MARTUNIFIER

SMARTUNIFIER Demo Guide

Release 1.4.0

Amorph Systems GmbH

Apr 25, 2022

TABLE OF CONTENTS

1	About SMARTUNIFIER Demonstrator 1.1 1.1 What is SMARTUNIFIER Demonstrator 1.1.1 Components 1.1.2 Data Flow Diagram 1.1.3 Demonstrator Artefacts Structure 1.1.4 Simulation Process Flow Diagram 1.2 What is SMARTUNIFIER 1.3 What does SMARTUNIFIER do
2	Installation Overview 2.1 System Requirements 2.2 Windows 2.1 Step 1 Install the SMAPTUNIEUP and the Decker Components
	2.2.1 Step 1 - Install the SMART UNIFIER and the Docker Components. 14 2.2.2 Uninstalling 14 2.2.3 Step 2 - Running the SMARTUNIFIER 14 2.2.4 Running SMARTUNIFIER as an Application 14
	2.2.5 Running SMARTUNIFIER as a Service 11 2.3 Linux 12 2.3.1 Step 1 - Install SMARTUNIFIER and Docker Components: 12 2.3.2 Step 2 - Running SMARTUNIFIER: 18
3	Instance Deployment 19 3.1 Run the Instance 19
4	Visualization Tools 24 4.1 Node-Red 24 4.1.1 Access 24 4.2 Grafana 24
5	Start the Simulation Scenario325.1Input Data335.2Start the Simulation335.2.1Release Order345.2.2Start Order345.2.3Start Failure365.3Instance Setup38

5.3.1	Information Models	38
5.3.2	Mappings	45

Integrate perfectly your Production-IT using



MORPH.pro SMARTUNIFIER

CHAPTER

ABOUT SMARTUNIFIER DEMONSTRATOR

- Learn about the *Demonstrator*.
- Learn about the *SMARTUNIFIER* connectivity platform.

1.1 What is SMARTUNIFIER Demonstrator

SMART**UNIFIER** Demonstrator is a package that allows users to simulate the connection between a MES system and a production equipment. The package contains all the necessary tools to run a complete communication scenario out of the box.

1.1.1 Components

The Demonstrator package contains the following components:

- **SMARTUNIFIER Manager** a modern web application to create SMART**UNIFIER** Instances that enable the communication between the MES system and the Equipment simulator.
- **Node-Red** a Docker image containing the NodeRed application preconfigured to act as a MES Simulator.
- **Equipment Simulator** a Docker image that contains a custom application that simulates a real PLC from a production equipment.
- **MQTT Broker** a Docker image containing the Moquette application acting as a message bus server.
- **Influx Database** a Docker image containing the Influx Database where the data coming in from the Equipment Simulator will be stored and used to build visualizations.
- **Grafana** a Docker image containing a preconfigured Grafana application that has a built-in dashboard displaying the key parameters sent by the Equipment Simulator.

1.1.2 Data Flow Diagram

SMARTUNIFIER Projects



1.1.3 Demonstrator Artefacts Structure

The following table shows the SMART**UNIFIER** artefacts that are used to create this demo:

Group	Туре	Name	Description
su.demo.dasl	h loaíod mation	Analytics	Stores data from the PLC on InfluxDb
	Models	MESSimula-	Represents the MES Simulator structure
		tor	
		SiemensS7PL	CRepresents the PLC blocks structure
	Communica-	SiemensS7PL	CRepresent the PLC communication protocol
	tion Channels	PLCToIn-	Is used to transmit data from PLC to InfluxDb
		fluxDb	
		MESSimula-	Is used to transmit data from MES Simulator to
		tor	MQTT broker
	Mappings	PLCToIn-	Defines when and how to extract data from the
		fluxDb	PLC and store it on the InfluxDb
		PLCToMES-	Defines data exchange between the PLC and the
		Simulator	MES Simulator
	Device Types	SUDevice-	Represents the template for the SMARTUNIFIER
		Туре	Instance
	Instances	SUInstance	Represents the configuration for the runnable
			application

1.1.4 Simulation Process Flow Diagram

Release/Start - Order Request



Equipment Response - Notify Product Produced

						SMARTUnifier					
MES Simulator	MQTT	Broker	MES Simulator Chanr	nel JSON - via MQTT	Mapping]	Influx Datab	ase Channel	Equipment Channel OPC-UA	Influx Database	Equipment Simulator
Send Message - Product Produced (Modul	le A/B/C)	Execut	es Rule - Product Produc	ed (Module A/B/C)	-Tri Exe	iggers Rule - Product Produced (Mr cutes Rule - Product Produced (Mr	odule A/B/C)	Send Message	 Notify - Prod. Product Produced (Module A/B/ 	at Produced (Module A	VB/C)
MES Simulator	MQTT	Broker	MES Simulator Chanr	nel JSON - via MQTT	Mapping)	Influx Datab	ase Channel	Equipment Channel OPC-UA	Influx Database	Equipment Simulator



Introduced Failure

1.2 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.



