logging Configuration



## 5.9 Extensions

### 5.9.1 AWS SiteWise Model Export Plugin

This extension allows you to export an SMARTUNIFIER Information Model to AWS IoT SiteWise.

#### How to access

To access the AWS SiteWise extension, click on the Account icon (1), go to the Administrative option (2) and select the Extensions (3).

	MORPH.pro		SMA	RT <b>UNIFIER </b>	8
>					1
Ŀ					
***					
<b>«··</b> >	Integrate perfectly your		Hi,	Unifier Administrator!	
đ	Production- IT using		20	Account	
۲			Ę;	Re-index Repository	1
A			Ð	Dark Theme	
-	SMARTUNIFIER		÷	Advanced UI	
	DECENTRALIZED INDUSTRIAL CONNECTIVITY PLATFORM	🚓 User Management	2	Administrative	· 2
		Extentions	0	About SMARTUNIFI	ER
		🗞 Credential Management	€	Sign Out	
	<b>MORPH.</b> pro	Docker Java Image Manager			
	SMARTUNIEFR	코는 Logging Configurations			
		$\dot{\chi}_{A}$ Translations Manager			
		Backup			
		€ Restore			

### Then select the **configuration** button of the (4)

stentions					Q + 5-
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

#### Prerequisite

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.

## Configuration

Follow the steps described below to export a the SMART**UNIFIER** Information Model:

- Configuration of the extension (1):
  - Select the region of the AWS Iot SiteWise service you are using
  - Enter the access key id and the secret access key id
  - Select the Information Model you want to export
- Click on the **Run** button to execute the export (2)

🇯 AWS SiteWise Model Export Plugin		2 ► 8
D	Region * eu-central-1 Access key id * AKIARU7XHLCLZDNSZZVT Secret access key id * Model * su.demo.dashboard.Analytics.latest	

### 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 SMARTUNIFIER 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 SMARTUNIFIER 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.