1.3 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 SMART**UNIFIER** it is possible to communicate unidirectional or bidirectional to each Communication Partner. I.e., messages and events can be sent and received at the same time.
- SMARTUNIFIER is able to 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, Webservices 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, SMART**UNIFIER** enables the same view to data regardless of the protocol or format being used to physically connect an equipment, device or IT system.
- A big advantage of SMARTUNIFIER is, that in many cases there is no need for coding when providing connectivity between different Communication Partners. SMARTUNIFIER Mappings enable users to assign data sources to data targets via drag and drop.

CHAPTER

INSTALLATION OVERVIEW

Installing SMART**UNIFIER** on a host is the first step in realizing your connectivity scenario.

- 1. See the system requirements prior to the installation.
- 2. Amorph Systems supports using SMARTUNIFIER on the following operating systems:
 - Windows
 - Linux

2.1 System Requirements

Minimum requirements for running the SMARTUNIFIER Manager

- Computer and Processor: 1 GHz or faster, x86-bit- or x64-bit-processor.
- Memory: 512 MB RAM.
- Hard Disk / SSD: 1 GB free space.
- Display PC (Engineering, Dashboard): 1280 x 1024 Resolution.
- Mobile Devices (Dashboard): Apple iPhone 6 or higher, Android.
- **Operating System**: Windows 10, Windows 8, Windows 7, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Linux, MacOS For an optimal user experience always use the newest version of the operating system.
- Browser: Latest version of Chrome, Microsoft Edge/Internet Explorer, Firefox.
- Other: Latest version of Docker Daemon (including Docker-compose). For more details follow the on-screen instructions from https://docs.docker.com/compose/install/.

2.2 Windows

2.2.1 Step 1 - Install the SMARTUNIFIER and the Docker Components:

- Move the installation package to a suitable location.
- Open "Docker Desktop" application.
- Run the **.exe** file to start the setup.
- Click on the "Browse" button (1) to change the location of the installation.
- Select the "Next" button (2) to continue.

😽 Setup - SMARTUNIFIER version 1.2.0	_		×
Select Destination Location Where should SMARTUNIFIER be installed?			
Setup will install SMARTUNIFIER into the following folder.			
To continue, click Next. If you would like to select a different folder, o	dick E	Browse.	
C:\SMARTUNIFIER		Browse	1
At least 474.7 MB of free disk space is required.			
2 Nex	t	Can	cel

• Enter the local IP address (3). Select the "Next" button (4) to continue.



• Click on the "Install" button (5) to start the installation.



• Check the box (6) to create SMARTUNIFIER Service (optional).



- Select the "Finish" button (7) to finallize installing the SMARTUNIFIER. The console will open and the extraction of the Docker components starts.
- After the console is closed, the Docker components are installed in a few seconds:
 - moquette
 - nodered
 - grafana
 - influxdb
 - simulator



If a Docker container is deleted, it can be redeployed. From the **scripts** folder (install_location/SMARTUNIFIER/scripts) open the console (CMD) and execute:

• to deploy InfluxDb container:

deploy.bat influxdb

• to deploy Grafana container:

deploy.bat grafana

• to deploy Node-Red container:

deploy.bat nodered

• to deploy MQTT broker container:

deploy.bat moquette

• to deploy Equipment Simulator container:

deploy.bat simulator

• to deploy all containers:

deploy.bat

2.2.2 Uninstalling

To undeploy a docker container, from the **scripts** folder (install_location/SMART**UNIFIER**/scripts) open the console(CMD) and execute:

• to remove InfluxDb container:

cleanup.bat influxdb

• to remove Grafana container:

cleanup.bat grafana

• to remove Node-Red container:

cleanup.bat nodered

• to remove MQTT broker container:

cleanup.bat moquette

• to remove Equipment Simulator container:

cleanup.bat simulator

• to remove all containers:

cleanup.bat

Follow the steps below to uninstall all the Demonstrator components (folders, files, Docker container):

- Make sure the Demonstrator Instance is NOT running.
- Make sure the "Docker Desktop" application IS running.
- There are two options to uninstall the Demonstrator:
- 1. Using Windows System settings section:
- Go to "Add or remove programs" and uninstall SMARTUNIFIER application.
- 2. Run the **unins000.exe** file from the "SMART**UNIFIER**" folder.
- Select the "Yes" button (1) to confirm.



• Select the "OK" button (2) to finish.



All the Demonstrator components are removed.

2.2.3 Step 2 - Running the SMARTUNIFIER

SMART**UNIFIER** can be started as an application or as a service.

2.2.4 Running SMARTUNIFIER as an Application

• If SMARTUNIFIER was not installed as a service, execute the UnifierManager.bat script located in the installation folder. Afterwards the SMARTUNIFIER Manager Console appears on the screen.

💽 UnifierManager - C:\SmartUnifier\\bin\adaptermanagerweb.bat — 🗌 🛛	<
2021-04-06 12:17:50,677 - [debug] - o.h.v.i.e.r.TraversableResolvers - Cannot find javax.persistence.Persistence on cla	s ^
spath. Assuming non JPA 2 environment. All properties will per default be traversable.	
2021-04-06 12:17:50,677 - [debug] - o.h.v.i.e.ConfigurationImpl - Setting custom ConstraintValidatorFactory of type pla	У
Judia.valuation.perauleconstraintevaluation actory	
zozi-ood uz.ir.jojoro - (ueug) - olinivi.e.com zgu altonimpi - Setting tustom messageinterpolator or type orginizern ta validator messagainterpolation DaramaterMaccaraInterpolator	
(e,valuato) increase anter polation ranameter message incer polaton 2021.04.06 12:17:06 672 - [dabua] - o b v i v c ValidationYmlDancan - Trving to load META-TNE/validation vml for YML ba	-
A Validator configuration	2
eu variadon com gu alion. 2021 A4 65 12:17:56 679 [dobug] o b v i v c PocounceloadonHelpon Trving to load META INE/validation vml via ICCL	
201-04-06 12:17:50,600 - [debug] - 0.1.V.I.X.C.ResourceLoaderHeiper - Trying to load META-INF/Validation.Xml via Heiban	n
zoriot-ot-ou initiationi, sino laborationi via niceroadenerper - niving to road Merk-Im/Varidation. Ami via nicer	
act variants is a structure of the state of	n
based configuration only	
2021.04.10.17.56 (5%) - [dabug] - o b v i e ValidatorEactoryImn] - HV000324. Using ong bibernate validator messagaint	4
zozi-oo-o iz.ir.sojod [leoug] - o.n.v.z.e.validatorEstonv.crond macsage internalator	-
791201.06 12:17:50 684 - [debug] - o b v i e ValidatorEstonyImm] - HV000234: Using org hibernate validator internal e	n
gine resolver TraverseAllTraverseAllResolver as ValidatoFactory.sconed traversable resolver	
2021-04-06 12:17:50-684 - [debug] - o.h.v.j.e.ValidatorFactoryImpl - HV00234: Using org.hibernate.validator.internal.u	t
il ExecutableParameterNameProvider as ValidatorEactory-scoped parameter name provider	
2021-04-06 12:17:50.684 - [debug] - o.b.v.i.e.ValidatorEactoryTmp] - HV000234: Using org.bibernate.validator.internal.e	n
gine.DefaultClockProvider as ValidatorFactory-scoped clock provider.	
2021-04-06 12:17:50.684 - [debug] - o.h.v.i.e.ValidatorFactorvImpl - HV000234: Using org.hibernate.validator.internal.e	n
gine.scripting.DefaultScriptEvaluatorFactory as ValidatorFactory.scoped script evaluator factory.	
2021-04-06 12:17:50.691 - [info] - play.api.Play - Application started (Prod) (no global state)	
2021-04-06 12:17:50.759 - [debug] - c.t.s.a.AkkaSSLConfig - Initializing AkkaSSLConfig extension	
2021-04-06 12:17:50,760 - [debug] - c.t.s.a.AkkaSSLConfig - buildHostnameVerifier: created hostname verifier: com.types	а
fe.sslconfig.ssl.DefaultHostnameVerifier@1c8e2850	
2021-04-06 12:17:51,284 - [debug] - a.i.TcpListener - Successfully bound to /0:0:0:0:0:0:0:0:0 <u>:0:0:0000</u>	
2021-04-06 12:17:51,297 - [info] - p.c.s.AkkaHttpServer - Listening for HTTP on /0:0:0:0:0:0:0:0:0:09000	

- After successfully starting up the SMARTUNIFIER Manager, it can be accessed by opening an Internet Browser (e.g., Chrome or Firefox) and navigating to http://localhost:9000. Use the administrator credentials to login:
- Username: admin
- Password: admin

Note: The console is for information purposes only. It can be moved to any suitable location on your screen or it can be hidden. Nevertheless, do not close it, because the related processes will also be terminated.

2.2.5 Running SMARTUNIFIER as a Service

- If SMARTUNIFIER was installed as a service, the service is already running.
- To check open "Services" in Windows (press the "Windows" button and type "services") and search "SMARTUNIFIER" from the list (1).

Services					-	- 🗆	\times
File Action V	fiew Help						
(+ +) 📰 🗄	3 🖸 📑 🛛 📰 🕨 🔳 🕪						
🔍 Services (Loca	Services (Local)	-					
	SMARTUNIFIER	Name	Description	Status	Startup Type	Log On As	
	Stop the service Pause the service <u>Restart</u> the service	System Interface Foundation Service System Guard Runtime Monitor Broker System Events Broker System Event Notification Service	The Lenovo System Interface Foundation Service provides interfaces for key features such as: system power Monitors and attests to the integrity of the Windows platform. Coordinates execution of background work for WinRT application. If this service is stopped or disabled, the Monitors system events and notifies subscribers to COM+ Event System of these events.	Running Running Running Running	Automatic Automatic (Automatic (T Automatic	Local Syster Local Syster Local Syster Local Syster	n n n
	Description: SMARTUNIFIER Manager Service	🤹 SysMain 🖏 Sync Host_3693ae 🥨 Storage Tiers Management	Maintains and improves system performance over time. This service synchronizes mail, contacts, calendar and various other user data. Mail and other applications Optimizes the placement of data in storage tiers on all tiered storage spaces in the system.	Running Running	Automatic Automatic (Manual	Local Syster Local Syster Local Syster	n n m
		Storage Service Still Image Acquisition Events Steam Client Service	Provides enabling services for storage settings and external storage expansion Launches applications associated with still image acquisition events. Steam Client Service monitors and updates Steam content	Running	Automatic (Manual Manual	Local Syster Local Syster Local Syster	n n m
		State Repository Service SSDP Discovery Spot Verifier Spot Verifier Spatial Data Service Schware Data tion	Provide required infrastructure support for the application model. Discovers networked devices and services that use the SSDP discovery protocol, such as UPnP devices. Also Verifies potential file system corruptions. This service is used for Spatial Perception scenarios	Running Running	Manual Manual Manual (Trig Manual	Local Syster Local Servic Local Syster Local Servic	n e n
		Solver Froection Solve	Receives the owninesa, meaning in an environment of digital literbes for Windows and Windows applicatu Receives the messages generated by local or remote Simple Network Management Protocol (SNMP) agent SMARTUNIFIER Manager Service Allows the system to be configured to lock the user desktop upon smart card removal.	Running	Automatic (Manual Automatic Manual	Local Servic Local System Local System	n m
		Smart Card Device Endmeration Service	Manages access to smart cards read by this computer. If this service is stopped, this computer will be unabl	Running	Automatic (T	Local Servic	e

- Open an Internet Browser (e.g., Chrome or Firefox) and navigating to http://localhost:9000. Use the administrator credentials to login:
- Username: admin
- Password: admin

2.3 Linux

2.3.1 Step 1 - Install SMARTUNIFIER and Docker Components:

- Move the installation package to a suitable location. Make sure the path to the directory does not include any white spaces!
- Extract the **.tar.gz**-archive.

```
tar -xvzf SMARTUNIFIER-Manager-linux-x64.tar.gz
```

• To change the deploy file properties and to convert it's format to Unix, open the terminal from the **scripts** folder (install_location/SMARTUNIFIER/scripts) and execute the following commands:

chmod 775 deploy.sh

dos2unix deploy.sh

• To deploy the Docker components execute the following commands (terminal opened from the **scripts** folder):

sh deploy.sh

- During the Docker components deploy, the input of the local IP address is required. To get the local IP run the command ifconfig.
- After entering the local IP address, the Docker components are deployed in a few seconds:

- moquette
- nodered
- grafana
- influxdb
- simulator

2.3.2 Step 2 - Running SMARTUNIFIER:

• Start the SMARTUNIFIERManager by executing in a terminal the following commands:

chmod +x UnifierManager.sh

./UnifierManager.sh

- After successfully starting the SMARTUNIFIER Manager, it can be accessed by opening an Internet Browser (e.g., Chrome or Firefox) and navigating to http://localhost:9000. Use the administrator credentials to login:
- Username: admin
- Password: admin

Note: The console is for information purposes only. It can be moved to any suitable location on your screen or it can be hidden. Nevertheless, do not close it, because the related processes will also be terminated.

CHAPTER

THREE

INSTANCE DEPLOYMENT

In order to run the Demonstrator you need to *Deploy and Start* the instance.

3.1 Run the Instance

The communication between the MES system (Node-Red) and the Equipment Simulator is facilitated by the SMART**UNIFIER** Instance.

First add the Instance to a local deployment:

- Open "Deployments" section (1).
- Click on the "Add" button (2).
- Select the "Local" option (3).



- Select the "Instance" from the dropdown (4).
- Select "Info" from the "Log File Configuration" dropdown (5).
- Check "Enable Encryption" (6) to have all credentials encrypted in the configuration files.
- Enable "Protected" (7) in order to secure the deployment A confirmation will be required for further changes on the deployment (e.g., deploy, undeploy, start, stop).

Instance *			
su.demo.dashboard:SUInstand	ce:latest		4
Log File Configuration *			
Info			5
Enable Encryption			
_			
Same folder as deployment			
Custom Path:			

Then run the Instance:

- Open the "Deployments" section (1).
- Click the "Deploy" button (2).

>	â	Deployment						Q	Ľ	+	Ð	\$
Ŀ	> ~	Group 🛧	Name	Version	Deployment Type	State						
***	P FILTEF	su.demo.dashboard	SUInstance	1.0.0	Local	NotDeployed		Ŷ		Ξ۲	/	Î
	GROUI							2				
<>								-				
B												
۵												
٨												
		-										

• Enter the Instance name (3) to confirm the action and click the "Ok" button (4).

Protected Instance

Enter the instance name to continue.



• Click the "Start" button (5) to run the Instance.

â,	Deployment							Q	Ľ	+	£	\$
> ~	Group 个	Name	Version	Deployment Type	State							
P FILTEF	su.demo.dashboard	SUInstance	1.0.0	Local	Stopped		1	$\widehat{\tau}_{\star}$	86	Ξ×	/	Î
INOF						5						

• Enter the Instance name (6) to confirm the action and click the "Ok" button (7).

Protected Instance Enter the instance name to continue. Instance Name Sulnstance



• The Instance is running (8).

â	Deployment						Q	Ľ	+	£	\$
> ~	Group 个	Name	Version	Deployment Type	State						
P FILTEF	su.demo.dashboard	SUInstance	1.0.0	Local	Started		$\widehat{\gamma}_{\rm x}$	88	Ξ×	/	Î
GROU					8						

CHAPTER

FOUR

VISUALIZATION TOOLS

The Demonstrator visualization tools are:

- *Node-Red* application preconfigured to act as a MES Simulator.
- *Grafana* application that has a built-in dashboard displaying the key parameters sent by the Equipment Simulator.

4.1 Node-Red

Node-RED is a tool for programming visually. It displays relations, functions and allows the user to program without having to code. Node-Red is a browser-based flow editor where you can add or remove nodes and wire them together in order to make them communicate with each other.

In the current demonstrator, Node-Red is preconfigured to act as a MES Simulator in order to provide commands/orders to the Equipment Simulator.

4.1.1 Access

Follow the steps below to access Node-Red:

• The Node-Red Docker container must be running.



• Open an Internet Browser (e.g., Chrome or Firefox) and navigate to http://localhost:1880/ui.

				n Smart Unifier	\equiv Incoming Data from	
Presure	Temperature	t	Info about Product	Failure	Order	
Module A	Module A		JobState	PercentFailure	Equipmentid 4-SWC2	
			Order	STARTFAILURE	OrderNumber Ord154	
			Product		ProductNumber Mv5	
			Module A Part		Quantity 10	
			Module B Part		RELEASEORDER	
Module B	Module B		Module C Part		STARTORDER	
			Job Started at:	-		
			Job Finished at:			
		sec	Time Difference			
Module C	Module C					
Module B Module C	Module B Module C	sec	Module A Part Module C Part Job Started at: Job Finished at: Time Difference		10 RELEASEORDER STARTORDER	

4.2 Grafana

Grafana is an open source analytics and interactive visualization web application. It provides charts, graphs and alerts for the web when connected to the supported data sources.

In the current demonstrator, Grafana is used to display the key parameters sent by the Equipment Simulator.

Access Grafana

Follow the steps bellow to access Grafana:

• Grafana Docker container must be running.



• Open an Internet Browser (e.g., Chrome or Firefox) and navigate to http://localhost:3333/. For "Username" and "Password" use **admin**.

Image: constraint of the startMelcome to GrafanaDon't get in the way of the data	
Email or username	
email or username	
Password	
password	
Log in	
Forgot your password?	

• Change the "Password" or just select the "Skip" button (1).

Welcome to Grafana	
Don't get in the way of the data	
New password	
Confirm new password	
Submit	
Skip 1	

• Select the "Home" button (2).

Ø	III Home 2				 E
Q +	Welcome to Grafana			Need help? Documentation	Tutorials Community Public Slack
88					Remove this name
Ø	Basic				Nellove Lins parter
4	The steps below will guide you to quickly	DITORIAL DATA SOURCE AND DASHBOARDS	Add your first data source	COMPLETE Create your first dashboard	
© 0	finish setting up your Grafana installation.	Gratana Tundamentals Set up and understand Grafana if you have no prior experience. Iudorial guides you through the entire process and covers the "D source" and "Dashboards" steps to the right.	This ata		
		ت	Learn how in the docs 🖉	Learn how in the docs 🖉	
	Dashb	boards		Latest from the blog	
			Thanks to its low cost and small size, the Raspberr showcased Raspberry Pi projects that use Grafana can follow this tutorial on how to install Grafana on check out Grafana Cloud, which is designed to be th	y Pi has become a popular hobbyst tool for running all sorts of a for monitoring homelab security and 3D printing. To get started your Raspberry Pi. If you want to use Grafana without having to he easiest way to get started creating dashboards and observing	oftware experiments. On our blog, we've with your own monitoring projects, you go through a full installation process, g metrics, logs, and traces.
			I'm a frontend software engineer at Grafana Labs b helping people grow in their careers, i've helped org community, as well as Google developer student ev technologies within the region.	ased in Nigeria. Over the past couple of years, driven by my inter anize developer meetups in Africa for organizations such as For rents. We are beginning to see a great surge in the amount of intr	est in building tech communities and loop Africa and Andela learning erest in open source cloud native
			One of the biggest challenges with data visualizatio some form of cloud-hosted solution. Traditionally it Grafana Cloud and the wider open source tooling in specialized systems and into battle-tested, open so	on for complicated software systems is getting quick access to t has required quite a bit of middleware and upfront setup with a firstructures such as Telegring, it is much easier to siphon off da surce metric and visualization tools such as Prometheus and Gra	he underlying data and connecting it to dditional tooling. But with the advent of ta from complicated and highly ifana.
0					
Q.			Setting up Prometheus to scrape your targets for m	netrics is usually just one part of your larger observability strateg	y. The other piece in the equation is

• Select "Siemens" options (3).

@	Search dashboards by name	×
Q		
+		
88	(2) General Sjørnons	
Ø		
¢		
6)		
Ū		

• The Grafana Siemens Dashboard is visible.

Ø	嘂 Siemens 쇼 ペ							ılıl* 🖨	© 🖵 ©	Last 5 mi	nutes v Q G	5 s ∨
Q +	Order No		Product No	Order QTY		Last Part Started	Parts OK		Parts Failed		Order Status	
88			Module A					Module B				
Ø		Temperature		Order Data			Temperature				Order Data	
¢				Metric							Metric	
\$												
Ø												
		Pressure					Pressure					
				N	lodule	e C						
				Temperature						Metric	Order Data	
				Pressure								
8												
0												

CHAPTER

START THE SIMULATION SCENARIO

The simulation scenario flow:

- 1. Input Data the order details.
- 2. Release Order sending the order details to the Equipment Simulator.
- 3. Start Order the Equipment Simulator starts the production.
- 4. *Start Failure* the Equipment Simulator starts the production of failed parts.

5.1 Input Data

The input data is structured into two categories, as seen in the Node-Red dashboard (MES Simulator):

1. Order:

- EquipmentId
- OrderNumber
- ProductNumber
- Quantity
- 2. Failure:
 - PercentFailure



5.2 Start the Simulation

The Demonstrator use case illustrates the connection between a MES system and a production equipment, connection facilitated by the SMART**UNIFIER**.

To run the simulation, three steps are required:

- Release Order
- Start Order
- Start Failure

Each simulation step can be initiated from the Node-Red Dashboard (MES Simulator). The simulation results will be visible in Grafana Dashboard.

5.2.1 Release Order

To start the simulation scenario, first input the order data (1) and release the order (2) from the Node-Red dashboard (MES Simulator).



Releasing of the order posts a message on the MQTT Broker that is picked up by the SMART**UNIFIER** Instance and sent to the Equipment Simulator.

5.2.2 Start Order

Second step is to Start the order (3) from the Node-Red dashboard. A new message is posted on the MQTT Broker and via the SMARTUNIFIER Instance sent to the Equipment Simulator instructing it to start the production.

	Order
	EquipmentId 4-SWC2
	OrderNumber Ord154
	ProductNumber Mv5
	Quantity 100
	RELEASEORDER
3	STARTORDER

The Equipment Simulator is built to have 3 production modules. The produced parts start in Module A continue production in Module B and finish on Module C. Once the Equipment Simulator starts to produce parts every Module will mark the production of a part.



The change that happens on each module is picked up by the SMARTUNIFIER and data is sent

simultaneously towards MQTT Broker and Influx Database in order to allow the follow up of the production using Grafana.

Ø	器 Siemens ☆ ペ						🛛 🕐 Last 5 minutes 🗸 🗧	Q 5s v
	Order No	Product No	Order QTY	Last Part Started	Parts OK	Parts Failed	Order Status	
Q +	Ord154	Mv5	100	34	34	0	OK	
88		Module A			Module B			
Ø	Tempera	ture	Order Data		Temperature		Order Data	
Å			Metric Value				Metric	Value
÷	80		ModuleA.LastPartNOTOK No data	90			ModuleB.LastPartNOTOK	No data
0			ModuleA.LastPartOK 34.00				ModuleB.LastPart0K	33.00
Ø	60 09:45:00 09:45:30 09:46:00 09:46:30 09:47:00	09:47:30 09:48:00 09:48:30 09:49:00 09:49	30 ModuleA.PercentFailure No data				ModuleB.PercentFailure	No data
	- ModuleA.Temperature		ModuleA.Pressure 170.00	60 09:45:00 09:45:30 09:46:00	09:46:30 09:47:00 09:47:30 09:48:00	09:48:30 09:49:00 09:49:30	ModuleB.Pressure	150.00
	Pressu		ModuleA.Quantity 100.00		Pressure		ModuleB.Quantity	100.00
			ModuleA.Temperature 80.00				ModuleB.Temperature	82.00
	125 09.4510 09.4510 09.4610 09.4610 09.4710	02-47-33 02-48-00 02-48-33 02-49-00 02-49	20	100 094500 094530 094600	094630 094700 094730 094800	ng 48-30 ng 49-00 ng 49-30		
	- ModuleA Pressure			- Module8.Pressure				
			Modu					
			Temperature				Order Data	
	110					Metric		Value
	90					Moduk	C.LastPartNOTOK	
	80					Module	C.LastPartOK	32.00
	00 00-00-00-00-00-00-00-00-00-00-00-00-0	094550 094600 094610 094620 094620 094640	0946-50 0947-00 0947-10 0947-20 0947-30 0947	E40 004250 004800 004810 004800 004	10-00 00-40-40 00-40-50 00-40-00 00-40-10 00-	e-an neve-an Module	C.LastPartStarted	No data
	- ModuleC.Temperature					Module	C.Pressure	120.00
			Pressure			Moduk	C.Quantity	100.00
						Moduk	C.Temperature	83.00
	120				•••••	•••••		
8	80							
୭	094440 094450 094500 094510 094520 094530 094540	09:45:50 09:46:00 09:46:10 09:46:20 09:46:30 09:46:40	09.46.50 09.47.00 09.47.10 09.47.20 09.47.30 09.4	140 09:47:50 09:48:00 09:48:10 09:48:20 09:4	8.30 09:48:40 09:48:50 09:49:00 09:49:10 09:	9:20 09:49:30		

5.2.3 Start Failure

The Equipment Simulator is capable to simulate production of failed parts by accessing the option "StartFailure" from the Node-RED dashboard. Add a value (percentage of failed parts to be generated by the Equipment Simulator) to the input box (4) and click the "StartFailure" button (5).





All the data provided by the Equipment Simulator is stored using the SMART**UNIFIER** in the Influx database and the overall process can be viewed on the Grafana dashboard.



5.3 Instance Setup

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, selected Communication Channels and Information Models.

5.3.1 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.

The Information Model perspective lists the information models currently configured within the SMART**UNIFIER** Manager:

- 1. Analytics
- 2. MES Simulator
- 3. Siemens S7PLC

I	Information Models						
	Group 🛧	Name					
	su.demo.dashboard	Analytics					
	su.demo.dashboard	MESSimulator					
	su.demo.dashboard	SiemensS7PLC					

1. Analytics

The Analytics represents the Influx database Information Model that stores data from the PLC of the production equipment (Equipment Simulator). The data provided by the Equipment Simulator (PLC) is stored using the SMART**UNIFIER** in the Influx database and the overall process can be viewed on the Grafana dashboard. As seen below, the data is structured for each production module.



C Edit Model: su.demo.dashboard:Analytics

1. MES Simulator

The MES Simulator Information Model represents the MES structure. It maps the production process flow, as seen below.

C Edit Model: su.demo.dashboard:MESSimulator
M MESSimulator
ReleaseOrder [ReleaseOrderType]
V Quantity [Int]
V ProductNumber [String]
V OrderNumber [String]
V EquipmentId [String]
∧ E StartOrder [StartOrderType]
V Quantity [Int]
V OrderNumber [String]
V ProductNumber [String]
V EquipmentId [String]
NotifyProductProduced [NotifyProductProducedType]
V State [String]
V OrderNumber [String]
EquipmentId [String]
NotifyProcessData_A [NotifyProcessDataType]



1. Siemens S7 PLC

The Siemens S7 PLC Information Model represents the PLC blocks structure. As seen below, the Model contains the input and output data for each production module.





5.3.2 Mappings

The Mapping represents the SMART**UNIFIER** component that defines 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.

The Mapping perspective lists the Mappings currently configured within the SMART**UNIFIER** Manager:

- 1. PLC to InfluxDb
- 2. PLC to MES Simulator

Mappings						
	Group 🛧	Name	Version	Models		
	su.demo.dashboard	PLCToInfluxDb	1.0.0	Analytics, SiemensS7PLC		
	su.demo.dashboard	PLCToMESSimulator	1.0.0	MESSimulator, SiemensS7PLC		

1. PLC to InfluxDb

This Mapping defines when and how to extract data from the PLC of the production equipment (Equipment Simulator) and store it on the InfluxDb.

Edit Mapping: su.demo.dashboard:PLCToInfluxDb:latest ~

Configuration			
Group * su.demo.dashboard			
Name * PLCToInfluxDb			
Description Defines when and how to	extract data from the PLC and store it on the InfluxDb		
Models			+
Short name db	Information model identifier * su.demo.dashboard:Analytics:latest	~	Ô
Short name plc	Information model identifier * su.demo.dashboard:SiemensS7PLC:latest	•	Ō

Rules

Q

C ModuleC	1	đ
<pre>1 * plc.ActiveOrder.LastPartStarted mapTo {{variable => 2 3 * scala.util.Try { 4 * if (variable.value > 0){ 6 * db.ModuleC.send(event => {</pre>		^
<pre>8 event.ProductNo := plc.ActiveOrder.Order.Information.ProductNo 9 event.StartOrder := plc.ActiveOrder.OrderInformation.Quantity 10 event.Quantity := plc.ActiveOrder.OrderInformation.Quantity 11 event.LastPartStarted := variable</pre>		•
C ModuleA	1	đ
<pre>1 * plc.ActiveOrder.LastPartStarted mapTo {variable => 2 3 * scala.util.Try { 4</pre>		Â
<pre>5 * if (variable.value > 0){ 6 * db.NoduleA.send(event => { 7 event.OrderNo := plc.ActiveOrder.OrderInformation.OrderNo 8 event.ProductNo := plc.ActiveOrder.OrderInformation.ProductNo 9 event.StartOrder := plc.ActiveOrder.StartOrder 10 event.Quantity := plc.ActiveOrder.OrderInformation.Quantity 11 event.LastPartStarted := variable</pre>		Ţ
C ModuleB		٥
1 v plc.ActiveOrder.LastPartStarted mapTo {variable => 2 3 v scala.util.Try {		Î
4 5 * if (variable.value > 0){ 6 * db.ModuleB.send(event => {		
<pre>/ event.urderNo := plc.ActiveOrder.UrderInformation.UrderNo 8 event.ProductNo := plc.ActiveOrder.OrderInformation.ProductNo 9 event.StartOrder := plc.ActiveOrder.StartOrder 10 event.Quantity := plc.ActiveOrder.OrderInformation.Quantity 11 event.LastPartStarted := variable</pre>		•

As seen above, for each production module, the rules created with Scala code lines define how the data exchange takes place.

1. PLC to MES Simulator

This Mapping defines data exchange between the PLC of the production equipment (Equipment Simulator) and the MES Simulator (Node-Red Dashboard).

🎽 Edit Mapping: su.o	lemo.dashboard:PLCToMESSimulator:latest ~	
Configuration		
Group *		
su.demo.dashboard		
Name *		
PLCToMESSimulator		
Description		
Defines data exchange bet	ween the PLC and the MES Simulator	
Models		+
Short name	Information model identifier *	
flow	su.demo.dashboard:MESSimulator:latest	•
Short name	Information model identifier *	
plc	su.demo.dashboard:SiemensS7PLC:latest	• 1

As seen below, for each production process, the rules created with Scala code lines define how the data exchange takes place.

Rules			Q
Release_Order	1	t	Ĵ
<pre>1 * flow.ReleaseOrder mapTo {{ event => 2 2 3 1 c.NewOrder.OrderInformation.Quantity := event.Quantity.toInt 4 plc.NewOrder.OrderInformation.ProductNu := event.ProductNumber 5 plc.NewOrder.OrderInformation.OrderNo := event.OrderNumber 6 plc.NewOrder.Ready := true 7 plc.EquipmentInformation.EquipmentType := event.EquipmentId 8 cl</pre>	rNumb	er}	")
Start_Order	,	t	
1 * flow.StartOrder mapTo {{ event => 2 plc.ActiveOrder.StartOrder := true			
Notify_Product_Produced	1	t	Ĵ
<pre>1 * plc.ActiveOrder.State mapTo {{ variable => 2 logger.info(s"Active order state: \${variable.value} - Processing Finished") 3 * if (variable.value == 3) { 4 * flow.NotifyProductProduced.send(event => { 5</pre>			

Notify_Result_Data	1
<pre>2 logger.into(s"Active order state: \${variable.value} - Processing Finished") 3 * if (variable.value == 3) { 4 * flow.NotifyResultData.send(event => { 5 event.Result := "t.b.n" 6 event.State := variable.toStr 7 event.ProductNumber := plc.ActiveOrder.OrderInformation.ProductNo 8 event.OrderNumber := plc.ActiveOrder.OrderInformation.OrderNo 9 event.EquipmentId := "t.b.n" 10 } 11 } 12 } 13 * 13 * 13 * 13 * 13 * 13 * 13 * 13 *</pre>	
Notify_Process_Data_A	1
<pre>1 v plc.Processing.Module_A.Part mapTo {{ variable => 2 logger.info(s"Active order state: \${variable.value} - Processing") 3 v // if (variable.value == 2) { 4 v flow.NotifyProcessData_A.send(event => { 5 event.Module := "Module A" 6 event.Pressure := plc.Processing.Module_A.Pressure.toStr 7 event.Temperature := plc.Processing.Module_A.Temperature.toStr 8 event.Step := plc.Processing.Module_A.Step.toStr 9 event.ProductNumber := plc.ActiveOrder.OrderInformation.ProductNo 8 event.OrderNumber := plc.ActiveOrder.OrderInformation.OrderNo</pre>	
event.lastPartStarted := nlc.Processing.Module A.Part	
Vent.LastPartStarted := nlc.Processing.Module A.Part Notify_Process_Data_B	1
<pre>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>	1
<pre>Notify_Process_Data_B v plc.Processing.Module_B.Part mapTo { variable => logger.info(s"Active order state: \${variable.value} - Processing") v flow.NotifyProcessData_B.send(event => { event.Module := "Module B" event.Pressure := plc.Processing.Module_B.Pressure.toStr event.Temperature := plc.Processing.Module_B.Temperature.toStr event.ProductNumber := plc.ActiveOrder.OrderInformation.ProductNo event.OrderNumber := plc.ActiveOrder.OrderInformation.OrderNo event.orderSure := plc.ActiveOrder.OrderInformation.OrderNo </pre>	
<pre>>> Processing.Module_B.Part mapTo {{ variable => logger.info(s"Active order state: \${variable.value} - Processing") >>> logger.info(s"Active order state: \${variable.value} - Processing") >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>	,
<pre>Notify_Process_Data_B vent.lastPartStarted := pic.Processing.Module A.Part plc.Process_Data_B v plc.Process_Data_B. v plc.Processing.Module_B.Part mapTo [{ variable => logger.info(s"Active order state: \${variable.value} - Processing") flow.NotifyProcessData_B.send(event => { event.Wodule := "Module B" event.Pressure := plc.Processing.Module_B.Pressure.toStr event.Temperature := plc.Processing.Module_B.Step.toStr event.ProductNumber := plc.ActiveOrder.OrderInformation.ProductNo event.lastPartStarted := nlc.Processing.Module B.Part Notify_Process_Data_C event.Module := "Module C" event.Temperature := plc.Processing.Module_C.Pressure.toStr event.Step := plc.Processing.Module_C.Temperature.toStr event.Step := plc.Processing.Module_C.Temperature.toStr event.Step := plc.Processing.Module_C.Step.toStr event.OrderNumber := plc.ActiveOrder.OrderInformation.ProductNo event.Step := plc.Processing.Module_C.Step.toStr event.OrderNumber := plc.ActiveOrder.OrderInformation.ProductNo event.Step := plc.Processing.Module_C.Part event.OrderNumber := plc.ActiveOrder.OrderInformation.ProductNo event.Step := plc.Processing.Module_C.Part event.EastPartStarted := plc.Processing.Module_C.Part event.EastPartStarted := plc.Processing.Module_C.Part event.EquipmentId := plc.EquipmentInformation.EquipmentType }) </pre>	,

```
1 * flow.StartFailure mapTo {{ event =>
2
3 *
4
5
6
7
7
7
8
8
}
```